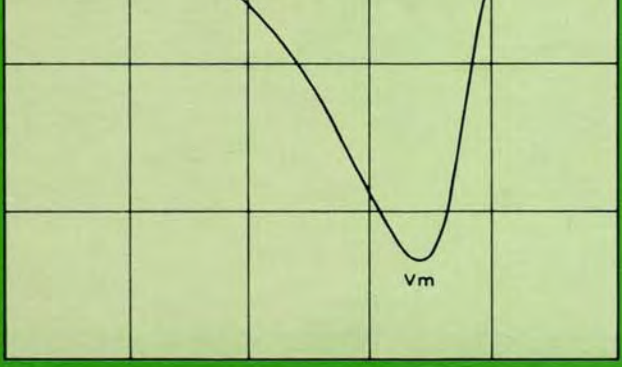
DATA HANDLING IN SCIENCE AND TECHNOLOGY

4

Advanced scientific computing in BASIC with applications in chemistry, biology and pharmacology

P. VALKÓ
S. VAJDA



ELSEVIER

DATA HANDLING IN SCIENCE AND TECHNOLOGY — VOLUME 4

Advanced scientific computing in BASIC with applications in chemistry, biology and pharmacology

DATA HANDLING IN SCIENCE AND TECHNOLOGY

Advisory Editors: B.G.M. Vandeginste, O.M. Kvalheim and L. Kaufman

Volumes in this series:

Volume 1 Microprocessor Programming and Applications for Scientists and Engineers by R.R. Smardzewski
Volume 2 Chemometrics: A textbook by D.L. Massart, B.G.M. Vandeginste, S.N. Deming, Y. Michotte and L. Kaufman
Volume 3 Experimental Design: A Chemometric Approach by S.N. Deming and S.N. Morgan
Volume 4 Advanced Scientific Computing in BASIC with Applications in Chemistry, Biology and Pharmacology by P. Valkó and S. Vajda

DATA HANDLING IN SCIENCE AND TECHNOLOGY — VOLUME 4

Advisory Editors: B.G.M. Vandeginste, O.M. Kvalheim and L. Kaufman

Advanced scientific computing in BASIC with applications in chemistry, biology and pharmacology

P. VALKÓ
Eötvös Loránd University, Budapest, Hungary

S. VAJDA
Mount Sinai School of Medicine, New York, NY, U.S.A.

ELSEVIER

Amsterdam — Oxford — New York — Tokyo 1989

ELSEVIER SCIENCE PUBLISHERS B.V.
Sara Burgerhartstraat 25
P.O. Box 211, 1000 AE Amsterdam, The Netherlands

Distributors for the United States and Canada:

ELSEVIER SCIENCE PUBLISHING COMPANY INC.
655, Avenue of the Americas
New York, NY 10010, U.S.A.

ISBN 0-444-87270-1 (Vol. 4) (software supplement 0-444-87217-X)
ISBN 0-444-42408-3 (Series)

© Elsevier Science Publishers B.V., 1989

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the publisher, Elsevier Science Publishers B.V./ Physical Sciences & Engineering Division, P.O. Box 330, 1000 AH Amsterdam, The Netherlands.

Special regulations for readers in the USA – This publication has been registered with the Copyright Clearance Center Inc. (CCC), Salem, Massachusetts. Information can be obtained from the CCC about conditions under which photocopies of parts of this publication may be made in the USA. All other copyright questions, including photocopying outside of the USA, should be referred to the publisher.

No responsibility is assumed by the Publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein.

Although all advertising material is expected to conform to ethical (medical) standards, inclusion in this publication does not constitute a guarantee or endorsement of the quality or value of such product or of the claims made of it by its manufacturer.

Printed in The Netherlands

CONTENTS

INTRODUCTION

This book is a practical introduction to scientific computing and offers BASIC subroutines, suitable for use on a personal computer, for solving a number of important problems in the areas of chemistry, biology and pharmacology. Although our text is advanced in its category, we assume only that you have the normal mathematical preparation associated with an undergraduate degree in science, and that you have some familiarity with the BASIC programming language. We obviously do not persuade you to perform quantum chemistry or molecular dynamics calculations on a PC , these topics are even not considered here. There are, however, important information - handling needs that can be performed very effectively. A PC can be used to model many experiments and provide information what should be expected as a result. In the observation and analysis stages of an experiment it can acquire raw data and exploring various assumptions aid the detailed analysis that turns raw data into timely information. The information gained from the data can be easily manipulated, correlated and stored for further use. Thus the PC has the potential to be the major tool used to design and perform experiments, capture results, analyse data and organize information.

Why do we use BASIC? Although we disagree with strong proponents of one or another programming language who challenge the use of anything else on either technical or purely emotional grounds, most BASIC dialects certainly have limitations. First, by the lack of local variables it is not easy to write multilevel, highly segmented programs. For example, in FORTRAN you can use subroutines as "black boxes" that perform some operations in a largely unknown way, whereas programming in BASIC requires to open these black boxes up to certain degree. We do not think, however, that this is a disadvantage for the purpose of a book supposed to teach you numerical methods. Second, BASIC is an interpretive language, not very efficient for programs that do a large amount of "number - crunching" or programs that are to be run many times. But the loss of execution speed is compensated by the interpreter's ability to enable you to interactively enter a program, immediately execute it and see the results without stopping to compile and link the program. There exists no more convenient language to understand how a numerical method works. BASIC is also superb for writing relatively small, quickly needed programs of less than 1000 program lines with a minimum programming effort. Errors can be found and corrected in seconds rather than in hours, and the machine can be immediately quizzed for a further explanation of questionable answers or for exploring further aspects of the problem. In addition, once the program runs properly, you can use a BASIC compiler to make it run faster. It is also important that on most PC's BASIC is usually very powerful for using all resources, including graphics, color, sound and communication devices, although such aspects will not be discussed in this book.

Why do we claim that our text is advanced? We believe that the methods and programs presented here can handle a number of realistic problems with the power and sophistication needed by professionals and with simple, step - by - step introductions for students and beginners. In spite of their broad range of applicability, the subroutines are simple enough to be completely understood and controlled, thereby giving more confidence in results than software packages with unknown source code.

Why do we call our subject scientific computing? First, we assume that you, the reader, have particular problems to solve, and do not want to teach you neither chemistry nor biology. The basic task we consider is extracting useful information from measurements via modelling, simulation and data evaluation, and the methods you need are very similar whatever your particular application is. More specific examples are included only in the last sections of each chapter to show the power of some methods in special situations and promote a critical approach leading to further investigation. Second, this book is not a course in numerical analysis, and we disregard a number of traditional topics such as function approximation, special functions and numerical integration of known functions. These are discussed in many excellent books, frequently with BASIC subroutines included. You will find here, however, some efficient and robust numerical methods that are well established in important scientific applications. For each class of problems we give an introduction to the relevant theory and techniques that should enable you to recognize and use the appropriate methods. Simple test examples are chosen for illustration. Although these examples naturally have a numerical bias, the dominant theme in this book is that numerical methods are no substitute for poor analysis. Therefore, we give due consideration to problem formulation and exploit every opportunity to emphasize that this step not only facilitates your calculations, but may help you to avoid questionable results. There is nothing more alien to scientific computing than the use of highly sophisticated numerical techniques for solving very difficult problems that have been made so difficult only by the lack of insight when casting the original problem into mathematical form.

What is in this book? It consists of five chapters. The purpose of the preparatory Chapter 1 is twofold. First, it gives a practical introduction to basic concepts of linear algebra, enabling you to understand the beauty of a linear world. A few pages will lead to comprehending the details of the two - phase simplex method of linear programming. Second, you will learn efficient numerical procedures for solving simultaneous linear equations, inversion of matrices and eigenanalysis. The corresponding subroutines are extensively used in further chapters and play an indispensable auxiliary role. Among the direct applications we discuss stoichiometry of chemically reacting systems, robust parameter estimation methods based on linear programming, as well as elements of principal component analysis.

Chapter 2 gives an overview of iterative methods of solving nonlinear equations and optimization problems of one or several variables. Though the one variable case is treated in many similar books, we include the corresponding simple subroutines since working with them may help you to fully understand the use of user supplied subroutines. For solution of simultaneous nonlinear equations and multivariable optimization problems some well established methods have been selected that also amplify the theory. Relative merits of different methods are briefly discussed. As applications we deal with equilibrium problems and include a general program for computing chemical equilibria of gaseous mixtures.

Chapter 3 plays a central role. It concerns estimation of parameters in complex models from relatively small samples as frequently encountered in scientific applications. To demonstrate principles and interpretation of estimates we begin with two linear statistical methods (namely, fitting a line to a set of points and a subroutine for multivariable linear regression), but the real emphasis is placed on nonlinear problems. After presenting a robust and efficient general purpose nonlinear least squares estimation procedure we proceed to more involved methods, such as the multiresponse estimation of Box and Draper, equilibrating balance equations and fitting error-in-variables models. Though the importance of these techniques is emphasized in the statistical literature, no easy-to-use programs are available. The chapter is concluded by presenting a subroutine for fitting orthogonal polynomials and a brief summary of experiment design approaches relevant to parameter estimation. The text has a numerical bias with brief discussion of statistical background enabling you to select a method and interpret results. Some practical aspects of parameter estimation such as near-singularity, linearization, weighting, reparametrization and selecting a model from a homologous family are discussed in more detail.

Chapter 4 is devoted to signal processing. Through in most experiments we record some quantity as a function of an independent variable (e.g., time, frequency), the form of this relationship is frequently unknown and the methods of the previous chapter do not apply. This chapter gives a summary of classical techniques for interpolating, smoothing, differentiating and integrating such data sequences. The same problems are also solved using spline functions and discrete Fourier transformation methods. Applications in potentiometric titration and spectroscopy are discussed.

The first two sections of Chapter 5 give a practical introduction to dynamic models and their numerical solution. In addition to some classical methods, an efficient procedure is presented for solving systems of stiff differential equations frequently encountered in chemistry and biology. Sensitivity analysis of dynamic models and their reduction based on quasy-steady-state approximation are discussed. The second central problem of this chapter is estimating parameters in ordinary differential equations. An efficient short-cut method designed specifically for PC's is presented and applied to parameter estimation, numerical deconvolution and input determination. Application examples concern enzyme kinetics and pharmacokinetic compartmental modelling.

Program modules and sample programs

For each method discussed in the book you will find a BASIC subroutine and an example consisting of a test problem and the sample program we use to solve it. Our main assets are the subroutines we call program modules in order to distinguish them from the problem dependent user supplied subroutines. These modules will serve you as building blocks when developing a program of your own and are designed to be applicable in a wide range of problem areas. To this end concise information for their use is provided in remark lines. Selection of available names and program line numbers allow you to load the modules in virtually any combination. Several program modules call other module(s). Since all variable names consist of two characters at the most, introducing longer names in your own user supplied subroutines avoids any conflicts. These user supplied subroutines start at lines 600, 700, 800 and 900 , depending on the need of the particular module. Results are stored for further use and not printed within the program module. Exceptions are the ones corresponding to parameter estimation, where we wanted to save you from the additional work of printing large amount of intermediate and final results. You will not find dimension statements in the modules, they are placed in the calling sample programs. The following table lists our program modules.

Table 1
Program modules

Identifier	Purpose	First line	Last line
M10	Vector coordinates in a new basis	1000	1044
M11	Linear programming two phase simplex method	1100	1342
M14	LU decomposition of a square matrix	1400	1460
M15	Solution of simultaneous linear equations backward substitution using LU factors	1500	1538
M16	Inversion of a positive definite symmetric matrix	1600	1656
M17	Linear equations with tridiagonal matrix	1700	1740
M18	Eigenvalues and eigenvectors of a symmetric matrix - Jacobi method	1800	1938
M20	Solution of a cubic equation - Cardano method	2000	2078
M21	Solution of a nonlinear equation bisection method	2100	2150
M22	Solution of a nonlinear equation regula falsi method	2200	2254
M23	Solution of a nonlinear equation secant method	2300	2354
M24	Solution of a nonlinear equation Newton-Raphson method	2400	2454
M25	Minimum of a function of one variable method of golden sections	2500	2548
M26	Minimum of a function of one variable parabolic interpolation - Brent's method	2600	2698
M30	Solution of simultaneous equations X=G(X) Wegstein method	3000	3074
M31	Solution of simultaneous equations F(X)=0 Newton-Raphson method	3100	3184
M32	Solution of simultaneous equations F(X)=0 Broyden method	3200	3336
M34	Minimization of a function of several variables Nelder-Mead method	3400	3564
M36	Minimization of a function of several variables Davidon-Fletcher-Powell method	3600	3794
M40	Fitting a straight line by linear regression	4000	4096
M41	Critical t-value at 95 % confidence level	4100	4156
M42	Multivariable linear regression weighted least squares	4200	4454
M45	Weighted least squares estimation of parameters in multivariable nonlinear models Gauss-Newton-Marquardt method	4500	4934
M50	Equilibrating linear balance equations by least squares method and outlier analysis	5000	5130

M52	Fitting an error-in-variables model of the form F(Z,P)=0 modified Patino-Leal - Reilly method	5200	5460
M55	Polynomial regression using Forsythe orthogonal polynomials	5500	5628
M60	Newton interpolation: computation of polynomial coefficients and interpolated values	6000	6054
M61	Local cubic interpolation	6100	6156
M62	5-point cubic smoothing by Savitzky and Golay	6200	6250
M63	Determination of interpolating cubic spline	6300	6392
M64	Function value, derivatives and definite integral of a cubic spline at a given point	6400	6450
M65	Determination of smoothing cubic spline method of C.H. Reinsch	6500	6662
M67	Fast Fourier transform Radix-2 algorithm of Cooley and Tukey	6700	6782
M70	Solution of ordinary differential equations fourth order Runga-Kutta method	7000	7058
M71	Solution of ordinary differential equations predictor-corrector method of Milne	7100	7188
M72	Solution of stiff differential equations semi-implicit Runge-Kutta method with backsteps Rosenbrock-Gottwald-Wanner	7200	7416
M75	Estimation of parameters in differential equations by direct integral method extension of the Himmelblau-Jones-Bischoff method	7500	8040

While the program modules are for general application, each sample program is mainly for demonstrating the use of a particular module. To this end the programs are kept as concise as possible by specifying input data for the actual problem in the DATA statements. Thus test examples can be checked simply by loading the corresponding sample program, carefully merging the required modules and running the obtained program. To solve your own problems you should replace DATA lines and the user supplied subroutines (if needed). In more advanced applications the READ and DATA statements may be replaced by interactive input. The following table lists the sample programs.

THE PROGRAM MODULES AND THE SAMPLE PROGRAMS ARE AVAILABLE ON DISKETTE, SUITABLE FOR MS-DOS COMPUTERS. THE DISKETTE CAN BE ORDERED SEPARATELY. PLEASE, SEE THE ORDER CARD IN THE FRONT OF THIS BOOK.

Table 2
Sample programs

Identifier	Example	Title	Modules called
EX112	1.1.2	Vector coordinates in a new basis	M10
EX114	1.1.4	Inversion of a matrix by Gauss-Jordan elimination	see EX112
EX12	1.2	Linear programming by two phase simplex method	M10,M11
EX132	1.3.2	Determinant by LU decomposition	M14
EX133	1.3.3	Solution of linear equations by LU decomposition	M14,M15
EX134	1.3.4	Inversion of a matrix by LU decomposition	M14,M15
EX14	1.4	Inversion of a positive definite symmetric matrix	M16
EX15	1.5	Solution of linear equations with tridiagonal matrix	M17
EX16	1.6	Eigenvalue-eigenvector decomposition of a sym. matrix	M18
EX182	1.8.2	Fitting a line - least absolute deviations	see EX12
EX183	1.8.3	Fitting a line - minimax method	see EX12
EX184	1.8.4	Analysis of spectroscopic data with background	see EX12
EX211	2.1.1	Molar volume by Cardano method	M20
EX212	2.1.2	Molar volume by bisection	M21
EX221	2.2.1	Optimum dosing by golden section method	M25
EX231	2.3.1	Reaction equilibrium by Wegstein method	M30
EX232	2.3.2	Reaction equilibrium by Newton-Raphson method	M14,M15,M31
EX241	2.4.1	Rosenbrock problem by Nelder-Mead method	M34
EX242	2.4.2	Rosenbrock problem by Davidon-Fletcher-Powell method	M36
EX253	2.5.3	Liquid-liquid equilibrium by Broyden method	M32
EX254	2.5.4	Chemical equilibrium of gaseous mixtures	M14,M15
EX31	3.1	Fitting a regression line	M40,M41
EX32	3.2	Multivariable linear regression - acid catalysis	M16,M18,M41,M42
EX33	3.3	Nonlinear LSQ parameter estimation - Bard example	M16,M18,M41,M45
EX37	3.7	Equilibrating linear balances	M16,M50
EX38	3.8	Error-in-variables parameter estimation - calibration	M16,M18,M41,M45, M52

EX39	3.9	Polynomial regression using Forsythe orthogonal polynomials	M55
EX3104	3.10.4	Van Laar parameter estimation (error-in-variables method)	M16,M18,M41,M45, M52
EX411	4.1.1	Newton interpolation	M60
EX413	4.1.3	Smoothed derivatives by Savitzky and Golay	M62
EX421	4.2.1	Spline interpolation	M63,M64
EX422	4.2.2	Smoothing by spline	M65
EX433	4.3.3	Application of FFT techniques	M67
EX511	5.1.1	Fermentation kinetics by Runge-Kutta method	M70
EX52	5.2	Solution of the Oregonator model by semi-implicit method	M14,M15,M72
EX53	5.3	Sensitivity analysis of a microbial growth process	M14,M15,M72
EX55	5.5	Direct integral parameter estimation	M14,M15,M16,M18, M41,M63,M72,M75
EX56	5.6	Direct integral identification of a linear system	M16,M18,M41,M42, M63
EX57	5.7	Input function determination to a given response	see EX56

Program portability

We have attempted to make the programs in this book as generally useful as possible, not just in terms of the subjects concerned, but also in terms of their degree of portability among different PC's. This is not easy in BASIC, since the recent interpreters and compilers are usually much more generous in terms of options than the original version of BASIC developed by John Kemeny and Thomas Kurtz. Standardization did not keep up with the various improvements made to the language. Restricting consideration to the common subset of different BASIC dialects would mean to give up some very comfortable enhancements introduced during the last decade, a price too high for complete compatibility. Therefore, we choose the popular Microsoft's BASIC that comes installed on the IBM PC family of computers and clones under the name (disk) BASIC, BASICA or GWBASIC. A disk of MS DOS (i.e., PC DOS) format, containing all programs listed in Tables 1 and 2 is available for purchase. If you plan to use more than a few of the programs in this book and you work with an IBM PC or compatible, you may find it useful to obtain a copy of the disk in order to save time required for typing and debugging. If you have the sample programs and the program modules on disk, it is very easy to run a test example. For instance, to reproduce Example 4.2.2 you should start your BASIC , then load the file "EX422.BAS", merge the file "M65.BAS" and run the program. In order to ease merging the programs they are saved in ASCII format on the disk. You will need a printer since the programs are written with LPRINT statements. If you prefer printing to the screen, you may change all the LPRINT statements to PRINT statements, using the editing facility of the BASIC interpreter or the more user friendly change option of any editor program.

Using our programs in other BASIC dialects you may experience some difficulties. For example, several dialects do not allow zero indices of an array, restrict the feasible names of variables, give +1 instead of -1 for a logical expression if it is true, do not allow the structure IF ... THEN ... ELSE, have other syntax for formatting a PRINT statement, etc. According to our experience, the most dangerous effects are connected with the different treatment of FOR ... NEXT loops. In some versions of the language the statements inside a loop are carried out once, even if the loop condition does not allow it. If running the following program

```
10 FOR I=2 TO 1
20  PRINT "IF YOU SEE THIS, THEN YOU SHOULD BE CAREFUL WITH YOUR BASIC"
30 NEXT I
```

will result in no output, then you have no reason to worry. Otherwise you will find it necessary to insert a test before each FOR ... NEXT loop that can be empty. For example, in the module M15 the loop in line 1532 is empty if 1 is greater than K - 1 (i.e., K < 2) , thus the line

```
1531 IF K<2 THEN 1534
```

inserted into the module will prevent unpredictable results.

We deliberately avoided the use of some elegant constructions as WHILE ... WEND structure, SWAP statement, ON ERROR condition and never broke up a single statement into several lines. Although this self-restraint implies that we had to give up some principles of structural programming (e.g., we used more GOTO statements than it was absolutely necessary), we think that the loss is compensated by the improved portability of the programs.

Note to the reader

Of course we would be foolish to claim that there are no bugs in such a large number of program lines. We tried to be very careful and tested the program modules on various problems. Nevertheless, a new problem may lead to difficulties that we overlooked. Therefore, we make no warranties, express or implied, that the programs contained in this book are free of error, or are consistent with any particular standard of merchantibility, or that they will meet your requirements for any particular application. The authors and publishers disclaim all liability for direct or consequential damages resulting from the use of the programs.

Acknowledgements

This book is partly based on a previous work of the authors, published in Hungarian by Müszaki Könyvkiadó, Budapest, in 1987. We wish to thank to the editor of the Hungarian edition, Dr J. Bendl, for his support. We also gratefully acknowledge the positive stimuli provided by Dr. P. Szepesváry, Prof. J.T. Clerc and our colleagues and students at the Eötvös Loránd University, Budapest. While preparing the present book, the second author was affiliated also with the Department of Chemistry at Princeton University, and he is indebted for the stimulating environment.

Chapter 1

COMPUTATIONAL LINEAR ALGEBRA

The problems we are going to study come from chemistry, biology or pharmacology, and most of them involve highly nonlinear relationships. Nevertheless, there is almost no example in this book which could have been solved without linear algebraic methods. Moreover, in most cases the success of solving the entire problem heavily depends on the accuracy and the efficiency in the algebraic computation.

We assume most readers have already had some exposure to linear algebra, but provide a quick review of basic concepts. As usual, our notations are

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_m \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & a_{1m} \\ a_{21} & a_{22} & \cdot & \cdot & \cdot & a_{2m} \\ \cdot & & & & & \\ \cdot & & & & & \\ \cdot & & & & & \\ a_{n1} & a_{n2} & \cdot & \cdot & \cdot & a_{nm} \end{bmatrix}, \tag{1.1}$$

where $\mathbf{x}$ is the m-vector of the elements $[\mathbf{x}]_i$, and $\mathbf{A}$ is the $n \times m$ matrix of the elements $[\mathbf{A}]_{ij} = a_{ij}$. Consider a scalar s, another m-vector $\mathbf{y}$, and an $m \times p$ matrix $\mathbf{B}$. The basic operations on vectors and matrices are defined as follows:

$$\left[\mathbf{x}+\mathbf{y}\right]_i = x_i+y_i, \quad \left[s\mathbf{x}\right]_i = sx_i, \quad \left[\mathbf{A}\mathbf{x}\right]_i = \sum_{j=1}^{m} a_{ij}x_{ij}, \quad \left[s\mathbf{A}\right]_{ij} = sa_{ij},$$
$$\left[\mathbf{A}\mathbf{B}\right]_{ij} = \sum_{k=1}^{m} a_{ik}b_{kj}, \quad \left[\mathbf{A}^T\right]_{ij} = a_{ji}, \qquad \mathbf{x}^T\mathbf{y} = \sum_{i=1}^{m} x_iy_i, \tag{1.2}$$

where $\mathbf{x}^T\mathbf{y}$ is called the scalar product of $\mathbf{x}$ and $\mathbf{y}$. We will also need the Euclidean norm or simply the length of $\mathbf{x}$, defined by $\|\mathbf{x}\| = (\mathbf{x}^T\mathbf{x})^{1/2}$.

The most important computational tasks considered in this chapter are as follows:

□ Solution of the matrix equation

$$\mathbf{A}\mathbf{x} = \mathbf{b} , \qquad (1.3)$$

where $\mathbf{A}$ is an $n \times m$ matrix of known coefficients, $\mathbf{b}$ is a known right-hand side vector of dimension n, and we want to find the m-vector $\mathbf{x}$ that satisfies (1.3).

- Calculation of the matrix $\mathbf{A}^{-1}$ which is the matrix inverse of an $n \times n$ square matrix $\mathbf{A}$, that is

  $$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I} , \qquad (1.4)$$

  where $\mathbf{I}$ is the $n \times n$ identity matrix defined by $[\mathbf{I}]_{ij} = 0$ for $i \neq j$, and $[\mathbf{I}]_{ii} = 1$.

- Let $\mathbf{a}$ and $\mathbf{b}$ be vectors of dimension n. The inequality $\mathbf{a} \leq \mathbf{b}$ means $a_i \leq b_i$ for all $i = 1,\ldots,n$. In the linear programming problem we want to find the m-vector $\mathbf{x}$ which will maximize the linear function

  $$z = \mathbf{c}^T\mathbf{x} \qquad (1.5)$$

  subject to the restrictions

  $$\mathbf{A}\mathbf{x} \leq \mathbf{b} , \quad \mathbf{x} \geq \mathbf{0} . \qquad (1.6)$$

  As we show in Section 1.2, a more general class of problems can be treated similarly.

- Solution of eigenvalue-eigenvector problems, where we find the eigenvalue λ and the eigenvector $\mathbf{u}$ of the square symmetric matrix $\mathbf{A}$ such that

  $$\mathbf{A}\mathbf{u} = \lambda\mathbf{u} . \qquad (1.7)$$

These problems are very important and treated in many excellent books, for example (refs. 1-6). Though the numerical methods can be presented as recipes, i.e. , sequences of arithmetic operations, we feel that their essence would be lost without fully understanding the underlying concepts of linear algebra, reviewed in the next section.

1.1 BASIC CONCEPTS AND METHODS

1.1.1 Linear vector spaces

The goal of this section is to extend some concepts of 3-dimensional space R^3 to n dimensions, and hence we start with R^3, the world we live in. Considering the components a_{11}, a_{21} and a_{31} of the vector $\mathbf{a}_1 = (a_{11}, a_{21}, , a_{31})^T$ as coordinates, $\mathbf{a}_1$ is shown in Fig. 1.1. In terms of these coordinates

$a_1 = a_{11}e_1 + a_{21}e_2 + a_{31}e_3$, where e_i denotes the i-th unit vector defined

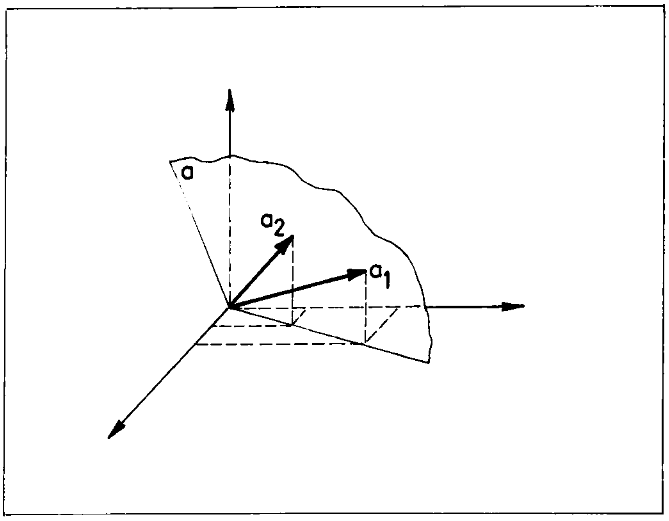


Fig. 1.1. Subspace in 3-dimensional space

by $[e_i]_i = 1$, $[e_i]_j = 0$, $i \neq j$. If s is a scalar and a_2 is a vector in R^3, then sa_1 and a_1+a_2 are also 3-dimensional vectors, and the vector space is closed under multiplication by scalars, and addition. This is the fundamental property of any linear vector space. Consider the vectors a_1 and a_2 in Fig. 1.1, which are not on the same line. The set of linear combinations $s_1a_1 + s_2a_2$, where s_1 and s_2 are arbitrary scalars, is a plane in R^3. If b_1 and b_2 are any vectors in this plane, then sb_1 and $b_1 + b_2$ are also in the plane, which is therefore closed under multiplication by scalars and addition. Thus the plane generated by all linear combinations of the form $s_1a_1 + s_2a_2$ is also a linear vector space, a 2-dimensional subspace of R^3. Any vector in this subspace is of the form $b = s_1a_1 + s_2a_2$, and hence can be described in terms of the coordinates $b = (s_1,s_2)^T$ in the coordinate system defined by the vectors a_1 and a_2 . We can, however, select another system of coordinates (e.g., two perpendicular vectors of unit length in the plane). If a_1 and a_2 are collinear, i.e., are on the same line, then the combinations $s_1a_1 + s_2a_2$ define only this line, a one dimensional subspace of R^3.

To generalize these well known concepts consider the n-vectors a_1, a_2, ...,

a_m , given by

$$a_1 = \begin{bmatrix} a_{11} \\ a_{21} \\ \cdot \\ \cdot \\ \cdot \\ a_{n1} \end{bmatrix}, \quad a_2 = \begin{bmatrix} a_{12} \\ a_{22} \\ \cdot \\ \cdot \\ \cdot \\ a_{n2} \end{bmatrix}, \quad . \; . \; ., \; a_m = \begin{bmatrix} a_{1m} \\ a_{2m} \\ \cdot \\ \cdot \\ \cdot \\ a_{nm} \end{bmatrix} . \qquad (1.8)$$

The linear combinations

$$b = s_1a_1 + s_2a_2 + \ldots + s_ma_m \qquad (1.9)$$

form a subspace of R^n which is said to be spanned by the vectors $a_1,\ldots,a_m$. We face a number of questions concerning the structure of this subspace. Do we need all vectors a_1, a_2, ..., a_m to span the subspace or some of them could be dropped? Do these vectors span the whole space R^n ? How to choose a system of coordinates in the subspace? The answers to these questions are based on the concept of linear independence. The vectors a_1, a_2, ..., a_m are said to be linearly independent if the equality

$$s_1a_1 + s_2a_2 + \ldots + s_ma_m = 0 \qquad (1.10)$$

implies $s_1 = s_2 = \ldots s_m = 0$. Otherwise the vectors a_1, a_2, ..., a_m are said to be linearly dependent. In this latter case we can solve (1.10) such that at least one of the coefficients is nonzero. Let $s_i \neq 0$, then a_i can be expressed from (1.10) as the linear combination

$$a_i = -\frac{s_1}{s_i}a_1 - \ldots - \frac{s_{i-1}}{s_i}a_{i-1} - \frac{s_{i+1}}{s_i}a_{i+1} - \ldots - \frac{s_m}{s_i}a_m \qquad (1.11)$$

of the other vectors in the system. It is now clear that we can restrict consideration to linearly independent vectors when defining a subspace. Assume that there exists only r independent vectors among a_1, a_2, ..., a_m, i.e., any set of $r+1$ vectors is linearly dependent. Then the integer r is said to be the rank of the vector system, and also define the dimension of the subspace spanned by these vectors.

Let $a_1,a_2,\ldots,a_r$ be a linearly independent subset of vectors $a_1,a_2,\ldots,a_m$ with rank r . Any vector in the subspace can be expressed as a linear combination of $a_1,a_2,\ldots,a_r$, thus these latter can be regarded to form a coordinate system in the subspace, also called a basis of the subspace. Since any such set of r linearly independent vectors forms a basis, it is obviously not unique.

If $r = n$, then the linearly independent vectors span the entire n-dimensional space. Again one can choose any n linearly independent vectors as a basis of the space. The unit vectors

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 0 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}, \quad \ldots, \quad e_n = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} \tag{1.12}$$

clearly are linearly independent. This is the canonical basis for R^n, and the components a_{ij} of the vectors (1.8) are coordinates in the canonical basis, if not otherwise stated.

1.1.2 <u>Vector coordinates in a new basis</u>

In practice a vector $\mathbf{a}_i$ is specified by its coordinates $(a_{1i}, a_{2i}, \ldots, a_{ni})^T$ in a particular basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$. For example the vectors (1.8) can be represented by the matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & \cdot & a_{1m} \\ a_{21} & a_{22} & \cdot & \cdot & \cdot & a_{2m} \\ \cdot & & & & & \\ \cdot & & & & & \\ \cdot & & & & & \\ a_{n1} & a_{n2} & \cdot & \cdot & \cdot & a_{nm} \end{bmatrix}, \tag{1.13}$$

where the coordinates a_{ij} do not necessarily correspond to the canonical basis. It will be important to see how the coordinates change if the vector $\mathbf{b}_p$ of the starting basis is replaced by $\mathbf{a}_q$. We first write the intended new basis vector $\mathbf{a}_q$ and any further vector $\mathbf{a}_j$ as

$$\mathbf{a}_q = a_{1q}\mathbf{b}_1 + a_{2q}\mathbf{b}_2 + \ldots + a_{pq}\mathbf{b}_p + \ldots + a_{nq}\mathbf{b}_n \tag{1.14}$$

$$\mathbf{a}_j = a_{1j}\mathbf{b}_1 + a_{2j}\mathbf{b}_2 + \ldots + a_{pj}\mathbf{b}_p + \ldots + a_{nj}\mathbf{b}_n \,. \tag{1.15}$$

If $a_{pq} \neq 0$, then from (1.14)

$$\mathbf{b}_p = -\frac{a_{1q}}{a_{pq}}\mathbf{b}_1 - \frac{a_{2q}}{a_{pq}}\mathbf{b}_2 - \ldots - \frac{a_{p-1,q}}{a_{pq}}\mathbf{b}_{p-1} + \frac{1}{a_{pq}}\mathbf{a}_q - \frac{a_{p+1,q}}{a_{pq}}\mathbf{b}_{p+1} - \ldots - \frac{a_{nq}}{a_{pq}}\mathbf{b}_n \,. \tag{1.16}$$

Introducing this expression of $\mathbf{b}_p$ into (1.15) and rearranging we have

$$\mathbf{a}_j = \left[a_{1j} - \frac{a_{pj}}{a_{pq}}a_{1q}\right]\mathbf{b}_1 + \left[a_{2j} - \frac{a_{pj}}{a_{pq}}a_{2q}\right]\mathbf{b}_2 + \dots + \left[a_{p-1,j} - \frac{a_{pj}}{a_{pq}}a_{p-1,q}\right]\mathbf{b}_{p-1} +$$

$$+ \left[\frac{a_{pj}}{a_{pq}}\right]\mathbf{a}_q + \left[a_{p+1,j} - \frac{a_{pj}}{a_{pq}}a_{p+1,q}\right]\mathbf{b}_{p+1} + \dots + \left[a_{nj} - \frac{a_{pj}}{a_{pq}}a_{nq}\right]\mathbf{b}_n \ . \qquad (1.17)$$

Since (1.17) gives $\mathbf{a}_j$ as a linear combination of the vectors $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{p-1}, \mathbf{a}_q, \mathbf{b}_{p+1}, \dots, \mathbf{b}_n$, its coefficients

$$a'_{ij} = a_{ij} - \frac{a_{iq}}{a_{pq}}a_{pj} \quad \text{for} \quad i \neq p \qquad \text{and} \qquad a'_{pj} = \frac{a_{pj}}{a_{pq}} \qquad (1.18)$$

are the coordinates of $\mathbf{a}_j$ in the new basis. The vector $\mathbf{b}_p$ can be replaced by $\mathbf{a}_q$ in the basis if and only if the pivot element (or pivot) a_{pq} is nonzero, since this is the element we divide by in the transformations (1.18).

The first BASIC program module of this book performs the coordinate transformations (1.18) when one of the basis vectors is replaced by a new one.

<u>Program module M10</u>

```
1000 REM ****************************************************
1002 REM *        VECTOR COORDINATES IN A NEW BASIS         *
1004 REM ****************************************************
1006 REM INPUT:
1008 REM     N      DIMENSION OF VECTORS
1010 REM     M      NUMBER OF VECTORS
1012 REM     IP     ROW INDEX OF THE PIVOT
1014 REM     JP     COLUMN INDEX OF THE PIVOT
1016 REM    A(N,M)  TABLE OF VECTOR COORDINATES
1018 REM OUTPUT:
1020 REM    A(N,M)  VECTOR COORDINATES IN THE NEW BASIS
1022 A=A(IP,JP)
1024 FOR J=1 TO M :A(IP,J)=A(IP,J)/A :NEXT J
1026 FOR I=1 TO N
1028  IF I=IP THEN 1038
1030  A=A(I,JP) :IF A=0 THEN 1038
1032  FOR J=1 TO M
1034   IF A(IP,J)<>0 THEN A(I,J)=A(I,J)-A(IP,J)*A
1036  NEXT J
1038 NEXT I
1040 A(0,A(IP,0))=0 :A(IP,0)=JP :A(0,JP)=IP
1042 RETURN
1044 REM ****************************************************
```

The vector coordinates (1.13) occupy the array A(N,M). The module will replace the IP-th basis vector by the JP-th vector of the system. The pivot element is A(IP,JP). Since the module does not check whether A(IP,JP) is nonzero, you should do this when selecting the pivot. The information on the current basis is stored in the entries A(0,J) and A(I,0) as follows:

$$A(I,\emptyset) = \begin{cases} \emptyset \text{ if the I-th basis vector is } \mathbf{e}_i \\ J \text{ if the I-th basis vector is } \mathbf{a}_j \end{cases}$$

$$A(\emptyset,J) = \begin{cases} \emptyset \text{ if } \mathbf{a}_j \text{ is not present in basis} \\ I \text{ if } \mathbf{a}_j \text{ is the I-th basis vector.} \end{cases}$$

The entry A(Ø,Ø) is a dummy variable.

If the initial coordinates in array A correspond to the canonical basis, we set A(I,Ø) = A(Ø,J) = Ø for all I and J . Notice that the elements A(Ø,J) can be obtained from the values in A(I,Ø), thus we store redundant information. This redundancy, however, will be advantageous in the programs that call the module M1Ø.

Example 1.1.2 Transformation of vector coordinates.

Assume that the vectors

$$\mathbf{a}_1 = \begin{bmatrix} 2 \\ 1 \\ -1 \\ 3 \\ 1 \end{bmatrix}, \mathbf{a}_2 = \begin{bmatrix} -1 \\ 2 \\ -2 \\ 1 \\ -3 \end{bmatrix}, \mathbf{a}_3 = \begin{bmatrix} 2 \\ -1 \\ 3 \\ 1 \\ 5 \end{bmatrix}, \mathbf{a}_4 = \begin{bmatrix} -2 \\ 1 \\ -5 \\ -1 \\ -7 \end{bmatrix}, \mathbf{a}_5 = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 3 \\ 2 \end{bmatrix}, \mathbf{a}_6 = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \\ 3 \end{bmatrix} \qquad (1.19)$$

are initially given by their coordinates in the canonical basis. We will replace the first basis vector $\mathbf{e}_1$ by $\mathbf{a}_1$, and compute the coordinates in the new basis $\mathbf{a}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, $\mathbf{e}_4$, $\mathbf{e}_5$, using the following main program as follows.

```
100 REM ------------------------------------------------------------
102 REM EX. 1.1.2. VECTOR COORDINATES IN A NEW BASIS
104 REM MERGE M10
106 REM ---------- DATA
108 REM  (VECTOR DIMENSION, NUMBER OF VECTORS)
110 DATA  5, 6
112 DATA  2,-1, 2,-2, 1, 1
114 DATA  1, 2,-1, 1, 2, 3
116 DATA -1,-2, 3,-5, 1, 2
118 DATA  3, 1, 1,-1, 3, 4
120 DATA  1,-3, 5,-7, 2, 3
200 REM ---------- READ DATA
202 READ N,M
204 DIM A(N,M)
206 FOR I=1 TO N :FOR J=1 TO M :READ A(I,J) :NEXT J :NEXT I
208 V$=STRING$(8*(M+1),"-")
210 LPRINT "COORDINATES IN CANONICAL BASIS"
```

```
212 REM ---------- PRINT COORDINATES
214 LPRINT V$
216 LPRINT "vector j";
218 FOR J=1 TO M :LPRINT TAB(J*8+4);J; :NEXT J :LPRINT
220 LPRINT " i basis"
222 LPRINT V$
224 FOR I=1 TO N
226  K=A(I,0)
228  IF K>0 THEN LPRINT USING " #  a#  ";I,K; ELSE LPRINT USING" #  e#  ";I,I;
230  FOR J=1 TO M :LPRINT USING " ###.###";A(I,J); :NEXT J :LPRINT
232 NEXT I
234 LPRINT V$ :LPRINT
236 REM ---------- SELECT MODE
238 INPUT "t(transformation),r(row interchange) or s(stop)";A$
240 A$=CHR$(32 OR ASC(A$))
242 IF A$="t" THEN 246 ELSE IF A$="r" THEN 260
244 IF A$="s" THEN 276 ELSE 238
246 REM ---------- TRANSFORMATION
248 INPUT "row index (IP) and column index (JP) of the pivot:";IP,JP
250 IF IP<1 OR IP>N OR JP<1 OR JP>M THEN PRINT "unfeasible" :GOTO 236
252 IF ABS(A(IP,JP))>.000001 THEN 256
254 PRINT "zero or nearly zero pivot" :GOTO 236
256 LPRINT "PIVOT ROW:";IP;" COLUMN:";JP
258 GOSUB 1000 :GOTO 212
260 REM ---------- CHANGE TWO ROWS
262 INPUT "enter i1,i2 to interchange rows i1 and i2" ;I1,I2
264 IF I1<1 OR I1>N OR I2<1 OR I2>N THEN PRINT "unfeasible" :GOTO 236
266 IF A(I1,0)=0 OR A(I2,0)=0 THEN PRINT "unfeasible" :GOTO 236
268 LPRINT "ROWS INTERCHANGED:";I1;",";I2
270 FOR J=0 TO M :A=A(I1,J) :A(I1,J)=A(I2,J) :A(I2,J)=A :NEXT J
272 A(0,A(I1,0))=I1 :A(0,A(I2,0))=I2
274 GOTO 212
276 REM ---------- STOP
278 STOP
```

The program reads the dimension N , the number M of the vectors, and the array A(N,M) of coordinates in the canonical basis, all from DATA statements. The coordinates are read row-by-row, i.e., we specify the first coordinates in all vectors and proceed by coordinates. The program first prints the starting coordinates:

```
COORDINATES IN CANONICAL BASIS
--------------------------------------------------------
vector j    1       2       3       4       5       6
 i basis
--------------------------------------------------------
 1  e1     2.000  -1.000   2.000  -2.000   1.000   1.000
 2  e2     1.000   2.000  -1.000   1.000   2.000   3.000
 3  e3    -1.000  -2.000   3.000  -5.000   1.000   2.000
 4  e4     3.000   1.000   1.000  -1.000   3.000   4.000
 5  e5     1.000  -3.000   5.000  -7.000   2.000   3.000
--------------------------------------------------------
```

There are now three options to proceed: transformation (t), row interchange (r) or stop (s). You can select one of these options by entering the appropriate character.

In this example we perform a transformation, and hence enter "t". Then the row index and the column index of the pivot element are required. We enter "1,1" and the program returns the new coordinates:

```
PIVOT ROW: 1  COLUMN: 1
---------------------------------------------------------
vector j   1      2      3      4      5      6
 i basis
---------------------------------------------------------
 1  a1    1.000  -0.500  1.000  -1.000  0.500  0.500
 2  e2    0.000   2.500 -2.000   2.000  1.500  2.500
 3  e3    0.000  -2.500  4.000  -6.000  1.500  2.500
 4  e4    0.000   2.500 -2.000   2.000  1.500  2.500
 5  e5    0.000  -2.500  4.000  -6.000  1.500  2.500
---------------------------------------------------------
```

1.1.3 Solution of matrix equations by Gauss-Jordan elimination

To solve the simultanous linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1.20}$$

recall that the coefficients in $\mathbf{A}$ can be regarded as the coordinates of the vectors $\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_m$ (i.e., the columns of $\mathbf{A}$) in the canonical basis. Therefore, (1.20) can be written as

$$x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \ldots + x_m\mathbf{a}_m = \mathbf{b} \tag{1.21}$$

with unknown coefficients x_1, x_2, ..., x_m . There exist such coefficients if and only if $\mathbf{b}$ is in the subspace spanned by the vectors $\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_m$, i.e., the rank of this system equals the rank of the extended system $\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_m,\mathbf{b}$.

For simplicity assume first that $\mathbf{A}$ is a square matrix (i.e., it has the same number n of rows and columns), and $\mathrm{rank}(\mathbf{A}) = n$. Then the columns of $\mathbf{A}$ form a basis, and the coordinates of $\mathbf{b}$ in this basis can be found replacing the vectors $\mathbf{e}_1,\mathbf{e}_2,\ldots,\mathbf{e}_n$ by the vectors $\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_n$, one-by-one. In this new basis matrix $\mathbf{A}$ is the identity matrix. The procedure is called Gauss-Jordan elimination. As we show in the following example, the method also applies if $n \neq m$.

Example 1.1.3 General solution of a matrix equation by Gauss-Jordan elimination

Find all solutions of the simultaneous linear equations

$$\begin{array}{rrrrrcr} 2x_1 & -x_2 & +2x_3 & -2x_4 & +x_5 & = & 1 \\ x_1 & +2x_2 & -x_3 & +x_4 & +2x_5 & = & 3 \\ -x_1 & -2x_2 & +3x_3 & -5x_4 & +x_5 & = & 2 \\ 3x_1 & +x_2 & +x_3 & -x_4 & +3x_5 & = & 4 \\ x_1 & -3x_2 & +5x_3 & -7x_4 & +2x_5 & = & 3 \end{array} \qquad (1.22)$$

The columns of the coefficient matrix $\mathbf{A}$ in eqn. (1.22) are the vectors $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, $\mathbf{a}_4$, and $\mathbf{a}_5$ in (1.19), whereas the right-hand side $\mathbf{b}$ equals $\mathbf{a}_6$. Therefore the problem can be solved by replacing further vectors of the current basis in the previous example. Replacing $\mathbf{e}_2$ by $\mathbf{a}_2$ and then $\mathbf{e}_3$ by $\mathbf{a}_3$ we obtain the following coordinates:

```
PIVOT ROW: 2  COLUMN: 2
--------------------------------------------------------
vector j    1        2        3        4        5       6
 i basis
--------------------------------------------------------
 1  a1     1.000    0.000    0.600   -0.600    0.800    1.000
 2  a2     0.000    1.000   -0.800    0.800    0.600    1.000
 3  e3     0.000    0.000    2.000   -4.000    3.000    5.000
 4  e4     0.000    0.000    0.000    0.000    0.000    0.000
 5  e5     0.000    0.000    2.000   -4.000    3.000    5.000
--------------------------------------------------------

PIVOT ROW: 3  COLUMN: 3
--------------------------------------------------------
vector j    1        2        3        4        5       6
 i basis
--------------------------------------------------------
 1  a1     1.000    0.000    0.000    0.600   -0.100   -0.500
 2  a2     0.000    1.000    0.000   -0.800    1.800    3.000
 3  a3     0.000    0.000    1.000   -2.000    1.500    2.500
 4  e4     0.000    0.000    0.000    0.000    0.000    0.000
 5  e5     0.000    0.000    0.000    0.000    0.000    0.000
--------------------------------------------------------
```

According to this last table, the vectors $\mathbf{a}_4,\mathbf{a}_5$ and $\mathbf{a}_6$ are expressed as linear combinations of the vectors $\mathbf{a}_1,\mathbf{a}_2$ and $\mathbf{a}_3$ of the current basis. Thus the rank of the coefficient matrix of eqn. (1.22) and the rank of the extended system (1.19) are both 3, and we need only to interpret the results. From the last column of the table

$$\mathbf{a}_6 = \mathbf{b} = -0.5\mathbf{a}_1 + 3\mathbf{a}_2 + 2.5\mathbf{a}_3 \ , \qquad (1.23)$$

and hence $\mathbf{x} = (-0.5,\ 3,\ 2.5,\ 0,\ 0)^T$ is a solution of (1.22). To obtain the general solution, i.e., the set of all solutions, we will exploit that $\mathbf{a}_4$ and $\mathbf{a}_5$ are also given in terms of the first three vectors:

$$\mathbf{a}_4 = 0.6\mathbf{a}_1 - 0.8\mathbf{a}_2 - 2\mathbf{a}_3 \qquad (1.24)$$

$$\mathbf{a}_5 = -0.1\mathbf{a}_1 + 1.8\mathbf{a}_2 + 1.5\mathbf{a}_3 \ . \qquad (1.25)$$

Choosing arbitrary values for x_4 and x_5 , eqns. (1.23-1.25) give

$$b = (-0.5 - 0.6x_4 + 0.1x_5)a_1 + (3 + 0.8x_4 - 1.8x_5)a_2 + (2.5 + 2x_4 - 1.5x_5)a_3 + \\ + x_4a_4 + x_5a_5 \ . \qquad (1.26a)$$

Therefore, the general solution is given by

$$\begin{aligned} x_1 &= -0.5 - 0.6x_4 + 0.1x_5 \\ x_2 &= 3 + 0.8x_4 - 1.8x_5 \qquad (1.26b) \\ x_3 &= 2.5 + 2x_4 - 1.5x_5 \ . \end{aligned}$$

Since (1.26b) gives the solution at arbitrary x_4 and x_5 , these are said to be "free" variables, whereas the coefficients x_1, x_2 and x_3 of the current basis vectors a_1, a_2 and a_3 , respectively, are called basis variables. Selecting another basis, the "free" variables will be no more x_4 and x_5, and hence we obtain a general solution that differs from (1.26). We emphasize that the set of solutions **x** is obtained by evaluating (1.26) for all values of the "free" variables. Though another basis gives a different algebraic expression for x , it may be readily verified that we obtain the same set of values and thus the general solution is independent of the choice of the basis variables.

In linear programming problems we will need special solutions of matrix equations with "free" variables set to zero. These are called basic solutions of a matrix equation, where rank(**A**) is less than the number of variables. The coefficients in (1.23) give such a basic solution. Since in this example the two "free" variables can be chosen in $\binom{5}{2} = 10$ different ways, the equation may have up to 10 different basic solutions.

In Examples 1.1.2 and 1.1.3 we did not need the row interchange option of the program. This option is useful in pivoting, a practically indispensable auxiliary step in the Gauss-Jordan procedure, as will be discussed in the next section. While the Gauss-Jordan procedure is a straightforward way of solving matrix equations, it is less efficient than some methods discussed later in this chapter. It is, however, almost as efficient as any other method to calculate the inverse of a matrix, the topics of our next section.

<u>Exercises</u>

□ Select a different basis in Example 1.1.3 and show that the basic solution corresponding to this basis can be obtained from (1.26) as suitable values of x_4 and x_5.

□ Replace the last element of the right-hand side vector b in (1.24) by 4.

We will run into trouble when trying to solve this system. Why?

1.1.4 Matrix inversion by Gauss-Jordan elimination

Consider the $n \times n$ square matrix A and find its inverse A^{-1} defined by

$$AA^{-1} = I \ . \qquad (1.27)$$

Let $\bar{a}_i = (\bar{a}_{1i}, \bar{a}_{2i}, \ldots \bar{a}_{ni})^T$ denote the i-th column vector of A^{-1} (i.e., the set of coordinates in the canonical basis), then by (1.27)

$$\bar{a}_{1i}a_1 + \bar{a}_{2i}a_2 + \ldots + \bar{a}_{ni}a_n = e_i \ , \qquad (1.28)$$

where e_i is the i-th unit vector. According to (1.28), the vector $\bar{a}_i$ is given by the coordinates of the unit vector e_i in the basis $a_1, a_2, \ldots, a_n$, the column vectors of A. Thus we can find A^{-1} replacing the canonical vectors $e_1, e_2, \ldots, e_n$ by the vectors $a_1, a_2, \ldots, a_n$ in the basis one-by-one. In this new basis A is reduced to an identity matrix, whereas the coordinates of $e_1, e_2, \ldots, e_n$ form the columns of A^{-1}. If $\mathrm{rank}(A) < n$, then A is said to be singular, and its inverse is not defined. Indeed, we are then unable to replace all unit vectors of the starting basis by the columns of A.

Example 1.1.4 Inversion of a square matrix by Gauss-Jordan elimination.

To calculate the inverse of the matrix

$$A = \begin{bmatrix} 5 & 3 & -1 & 0 \\ 2 & 0 & 4 & 1 \\ -3 & 3 & -3 & 5 \\ 0 & 6 & -2 & 3 \end{bmatrix}$$

consider the vectors $a_1, a_2, a_3, a_4, e_1, e_2, e_3$ and e_4, where a_j is the j-th column vector of A. These coordinates are listed in the new DATA statements of the main program we used in the previous examples:

```
100 REM ------------------------------------------------------------
102 REM EX. 1.1.4. INVERSION OF A MATRIX BY GAUSS-JORDAN ELIMINATION
104 REM MERGE M10
106 REM ---------- DATA
108 REM (VECTOR DIMENSION, NUMBER OF VECTORS)
110 DATA 4,8
112 DATA  5, 3,-1, 0,  1, 0, 0, 0
114 DATA  2, 0, 4, 1,  0, 1, 0, 0
116 DATA -3, 3,-3, 5,  0, 0, 1, 0
118 DATA  0, 6,-2, 3,  0, 0, 0, 1
120 REM ---------- FROM HERE THE SAME AS THE PROGRAM OF EX. 1.1.2
```

Replacing the canonical basis vectors by a_1,a_2,a_3, and a_4 we obtain the following table of coordinates:

```
PIVOT ROW: 4  COLUMN: 4
----------------------------------------------------------------------
vector j   1      2      3      4      5      6      7      8
 i basis
----------------------------------------------------------------------
 1  a1    1.000  0.000  0.000  0.000  0.235  0.044  0.088 -0.162
 2  a2    0.000  1.000  0.000  0.000 -0.108 -0.010 -0.186  0.314
 3  a3    0.000  0.000  1.000  0.000 -0.147  0.191 -0.118  0.132
 4  a4    0.000  0.000  0.000  1.000  0.118  0.147  0.294 -0.206
----------------------------------------------------------------------
```

The last 4 columns of this table form A^{-1}.

In Example 1.1.4 we could replace e_i by the vector a_i in the basis for all i. Matrix inversion (or solution of a matrix equation) is, however, not always as simple. Indeed, we run into trouble if we want to replace e_i by a_i, but the desired pivot element a_{ii} is zero. This does not mean that A is singular if e_i can be replaced by another vector, say a_j. If the matrix is nonsingular, we will be able to include also a_i into the basis later on. Altering the order of entering vectors we interchange the rows of A^{-1}. The true order of rows can be restored by the row interchange option of the program. (Note that a row cannot be moved if the corresponding basis vector is still the canonical one.)

The next diagonal element is not necessarily the best choice for the pivot, even when nonzero. By (1.18), the current vector coordinates are modified by quantities proportional to the ratio a_{pj}/a_{pq}. The magnitude of a_{pq}, the intended pivot element, may be small, and divison by it is undesirable in the presence of roundoff errors, inherent to any computation. This is particulary important in the inversion of large matrices, where such errors may accumulate. An obvious counter-measure is picking the largest (in magnitude) available element of the next row as the pivot. This procedure is called partial pivoting. A more involved procedure is full pivoting, where the pivot is the largest (in magnitude) available element, not necessarily in the next row.

Exercises

- Calculate the inverse of A in Example 1.1.4 by different pivoting strategies. Save the inverse in an array and check its accuracy by evaluating the matrix product AA^{-1}.
- Replace the last row of A in Example 1.1.4 by (0, 6, -8, 4) and try to calculate the inverse.

1.2 LINEAR PROGRAMMING

We begin by solving a simple blending problem, a classical example in linear programming.

To kinds of row materials, A and B , are used by a manufacturer to produce products I and II. To obtain each unit of product I he blends 1/3 unit of A and 2/3 unit of B , whereas for each unit of product II he needs 5/6 unit of A and 1/6 unit of B . The available supplies are 30 units of A and 16 units of B . If the profit on each unit of product I is 100 ECU (European Currency Unit) and the profit on each unit of product II is 200 ECU, how many units of each product should be made to maximize the profit?

Let x_1 and x_2 denote the number of units of product I and II, respectively, being produced. By the limited supply of A we must have

$$\frac{1}{3}x_1 + \frac{5}{6}x_2 \le 30 \tag{1.29a}$$

whereas the supply of B gives

$$\frac{2}{3}x_1 + \frac{1}{6}x_2 \le 16 \ . \tag{1.29b}$$

In addition, the number of units of a product must be nonnegative:

$$x_1 \ge 0, \ X_2 \ge 0 \ . \tag{1.29c}$$

Now we want to maximize the objective function (i.e., the profit) given by

$$z = 100x_1 + 200x_2 \tag{1.30}$$

subject to the constraints (1.29).

As shown in Fig. 1.2 , to solve this problem we need only analytical geometry. The constraints (1.29) restrict the solution to a convex polyhedron in the positive quadrant of the coordinate system. Any point of this region satisfies the inequalities (1.29), and hence corresponds to a feasible vector or feasible solution. The function (1.30) to be maximized is represented by its contour lines. For a particular value of z there exists a feasible solution if and only if the contour line intersects the region. Increasing the value of z the contour line moves upward, and the optimal solution is a vertex of the polyhedron (vertex C in this example), unless the contour line will include an entire segment of the boundary. In any case, however, the problem can be solved by evaluating and comparing the objective function at the vertices of the polyhedron.

To find the coordinates of the vertices it is useful to translate the inequality constraints (1.29a -1.29b) into the equalities

$$\frac{1}{3}x_1 + \frac{5}{6}x_2 + x_3 \qquad = 30 \tag{1.31a}$$

$$\frac{2}{3}x_1 + \frac{1}{6}x_2 \qquad + x_4 = 16 \tag{1.31b}$$

by introducing the so called slack variables x_3 and x_4 which must be also nonnegative. Hence (1.29c) takes the form

$$x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0 . \qquad (1.31c)$$

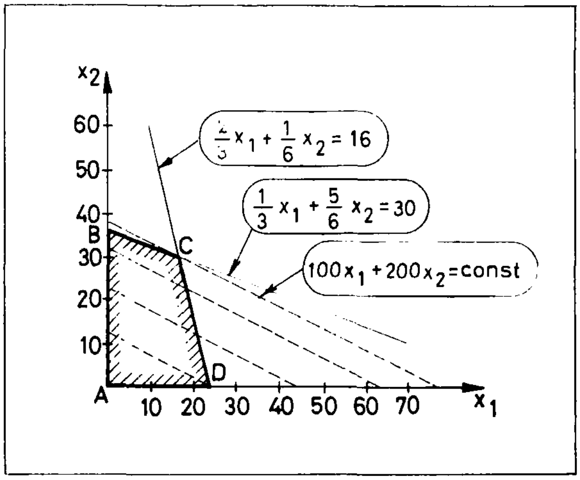


Fig. 1.2. Feasible region and contour lines of the objective function

The slack variables do not influence the objective function (1.30) but for convenience we can include them with zero coefficients.

We consider the equality constraints (1.31a-1.31b) as a matrix equation, and generate one of its basic solution with "free" variables beeing zero. A basic solution is feasible if the basis variables take nonnegative values. It can be readily verified that each feasible basic solution of the matrix equation (1.31a-1.31b) corresponds to a vertex of the polyhedron shown in Fig. 1.2. Indeed, $x_1 = x_2 = 0$ in point A , $x_1 = x_3 = 0$ in point B , $x_3 = x_4 = 0$ in point C , and $x_2 = x_4 = 0$ in point D . This is a very important observation, fully exploited in the next section.

1.2.1 Simplex method for normal form

By introducing slack variables the linear programming problem (1.5-1.6) can be translated into the normal form

$$\begin{aligned} &\mathbf{Ax} = \mathbf{b}\ ,\ (\ \mathbf{b} \geq \mathbf{0}\ ) \\ &\mathbf{x} \geq \mathbf{0}\ , \qquad\qquad (1.32) \\ &z = \mathbf{c}^T\mathbf{x} \dashrightarrow \max\ , \end{aligned}$$

where we have n constraints, $m+n$ variables and $\mathbf{A}$ denotes the (extended) coefficient matrix of dimensions $n\times(m+n)$. (Here we assume that the right-hand side is nonnegative - a further assumption to be relaxed later on.) The key to solving the original problem is the relationship between the basic solutions of the matrix equation $\mathbf{Ax} = \mathbf{b}$ and the vertices of the feasible polyhedron. An obvious, but far from efficient procedure is calculating all basic solutions of the matrix equation and comparing the values of the objective function at the feasible ones.

The simplex algorithm (refs.7-8) is a way of organizing the above procedure much more efficiently. Starting with a feasible basic solution the procedure will move into another basic solution which is feasible, and the objective function will not decrease in any step. These advantages are due to the clever choice of the pivots.

A starting feasible basic solution is easy to find if the original constraints are of the form (1.6) with a nonnegative right-hand side. The extended coefficient matrix $\mathbf{A}$ in (1.32) includes the identity matrix (i.e., the columns of $\mathbf{A}$ corresponding to the slack variables $x_{m+1},\ldots,x_{m+n}$.) Consider the canonical basis and set $x_i = 0$ for $i = 1,\ldots,m$, and $x_{m+i} = b_i$ for $i = 1,..,n$. This is clearly a basic solution of $\mathbf{Ax} = \mathbf{b}$, and it is feasible by the assumption $b_i \geq 0$. Since the starting basis is canonical, we know the coordinates of all the vectors in this basis. As in Section 1.1.3 , we consider the right-hand side $\mathbf{b}$ as the last vector $\mathbf{a}_M = \mathbf{b}$, where $M = m+n+1$.

To describe one step of the simplex algorithm assume that the vectors present in the current basis are $\mathbf{a}_{B1}$, $\mathbf{a}_{B2}$, ..., $\mathbf{a}_{Bn}$. We need this indirect notation because the indices B1, B2, ..., Bn are changing during the steps of the algorithm. They can take values from 1 to $m+n$. Similarly, we use the notation c_{Bi} for the objective function coefficient corresponding to the i-th basis variable. Assume that the current basic solution is feasible, i.e, the coordinates of $\mathbf{a}_M$ are nonnegative in the current basis. We first list the operations to perform:

(i) Compute the indicator variables $z_j - c_j$ for all $j = 1,\ldots,m+n$ where z_j is defined by

$$z_j = \sum_{i=1}^{n} a_{ij}c_{Bi}\ . \qquad (1.33)$$

The expression (1.33) can be computed also for $j = M$. In this case it gives the current value of the objective function, since the "free" variables vanish and a_{iM} is the value of the i-th basis variable.

(ii) Select the column index q such that $z_q - c_q \leq z_j - c_j$ for all $j = 1, \ldots, m+n$, i.e., the column with the least indicator variable value. If $z_q - c_q \geq 0$, then we attained the optimal solution, otherwise proceed to step (iii).

(iii) If $a_{iq} \leq 0$ for each $i = 1,\ldots,n$, (i.e., there is no positive entry in the selected column), then the problem has no bounded optimal solution. Otherwise proceed to step (iv).

(iv) Locate a pivot in the q-th column, i.e.,select the row index p such that $a_{pq} > 0$ and $a_{pM}/a_{pq} \leq a_{iM}/a_{iq}$ for all $i = 1,\ldots,n$ if $a_{iq} > 0$.

(v) Replace the p-th vector in the current basis by a_q , and calculate the new coordinates by (1.18).

To understand why the algorithm works it is convenient to consider the indicator variable $z_j - c_j$ as loss minus profit. Indeed, increasing a "free" variable x_j from zero to one results in the profit c_j . On the other hand, the values of the current basis variables $x_{Bi} = a_{iM}$ must be reduced by a_{ij} for $i = 1,\ldots,n$ in order to satisfy the constraints. The loss thereby occuring is z_j. Thus step (ii) of the algorithm will help us to move to a new basic solution with a nondecreasing value of the objective function.

Step (iv) will shift a feasible basic solution to another feasible basic solution. By (1.18) the basis variables (i.e., the current coordinates of the right-hand side vector a_M) in the new basis are

$$a'_{iM} = a_{iM} - \frac{a_{iq}}{a_{pq}} a_{pM} . \tag{1.34}$$

Since the previous basic solution is feasible, $a_{iM} \geq 0$. If $a_{iq} \leq 0$, then $a'_{iM} \geq 0$ follows. However, $a'_{iM} \geq 0$ in any case, since we selected $a_{pq} \geq 0$ to satisfy $a_{pM}a_{iq}/a_{pq} \leq a_{iM}$ for all i corresponding to positive a_{iq}. According to a "dynamic" view of the process, we are increasing a previously "free" variable until one of the previous basic variables is driven to zero.

If there is no positive entry in the q-th column, then none of the previous basic variables will decrease and we can increase the variable x_j indefinitely, yielding ever increasing values of the objective function. Detecting this situation in step (ii), there is no reason to continue the procedure.

Replacing the p-th basis vector by a_q in step (v), the new value z'_M of the objective function will be

$$z'_M = \sum_{\substack{i=1\\(i\neq p)}}^{n} a'_{iM}c_{Bi} + a'_{pM}c_q \ . \tag{1.35}$$

By (1.18) we can express the new coordinates a'_{iM} and a'_{pM} in terms of the old ones, resulting in the relationship

$$z'_M = z_M - \frac{z_q - c_q}{a_{pq}} a_{pM} \ . \tag{1.36}$$

Since $z_q - c_q$ is negative and a_{pq} is positive, the sign of the change in the objective function depends on the sign of a_{pM}. This latter might be positive (increasing the objective function value) or zero (resulting in no change of the objective function value).

It remains to show that $z_q - c_q \geq 0$ really indicates the optimal solution. This requires a somewhat deeper analysis. Let $\mathbf{B}$ denote the n×n matrix formed by the column vectors $a_{B1}, a_{B2}, \ldots, a_{Bn}$. We have to show that for every feasible solution $\mathbf{y}$, the objective function does not increase, i.e.,

$$\mathbf{c}^T_B \mathbf{B}^{-1}\mathbf{b} \geq \mathbf{c}^T\mathbf{y} \ . \tag{1.37}$$

We will exploit the fact that all indicator variables are nonnegative:

$$z_j \geq c_j, \ j = 1,2,\ldots,m+n \ . \tag{1.38}$$

By virtue of the definition (1.33)

$$z_j = \mathbf{c}^T_B \mathbf{B}^{-1}\mathbf{a}_j, \ j = 1,2,\ldots m+n \ . \tag{1.39}$$

Using this expression in (1.38) and multiplying the j-th inequality by the nonnegative y_j gives m+n inequalities whose sum is

$$\sum_{j=1}^{m+n} \mathbf{c}^T_B \mathbf{B}^{-1}\mathbf{a}_j y_j \geq \mathbf{c}^T\mathbf{y} \ . \tag{1.40}$$

Since $\mathbf{y}$ is the solution of the matrix equation, $\sum_{j=1}^{m+n} \mathbf{a}_j y_j = \mathbf{b}$. Introducing this equality into (1.40) gives the inequality (1.37) that we wanted to prove.

Similarly to the derivation of (1.35) and (1.36) one can easily show that

$$z'_j - c_j = z_j - c_j - \frac{z_q - c_q}{a_{pq}} a_{pj} \quad . \qquad (1.41)$$

Thus the coordinate transformations (1.18) apply also to the indicator variables and to the objective function. On the basis of this observation it is convenient to perform all calculations on a matrix extended by the z_j-c_j values and the objective function value as its last row. This extended matrix is the so-called simplex tableau.

If the j-th column is in the basis then $z_j-c_j = 0$ follows, but an entry of the last row of the simplex tableau may vanish also for a column that is not in the basis. If this situation occures in the optimal simplex tableau then the linear programming problem has several optimal basic solutions. In our preliminary example this may happen when contour lines of the objective function are parallel to a segment of the boundary of the feasible region.

The simplex algorithm will reach the optimal solution in a finite number of steps if the objective function is increased in each of them. In special situations, however, the objective function value may be the same in several consecutive steps and we may return to the same basis, repeating the cycle again. The analysis of cycling is a nice theoretical problem of linear programming and the algorithms can be made safe against it. It is very unlikely, however, that you will ever encounter cycling when solving real-life problems.

1.2.2 Reducing general problems to normal form. The two-phase simplex method

In this section we state a much more general linear programming problem, introducing notations which will be used also in our linear programming module. Let NV be the number of variables, denoted by $x_1, x_2, \ldots, x_{NV}$. The NE constraints are of the form

$$\begin{array}{l}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1,NV}x_{NV} \ \{ \le, =, \ge \} \ a_{1,NV+1} \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2,NV}x_{NV} \ \{ \le, =, \ge \} \ a_{2,NV+1} \\
\cdot \\
\cdot \\
\cdot \\
a_{NE,1}x_1 + a_{NE,2}x_2 + \ldots + a_{NE,NV}x_{NV} \ \{ \le, =, \ge \} \ a_{NE,NV+1}
\end{array} \qquad (1.42)$$

where we adopt the notation $\{ \le, =, \ge \}$ to emphasise that any one of these relation signs can be used in a constraint. As before, our primary constraint are

$$x_1 \ge 0, \ x_2 \ge 0, \ \ldots, \ x_{NV} \ge 0, \qquad (1.43)$$

but now we do not require the entries of the right-hand side vector to be nonnegative. The problem is either to maximize or minimize the objective function

$$c_1x_1 + c_2x_2 + \ldots + c_{NV}x_{NV} \dashrightarrow \left\{ \begin{matrix} \max \\ \min \end{matrix} \right\} . \qquad (1.44)$$

This generalized problem can easily be translated to the normal form by the following tricks.

- If the right-hand side is negative, multiply the constraint by (-1).
- As discussed, a constraint with $\leq$ is transformed into an equality by adding a (nonnegative) slack variable to its left-hand side. The same can be done in an inequality with $\geq$, this time by substracting a (nonnegative) slack variable from its left-hand side.
- The problem of locating the minimum is translated to the normal (maximization) problem by changing the sign of the objective function coefficients.

With inequality constraints of the form $\leq$ only, the columns corresponding to the slack variables can be used as a starting basis. This does not work for the generalized problem, and we must proceed in two phases.

In the first phase we invent futher variables to create an identity matrix within the coefficient matrix **A**. We need, say, r of these, called artificial variables and denoted by $s_1,s_2,\ldots,s_r$. Exactly one non-negative artificial variable is added to the left-hand side of each constraint with the sign $=$ or $\geq$. A basic solution of this extended matrix equation will be a basic solution of the original equations if and only if $s_1 = s_2 = \ldots = s_r = 0$. We try to find such a solution by applying the simplex algorithm itself. For this purpose replace the original objective function by $z_I = - \sum_{i=1}^{r} s_i$, which is then maximized. This can obviously be done by the simplex algorithm described in the previous section. The auxiliary linear programming problem of the first phase always has optimal solution where either $z_I < 0$ or $z_I = 0$. With $z_I < 0$ we are unable to eliminate all the artificial variables and the original problem has no feasible solution. With $z_I = 0$ there may be two different situations. If $z_I = 0$ and there are no artificial variables among the basic variables, then we have a feasible basic solution of the original problem. It may happen, however, that $z_I = 0$ but there is an artificial variable among the basic variables, obviously with zero value. If there is at least one nonzero entry in the corresponding row of the tableau then we can use it as a pivot to replace the artificial vector still in the basis. If all entries are zero in the corresponding row, we can simply drop it, since the constraint is then a linear combination of the others.

After completing the first phase we have a feasible basic solution. The second phase is nothing else but the simplex method applied to the normal form. The following module strictly follows the algorithmic steps described.

Program module M11

```
1100 REM ****************************************************
1102 REM *             LINEAR PROGRAMMING                   *
1104 REM *          TWO-PHASE SIMPLEX METHOD                *
1106 REM ****************************************************
1108 REM INPUT:
1110 REM      NV     NUMBER OF VARIABLES
1112 REM      NE     NUMBER OF CONSTRAINTS
1114 REM      E$     PROBLEM TYPE:  'MAX'  OR  'MIN'
1116 REM    E$(NE)   TYPE OF CONSTRAINTS: 'LE','EQ' OR 'GE'
1118 REM    A(.,.)   INITIAL SIMPLEX TABLEAU
1120 REM             A(1...NE,1...NV) CONSTRAINT MATRIX COEFFICIENTS
1122 REM             A(1...NE,NV+1)   CONSTRAINT RIGHT HAND SIDES
1124 REM    C(NV)    OBJECTIVE FUNCTION COEFFICIENTS
1126 REM OUTPUT:
1128 REM      ER     STATUS FLAG
1130 REM                0  OPTIMUM FOUND
1132 REM                1  NO FEASIBLE SOLUTION
1134 REM                2  NO FINITE OPTIMUM
1136 REM                3  ERRONEOUS CHARACTERS IN  E$(.)  OR  E$
1138 REM      N      NUMBER OF ROWS IN FINAL SIMPLEX TABLEAU, N=NE+1
1140 REM      M      NUMBER OF COLUMNS IN FINAL SIMPLEX TABLEAU,M=NV+LE+GE+1
1142 REM    A(N,M)   FINAL SIMPLEX TABLEAU
1144 REM             OPTIMUM VALUE OF THE J-TH VARIABLE
1146 REM                 0         IF A(0,J)=0 ,
1148 REM             A(A(0,J),M)   OTHERWISE
1150 REM             OPTIMUM OBJECTIVE FUNCTION VALUE IS E*A(N,M)
1152 REM
1154 REM MODULE CALLED: M10
1156 REM ---------- INITIAL VALUES
1158 LE=0 :EQ=0 :GE=0 :EP=.000001 :EN=EP :MA=1E+30
1160 REM ---------- CHECK INPUT DATA
1162 FOR I=1 TO NE
1164  IF A(I,NV+1)>=0 THEN 1170
1166   IF E$(I)="LE" THEN E$(I)="GE" :GOTO 1170
1168   IF E$(I)="GE" THEN E$(I)="LE"
1170  IF E$(I)="LE" OR E$(I)="EQ" OR E$(I)="GE" THEN 1174
1172   ER=3 : GOTO 1340
1174  EQ=EQ-(E$(I)="EQ")
1176  LE=LE+(A>=0)*(E$(I)="LE")+(A<0)*(E$(I)="GE")
1178  GE=GE+(A>=0)*(E$(I)="GE")+(A<0)*(E$(I)="LE")
1180 NEXT I
1182 IF E$<>"MAX" AND E$<>"MIN" THEN ER=3 : GOTO 1340
1184 M=NV+LE+EQ+2*GE+1 :N=NE+1
1186 REM --------------------------------------------------------------
1188 PRINT "SOLUTION OF THE ACTUAL PROBLEM REQUIRES DIM A(";N;",";M;")"
1190 REM --------------------------------------------------------------
```

```
1192 REM ---------- FILL SIMPLEX TABLEAU
1194 JV=NV :JA=NV+LE+GE
1196 FOR I=1 TO NE
1198  E=(E$(I)="GE")-(E$(I)="LE")
1200  A=A(I,NV+1) :IF A>=0 THEN 1204
1202  A=-A : FOR J=1 TO NV :A(I,J)=-A(I,J) :NEXT J :E=-E
1204  FOR J=NV+1 TO M-1 :A(I,J)=0 :NEXT J :A(I,M)=A
1206  IF E=0 THEN 1210
1208   JV=JV+1 :A(I,JV)=E :IF E>0 THEN A(0,JV)=I :A(I,0)=JV
1210  IF E>0 THEN 1214
1212   JA=JA+1 :A(I,JA)=1 :A(0,JA)=I :A(I,0)=JA
1214 NEXT I
1216 REM ---------- PHASE 1
1218 IF EQ+GE=0 THEN 1284
1220 REM ---------- ---------- Z-C VALUES
1222 FOR J=1 TO M
1224  IF A(0,J)<>0 THEN 1230
1226   A(N,J)=0
1228   FOR I=1 TO NE :A(N,J)=A(N,J)+A(I,J)*(A(I,0)>NV+LE+GE) :NEXT I
1230 NEXT J
1232 IF A(N,M)>=-EP THEN 1266
1234 REM ---------- ---------- CHECK FEASIBILITY
1236 MI=0
1238 FOR J=1 TO M-1
1240  IF A(N,J)<MI THEN MI=A(N,J) :JP=J
1242 NEXT J
1244 IF MI=0 THEN ER=1 : GOTO 1340
1246 REM ---------- ---------- CHANGE BASIS
1248 MI=MA
1250 FOR I=1 TO NE
1252  IF A(I,JP)<=EP THEN 1256
1254   IF A(I,M)/A(I,JP)<MI THEN MI=A(I,M)/A(I,JP) :IP=I
1256 NEXT I
1258 GOSUB 1000 :EP=EP+EN
1260 REM ---------- ---------- TERMINATION CONDITION
1262 IF A(N,M)<-EP THEN 1236
1264 REM ---------- ---------- ELIMINATION OF ARTIFICIAL VARIABLES
1266 FOR IP=1 TO NE
1268  IF A(IP,0)<=NV+LE+GE THEN 1280
1270  FOR JP=1 TO NV+LE+GE
1272   IF ABS(A(IP,JP))>=EP THEN  GOSUB 1000 : EP=EP+EN :GOTO 1280
1274   A(IP,JP)=0
1276  NEXT JP
1276  A(IP,0)=0 :A(IP,M)=0
1280 NEXT IP
1282 REM ---------- PHASE 2
1284 FOR J=1 TO NV : A(N,J)=C(J) :NEXT J
1286 E=(E$="MIN")-(E$="MAX")
1288 M=NV+LE+GE+1
1290 A(0,M)=0
1292 FOR J=NV+1 TO M :A(N,J)=0 :NEXT J
1294 FOR I=1 TO NE :A(I,M)=A(I,M+EQ+GE) :NEXT I
1296 REM ---------- ----------- Z-C VALUES
1298 FOR J=1 TO M
1300  IF A(0,J)>0 THEN 1306
1302   A(N,J)=-E*A(N,J)
1304   FOR I=1 TO NE :A(N,J)=A(N,J)+E*A(I,J)*A(N,A(I,0)) :NEXT I
1306 NEXT J
1308 FOR I=1 TO NE : A(N,A(I,0))=0 :NEXT I
```

```
310 REM ---------- ----------- CHECK OPTIMALITY
1312 MI=-EP
1314 FOR J=1 TO M-1
1316  IF A(N,J)<MI THEN MI=A(N,J) :JP=J
1318 NEXT J
1320 IF MI=-EP THEN ER=0 : GOTO 1340
1322 REM ---------- ----------- CHANGE BASIS
1324 MI=MA
1326 FOR I=1 TO NE
1328  IF A(I,JP)<=EP THEN 1332
1330   IF A(I,M)/A(I,JP)<MI THEN MI=A(I,M)/A(I,JP) :IP=I
1332 NEXT I
1334 REM ---------- ----------- NO FINITE OPTIMUM OR CONTINUE
1336 IF MI=MA THEN ER=2 : GOTO 1340
1338 GOSUB 1000 :EP=EP+EN :GOTO 1312
1340 RETURN
1342 REM ####################################################
```

The remarks in the module tell you how to specify the input. Notice that any right-hand coefficient may be negative, and you do not have to group the constraints depending on their relation signs. What you should do is simply to write the character sequences "LE", "EQ" or "GE" into the entries of the vector E$() for the constraints $\le$, =, and $\ge$, respectively. Depending on what you want, put the character sequences "MAX" or "MIN" into the non-vector variable E$.

You may face, however, difficulties in selecting the physical dimensions of the array A in your calling program, since this array stores the simplex tableau in both phases. We will present a main program that selects these dimensions for you. If you want to call the module from your own program, you should specify dimensions that are large enough. BASIC does not care about the extra space occupied. If you do not know what large enough means in your particular problem, you may watch the screen, since the module will output the dimensions of the array A actually required.

On output the flag ER will tell you the outcome. The return value ER = 0 indicates an optimal solution is found. In this case the solution is stored in the M-th column of A , where the value of M is determined also by the module. To find the results, however, you need to know which vectors are in the final basis, and also the positions of these vectors in the tableau. The coordinate transformations are performed by the module M10, and hence this information is stored in the entries A(0,J) , as described in Section 1.1.2.

You may wish to follow the steps of the procedure and print the indices IP,JP of the pivot. This can be done in the module M10. The current value of the objective function may be obtained by printing the product E*A(N,M).

While our test example is very simple, the module enables you to solve much larger problems, in principle constrained only by the storage capacity provided by your BASIC interpreter or compiler. As emphasized in Section 1.1.4, in a sequence of coordinate transformations we accumulate round-off errors. When selecting a pivot element the test for inequality with zero actually is a test against a small parameter whose value is increased in each step to compensate the accumulation of errors. Nevertheless, you may encounter problems with detecting convergence if there are order of magnitude differences in the coefficient matrix. Therefore, it is advisable to perform some scaling of the constraints and the variables before solving a larger problem. You may multiply all coefficients and the right-hand side of a constraint by a scaling factor. Similarly, you may multiply all coefficients in a column of **A** and the corresponding coefficient in the objective function, but in this case after solving the problem the corresponding variable must also be multiplied by the same factor.

Eample 1.2 Solution of the blending problem

Though we solve here only the simple blending problem (1.29-1.30) by calling the module M11, we present a main program which, apart from the specific input in its DATA statements, is rather general and performs a number of auxiliary operations. In particular, it reads the problem, calculates the dimensions, calls the module, locates and prints out the results. Later on we will solve other problems by this program, replacing only the data lines.

```
100 REM -----------------------------------------------------
102 REM EX. 1.2. LINEAR PROGRAMMING BY TWO PHASE SIMPLEX METHOD
104 REM MERGE M10,M11
106 REM DATA
108 REM (NUMBER OF VARIABLES, NUMBER OF CONSTRAINTS)
110 DATA 2,2
112 REM CONSTRAINTS
114 DATA  0.333333,  0.833333, LE,  30
116 DATA  0.666667,  0.166667, LE,  16
118 REM OBJECTIVE FUNCTION:
120 DATA  100,       200,       MAX
200 REM ---------- CHECK DATA AND COMPUTE DIMENSIONS
202 LE=0 :EQ=0 :GE=0 :READ NV,NE
204 FOR I=1 TO NE
206  FOR J=1 TO NV :READ A :NEXT J: READ E$,A
208  IF E$="LE" OR E$="EQ" OR E$="GE" THEN 212
210   LPRINT "ERROR IN CONSTRAINT No.";I :GOTO 324
212  IF E$="EQ" THEN EQ=EQ+1 :GOTO 222
214  IF E$="GE" THEN 220
216  IF A>=0 THEN LE=LE+1 ELSE GE=GE+1
218  GOTO 222
220  IF A>=0 THEN GE=GE+1 ELSE LE=LE+1
222 NEXT I
224 FOR J=1 TO NV :READ A: NEXT J :READ E$
226 IF E$="MAX" OR E$="MIN" THEN 230
228 LPRINT "ERROR IN OBJECTIVE FUNCTION SPECIFICATION" :GOTO 324
230 M=NV+LE+EQ+2*GE+1 :N=NE+1
232 DIM A(N,M),C(NV),E$(NE)
```

```
234 REM ---------- FILL INITIAL SIMPLEX TABLEAU
236 RESTORE
238 READ NV,NE
240 FOR I=1 TO NE
242  FOR J=1 TO NV :READ A(I,J) :NEXT J
244  READ E$(I),A(I,NV+1)
246 NEXT I
248 FOR J=1 TO NV :READ C(J) :NEXT J
250 READ E$
252 REM ---------- CALL LP MODULE
254 GOSUB 1100
256 LPRINT
258 LPRINT TAB(10);"LINEAR PROGRAMMING BY TWO PHASE SIMPLEX METHOD"
260 LPRINT :LPRINT :LPRINT
262 IF ER=1 THEN LPRINT "NO FEASIBLE SOLUTION" :GOTO 324
264 IF ER=2 THEN LPRINT "NO FINITE ";E$;"IMUM" :GOTO 324
266 LPRINT :LPRINT  "EVALUATION OF CONSTRAINTS" :LPRINT
268 V$=STRING$(62,"-") :V1$=STRING$(54,"-")
270 LPRINT V$
272 LPRINT " I  TYPE  L.H.S.        R.H.S     SLACK           SHADOW PRICE"
274 LPRINT V$
276 RESTORE :READ NV,NE :JV=NV
278 FOR I=1 TO NE
280  B=0
282  FOR J=1 TO NV
284   READ A :K=A(0,J) :IF K=0 THEN X=0 ELSE X=A(K,M)
286   B=B+A*X
288  NEXT J :READ E$,A
290  T=ABS(A-B) :IF T<=EP*ABS(A) THEN T=0
292  LPRINT I;TAB(6);E$;TAB(10);B;TAB(24);A;TAB(34);T;
294  IF E$(I)<>"EQ" THEN JV=JV+1 :IF A(0,JV)=0 THEN LPRINT TAB(50);A(N,JV);
296  LPRINT
298 NEXT I
300 LPRINT V$ :LPRINT
302 LPRINT :LPRINT "           OPTIMUM SOLUTION" :LPRINT
304 LPRINT V1$
306 LPRINT " j"," Xj","  Cj","   Cj*Xj"
308 LPRINT V1$
310 FOR J=1 TO NV
312  READ C :K=A(0,J) :IF K>0 THEN X=A(K,M) ELSE X=0
314  LPRINT J;TAB(15)X;TAB(30)C;TAB(45)C*X
316 NEXT J
318 READ E$ :A=A(N,M) :IF E$="MIN" THEN A=-A
320 LPRINT V1$ :LPRINT
322 LPRINT "OBJECTIVE FUNCTION ";E$;"IMUM VALUE .......... ";A :LPRINT
324 STOP
```

The DATA statements contain the input data in the following order:

- the number of variables and the number of constraints;
- for each constraint the coefficients, the type of the constraint ("LE","EQ" or "GE") and the right-hand side;
- the objective function coefficients and the type of the problem ("MAX" or "MIN").

The program output for this example is as follows.

```
        LINEAR PROGRAMMING BY TWO PHASE SIMPLEX METHOD

EVALUATION OF CONSTRAINTS

------------------------------------------------------------
 I  TYPE  L.H.S.        R.H.S     SLACK           SHADOW PRICE
------------------------------------------------------------
 1   LE   30            30        0               233.3334
 2   LE   16            16        0               33.33341
------------------------------------------------------------


           OPTIMUM SOLUTION

----------------------------------------------------
 j            Xj             Cj             Cj*Xj
----------------------------------------------------
 1            16.66664       100            1666.664
 2            29.33337       200            5866.674
----------------------------------------------------

OBJECTIVE FUNCTION MAXIMUM VALUE .........  7533.337
```

According to these results the slack variables vanish in the constraints 1 and 2, which are of type $\le$. Therefore, the optimal solution is on the boundary defined by these two constraints. Such constraints are said to be active ones. In physical terms it means that the available supplies of raw material A and B are both exhausted. The optimal strategy is producing 16.7 units of product I and 29.3 units of product II.

Our results include the shadow prices for each active constraint. A shadow price can be regarded as the change in the optimal value of the objective function following the increase of the right-hand side of the constraint by one unit. (Strictly speaking you may obtain even larger change in the objective function if the optimal basis will not remain the same.) In the given example it is advantageous to increase the supply of A if its market price is less than 233.3 ECU/unit. The raw material B is much less valuable in the given situation. You can learn more about shadow prices by reading on the concept of duality in linear programming, e.g., in (ref. 8).

<u>Exercise</u>

□ Solve the blending problem with objective functions

$z = 100x_1 + 250x_2$

and

$z = 100x_1 + 300x_2$,

both by the program and by geometrical considerations.

1.3 LU DECOMPOSITION

In this section we restrict considerations to an $n \times n$ nonsingular matrix **A** . As shown in Section 1.1, the Gauss-Jordan elimination translates **A** into the identity matrix I . Selecting off-diagonal pivots we interchange some rows of I , and obtain a permutation matrix **P** instead, with exactly one element 1 in each row and in each column, all the other entries beeing zero. Matrix **P** is called permutation matrix, since the operation **PA** will interchange some rows of **A** .

We can save some efforts reducing **A** into a triangular matrix and not all the way to the identity matrix. More generally, we will write **A** as

$$\mathbf{PA} = \mathbf{LU} \, , \tag{1.45}$$

where P is a permutation matrix, L is a lower triangular (has elements only in the diagonal and below), U is upper triangular (has elements only on the diagonal and above), and $[L]_{ii} = 1$.

The decomposition will be performed by Gaussian elimination. This classical method can easily be understood by solving an example.

1.3.1 Gaussian elimination

We solve the matrix equation (ref. 9)

$$\begin{bmatrix} 5 & 3 & -1 & 0 \\ 2 & 0 & 4 & 1 \\ -3 & 3 & -3 & 5 \\ 0 & 6 & -2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 11 \\ 1 \\ -2 \\ 9 \end{bmatrix} \tag{1.46}$$

by reducing its coefficient matrix to an upper triangular one. Therefore, let us first eliminate x_1 from equations 2 and 3, multiplying the first equatation by factors (2/5) and (-3/5) , respectively, and then substracting from equations 2 and 3 . The resulting equation is

$$\begin{bmatrix} 5 & 3 & -1 & 0 \\ 0 & -1.2 & 4.4 & 1.0 \\ 0 & 4.8 & -3.6 & 5.0 \\ 0 & 6.0 & -2.0 & 3.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 11 \\ -3.4 \\ 4.6 \\ 9.0 \end{bmatrix} \tag{1.47}$$

The pivot (i.e., the element we divide by) in this step was 5 , and the factors (2/5, -3/5, 0) = (0.4, -0.6, 0) are called multipliers. We perform partial pivoting (see Section 1.1.4) and pick $[A]_{4,2} = 6.0$ as the next pivot instead of the diagonal element $[A]_{2,2} = -1.2$. This choice implies interchanging rows 2 and 4 (and also the corresponding right-hand side entries). Using the multipliers $(4.8/6.0,\ -1.2/6.0) = (0.8,\ -0.2)$ we have

$$\begin{bmatrix} 5 & 3 & -1 & 0 \\ 0 & 6.0 & -2.0 & 3.0 \\ 0 & 0 & -2.0 & 2.6 \\ 0 & 0 & 4.0 & 1.6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 11 \\ 9.0 \\ -2.6 \\ -1.6 \end{bmatrix} . \tag{1.48}$$

The next pivot will be $[A]_{4,3} = 4.0$, thereby interchanging rows 3 an 4. To eliminate x_3 from equation 3 we need the single multiplier $-2.0/4.0 = -0.5$, and obtain the matrix in the desired upper triangular form:

$$\begin{bmatrix} 5 & 3 & -1 & 0 \\ 0 & 6.0 & -2.0 & 3.0 \\ 0 & 0 & 4.0 & 1.6 \\ 0 & 0 & 0 & 3.4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 11 \\ 9.0 \\ -1.6 \\ -3.4 \end{bmatrix} . \tag{1.49}$$

Equations (1.49) are very easy to solve. Indeed, $x_4 = -1$ is already isolated in equation 4. Proceeding with this value to equation 3 gives $x_3 = 0$. Then we move to equation 2 with x_3 and x_4 known. The procedure is called backsubstitution and gives the solution vector $x = (1.0,\ 2.0,\ 0.0,\ -1.0)^T$.

1.3.2 Performing the LU decomposition

The Gaussian elimination also enables us to decompose the matrix in (1.46). We already have the upper triangular in (1.49). To form the permutation matrix P we will interchange those rows of the identity matrix I that have been interchanged in **A** in the course of the Gaussian elimination. Let (i,k_i) denote the operation of interchanging rows i and k_i in the i-th step, then what we did is (1,1), (2,4) and (3,4) . These operations applied to the identity matrix I result in the permutation matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \tag{1.50}$$

The lower triangular matrix L can be constructed from the multipliers used in the elimination steps if we adjust them according to the rows interchanged. Taking into account that for the row of the pivot the multiplier is necessarily 1.0 (i.e., this row remains unchanged), in the three steps of the Gaussian elemination the multipliers were (1.0, 0.4, -0.6, 0.0), (1.0, 0.8,- 0.2) and

(1.0, -0.8). In the second elimination step we performed the interchange (2,4), and hence write the previous multipliers in the order (1.0, 0.0, -0.6, 0.4). In step 3 the interchange was (3,4), which will affect all the previous multipliers, resulting in (1.0, 0.0, 0.4, -0.6) and (1.0, -0.2, 0.8), whereas (1.0,-0.5), used in this last step, remains unchanged. We put these vectors into the lower triangular of a matrix and obtain

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0.4 & -0.2 & 1 & 0 \\ -0.6 & 0.8 & -0.5 & 1 \end{bmatrix} . \qquad (1.51)$$

It can be readily verified that the matrices (1.46), (1.49), (1.50) and (1.51) satisfy the relation (1.45). Since L is constructed from multipliers, on the basis of the Gaussian elimination algorithm you will understand why the method works.

Now we present a module for the LU decomposition and apply it to compute the determinant of A . As is well known, det(A) is a number, defined by

$$\det(A) = \Sigma(-1)^h(a_{1,k1} \times a_{2,k2} \times \ldots \times a_{n,kn}) \qquad (1.52)$$

where the indices k_i are selected so that there is exactly one element from each column of A in each term of the sum, and we add all the possible combinations. Therefore, the number of terms in (1.52) is n! . In each term the indices k_i take the values 1,2,...,n in different orders. Finally, h in (1.52) is the number of pairwise permutations required to bring all indices k_i into the order 1,2,...,n .

Since det(A) = 0 if and only if A is singular, it provides a convenient way of checking singularity. Determinants have traditionally been used also for solving matrix equations (ref. 10), but both the Gauss-Jordan method and the Gaussian elimination are much more efficient. The determinant itself can easily be calculated by LU decomposition. For the decomposed matrix (1.45)

$$\det(A) = \frac{1}{\det(P)} \det(L) \det(U) . \qquad (1.53)$$

For a triangular matrix the only nonzero term in (1.52) is the product of the diagonal elements. Therefore, det(L) = 1, and $\det(U) = \prod_{i=1}^{n} [U]_{ii}$. There is also only a single nonzero entry in det(P) , so that det(P) = $\pm$1. Since det(I) = 1 , det(P) = +1 if the number of row interchanges translating I into P is even, and det(P)= -1 if this number is odd.

The following module for LU decomposition of an n×n matrix A is based on the algorithm in (ref. 1).

Program module M14

```
1400 REM ****************************************************
1402 REM *       LU DECOMPOSITION OF A SQUARE MATRIX        *
1404 REM ****************************************************
1406 REM INPUT:
1408 REM      N      DIMENSION OF MATRIX
1410 REM    A(N,N)   MATRIX
1412 REM OUTPUT:
1414 REM      ER     STATUS FLAG
1416 REM                0  SUCCESSFUL DECOMPOSITION
1418 REM                1  SINGULAR MATRIX
1420 REM    A(N,N)   MATRIX FACTORS IN PACKED FORM
1422 REM
1424 A(0,0)=1
1426 FOR K=1 TO N-1
1428  M=K
1430  FOR I=K+1 TO N
1432   IF ABS(A(I,K))>ABS(A(M,K)) THEN M=I
1434  NEXT I
1436  A(K,0)=M :A=A(M,K)
1438  IF M>K THEN A(0,0)=-A(0,0) :A(M,K)=A(K,K) :A(K,K)=A
1440  IF A<>0 THEN A=1/A ELSE ER=1 :GOTO 1458
1442  FOR I=K+1 TO N :A(I,K)=-A(I,K)*A :NEXT I
1444  FOR J=K+1 TO N
1446   A=A(M,J) :A(M,J)=A(K,J) :A(K,J)=A
1448   IF A=0 THEN 1452
1450    FOR I=K+1 TO N :A(I,J)=A(I,J)+A(I,K)*A :NEXT I
1452  NEXT J
1454 NEXT K
1456 IF A(N,N)=0 THEN ER=1 ELSE ER=0
1458 RETURN
1460 REM ****************************************************
```

Since only n-1 elimination steps are required for the decomposition, it can be performed also for matrices with rank(**A**) = n-1. For simplicity, however, the module will return the flag ER with value 1 if **A** is singular.

The decomposition is "in place", i.e., all results are stored in the locations that matrix **A** used to occupy. The upper triangular matrix **U** will replace the diagonal elements of **A** and the ones above, whereas **L** is stored in the part below the diagonal, the unit elements of its diagonal being not stored. We will then say that the matrices are in "packed" form. The permutation matrix **P** is not stored at all. As discussed, the row interchanges can be described by n-1 pairs (i,k_i), and all information is contained in a permutation vector k_1, k_2, ..., k_{n-1} that will occupy the entries A(1,0), A(2,0) ,..., A(N-1,0) . The module writes +1 or -1 into A(0,0) depending on the number of row interchanges.

Example 1.3.2 Determinant of a matrix by LU decomposition

The following program performs LU decomposition of the matrix in (1.46) and calculates its determinant.

```
100 REM ---------------------------------------------------------
102 REM EX. 1.3.2. DETERMINANT BY LU DECOMPOSITION
104 REM MERGE M14
106 REM ---------- DATA
108 REM (DIMENSION OF MATRIX)
110 DATA 4
112 DATA  5, 3,-1, 0
114 DATA  2, 0, 4, 1
116 DATA -3, 3,-3, 5
118 DATA  0, 6,-2, 3
200 REM ---------- READ DATA
202 READ N
204 DIM A(N,N)
206 FOR I=1 TO N :FOR J=1 TO N
208  READ A(I,J)
210 NEXT J :NEXT I
212 REM ---------- CALL DECOMPOSITION MODULE
214 GOSUB 1400
216 IF ER=1 THEN D=0 :GOTO 222
218 D=A(0,0)
220 FOR I=1 TO N :D=D*A(I,I) :NEXT I
222 LPRINT "DETERMINANT ...................... ";D
224 LPRINT
226 LPRINT "LU DECOMPOSITION IN PACKED FORM"
228 V$=STRING$(8*(N+1),"-")
230 LPRINT V$
232 LPRINT USING "  ##    ";A(0,0)
234 FOR I=1 TO N
236  LPRINT USING "  ##    ";A(I,0);
238  FOR J=1 TO N :LPRINT USING " ###.###";A(I,J); :NEXT J
240  LPRINT
242 NEXT I
244 LPRINT V$
246 LPRINT
248 STOP
```

The determinant is computed by $\det(A) = \prod_{I=Ø}^{N} A(I,I)$, where A(Ø,Ø) affects only the sign. The resulting matrix is printed as stored, in a packed form. It it is easy to recognise **U** . The elements of **L** are stored with opposite signs, and in the order they originally appeared in the Gaussian elimination.

```
DETERMINANT ...................... 408

LU DECOMPOSITION IN PACKED FORM
----------------------------------------
  1
  1      5.000   3.000  -1.000   0.000
  4     -0.400   6.000  -2.000   3.000
  4      0.600  -0.800   4.000   1.600
  0      0.000   0.200   0.500   3.400
----------------------------------------
```

1.3.1 Solution of matrix equations

We can use the LU decomposition to solve the equation $\mathbf{Ax} = \mathbf{b}$ very efficiently, where $\mathbf{A}$ is a nonsingular square matrix. Multiplying the equation by a permutation matrix $\mathbf{P}$ we have $\mathbf{PAx} = \mathbf{Pb}$, and hence $\mathbf{LUx} = \mathbf{Pb}$ by (1.45) . This last equation is very easy to solve, first by solving for a vector $\mathbf{d}$ such that

$$\mathbf{Ld} = \mathbf{Pb} \tag{1.54}$$

and then solving

$$\mathbf{Ux} = \mathbf{d} \ . \tag{1.55}$$

Since both $\mathbf{L}$ and $\mathbf{U}$ are triangular matrices, (1.54) and (1.55) are solved by the simplest backsubstitution except for taking into account the right-hand side interchanges in (1.54). The next module performs these calculations.

Program module M15

```
1500 REM ****************************************************
1502 REM *   SOLUTION OF SIMULTANEOUS LINEAR EQUATIONS      *
1504 REM *    BACKWARD SUBSTITUTION USING LU FACTORS        *
1506 REM ****************************************************
1508 REM INPUT:
1510 REM      N       NUMBER OF EQUATIONS
1512 REM    A(N,N)    LU DECOMPOSITION OF THE COEFFICIENT MATRIX
1514 REM     X(N)     RIGHT HAND SIDE
1516 REM OUTPUT:
1518 REM     X(N)     SOLUTION
1520 FOR K=1 TO N-1
1522  I=A(K,0) :A=X(I) :X(I)=X(K) :X(K)=A
1524  FOR I=K+1 TO N :X(I)=X(I)+A(I,K)*A :NEXT I
1526 NEXT K
1528 FOR K=N TO 1 STEP -1
1530  X(K)=X(K)/A(K,K)
1532  FOR I=1 TO K-1 :X(I)=X(I)-A(I,K)*X(K) :NEXT I
1534 NEXT K
1536 RETURN
1538 REM ****************************************************
```

On input the array A contains the decomposed matrix as given by the module M14, and the right-hand side coefficients are placed into the vector X. On output, this vector will store the solution. There is nothing to go wrong in backsubstitution if the previous decomposition was successful, and hence we dropped the error flag.

Example 1.3.3 Solution of simultaneous linear equations by LU decomposition

We use the following main program to solve (1.46) by the modules M14 and M15:

```
100 REM ------------------------------------------------------------
102 REM EX. 1.3.3. SOLUTION OF LINEAR EQUATIONS BY LU DECOMPOSITION
104 REM MERGE M14,M15
106 REM ---------- DATA
108 REM (NUMBER OF EQUATIONS)
110 DATA 4
112 DATA  5, 3,-1, 0, =, 11
114 DATA  2, 0, 4, 1, =,  1
116 DATA -3, 3,-3, 5, =, -2
118 DATA  0, 6,-2, 3, =,  9
200 REM ---------- READ DATA
202 READ N
204 DIM A(N,N),X(N)
206 FOR I=1 TO N
208  FOR J=1 TO N :READ A(I,J) :NEXT J
210  READ A$,X(I)
212 NEXT I
214 REM ---------- CALL DECOMPOSITION MODULE
216 GOSUB 1400
218 IF ER=1 THEN LPRINT "COEFFICIENT MATRIX IS SINGULAR" :GOTO 238
220 REM ---------- CALL SOLUTION MODULE
222 GOSUB 1500
224 LPRINT "SOLUTION OF THE SYSTEM OF LINEAR EQUATIONS" :LPRINT
226 V$=STRING$(16,"-")
228 LPRINT V$
230 LPRINT " I       X(I)"
232 LPRINT V$
234 FOR I=1 TO N :LPRINT USING "##    ##.####"; I,X(I) :NEXT I
236 LPRINT V$ :LPRINT
238 STOP
```

The coefficients and the right-hand sides are separated by the character "=" but you can use any other character sequence as a separator. The resulting output agrees with our hand calculations:

```
SOLUTION OF THE SYSTEM OF LINEAR EQUATIONS

----------------
 I      X(I)
----------------
 1     1.0000
 2     2.0000
 3     0.0000
 4    -1.0000
----------------
```

A special property of solving a matrix equation in this way is that the LU decomposition does not involve the right-hand side vector **b** , in contrast both to the Gauss-Jordan method and to the Gaussian elimination. This is particularly advantageous when solving several matrix equations with the same coefficient matrix **A** , as we do in the next section.

1.3.4 Matrix Inversion

As discussed in Section 1.1.4, to calculate the inverse of the $n \times n$ matrix **A** one solves n matrix equations $\mathbf{Ax} = \mathbf{e}_i$, $i = 1,2,\dots,n$, and hence the LU decomposition is particularly advantageous. You must, however, never compute $\mathbf{A}^{-1}$ only to obtain the solution of the matrix equation $\mathbf{Ax} = \mathbf{b}$ in the form $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ since the method applied in Example 1.3.3 is more efficient.

Example 1.3.4 Inversion of a square matrix

We find the inverse of the matrix in (1.46). On input, the original matrix is stored in the array A , and its inverse will occupy the array B on output. Performing LU decomposition by the module M14, the original matrix will be destroyed. The program and the output are as follows:

```
100 REM -------------------------------------------------------
102 REM EX. 1.3.4. INVERSION OF A MATRIX BY LU DECOMPOSITION
104 REM MERGE M14,M15
106 REM ---------- DATA
108 REM (DIMENSION OF MATRIX)
110 DATA 4
112 DATA  5, 3,-1, 0
114 DATA  2, 0, 4, 1
116 DATA -3, 3,-3, 5
118 DATA  0, 6,-2, 3
200 REM ---------- READ DATA
202 READ N
204 DIM A(N,N),B(N,N),X(N)
206 FOR I=1 TO N :FOR J=1 TO N
208  READ A(I,J)
210 NEXT J :NEXT I
212 REM ---------- CALL DECOMPOSITION MODULE
214 GOSUB 1400
216 IF ER=1 THEN LPRINT "MATRIX IS SINGULAR" :GOTO 246
218 REM ---------- CALL SOLUTION MODULE
220 FOR J=1 TO N
222  FOR I=1 TO N :X(I)=-(I=J) :NEXT I
224  GOSUB 1500
226  FOR I=1 TO N :B(I,J)=X(I) :NEXT I
228 NEXT J
230 V$=STRING$(8*N,"-")
232 LPRINT "INVERSE MATRIX:" :LPRINT
234 LPRINT V$
236 FOR I=1 TO N
238  FOR J=1 TO N :LPRINT USING " ###.###";B(I,J); :NEXT J
240  LPRINT
242 NEXT I
244 LPRINT V$ :LPRINT
246 STOP
```

```
INVERSE MATRIX:

--------------------------------
  0.235  0.044  0.088 -0.162
 -0.108 -0.010 -0.186  0.314
 -0.147  0.191 -0.118  0.132
  0.118  0.147  0.294 -0.206
--------------------------------
```

You may find interesting to compare the results with the output in Example 1.1.4.

1.4 INVERSION OF A SYMMETRIC POSITIVE DEFINITE MATRIX

As you learned in the previous sections, LU decomposition with built-in partial pivoting, followed by backsubstitution is a good method to solve the matrix equation $\mathbf{A}\mathbf{x} = \mathbf{b}$. You can use, however, considerable simpler technics if the matrix $\mathbf{A}$ has some special structure. In this section we assume that $\mathbf{A}$ is symmetric (i.e., $\mathbf{A}^T = \mathbf{A}$), and positive definite (i.e., $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$; you will encounter the expression $\mathbf{x}^T\mathbf{A}\mathbf{x}$ many times in this book, and hence we note that it is called quadratic form.) The problem considered here is special, but very important. In particular, estimating parameters in Chapter 3 you will have to invert matrices of the form $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ many times, where $\mathbf{X}$ is an $n \times m$ matrix. The matrix $\mathbf{X}^T\mathbf{X}$ is clearly symmetric, and it is positive definite if the columns of $\mathbf{X}$ are linearly independent. Indeed, $\mathbf{x}^T(\mathbf{X}^T\mathbf{X})\mathbf{x} = (\mathbf{X}\mathbf{x})^T(\mathbf{X}\mathbf{x}) \geq 0$ for every $\mathbf{x}$ since it is a sum of squares. Thus $(\mathbf{X}\mathbf{x})^T(\mathbf{X}\mathbf{x}) = 0$ implies $\mathbf{X}\mathbf{x} = 0$ and also $\mathbf{x} = \mathbf{0}$ if the columns of $\mathbf{X}$ are linearly independent.

A positive definite symmetric matrix $\mathbf{A}$ can be decomposed in the form $\mathbf{A} = \mathbf{H}\mathbf{H}^T$ where $\mathbf{H}$ is a lower triangular matrix, by the method of Cholevsky. An interesting application is to decompose the inverse in the form $\mathbf{A}^{-1} = \mathbf{Q}^T\mathbf{Q}$, where $\mathbf{Q}$ is an upper triangular matrix, easily obtainable from $\mathbf{H}$. We will need such a decomposition when dealing with error-in-variables models in Chapter 3 . You may find details of the algorithm in (ref. 2), and the corresponding BASIC statements in the module M52 of Chapter 3. Here we provide a module based on Gaussian elimination, for inverting a positive definite matrix.

The method (ref. 2) is based on solving the matrix equation $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{y}$ is not a fixed right-hand side, but a vector of variables $y_1, y_2, \ldots, y_n$ with completely "free" values. To solve the equation for $\mathbf{x}$ in terms of $\mathbf{y}$ notice that $a_{11} \neq 0$ due to positive definiteness of $\mathbf{A}$, since $a_{11} = (\mathbf{e}_1)^T\mathbf{A}\mathbf{e}_1$. We can therefore solve the first equation for x_1, and replace x_1 by the resulting expression in the other equations:

$$x_1 = a'_{11}y_1 + a'_{12}x_2 + \ldots + a'_{1n}x_n$$

$$y_2 = a'_{21}y_1 + a'_{22}x_2 + \ldots + a'_{2n}x_n$$

$$\begin{matrix} \cdot \\ \cdot \\ \cdot \end{matrix} \qquad (1.56)$$

$$y_n = a'_{n1}y_1 + a'_{n2}x_2 + \ldots + a'_{nn}x_n$$

where the new coefficients are:

$$a'_{11} = 1/a_{11} \; ,$$

$$a'_{1j} = -a_{1j}/a_{11} \; , \qquad (\; j = 2, 3, \ldots, n \;) \; ,$$

$$a'_{i1} = a_{i1}/a_{11} \; , \qquad (\; i = 2, 3, \ldots, n \;) \; ,$$

$$a'_{ij} = a_{ij} - a_{i1}a_{1j}/a_{11} \; , \qquad (\; i,j = 2, 3, \ldots, n \;) \; .$$

To proceed we have to assume that $a'_{22} \neq 0$. It can be shown that this follows from the positive definiteness of $\mathbf{A}$, (see ref. 2). If $a'_{22} \neq 0$ then we solve the second equation of (1.56) for x_2 in terms of $y_1, x_3, \ldots, x_n$, and replace x_2 with the resulting expression in all the other equations. Since $\mathbf{A}$ is positive definite, we can perform all the elimination steps in this way and obtain $\mathbf{x}$ in terms of $\mathbf{y}$ as

$$\mathbf{x} = \mathbf{By} \; . \qquad (1.57)$$

Since $\mathbf{y} = \mathbf{Ax}$ according to the original equation, $\mathbf{B} = \mathbf{A}^{-1}$ follows, thus we obtain the inverse in place of the original matrix.

Though the procedure appears to be special, you will notice that it is essentially a Gaussian elimination without pivoting.

The following module is based on the algorithm in (ref. 2). Its concise structure is due to cyclic renumbering of both groups of variables, so that always x_1 is expressed and always from the first equation.

Program module M16

```
1600 REM ####################################################
1602 REM #INVERSION OF A POSITIVE DEFINITE SYMMETRIC MATRIX#
1604 REM ####################################################
1606 REM INPUT:
1608 REM      N       DIMENSION OF MATRIX
1610 REM    A(N,N)    MATRIX
1612 REM              (ONLY LOWER TRIANGLE IS USED)
1614 REM OUTPUT;
1616 REM      ER      STATUS FLAG
1618 REM                 0  SUCCESSFUL INVERSION
1620 REM                 1  MATRIX IS NOT POSITIVE DEFINITE
1622 REM    A(N,N)    INVERSE MATRIX
1624 REM              (INCLUDING THE UPPER TRIANGLE)
1626 FOR K=N TO 1 STEP -1
1628  IF A(1,1)<=0 THEN ER=1 :GOTO 1654
1630  A(0,N)=1/A(1,1)
1632  FOR I=2 TO N
1634   A=A(I,1)*A(0,N)
1636   IF I>K THEN A(0,I-1)=A ELSE A(0,I-1)=-A
1638   FOR J=2 TO I :A(I-1,J-1)=A(I,J)+A(I,1)*A(0,J-1) :NEXT J
1640  NEXT I
1642  FOR I=1 TO N :A(N,I)=A(0,I) :NEXT I
1644 NEXT K
1646 FOR I=1 TO N-1
1648  FOR J=I+1 TO N :A(I,J)=A(J,I) :NEXT J
1650 NEXT I
1652 ER=0
1654 RETURN
1656 REM ####################################################
```

Since A is symmetric, on input it is sufficient to store its corresponding portion in the lower triangular part of the array A , including the diagonal. The inverse is also symmetric, but on output it will occupy the entire matrix, since ithis i advantageous for further use. The zero-th row of array A is used as a vector of auxiliary variables, so do not store your own data here. If the matrix is not positive definite, the module will return the flag ER = 1. As we discussed, for a matrix of the form X^TX this implies singularity. For a general symmetric matrix, however, the return value ER = 1 does not necessarily imply its singularity, and you can still try to use the modules M14 and M15 in order to invert the matrix.

Example 1.4 Inversion of a positive definite matrix

Our test problem will involve the Hilbert matrix of order 6, defined by

$$[H_6]_{ij} = 1/(i + j - 1) , \qquad i,j = 1, 2, ..., 6 . \tag{1.58}$$

Hilbert matrices (and obviously their inverses) are positive definite and are frequently used for testing algebraic procedures (ref. 1). We present a main program, with the lower triangular of the inverse of H_6 in the DATA statements.

```
100 REM ----------------------------------------------------------
102 REM EX. 1.4. INVERSION OF A POSITIVE DEFINITE SYMMETRIC MATRIX
104 REM MERGE M16
106 REM ---------- DATA
108 REM (DIMENSION OF MATRIX)
110 DATA 6
112 REM (LOWER TRIANGULAR PART)
114 DATA    36
116 DATA  -630,    14700
118 DATA  3360,   -88200,   564480
120 DATA -7560,   211680, -1411200,   3628800
122 DATA  7560,  -220500,  1512000,  -3969000,  4410000
124 DATA -2772,    83160,  -582120,   1552320, -1746360,  698544
200 REM ---------- READ DATA
202 READ N
204 DIM A(N,N)
206 FOR I=1 TO N :FOR J=1 TO I
208  READ A(I,J)
210 NEXT J :NEXT I
212 REM ---------- CALL INVERSION MODULE
214 GOSUB 1600
216 IF ER=0 THEN 220
218 LPRINT "MATRIX IS SINGULAR" :GOTO 236
220 LPRINT "INVERSE MATRIX:" :LPRINT
222 V$=STRING$(9*N,"-")
224 LPRINT V$
226 FOR I=1 TO N
228  FOR J=1 TO N :LPRINT USING " ###.####";A(I,J); :NEXT J
230  LPRINT
232 NEXT I
234 LPRINT V$ :LPRINT
236 STOP
```

We expect to obtain elements that satisfy (1.58). The program output is:

```
INVERSE MATRIX:

------------------------------------------------------
   0.9995   0.4996   0.3330   0.2497   0.1997   0.1664
   0.4996   0.3330   0.2497   0.1997   0.1664   0.1426
   0.3330   0.2497   0.1997   0.1664   0.1426   0.1248
   0.2497   0.1997   0.1664   0.1426   0.1248   0.1109
   0.1997   0.1664   0.1426   0.1248   0.1109   0.0998
   0.1664   0.1426   0.1248   0.1109   0.0998   0.0908
------------------------------------------------------
```

As you see, the elements are accurate only up to three digits. To get more accurate results, you may repeat the same calculation in double precision inserting the BASIC line:

```
99 DEFDBL A
```

We will return to the problem of numerical accuracy in Sections 1.7 and 1.8.6. Here we only note that similar problems may arise even with full pivoting. (You may try it using the program of Example 1.1.2 .)

1.5 TRIDIAGONAL SYSTEM OF EQUATIONS

Another special case of the matrix equation **Ax=b** is the one with **A** beeing tridiagonal, i.e., having nonzero elements only on the diagonal plus or minus one column. For example, the equations

$$\begin{array}{rcl} 4x_1 + 2x_2 & & = 1 \\ x_1 + 4x_2 + x_3 & & = 2 \\ x_2 + 4x_3 + x_4 & & = 3 \\ x_3 + 4x_4 + x_5 & & = 4 \\ 2x_4 + 4x_5 & & = 5 \end{array} \qquad (1.59)$$

form a tridiagonal system. To devise a very simple algorithm for solving equations of this form we need a further special property called diagonal dominance. The coefficient matrix **A** is diagonally dominant if

$|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for all i, i.e., each diagonal element is sufficiently large in magnitude. As in the previous section, these assumptions are restrictive, but satisfied in a number of important applications. For example, we solve tridiagonal systems of linear equations when interpolating by spline functions in Section 5.3, and a similar problem arises in modelling distillation columns (the latter is not treated in this book).

The Gaussian elimination can be used without pivoting because of diagonal dominance (ref. 1). Due to the many zeros the algorithm (sometimes called Thomas algorithm) is very easy to implement:

Program module M17

```
1700 REM ****************************************************
1702 REM *    LINEAR EQUATIONS WITH TRIDIAGONAL MATRIX      *
1704 REM ****************************************************
1706 REM INPUT:
1708 REM      N      NUMBER OF EQUATIONS
1710 REM A(N),B(N),C(N),D(N)
1712 REM             COEFFICIENTS AND RIGHT HAND SIDE IN
1714 REM               THE I-TH EQUATION OF THE FORM
1716 REM             A(I)*X(I-1)+B(I)*X(I)+C(I)*X(I+1)=D(I)
1718 REM OUTPUT:
1720 REM     X(N)    SOLUTION
1722 REM AUXILIARY ARRAY:
1724 REM     P(N)
1726 P(1)=B(1) :X(1)=D(1)
1728 FOR I=2 TO N
1730  Q=A(I)/P(I-1) :P(I)=B(I)-Q*C(I-1) :X(I)=D(I)-Q*X(I-1)
1732 NEXT I
1734 X(N)=X(N)/P(N)
1736 FOR I=N-1 TO 1 STEP -1 :X(I)=(X(I)-C(I)*X(I+1))/P(I) :NEXT I
1738 RETURN
1740 REM ****************************************************
```

The 3 nonzero entries of the i-th row of the coefficient matrix occupy the variables A(I), B(I) and C(I) ,so that A(1) and C(N) are not used. We need an auxiliary vector P() of N elements. Since there is no pivoting, you may experience overflow (i.e., division by a too small pivot) even for a nonsingular matrix, if it is not diagonally dominant.

Example 1.5 Solution of a tridiagonal matrix equation

The matrix in (1.59) is diagonally dominant, and we can use module M17 to solve the equation. As in example 1.3.3, we separate the coefficients from the right hand side by the character "=" in each DATA line.

```
100 REM ------------------------------------------------------------
102 REM EX. 1.5. SOLUTION OF LINEAR EQUATIONS WITH TRIDIAGONAL MATRIX
104 REM MERGE M17
106 REM ---------- DATA
108 REM (NUMBER OF EQUATIONS)
110 DATA 5
112 DATA 4,2,      =,1
114 DATA 1,4,1,    =,2
116 DATA   1,4,1,  =,3
118 DATA     1,4,1,=,4
120 DATA       2,4,=,5
200 REM ---------- READ DATA
202 READ N
204 DIM A(N),B(N),C(N),D(N),X(N),P(N)
206 FOR I=1 TO N
208  IF I>1 THEN READ A(I)
210  READ B(I)
212  IF I<N THEN READ C(I)
214  READ A$,D(I)
216 NEXT I
218 REM ---------- CALL SOLUTION MODULE
220 GOSUB 1700
222 LPRINT "SOLUTION:" :LPRINT
224 V$=STRING$(16,"-")
226 LPRINT " I      X(I)"
228 LPRINT V$
230 FOR I=1 TO N :LPRINT I;TAB(6);X(I) :NEXT I
232 LPRINT V$ :LPRINT
234 STOP
```

The results are as follows:

```
SOLUTION:

 I      X(I)
----------------
 1    7.142857E-02
 2    .3571429
 3    .5
 4    .6428571
 5    .9285713
----------------
```

1.6 EIGENVALUES AND EIGENVECTORS OF A REAL SYMMETRIC MATRIX

In Section 1.1 we defined the eigenvalue λ and the eigenvector u of the $n \times n$ matrix A to satisfy the matrix equation

$$(A-\lambda I)u = 0 \ . \qquad (1.60)$$

This is a homogeneous set of linear equations (i.e., its righ-hand side is zero), and has a nonzero solution if and only if the columns of $(A-\lambda I)$ are linearly dependent. Thus

$$\det(A-\lambda I) = 0 \ , \qquad (1.61)$$

which is said to be the characteristic equation of A . By the definition (1.52) of the determinant, the left-hand side of (1.61), if expanded, is a polynomial of degree n in λ whose n roots are the eigenvalues of A . If A is symmetric, all eigenvalues $\lambda_1,\lambda_2,\ldots,\lambda_n$ are real (ref. 10). The i-th eigenvector u_i can be obtained by solving the equation

$$(A-\lambda_i I)u_i = 0 \ . \qquad (1.62)$$

The solution of this equation is not unique. The eigenvector will be, however, uniquely defined if we prescribe its length, e.g., by the constraint $\|u_i\|^2 = u^T_i u_i = 1$, and specify the sign of its first nonzero element. An eigenvector of unit length is called normalized eigenvector. Assume that all eigenvalues $\lambda_1,\lambda_2,\ldots,\lambda_n$ of A are different, i.e., the characteristic equation has no repeated roots. Then the eigenvectors $u_1,u_2,\ldots,u_n$ are linearly independent and form a basis of the n dimensional space. Furthermore, the eigenvectors are pairwise orthogonal, and the set $u_1,u_2,\ldots u_n$ of normalized eigenvectors is said to be orthonormal, which means the property

$$u^T_i u_i = \begin{cases} 1 \text{ if } i=j \\ 0 \text{ otherwise.} \end{cases} \qquad (1.63)$$

Consider the matrix $B = T^{-1}AT$, where T is an $n \times n$ nonsingular matrix, and find the eigenvalues of B . Since $\det(T)\det(T^{-1}) = 1$,

$$\det(B-\lambda I) = \det(T^{-1}(A-\lambda I)T) = \det(A-\lambda I) \ . \qquad (1.64)$$

Thus each eigenvalue of B is an eigenvalue of A and vice versa. In this case the matrices A and B are said to be similar and T is called similarity transformation.

An important application of eigenanalysis is the diagonalization of a (symmetric) matrix A . Let U denote the matrix whose columns are the normalized eigenvectors $u_1,u_2,\ldots,u_n$. By the definition (1.60) we have

$$AU = UD \qquad (1.66)$$

where D denotes the $n \times n$ diagonal matrix with the diagonal elements $\lambda_1, \lambda_2, \ldots, \lambda_n$. The matrix U of eigenvectors is nonsingular, and by virtue of (1.63), $U^{-1} = U^T$ (i.e., $U^TU = I$). Therefore, from (1.65) we obtain

$$U^TAU = D \ . \qquad (1.66)$$

The eigenvalues of A can be find by solving the characteristic equation of (1.61). It is much more efficient to look for similarity transformations that will translate A into the diagonal form with the eigenvalues in the diagonal. The Jacobi method involves a sequence of orthonormal similarity transformations $T_1, T_2, \ldots$ such that $A_{k+1} = T^T{}_k A_k T_k$. The matrix T_k differs from the identity matrix only in four elements: $t_{pp} = t_{qq} = \cos z$ and $t_{pq} = -t_{qp} = \sin z$. We can chose a value for z such that $\left[A_{k+1}\right]_{pq} = 0$, but the transformation may "bring back" some off-diagonal elements, annihilated in the previous steps. Nevertheless, the diagonal form (1.66) may be approximated with a desired accuracy after sufficiently large number k of steps. The diagonal elements of A_k will then converge to the eigenvalues and the accumulated product $T_1T_2\ldots T_k$ to the matrix U of the eigenvectors. In the classical Jacobi iteration always the largest (in magnitude) off-diagonal element is annihilated and the search for it is time consuming. A better strategy is to annihilate the first off-diagonal element which is larger than a certain threshold, and decrease the threshold when no more such element is found (ref. 11). This is the basis of the following program module:

Program module M18

```
1800 REM ****************************************************
1802 REM *   EIGENVALUES AND EIGENVECTORS OF A SYMMETRIC    *
1804 REM *            MATRIX  -  JACOBI METHOD              *
1806 REM ****************************************************
1808 REM INPUT:
1810 REM      N        DIMENSION OF MATRIX
1812 REM   A(N,N)      MATRIX
1814 REM               (ONLY LOWER TRIANGLE IS USED)
1816 REM OUTPUT:
1818 REM   U(0,J)      J=1 TO N, EIGENVALUES
1820 REM               (IN DECREASING ORDER)
1822 REM   U(I,J)      I=1 TO N, J-TH EIGENVECTOR
1824 REM               ( LOWER TRIANGLE OF MATRIX A(.,.) IS OVERWRITTEN )
1826 FOR I=1 TO N :FOR J=1 TO N
1828  U(I,J)=-(I=J)
1830 NEXT J :NEXT I
1832 V=0
1834 FOR I=2 TO N :FOR J=1 TO I-1
1836  V=V+ABS(A(I,J))
1838 NEXT J :NEXT I
```

```
1840 IF V=0 THEN 1922
1842 V0=V/N/N*.000005 :V1=0
1844 V=V/N
1846 FOR I0=2 TO N :FOR J0=1 TO I0-1
1848  IF ABS(A(I0,J0))<=V THEN 1916
1850  V1=1
1852  IF A(J0,J0)=A(I0,I0) THEN T=1 :GOTO 1862
1854  IF A(J0,J0)>A(I0,I0) THEN V2=1 ELSE V2=-1
1856  V3=ABS(A(J0,J0)-A(I0,I0))
1858  V4=SQR((A(J0,J0)-A(I0,I0))^2+4*A(I0,J0)^2)
1860  T=2*A(I0,J0)*V2/(V3+V4)
1862  C=1/SQR(1+T^2)
1864  S=T*C
1866  C1=C^2 :S1=S^2 :T1=T^2
1868  V5=A(I0,I0)
1870  A(I0,I0)=C1*(V5-2*T*A(I0,J0)+T1*A(J0,J0))
1872  A(J0,J0)=C1*(A(J0,J0)+2*T*A(I0,J0)+T1*V5)
1874  A(I0,J0)=0
1876  FOR J=1 TO J0-1
1878   V5=-S*A(J0,J)+C*A(I0,J)
1880   A(J0,J)=C*A(J0,J)+S*A(I0,J)
1882   A(I0,J)=V5
1884  NEXT J
1886  FOR I=J0+1 TO I0-1
1888   V5=-S*A(I,J0)+C*A(I0,I)
1890   A(I,J0)=C*A(I,J0)+S*A(I0,I)
1892   A(I0,I)=V5
1894  NEXT I
1896  FOR I=I0+1 TO N
1898   V5=-S*A(I,J0)+C*A(I,I0)
1900   A(I,J0)=C*A(I,J0)+S*A(I,I0)
1902   A(I,I0)=V5
1904  NEXT I
1906  FOR I=1 TO N
1908   V5=C*U(I,I0)-S*U(I,J0)
1910   U(I,J0)=S*U(I,I0)+C*U(I,J0)
1912   U(I,I0)=V5
1914  NEXT I
1916 NEXT J0 :NEXT I0
1918 IF V1=1 THEN V1=0 :GOTO 1846
1920 IF V>=V0 THEN 1844
1922 REM ---------- SORT IN DECREASING ORDER
1924 FOR I=1 TO N :U(0,I)=A(I,I) :NEXT I
1926 FOR I=2 TO N :L=I
1928  IF U(0,L-1)>=U(0,L) THEN 1934
1930   FOR J=0 TO N :T=U(J,L-1) :U(J,L-1)=U(J,L) :U(J,L)=T :NEXT J
1932   IF L>2 THEN L=L-1 :GOTO 1928
1934 NEXT I
1936 RETURN
1938 REM ****************************************************
```

Since the matrix A is symmetric, on input we store only its lower triangular portion (including the diagonal) in the corresponding entries of the array A. This matrix will be destroyed during calculations. The resulting eigenvalues occupy the zeroth row of array U, in decreasing order. The corresponding eigenvectors are stored in the corresponding column of array U. (You can monitor the iteration by printing the actual value of the threshold stored in the variable V, for instance in line 1845.)

<u>Example 1.6</u> Eigenvalues and eigenvectors of a symmetric matrix

The eigenanalysis of the matrix (ref. 2)

$$A = \begin{bmatrix} 10 & 1 & 2 & 3 & 4 \\ 1 & 9 & -1 & 2 & -3 \\ 2 & -1 & 7 & 3 & -5 \\ 3 & 2 & 3 & 12 & -1 \\ 4 & -3 & -5 & -1 & 15 \end{bmatrix}$$

is carried out by the following main program:

```
100 REM ------------------------------------------------------------
102 REM EX. 1.6. EIGENVALUE-EIGENVECTOR DECOMPOSITION OF A SYM. MATRIX
104 REM MERGE M18
106 REM ---------- DATA
108 REM (DIMENSION OF MATRIX)
110 DATA 5
112 REM (LOWER TRIANGULAR PART)
114 DATA  10
116 DATA   1, 9
118 DATA   2,-1, 7
120 DATA   3, 2, 3, 12
122 DATA   4,-3,-5, -1, 15
200 REM ---------- READ DATA
202 READ N
204 DIM A(N,N),U(N,N)
206 FOR I=1 TO N :FOR J=1 TO I
208  READ A(I,J)
210 NEXT J :NEXT I
212 REM ---------- CALL JACOBI MODULE
214 GOSUB 1800
216 REM ----------LPRINT RESULTS
218 V$=STRING$(13*N,"-")
220 LPRINT "EIGENVALUES:"
222 LPRINT V$
224 FOR J=1 TO N :LPRINT USING " #.#####^^^^ ";U(0,J); :NEXT J
226 LPRINT :LPRINT V$
228 LPRINT :LPRINT "EIGENVECTORS:"
230 LPRINT V$
232 FOR J=1 TO N :LPRINT USING "     u#     ";J; :NEXT J
234 LPRINT
236 FOR I=1 TO N
238  FOR J=1 TO N :LPRINT USING "  ##.######  ";U(I,J); :NEXT J
240 LPRINT
242 NEXT I
244 LPRINT V$ :LPRINT
246 STOP
```

The program output is as follows.

```
EIGENVALUES:
---------------------------------------------------------------------
 0.19175E+02  0.15809E+02  0.93656E+01  0.69948E+01  0.16553E+01
---------------------------------------------------------------------

EIGENVECTORS:
---------------------------------------------------------------------
     u1            u2            u3            u4            u5
   0.174504      0.623703     -0.052151      0.654093     -0.387297
  -0.247303      0.159101      0.859964      0.199681      0.366221
  -0.361642      0.227297     -0.505575      0.256510      0.704377
  -0.264412      0.692684     -0.000201     -0.660403     -0.118926
   0.841244      0.232823      0.046219     -0.174280      0.453423
---------------------------------------------------------------------
```

<u>Exercise</u>

□ Check the results on the basis of (1.65).

1.7 ACCURACY IN ALGEBRAIC COMPUTATIONS. ILL-CONDITIONED PROBLEMS

Solving the matrix equation $\mathbf{Ax} = \mathbf{b}$ by LU decomposition or by Gaussian elimination you perform a number of operations on the coefficient matrix (and also on the right-hand side vector in the latter case). The precision in each step is constrained by the precision of your computer's floating-point word that can deal with numbers within certain range. Thus each operation will introduce some round-off error into your results, and you end up with some residual $\mathbf{r} = \mathbf{A}\bar{\mathbf{x}} - \mathbf{b} \neq \mathbf{0}$, where $\bar{\mathbf{x}}$ is the numerical solution of the equation. You have seen that pivoting will decrease the round-off errors and hence the residual $\mathbf{r}$. You can also decrease the errors by using double-precision variables thereby increasing the range of your floating-point arithmetics.

Another promising way to reduce the residual $\mathbf{r}$ is to perform an iterative improvement of the solution. The equations we use are $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{A}\bar{\mathbf{x}} - \mathbf{b} = \mathbf{r}$. Substracting the first equation from the second one gives $\mathbf{Ae} = \mathbf{r}$, where $\mathbf{e} = \bar{\mathbf{x}} - \mathbf{x}$ is the error in the solution $\bar{\mathbf{x}}$. We have two expressions for $\mathbf{r}$ that yield the equation $\mathbf{Ae} = \mathbf{A}\bar{\mathbf{x}} - \mathbf{b}$ with known terms on the right-hand side, since $\bar{\mathbf{x}}$ is the solution we want to improve. We need only to solve this equation for $\mathbf{e}$ and to get the improved solution $\mathbf{x} = \bar{\mathbf{x}} - \mathbf{e}$. Of course, neither $\mathbf{e}$ can be computed without error, but it will certainly reduce the error in $\mathbf{x}$. We can repeat this step iteratively until all elements of $\mathbf{r}$

will be indistinguishable from zero, which obviously means the machine epsilon of the computer we use. It is highly advisable to calculate at least the product $\mathbf{A}\tilde{\mathbf{x}}$ in double precision.

While the residual $\mathbf{r}$ can be considerable reduced by iterative improvement, in many problems this does not mean that the residual error $\mathbf{e}$ will be also small. To relate $\mathbf{e}$ to $\mathbf{r}$, define the norm $\|\mathbf{A}\|$ of the square matrix $\mathbf{A}$ by

$$\|\mathbf{A}\| = \max_{\|\mathbf{x}\|\neq 0} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\| \tag{1.67}$$

which is a straightforward extension of the norm of a vector as defined in Section 1.1. According to (1.67)

$$\|\mathbf{Ax}\| \le \|\mathbf{A}\|\|\mathbf{x}\| \tag{1.68}$$

for all $\mathbf{A}$ and $\mathbf{x}$. Since $\mathbf{r} = \mathbf{Ae}$ and $\mathbf{A}$ is nonsingular, $\mathbf{e} = \mathbf{A}^{-1}\mathbf{r}$, and by (1.68)

$$\|\mathbf{e}\| \le \|\mathbf{A}^{-1}\|\|\mathbf{r}\| \ . \tag{1.69}$$

Since $\mathbf{b} = \mathbf{Ax}$,

$$\|\mathbf{b}\| \le \|\mathbf{A}\|\|\mathbf{x}\| \ . \tag{1.70}$$

Multiplying the two last inequalities and rearranging, for $\mathbf{b} \neq \mathbf{0}$ we have

$$\frac{\|\mathbf{e}\|}{\|\mathbf{x}\|} \le \|\mathbf{A}\|\|\mathbf{A}^{-1}\| \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} , \tag{1.71}$$

the desired relationship between the relative residual $\|\mathbf{r}\|/\|\mathbf{b}\|$ and the relative error $\|\mathbf{e}\|/\|\mathbf{r}\|$, where the $\|\mathbf{A}\|\|\mathbf{A}^{-1}\|$ is called the condition number of $\mathbf{A}$, denoted by cond($\mathbf{A}$) . By (1.71) cond($\mathbf{A}$) is the relative error magnification factor, and its value is at least one. If it is very large, the relative error in $\mathbf{x}$ will be large in spite of carefully reducing the residual $\mathbf{r}$ by one of the methods discussed. Such problems are said to be ill-conditioned or nearly singular, and can only be solved by sophisticated regularization methods (ref. 12). The basic idea of regularization is replacing $\mathbf{A}$ by a sequence of matrices $\mathbf{A}_1$, $\mathbf{A}_2$, ... such that cond($\mathbf{A}_i$) < cond($\mathbf{A}$) . The matrices $\mathbf{A}_i$ approximate $\mathbf{A}$, but we constrain cond($\mathbf{A}_i$) by a suitable upper bound. In practice it is far from easy to select a reasonable termination condition.

As a numerical analyst you may have to solve inherently ill-conditioned problems, but in scientific computing there are further opportunities. Neglecting or coupling unimportant variables, seeking further constraints or devising new experiments for further information may help you to derive a "better" model and avoid near-singularity in computations. While this is one of the basic ideas of scientific computing, it is too general to be useful, and we can give you further suggestions only in particular applications (e.g., in Chapter 3).

1.8 APPLICATIONS AND FURTHER PROBLEMS

1.8.1 Stoichiometry of chemically reacting species

While linear algebraic methods are present in almost every problem, they also have a number of direct applications. One of them is formulating and solving balance equations for extensive quantities such as mass and energy. A particularly nice application is stoichiometry of chemical systems, where you will discover most of the the basic concepts of linear algebra under different names.

We consider a closed system with k species denoted by $M_1, M_2, \ldots, M_k$. Let n_i denote the quantity of species M_i expressed in moles. The k-vector $n = (n_1, n_2, \ldots, n_k)^T$ is called the mole vector and we are interested in its change $\Delta n = n - n^o$ with respect to an initial state n^o. Since the sytem is closed, the mole vector changes Δn are not arbitrary. Stoichiometry offers two ways to specify the set of admissible mole vector changes, i.e. the stoichiometric subspace. In particular applications (e.g. when calculating chemical equilibrium) one or the other approach might be more advantageous, so that we study their relation here.

The first approach is based on explicitly describing chemical reactions. We suppose that there are p reactions taking place in the system. The j-th reaction is described by equation of the form

$$\sum_{i=1}^{k} b_{ij}M_i = 0 , \qquad (1.72)$$

where the stoichiometric coefficients b_{ij} are negative for reactants (or so called left-hand species) and positive for products (or right-hand species) of the j-th reaction. The stoichiometric coefficients can be considered as the components of the reaction matrix (or stoichiometric matrix) $\mathbf{B}$ of dimension $k \times p$. If the system is closed, any mole vector change is due to chemical reactions, i.e.,

$$\Delta n = \mathbf{B}\xi , \qquad (1.73)$$

where the p-vector ξ is formed by the extents of individual reactions.Its j-th component $[\xi]_j$ measures how many moles of "left-hand side" have been transformed to "right-hand side" in the j-th reaction.

The concept of a closed system can also be introduced without considering reactions. Chemical species are built from building blocks called atoms. Define the atom matrix $\mathbf{A}$, where $[\mathbf{A}]_{ij}$ is the number of the i-th atom in the molecule of the j-th species M_j . If the number of different atoms is denoted by a then the atom matrix is of dimension $a\times k$. The quantities of atoms in the system can be calculated by summing up their quantities in each species, i.e., forming the product $\mathbf{An}$. These quantities remain unchanged if the system is closed, so that

$$\mathbf{A}\Delta\mathbf{n} = \mathbf{0} . \tag{1.74}$$

For a given system both (1.73) and (1.74) hold, and hence

$$\mathbf{AB}\xi = \mathbf{0} . \tag{1.75}$$

Since in eqn. (1.75) the reaction extent vector ξ can take arbitrary values,

$$\mathbf{AB} = \mathbf{0} , \tag{1.76}$$

where $\mathbf{0}$ is a null matrix of dimension $a\times p$.

Equation (1.76) expresses the fundamental relation between the atom matrix and the reaction matrix of a closed system. The matrices $\mathbf{A}$ and $\mathbf{B}$, however, result in the same stoichiometric subspace if and only if the subspace defined by (1.73) and the one defined by (1.74) are of the same dimension, in addition to the relation (1.76). We denote the dimension of the stoichiometric subspace by f also called the stoichiometric number of freedom . If the reaction matrix $\mathbf{B}$ is known, then $f = \mathrm{rank}(\mathbf{B})$, i.e., f is the number of linearly independent reactions. If the atom matrix $\mathbf{A}$ is known, then the stoichiometric number of freedom defined by (1.74) can be obtained from $f = k - \mathrm{rank}(\mathbf{A})$, i.e., f is the number of "free" variables in the general solution of the matrix equation (1.74).

There are the following two basic problems in stoichiometry:

(i) Given an atom matrix $\mathbf{A}$ construct a (virtual) reaction matrix $\bar{\mathbf{B}}$ that defines the same stoichiometric subspace and has a minimum number $\bar{p}$ of columns.

(ii) Given a reaction matrix $\mathbf{B}$ construct a (virtual) atom matrix $\bar{\mathbf{A}}$ that defines the same stoichiometric subspace and has a minimum number $\bar{a}$ of rows.

The solution of problem (i) involves the transformation of the basis. Starting from the canonical basis we replace r unit vectors by r column vectors of the matrix A, where $r = \mathrm{rank}(A)$. For notational simplicity let us renumber the species such that the first r columns $a_1, a_2, \ldots, a_r$ are in the resulting basis. Then the table of coordinates takes the form:

	a_1	a_2	$\ldots$	a_r	a_{r+1}	$\ldots$	a_k
a_1	1	0		0			
a_2	0	1		0			
.						$Y_{r,k-r}$	
.							
.							
a_r	0	0		1			
e_{r+1}	0	0		0	0		0
.							
e_a	0	0		0	0		0

where $Y_{r,k-r}$ contains the coordinates of vectors $a_{r+1}, a_{r+2}, \ldots, a_k$ in the current basis. We select $\bar{p} = k - r$ reactions in which species $M_{r+1}, M_{r+2}, \ldots, M_k$ are decomposed, respectively, into species $M_1, M_2, \ldots, M_r$ and obtain the reaction matrix:

$$\bar{B} = \begin{bmatrix} Y_{r,\bar{p}} \\ -I_{\bar{p}} \end{bmatrix} .$$

Interchanging the rows of $\bar{B}$ you can easily restore the original order of species.

To illustrate the above procedure consider the species CH_4, CH_3D, CH_2D_2, CHD_3 and CD_4 (ref. 15). Here $a = 3$, $k = 5$, and fixing the order of atoms as C, H and D gives the atom matrix

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 4 & 3 & 2 & 1 & 0 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix} .$$

After two transformations we arrive at the table of coordinates:

	a_1	a_2	a_3	a_4	a_5
a_1	1	3/4	1/2	1/4	0
a_5	0	1/4	1/2	3/4	1
e_3	0	0	0	0	0

From the table $r = 2$ and $f = 5 - 2 = 3$. The (virtual) reaction matrix $\bar{\mathbf{B}}$ with $k = 5$ rows and $\bar{p} = f = 3$ columns is given by

$$\bar{B} = \begin{bmatrix} 3/4 & 1/2 & 1/4 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1/4 & 1/2 & 3/4 \end{bmatrix} .$$

In more familiar chemical terms the following reactions have been constructed:

$$CH_3D = (3/4)CH_4 + (1/4)CD_4$$
$$CH_2D_2 = (1/2)CH_4 + (1/2)CD_4$$
$$CHD_3 = (1/4)CH_4 + (3/4)CD_4$$

Now we turn to problem (ii). Taking the transpose of eqn. (1.76) we obtain

$$\mathbf{B}^T\mathbf{A}^T = \mathbf{0} , \qquad (1.78)$$

where the null matrix $\mathbf{0}$ is of dimension $p \times a$. It is then follows from (1.78) that starting with $\mathbf{B}^T$ and repeating all the steps needed in problem (i) we arrive at $\bar{\mathbf{A}}^T$. The number of rows in $\bar{\mathbf{A}}$ will be $\bar{a} = k - \text{rank}(\mathbf{B})$.

To see how the method works, suppose six species M_1, M_2, ..., M_6 are known to take part in the reactions (ref. 15)

$$M_1 + 2M_2 = M_3 + 2M_4$$
$$M_2 + M_5 = M_6$$
$$M_1 + M_2 + M_6 = M_3 + 2M_4 + M_5 .$$

The transpose of the reaction matrix is then

$$\mathbf{B}^T = \begin{bmatrix} -1 & -2 & 1 & 2 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 1 \\ -1 & -1 & 1 & 2 & 1 & -1 \end{bmatrix} ,$$

and after two transformations we arrive at the table of coordinates:

	b^1	b^2	b^3	b^4	b^5	b^6
b^3	-1	-2	1	2	0	0
b^6	0	-1	0	0	-1	1
e^3	0	0	0	0	0	0

From the table rank(B) = 2 , $\bar{a} = 6 - 2 = 4$, and the virtual atom matrix is

$$\bar{A} = \begin{bmatrix} -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & -2 & 0 & 0 & -1 \\ 0 & 0 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} .$$

The matrix $\bar{A}$ imposes constraints on the mole vector change. In terms of mole numbers it means that

$$\begin{aligned} n_1 + n_3 &= \mathrm{const}_1 \\ n_2 + 2n_3 + n_6 &= \mathrm{const}_2 \\ 2n_3 - n_4 &= \mathrm{const}_3 \\ n_5 + n_6 &= \mathrm{const}_4 . \end{aligned}$$

These quantities are preserved like atoms in the given reactions and hence are called reaction invariants (ref. 16). In this example we found 4 linearly independent reaction invariants. It does not mean, however, that the species M_1, M_2, ..., M_6 are built necessarily from 4 atoms. In fact, introducing the species $M_1 = CH_4$, $M_2 = O_2$, $M_3 = CO_2$, $M_4 = H_2O$, $M_5 = H_2$, and $M_6 = H_2O_2$ the considered reactions are possible, although the number of atoms is only 3 . Based on the true atom matrix the number of stoichiometric freedom is $f = 6 - 3 = 3$, but the actual reactions do not span the possible stoichiometric subspace, and that is why a fourth reaction invariant appears.

1.8.2 Fitting a line by the method of least absolute deviations

We will discuss many times the problem of adjusting the parameters a and b of the linear function $y = ax + b$ in order to "fit" the line to the set $\{(x_i,y_i),\ i = 1,2,\ldots,m\}$ of observations. In this section the "best fit" will mean the least sum of the absolute deviations between observed and computed values of y , i.e., the minimum of the objective function

$$Q(a,b) = \sum_{i=1}^{m} |y_i - ax_i - b| . \tag{1.79}$$

This problem can be translated into one of linear programming. Introducing the variables $s_i \geq 0$ we first construct an equivalent constrained minimization problem given by

$$|y_i - ax_i - b| \le s_i, \quad i = 1,2,\ldots,m; \quad \sum_{i=1}^{m} s_i \dashrightarrow \min. \tag{1.80}$$

Each constraint in (1.80) can be splitted as

$$\begin{aligned} y_i - ax_i - b &\le s_i \\ y_i - ax_i - b &\ge -s_i \end{aligned} \tag{1.81}$$

Thus both the constraints and the objective function are linear.

The only remaining problem is that a and b are not necessarily nonnegative, as required in linear programming. Inventing further new variables a_1, a_2, b_1, $b_2 \ge 0$ and setting $a = a_1 - a_2$ and $b = b_1 - b_2$ will eliminate this difficulty, and we can finally formulate the linear programming problem as

$$\sum_{i=1}^{m} s_i \dashrightarrow \min ,$$

subject to

$$\left.\begin{aligned} b_1 - b_2 + x_i a_1 - x_i a_2 + s_i &\ge y_i \\ b_1 - b_2 + x_i a_1 - x_i a_2 - s_i &\le y_i \end{aligned}\right\} \quad i = 1,2,\ldots,m$$

$$a_1,\ a_{21},\ b_1,\ b_2,\ s_1,\ s_2,\ \ldots,\ s_m \ge 0 . \tag{1.82}$$

We apply the method to the data of Table 1.1, which gives the content of tar (x) and nicotine (y) in different sorts of cigarettes (ref. 17).

Table 1.1
Tar and nicotine content of cigarettes

No. of observation	1	2	3	4	5	6	7	8	9	10
Tar, mg	8.3	12.3	18.8	22.9	23.1	24.0	27.3	30.0	35.9	41.6
Nicotine, mg	0.32	0.46	1.10	1.32	1.26	1.44	1.42	1.96	2.23	2.20

As in the constraints (1.82), the variables will be listed in the order b_1, b_2, a_1, a_2, s_1, s_2, ..., s_{10}. To use the main program of Example 1.2, its DATA statements will be replaced by the following lines:

```
100 REM ----------------------------------------------------
102 REM EX. 1.8.2. FITTING A LINE  -  LEAST ABSOLUTE DEVIATIONS
104 REM MERGE M10,M11
106 REM ---------- DATA
108 REM (NUMBER OF VARIABLES, NUMBER OF CONSTRAINTS)
110 DATA 14,20
112 DATA  1,  -1,   8.3, - 8.3,  1, 0, 0, 0, 0, 0, 0, 0, 0, 0, GE,  0.32
114 DATA  1,  -1,   8.3, - 8.3, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, LE,  0.32
116 DATA  1,  -1,  12.3, -12.3,  0, 1, 0, 0, 0, 0, 0, 0, 0, 0, GE,  0.46
118 DATA  1,  -1,  12.3, -12.3,  0,-1, 0, 0, 0, 0, 0, 0, 0, 0, LE,  0.46
120 DATA  1,  -1,  18.8, -18.8,  0, 0, 1, 0, 0, 0, 0, 0, 0, 0, GE,  1.10
122 DATA  1,  -1,  18.8, -18.8,  0, 0,-1, 0, 0, 0, 0, 0, 0, 0, LE,  1.10
124 DATA  1,  -1,  22.9, -22.9,  0, 0, 0, 1, 0, 0, 0, 0, 0, 0, GE,  1.32
126 DATA  1,  -1,  22.9, -22.9,  0, 0, 0,-1, 0, 0, 0, 0, 0, 0, LE,  1.32
128 DATA  1,  -1,  23.1, -23.1,  0, 0, 0, 0, 1, 0, 0, 0, 0, 0, GE,  1.26
130 DATA  1,  -1,  23.1, -23.1,  0, 0, 0, 0,-1, 0, 0, 0, 0, 0, LE,  1.26
132 DATA  1,  -1,  24.0, -24.0,  0, 0, 0, 0, 0, 1, 0, 0, 0, 0, GE,  1.44
134 DATA  1,  -1,  24.0, -24.0,  0, 0, 0, 0, 0,-1, 0, 0, 0, 0, LE,  1.44
136 DATA  1,  -1,  27.3, -27.3,  0, 0, 0, 0, 0, 0, 1, 0, 0, 0, GE,  1.42
138 DATA  1,  -1,  27.3, -27.3,  0, 0, 0, 0, 0, 0,-1, 0, 0, 0, LE,  1.42
140 DATA  1,  -1,  30.0, -30.0,  0, 0, 0, 0, 0, 0, 0, 1, 0, 0, GE,  1.96
142 DATA  1,  -1,  30.0, -30.0,  0, 0, 0, 0, 0, 0, 0,-1, 0, 0, LE,  1.96
144 DATA  1,  -1,  35.9, -35.9,  0, 0, 0, 0, 0, 0, 0, 0, 1, 0, GE,  2.23
146 DATA  1,  -1,  35.9, -35.9,  0, 0, 0, 0, 0, 0, 0, 0,-1, 0, LE,  2.23
148 DATA  1,  -1,  41.6, -41.6,  0, 0, 0, 0, 0, 0, 0, 0, 0, 1, GE,  2.20
150 DATA  1,  -1,  41.6, -41.6,  0, 0, 0, 0, 0, 0, 0, 0, 0,-1, LE,  2.20
152 REM ( OBJECTIVE FUNCTION )
154 DATA  0,   0,    0,    0,     1, 1, 1, 1, 1, 1, 1, 1, 1, 1, MIN
156 REM ---------- FROM HERE THE SAME AS THE PROGRAM OF EX. 1.2.
```

The sample output of the program is:

```
          OPTIMUM SOLUTION

--------------------------------------------------
 j           Xj            Cj            Cj*Xj
--------------------------------------------------
 1           0             0             0
 2           .2484932      0             0
 3           6.849316E-02  0             0
 4           0             0             0
 5           0             1             0
 6           .1339727      1             .1339727
 7           6.082198E-02  1             6.082198E-02
 8           0             1             0
 9           7.369876E-02  1             7.369876E-02
 10          4.465751E-02  1             4.465751E-02
 11          .2013699      1             .2013699
 12          .1536985      1             .1536985
 13          1.958874E-02  1             1.958874E-02
 14          .4008219      1             .4008219
--------------------------------------------------

OBJECTIVE FUNCTION MINIMUM VALUE .......... 1.08863
```

Thus the estimates of the parameters are a = 0.06849 and b = -0.2485 .

We are almost sure that you solved similar problems by the method of least squares (also known as linear regression) and you know that it is computationally simpler than the procedure suggested here. The method of least absolute deviations is, however, more robust, i.e., it is less sensitive to the errors in observations (ref. 18). We will give you more details in Section 3.10.1.

1.8.3 Fitting a line by the minimax method

Now we solve the previous problem by minimizing the objective function

$$Q(a,b) = \max_{1\le i\le m} \left| y_i - ax_i - b \right| . \tag{1.83}$$

This procedure is also known as uniform or Chebyshev approximation. We have the introduce the single auxiliary variable $s \ge 0$ to translate the minimization of (1.83) into the problem

$$\left| y_i - ax_i - b \right| \le s, \; i = 1,2, \ldots, m \; ; \quad s \dashrightarrow \min . \tag{1.84}$$

Proceeding as in the previous section we obtain the linear programming problem

$s \dashrightarrow \min$,

subject to

$$\left.\begin{array}{l} b_1 - b_2 + x_i a_1 - x_i a_2 + s_i \ge y_i \\ b_1 - b_2 + x_i a_1 - x_i a_2 - s_i \le y_i \end{array}\right\} \; i = 1,2,\ldots,m$$

$$a_1,\, a_2,\, b_1,\, b_2,\, s \ge 0 . \tag{1.85}$$

The main program is now used with the DATA statements

```
100 REM -------------------------------------------------------
102 REM EX. 1.8.3. FITTING A LINE  -  MINIMAX METHOD
104 REM MERGE M10,M11
106 REM ---------- DATA
108 REM (NUMBER OF VARIABLES, NUMBER OF CONSTRAINTS)
110 DATA  5,20
112 DATA  1,  -1,   8.3, - 8.3,  1,  GE,  0.32
114 DATA  1,  -1,   8.3, - 8.3, -1,  LE,  0.32
116 DATA  1,  -1,  12.3, -12.3,  1,  GE,  0.46
118 DATA  1,  -1,  12.3, -12.3, -1,  LE,  0.46
120 DATA  1,  -1,  18.8, -18.8,  1,  GE,  1.10
122 DATA  1,  -1,  18.8, -18.8, -1,  LE,  1.10
124 DATA  1,  -1,  22.9, -22.9,  1,  GE,  1.32
126 DATA  1,  -1,  22.9, -22.9, -1,  LE,  1.32
128 DATA  1,  -1,  23.1, -23.1,  1,  GE,  1.26
130 DATA  1,  -1,  23.1, -23.1, -1,  LE,  1.26
132 DATA  1,  -1,  24.0, -24.0,  1,  GE,  1.44
134 DATA  1,  -1,  24.0, -24.0, -1,  LE,  1.44
136 DATA  1,  -1,  27.3, -27.3,  1,  GE,  1.42
138 DATA  1,  -1,  27.3, -27.3, -1,  LE,  1.42
```

```
140 DATA  1,  -1,  30.0, -30.0,  1,  GE,  1.96
142 DATA  1,  -1,  30.0, -30.0, -1,  LE,  1.96
144 DATA  1,  -1,  35.9, -35.9,  1,  GE,  2.23
146 DATA  1,  -1,  35.9, -35.9, -1,  LE,  2.23
148 DATA  1,  -1,  41.6, -41.6,  1,  GE,  2.20
150 DATA  1,  -1,  41.6, -41.6, -1,  LE,  2.20
152 REM ( OBJECTIVE FUNCTION )
154 DATA  0,   0,   0,    0,     1,  MIN
156 REM ---------- FROM HERE THE SAME AS THE PROGRAM OF EX. 1.2.
```

and gives the output

```
EVALUATION OF CONSTRAINTS

--------------------------------------------------------------
 I  TYPE  L.H.S.        R.H.S     SLACK           SHADOW PRICE
--------------------------------------------------------------
 1   GE   .6713311      .32       .3513311
 2   LE   .2224573      .32       9.754272E-02
 3   GE   .9088738      .46       .4488738
 4   LE   .46           .46       0               .1979525
 5   GE   1.294881      1.1       .1948805
 6   LE   .8460068      1.1       .2539933
 7   GE   1.538362      1.32      .2183617
 8   LE   1.089488      1.32      .230512
 9   GE   1.550239      1.26      .290239
10   LE   1.101365      1.26      .1586348
11   GE   1.603686      1.44      .163686
12   LE   1.154812      1.44      .2851877
13   GE   1.799659      1.42      .3796588
14   LE   1.350785      1.42      6.921494E-02
15   GE   1.96          1.96      0               .5
16   LE   1.511126      1.96      .4488737
17   GE   2.310376      2.23      8.037567E-02
18   LE   1.861502      2.23      .3684982
19   GE   2.648874      2.2       .4488738
20   LE   2.2           2.2       0               .3020477
--------------------------------------------------------------


           OPTIMUM SOLUTION

------------------------------------------------------
 j            Xj              Cj              Cj*Xj
------------------------------------------------------
 1            0               0               0
 2            4.600691E-02    0               0
 3            5.938567E-02    0               0
 4            0               0               0
 5            .2244369        1               .2244369
------------------------------------------------------

OBJECTIVE FUNCTION MINIMUM VALUE .......... .2244369
```

Thus a = 0.05939 and b = -0.4601 . In this case the shadow prices are also of interest and show that point 8 seems to be an outlier.

Notice that the methods presented in Sections 1.8.2 and 1.8.3 can be extended to estimate the parameters in multivariable functions that are linear in the parameters.

1.8.4 Analysis of spectroscopic data for mixtures with background absorption

Spectroscopy in the visible region is a classical method of determining the composition of species in solution if they have sufficiently different light-absorbing properties. The method is based on measuring light absorption at different wavelengths. If a_{ij} denote the molar absorption of the j-th component at the i-th wavelength, then the total light absorption is well described by the weighted sum $A_i = \sum_{i=1}^{n} a_{ij}x_j$, where n is the number of absorbing species in the solution, and x_j is the concentration of the j-th component. If the a_{ij}'s are known, observations A_1, A_2, ..., A_n at n appropriately selected wavelengths will enable us to find $\mathbf{x} = (x_1,x_2,\ldots,x_n)^T$ by solving a matrix equation. Since the A_i's are corrupted by measurement errors, it is better to have $m > n$ observations, and estimate $\mathbf{x}$ by the least squares method, i.e., minimizing the objective function

$$Q(x_1,x_2,\ldots,x_n) = \sum_{i=1}^{m} (A_i - \sum_{j=1}^{n} a_{ij}x_j)^2 . \qquad (1.86)$$

We run, however, into difficulty if there is an (n+1)-th, unidentified component in the mixture with unknown molar absorption cefficients. Then

$$\sum_{j=1}^{n} a_{ij}x_j \le A_i , \qquad (1.87)$$

and the best we can do is to minimize some error norm, e.g., (1.86) under constraints (1.87). Because of the absorption of the unknown component, the minimum of (1.86) is expected to be large, with large residual deviations between the observed and measured absorbances. As we will discuss in Section 3.10.1, in such situations we obtain better estimates of $\mathbf{x}$ by minimizing the sum of absolute deviations

$$Q(x_1,x_2,\ldots x_n) = \sum_{i=1}^{m} \left| A_i - \sum_{j=1}^{n} a_{ij}x_j \right| . \qquad (1.88)$$

In addition, this extremum is easier to find. Indeed, by (1.87) each deviation in (1.88) is nonnegative, and (1.88) can be replaced by the sum of deviations without taking absolute values. Furthermore, the sum of fixed A_i's does not change the value of x minimizing (1.88), and hence we obtain a linear programming problem with constraints (1.87) and $x_j \geq 0$ $(j = 1,2,\ldots,n)$, and with the objective function

$$\sum_{i=1}^{n} \left[\sum_{j=1}^{m} a_{ij}\right] x_j \longrightarrow \max . \qquad (1.89)$$

Our considerations are valid only for error-free observations since with errors in A_i the inequalities (1.87) are not necessarily true. It is far from easy to extend this method to the real situation. In (ref. 19) the authors increased each observed A_i values by the half-length of the confidence intervals (for definition see Chapter 3), i.e., replaced (1.87) by inequalities

$$\sum_{j=1}^{n} a_{ij} x \leq A_i + ts_i , \qquad (1.90)$$

where s_i is an estimate of the standard deviation of A_i, t is the value of the Student's t (say, at 0.05 probability level) with $r - 1$ degrees of freedom and r denotes the number of A_i's used to determine the standard error s_i. If there are no repeated observations, s_i can be the estimated precision of the spectroscopic measurement, but then there is some arbitrariness in selecting a reasonable value for t.

We are going to reproduce the example studied in (ref. 19), where the term ts_i has been replaced by a given percentage of A_i . The mixture consisted of α,γ,δ and ϵ isomers of hexachlorine-cyclohexane, for testing the method in known quantities. The absorption of the mixture was measured at 20 wavelengths, and the ϵ isomer was regarded as the unknown component, responsible for the background absorption. Therefore, only the specific absorbances of the α,γ and δ isomers were assumed to be known.

We use the main program of Example 1.2 to solve the linear programming problem. At $s_i=0$ the constraints have the form (1.87). Thus the coefficients in each DATA statement are the molar absorption coefficients at the corresponding wavelength, whereas the right-hand side is the observed absorbance A_i . The objective function coefficients are the sums in (1.89). You can easily reconstruct the data of (ref. 19) from the following DATA statements.

```
100 REM --------------------------------------------------------
102 REM EX. 1.8.4. ANALYSIS OF SPECTROSCOPIC DATA WITH BACKGROUND
104 REM MERGE M10,M11
106 REM -------- DATA
108 REM (NUMBER OF VARIABLES, NUMBER OF CONSTRAINTS)
110 DATA 3,20
112 REM (  ALFA       GAMMA     DELTA        OBSERVED)
114 DATA      0   ,   0     , .137484  ,LE,    0.755
116 DATA      0   , .182928,   0       ,LE,    0.913
118 DATA  .308334 ,   0     , .012791  ,LE,    1.106
120 DATA      0   ,   0     , .111669  ,LE,    2.938
122 DATA      0   ,   0     , .436536  ,LE,    2.49
124 DATA  .325941 ,   0     ,   0      ,LE,    1.219
126 DATA      0   , .120804,   0       ,LE,    0.63
128 DATA  .035649 , .424193, .041717  ,LE,    2.534
130 DATA  .062569 , .27824 , .072141  ,LE,    1.87
132 DATA  .325941 , .090488, .020052  ,LE,    1.673
134 DATA  .091088 , .238967, .00991   ,LE,    1.547
136 DATA  .109423 , .03376 , .00749   ,LE,    0.969
138 DATA  .049182 , .022966, .00749   ,LE,    1.367
140 DATA  .009458 , .004937, .195104  ,LE,    1.034
142 DATA  .1235   , .009875, .075598  ,LE,    0.959
144 DATA  .006256 , .052478, .299167  ,LE,    1.87
146 DATA  .209388 , .111273, .017632  ,LE,    1.738
148 DATA  .157004 , .038698, .015211  ,LE,    2.882
150 DATA  .022262 , .207503, .00749   ,LE,    1.136
152 DATA  .179558 , .022966, .052665  ,LE,    0.969
154 REM  (OBJECTIVE FUNCTION COEFFICIENTS)
156 DATA 2.01555  ,1.84008 ,1.52015   ,MAX
158 REM ---------- FROM HERE THE SAME AS THE PROGRAM OF EX. 1.2.
```

It is easy to modify the program to solve the problem assuming 5% and 10% errors in observations. Results are summarized in Table 1.2. The table also includes the unconstrained least squares estimates of x, i.e., the values minimizing (1.86) with $n = 3$ and $m = 20$. This latter result was obtained by inserting the appropriate data into the main program of Section 3.2 .

Table 1.2
Results for the spectroscopic problem with background

Isomer	True concentration %	Estimated concentration, %			
		linear programming			least squares
		$ts_i = 0\%$	$ts_i = 5\%$	$ts_i = 10\%$	
α	3.85	3.33	3.49	3.66	4.51
γ	4.88	4.67	4.90	5.14	4.89
δ	4.86	5.02	5.27	5.52	6.21

The procedure described here clearly gives somewhat better results at each assumed magnitude of errors than the least squares approach.

1.8.5 Canonical form of a quadratic response function

The conversion y in a chemical reaction was described by the empirical relationship (ref. 13)

$$y = 67.711 + 1.944x_1 + 0.906x_2 + 1.069x_3 - 1.539x^2_1 - 0.264x^2_2 - \\ - 0.676x^2_3 - 3.088x_1x_2 - 2.188x_1x_3 - 1.212x_2x_3 \qquad (1.91)$$

as a function of the temperature x_1, the feed concentration x_2 and the reaction time x_3. We want to know whether or not (1.91) has a maximum. More generally, we are interested in the geometric characterization of the quadratic function

$$y = a + \mathbf{b}^T\mathbf{x} + \mathbf{x}^T\mathbf{A}\mathbf{x} \qquad (1.92)$$

of n variables $\mathbf{x} = (x_1,x_2,\ldots,x_n)^T$, where $\mathbf{b}$ is an n-vector and $\mathbf{A}$ is an $n\times n$ symmetric matrix. From (1.91) in this example we have

$$\mathbf{b} = \begin{bmatrix} 1.944 \\ 0.906 \\ 1.069 \end{bmatrix} , \quad \mathbf{A} = \begin{bmatrix} -1.539 & -1.544 & -1.094 \\ -1.544 & -0.264 & -0.606 \\ -1.094 & -0.606 & -0.676 \end{bmatrix} . \qquad (1.93)$$

Any extremum point of (1.91) satisfies the equation

$$\frac{\partial y}{\partial \mathbf{x}} = \mathbf{b} + 2\mathbf{A}\mathbf{x} = \mathbf{0} . \qquad (1.94)$$

As we will see later, $\mathbf{A}$ is nonsingular, hence the only solution is $\mathbf{x}^o = -(1/2)\mathbf{A}^{-1}\mathbf{b}$. It may be a maximum, a minimum or a saddle point. We can slightly simplify the problem by setting $\mathbf{z} = \mathbf{x} - \mathbf{x}^o$, wich translates (1.92) into

$$y - y^o = \mathbf{z}^T\mathbf{A}\mathbf{z} , \qquad (1.95)$$

where y^o is the value of (1.91) at $\mathbf{x}^o$. This point $\mathbf{x}^o$ is the (global) maximum point of (1.91) if and only if $\mathbf{z}^T\mathbf{A}\mathbf{z} < 0$ for all nonzero $\mathbf{z}$, i.e., the matrix $\mathbf{A}$ is negative definite. We can easily check this property by diagonalizing $\mathbf{A}$. Let $\mathbf{U}$ denote the matrix formed by the normalized eigenvectors of $\mathbf{A}$. By (1.66), introducing the new variables $\mathbf{w} = \mathbf{U}^T\mathbf{z}$, (1.95) is reduced to

$$y - y^o = \sum_{i=1}^{n} \lambda_i w^2_i , \qquad (1.96)$$

where λ_i is the i-th eigenvalue of $\mathbf{A}$ and w_i is the i-th element of the vector $\mathbf{w}$, that is $w_i = \mathbf{u}^T{}_i\mathbf{z}$. Expression (1.96) gives the quadratic function (1.91) in its canonical form. This function has maximum at $\mathbf{x}^o$ if and only if $\lambda_i < 0$ for $i = 1,2,\dots,n$. Therefore, we perform the eigenvalue-eigenvector decomposition of matrix $\mathbf{A}$ by changing the DATA statements in the program presented for Example 1.6. The following results are obtained:

```
EIGENVALUES:
--------------------------------------
 0.77996E+00  -.68638E-01  -.31903E+01
--------------------------------------

EIGENVECTORS:
--------------------------------------
      u1           u2           u3
 -0.584927    -0.306329     0.751015
  0.804293    -0.338644     0.488295
  0.104748     0.889653     0.444460
--------------------------------------
```

What can we see from these results? The point $\mathbf{x}^o$ is not a maximum, since the first eigenvalue is positive. Selecting the canonical variables $w_1 \neq 0$, $w_2 = w_3 = 0$ we can increase the value of y . By orthogonality of the eigenvectors any step $\mathbf{x} - \mathbf{x}^o$ parallel to the first eigenvector $\mathbf{u}_1$ results in $w_1 \neq 0$ and $w_2 = w_3 = 0$.

To find the point $\mathbf{x}^o$ one can use LU decomposition and a backsubstitution. An other possibility is to apply the results of the eigenvalue-eigenvector decomposition directly. By eqn. (1.66)

$$\mathbf{A}^{-1} = \mathbf{U}\mathbf{D}^{-1}\mathbf{U}^T , \tag{1.97}$$

and hence (1.94) takes the form

$$\mathbf{x}^o = -(1/2)\mathbf{U}\mathbf{D}^{-1}\mathbf{U}^T\mathbf{b} . \tag{1.98}$$

Evaluating this expression is quite easy if taking into account that $\mathbf{D}^{-1}$ is a diagonal matrix with reciprocal values of the eigenvalues in its main diagonal. We leave to you to compute $\mathbf{x}^o$ and to show that the computed conversion is higher at the point $\mathbf{x} = \mathbf{x}^o + \mathbf{u}_1$ than at the point $\mathbf{x}^o$.

1.8.6 Euclidean norm and condition number of a square matrix

In Section 1.7 we emphasized the importance of the condition number cond($\mathbf{A}$), but did not tell you how to find it. Now we try to close this gap, first considering the norm $\|\mathbf{A}\|$ of a matrix. According to (1.67) to find $\|\mathbf{A}\|$ we have to maximize the function $\|\mathbf{A}\mathbf{x}\|^2 = \mathbf{x}^T(\mathbf{A}^T\mathbf{A})\mathbf{x}$ subject to the constraint $\|\mathbf{x}\| = 1$. This problem is easy to solve by writing $\mathbf{A}\mathbf{x}$ in the basis of the eigenvectors **U** of A^TA, thereby introducing the new variables $w = U^Tx$. Since the columns of **U** form an orthonormal system, $\|w\| = \|U^Tx\| = \|x\|$, and by (1.66)

$$\|Ax\|^2 = w^TDw \ , \tag{1.99}$$

where **D** is diagonal matrix with the eigenvalues of A^TA in its diagonal. The function (1.99) clearly will attain its maximum value if $w = u_1$, the eigenvector corresponding to the largest eigenvalue $\lambda_{max} = \lambda_1$ of A^TA, and hence

$$\|A\| = (\lambda_{max})^{1/2} \ . \tag{1.100}$$

Since A^TA is symmetric and positive semidefinite, λ_{max} is real and nonnegative. If the matrix is nonsingular (and hence positive definite) then $\lambda_{min} \neq 0$ and by (1.97)

$$\|A^{-1}\| = (\lambda_{min})^{-1/2} \ , \tag{1.101}$$

where $\lambda_{min} = \lambda_n$ is the smallest eigenvalue of A^TA. Therefore, by its definition

$$\mathrm{cond}(A) = (\lambda_{max}/\lambda_{min})^{1/2} \ . \tag{1.102}$$

We note that the values $\lambda^{1/2}{}_i$ are called the singular values of the matrix **A** and they can be determined directly from **A** , without forming A^TA . The corresponding numerical method called singular value decomposition is relatively complex but somewhat more accurate then the procedure described here, for details see (ref. 11).

For exercise find $\mathrm{cond}(H_6)$ of the Hilbert matrix H_6 defined by (1.69). Give a crude estimate of the relative errors of the columns of $H^{-1}{}_6$, if the floating-point numbers are stored to 7 digits.

1.8.7 Linear dependences in data

Observing a process, scientists and engineers frequently record several variables. For example, (ref. 20) presents concentrations of all species for the thermal isomerization of α-pinene at different time points. These species are α-pinene (y_1), dipentene (y_2), allo-ocimene (y_3), pyronene (y_4) and a dimer product (y_5). The data are reproduced in Table 1.3. In (ref. 20) a reaction scheme has also been proposed to describe the kinetics of the process. Several years later Box at al. (ref. 21) tried to estimate the rate coefficients of this kinetic model by their multiresponse estimation procedure that will be discussed in Section 3.6. They run into difficulty and realized that the data in Table 1.3 are not independent. There are two kinds of dependencies that may trouble parameter estimation:

Table 1.3
Concentrations in the thermal isomerization of α-pinene

Observation	Time, min	Concentration, mol %				
		y_1	y_2	y_3	y_4	y_5
1	1230	88.35	7.3	2.3	0.4	1.75
2	3050	76.4	15.6	4.5	0.7	2.8
3	4920	65.1	23.1	5.3	1.1	5.8
4	7800	50.4	32.9	6.0	1.5	9.3
5	10680	37.5	42.7	6.0	1.9	12.0
6	15030	25.9	49.1	5.9	2.2	17.0
7	22620	14.0	57.4	5.1	2.6	21.0
8	36420	4.5	63.1	3.8	2.9	25.7

(i) If one of the variables is difficult to measure, the experimenter may calculate its values from some known relationship, e.g., a balance equation . Let $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]$ denote the m×n observation matrix, where m is the number of observations , n is the number of variables and $\mathbf{y}_j$ is the j-th column of the observation matrix. The dependence is of the form

$$\sum_{j=1}^{n} v_j y_{ij} = \text{const} \tag{1.103}$$

for all $i = 1,2,\ldots,m$, where the v_j's are constant coefficients. The affine linear relationship (1.103) can be transformed into a linear one by centering the data, i.e., considering the deviations $x_{ij} = y_{ij} - \bar{y}_j$, where $\bar{y}_j = \left[\sum_{i=1}^{m} y_{ij}\right]/m$ is the average of the elements in the j-th column of $\mathbf{Y}$. Then the columns of the centered data matrix $\mathbf{X}$, defined by $[\mathbf{X}]_{ij} = x_{ij}$, are linearly dependent, and hence there exists an n-vector $\mathbf{u} \neq \mathbf{0}$ such that

$$\mathbf{X}\mathbf{u} = \mathbf{0} \ . \tag{1.104}$$

Multiplying (1.104) by $\mathbf{X}^T$ we have $(\mathbf{X}^T\mathbf{X})\mathbf{u} = \mathbf{0}$, and thus there exists an affine linear dependence of the form (1.103) among the columns of $\mathbf{Y}$ if and only if the matrix $\mathbf{X}^T\mathbf{X}$ has a $\lambda = 0$ eigenvalue. It is obvious that λ_{min} will equal not zero, but some small number because of the roundoff errors.

(ii) The second kind of dependence is somewhat weaker. In chemical systems the variables are required to satisfy a number of balance equations, e.g., stoichiometric relations. Therefore, certain affine linear relationships may exist among the expected values of the responses. In such cases the least eigenvalue λ_{min} of $\mathbf{X}^T\mathbf{X}$ will be larger then in the previous case, stemming from a linear dependence directly among the (centered) data, but still small.

We need some threshold values of λ_{min} in order to classify the corresponding linear dependence as (i) or (ii). According to Box at al. (ref. 21), in case (ii), i.e., a linear dependence in the expected values of the responses $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$, the expected value of the eigenvalue λ_{min} can be estimated by

$$E((\lambda_{min})^{(ii)}) = (m-1)\mathbf{u}^T\mathbf{C}_x\mathbf{u} \ , \qquad (1.105)$$

where $\mathbf{u}$ is the corresponding eigenvector of $\mathbf{X}^T\mathbf{X}$, and $\mathbf{C}_x$ is the $n \times n$ covariance matrix of measurement errors in the observations of $y_1, y_2, \ldots, y_n$. In practice it is usually difficult to find a reliable estimate of $\mathbf{C}_x$ (we will discuss this problem in Chapter 3), and we can get a crude estimate of

$E[(\lambda_{min})^{(ii)}]$ approximating $\mathbf{C}_x$ by $\mathbf{I}\bar{\sigma}^2{}_r$, where the average variance $\bar{\sigma}^2{}_r$ is estimated from the residual sum of squares following a least squares estimation procedure. These concepts will also be discussed in Chapter 3, and here we simply state that under the above approximation

$E[(\lambda_{min})^{(ii)}] \approx (m - 1)\bar{\sigma}^2{}_r$. To obtain a similar upper bound on $\lambda_{min}{}^{(i)}$ in the case (i), when there are only roundoff errors present, Box at al. (ref. 21) suggested to assume that the rounding error is distributed uniformly with range -0.5 to +0.5 of the last digit reported in the data. The rounding error variance $\bar{\sigma}^2{}_{re}$ is then given by the range squared divided by 12, and

$E((\lambda_{min})^{(i)}) \approx (m-1)\bar{\sigma}^2{}_{re}$. In Table 1.3 the concentration data are reported to the nearest 0.1 percent and therefore the range of the last reported digit is from -0.05 to +0.05 or 0.1. Thus, for class (i) of the dependences we have $E[(\lambda_{min})^{(i)}] \approx 7*(0.1)^2/12 \approx 0.006$. As we will show in Section 3.6, the average variance $\bar{\sigma}^2 \approx 0.6$ and hence the threshold for class (ii) is given by $E[(\lambda_{min})^{(i)}] \approx (m-1)\bar{\sigma}^2 \approx 4.2$.

We used the program given in Example 1.6 to calculate the eigenvalues and eigenvectors of $\mathbf{X}^T\mathbf{X}$, where $\mathbf{X}$ is the centered observation matrix from the data of Table 1.3. The program output is as follows.

```
EIGENVALUES:
--------------------------------------------------------------
 0.96629E+04  0.25830E+02  0.12194E+01  0.16679E-01  0.12790E-02
--------------------------------------------------------------

EIGENVECTORS:
--------------------------------------------------------------
     u1           u2           u3           u4           u5
  0.908725     0.056799    -0.295717     0.475478    -0.170909
 -0.540384    -0.223608    -0.610784     0.489190    -0.213592
 -0.012679    -0.612247     0.640183     0.434226    -0.163114
 -0.024106     0.003750    -0.009978     0.368382     0.929301
 -0.230667     0.756249     0.359945     0.458648    -0.186982
--------------------------------------------------------------
```

Since λ_4, λ_5 << 4.2, and both are close to the threshold $(\lambda_{min})^{(i)} = 0.006$, we expect to find two exact linear dependences in the data. From an exemination ofthe original paper (ref. 20) Box at al. (ref. 21) found that y_4 had been not measured because of experimental difficulties, but rather had been assumed to constitute 3% of the total conversion of α-pinene (y_4). That is, it was assumed that $y_4 = 0.03(100-y_1)$, which gives the exact affine linear relationship

$$(0.03)y_1 + (0)y_2 + (0)y_3 + (1)y_4 + (0)y_5 = 3 \qquad (1.106)$$

among the observations. The second such dependence, associated with λ_4, stems from the normalization of the data, forced to satisfy the balance equation

$$y_1 + y_2 + y_3 + y_4 + y_5 = 1 \ . \qquad (1.107)$$

The eigenvalue λ_3 is less than 4.2, but much larger than 0.006. Thus there is a further linear dependence, now among the expectations of the y's. This stems from the assumed reaction scheme, given later in Section 3.6, and is discussed there.

The form of the linear dependences (1.106) and (1.107) can be discovered by looking at the eigenvectors that correspond to the small eigenvalues λ_5 and λ_4, respectively. The only large element in u_5 corresponds to the variable y_4, and hence u_5 certainly stems from (1.106). According to (1.107) the eigenvalue u_4 is expected to have the form $u_4 = (v,v,v,v,v)^T$ with identical elements v. Since $\|u_4\|=1$, $v = \sqrt{5}/5 = 0.447$. The eigenvectors are, however, forced to be orthogonal. The projection of the theoretical eigenvector $(0.447, 0.447, 0.447, 0.447, 0.447)^T$ into the subspace orthogonal to u_5 gives the vector $(0.465, 0.468, 0.464, 0.363, 0.466)^T$, which is really close to the empirical eigenvector u_4. We will use similar mechanistic interpretations of eigenvectors in Section 3.5.2.

In this example we used some concepts that will be rigorously defined only in latter chapters. It is, however, difficult to avoid such flaws in structure when presenting applications of essentially algebraic methods, since the problems themselves usually come from other application areas.

1.8.8 Principal component and factor analysis

We generalize the discussion of the previous section by considering the $m \times n$ raw data matrix $\mathbf{Y}$, obtained by measuring the variables $y_1, y_2, \ldots, y_n$ at m sample points. Depending on the physical meaning of the data we apply some kind of preprocessing of the raw data to obtain the observation matrix $\mathbf{X}$ with the same number of rows and columns, respectively. In the previous section we considered centering as possible preprocessing. Another useful procedure is normalizing a column by the empirical standard deviation of the observations in the given column.

Principal component analysis is based on the *eigenvalue-eigenvector* decomposition of the $n \times n$ empirical covariance matrix $\mathbf{C}_x = \mathbf{X}^T\mathbf{X}$ (ref. 22-24). The eigenvalues are denoted by $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0$, where the last inequality follows from the presence of some random error in the data. Using the eigenvectors $u_1, u_2, \ldots, u_n$, define the new variables

$$
\begin{aligned}
z_1 &= \lambda^{-1/2}{}_1(u_{11}x_1 + u_{21}x_2 + \ldots + u_{n1}x_n) \\
z_2 &= \lambda^{-1/2}{}_2(u_{12}x_1 + u_{22}x_2 + \ldots + u_{n2}x_n) \\
&\cdot \\
&\cdot \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (1.110) \\
&\cdot \\
z_n &= \lambda^{-1/2}{}_n(u_{1n}x_1 + u_{2n}x_2 + \ldots + u_{nn}x_n)
\end{aligned}
$$

called principal components or abstract factors. We calculate the row vector $(z_1, z_2, \ldots, z_n)$ for each sample point, and construct the $m \times n$ principal component observation matrix $\mathbf{Z}$ from these rows. By (1.110) $\mathbf{Z}$ is given by

$$\mathbf{Z} = \mathbf{XUD}^{-1/2} , \qquad (1.111)$$

where $\mathbf{D}^{-1/2}$ is the diagonal matrix with the square roots of the reciprocal eigenvalues in its diagonal. You can readily verify that $\mathbf{C}_z = \mathbf{Z}^T\mathbf{Z} = \mathbf{I}$, and thus the principal components are uncorrelated and each variable z_i has the empirical variance 1. These are the important properties we will exploit.

Since $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, by (1.109) the observation matrix can be written as

$$\mathbf{X} = \mathbf{ZD}^{1/2}\mathbf{U}^T . \qquad (1.112)$$

Thus the variables x are represented by the linear combinations

$$x_1 = (\lambda^{1/2}{}_1u_{11})z_1 + (\lambda^{1/2}{}_2u_{12})z_2 + \dots + (\lambda^{1/2}{}_nu_{1n})z_n$$
$$x_2 = (\lambda^{1/2}{}_1u_{21})z_1 + (\lambda^{1/2}{}_2u_{22})z_2 + \dots + (\lambda^{1/2}{}_nu_{2n})z_n$$
$$\vdots \qquad (1.113)$$
$$x_n = (\lambda^{1/2}{}_1u_{n1})z_1 + (\lambda^{1/2}{}_2u_{n2})z_1 + \dots + (\lambda^{1/2}{}_nu_{nn})z_n$$

of the principal components. This expression is very informative. Each variable z_i has unit variance, and increasing the index i the corresponding principal components z_i will less and less influence the observed variables, according to the decreasing eigenvalues λ_i.

Principal components corresponding to small eigenvalues may give effects within the range of measurement errors. Having information on the magnitudes of these errors enables us to classify the principal components as primary and secondary ones. The simplest method of classification is considering a threshold on the eigenvalues, as we did in the previous section, but there exists a large number of more involved procedures (ref. 23). In some applications the selected primary principal components are rotated in order to form factors which can be better interpreted in physical terms. Sometimes one wants to know only the number of primary factors. For example, spectroscopic analysis of a number of mixtures containing the same components in different compositions will enable us to find the number of species without any further information on their properties.

Another important problem is to reproduce the observation matrix using only the primary factors, i.e., dropping some small terms in (1.113) that likely stem from measurement error.

Representing the data in terms of a small number of primary factors is a very efficient way of storing information. This approach is frequently used in spectroscopic libraries, designed to identify unknown species by comparing their spectra with ones filed in the library.

You will better understand the goals of factor analysis considering first the highly idealized situation with error-free observations and only $r < n$ linearly independent columns in the matrix **X** . As discussed in Section 1.1, all columns of **X** are then in an r-dimensional subspace, and you can write them as linear combinations of r basis vectors. Since the matrix $\mathbf{X}^T\mathbf{X}$ has now r nonzero eigenvalues, there are exactly r nonvanishing vectors in the matrix Z defined by (1.111), and these vectors form a basis for the subspace. The corresponding principal components $z_1, z_2, \dots, z_r$ are the coordinates in this basis. In the real life you have measurement errors, the columns of **X** are no more linearly dependent, and X^TX has $n - r$ small, but nonzero eigenvalues. Nevertheless, choosing the primary factors you select the subspace what is really important, and the primary factors are coordinates in the basis for this subspace.

Exercise

□ Reproduce the observation matrix in Section 1.8.7 using 1, 2, 3, and 4 , respectively, primary factors. Compute the sum of reproduction error squares for each case. Compare these sums with the following sums: $\lambda_2 + \lambda_3 + \lambda_4 + \lambda_5$, $\lambda_3 + \lambda_4 + \lambda_5$, $\lambda_4 + \lambda_5$ and λ_5, respectively.

REFERENCES

1 G.E. Forsythe and C.B. Moler, Computer Solution of Linear Algebraic Systems, Prentice Hall, Englewood Cliffs,N.J., 1967.

2 J.H. Wilkinson and C. Reinsch, Handbook for Automatic Computation, Vol. II, Linear Algebra, Springer Verlag, New York, 1971.

3 V.N. Fadeeva, Computational Methods of Linear Algebra, Dover Publications, Inc., New York, 1959.

4 A. Ralston and P. Rabinowitz, A First Course in Numerical Analysis, 2nd ed., McGraw-Hill, New York, 1978.

5 A. Ralston and H.S. Wilf (Editors) , Mathematical Methods for Digital Computers, Vol. 1, John Wiley, New York, 1964.

6 J. Stoer and R. Bulirsch, Introduction to Numerical Analysis, Springer Verlag, New York, 1980.

7 G.B. Dantzig, Linear Programming and Extensions. Princeton University Press, 1963.

8 G. Hadley, Linear Programming, Addison-Wesley P.C., Reading, Mass., 1963.

9 R.L. Johnston, Numerical Methods, A Software Approach, John Wiley, New York,1982.

10 G. Hadley, Linear Algebra. Addison-Wesley P.C., Reading, Mass., 1961.

11 G.H. Golub and C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, 1983.

12 A.N. Tihonov and V. Ya. Arsenin, Solution Methods of Ill-Conditioned Problems, Nauka, Moscow, 1979. (in Russian)

13 N.R. Amundson, Mathematical Methods in Chemical Engineering, Matrices and their Applications, Prentice-Hall, Englewood Cliffs,N.J., 1966.

14 R. Aris, Imtroduction to the Anaysis of Chemical Reactors, Prentice-Hall, Englewood Cliffs,N.J., 1961.

15 W.R. Smith and I. Missen, Chemical Reaction Equilibrium Analysis, John Wiley, New York, 1982.

16 K.V. Waller and P.M. Makila, Chemical reaction invariants and variants and their use in in reactor modeling, simulation and control, Ind. Eng. Chem. Proc. Des. Dev. 20 (1981) 1-11.

17 B. Carnahan and J.D. Wilkes, Digital Computing and Numerical Methods, John Wiley, New York, 1973.

18 A. Bloomfield and W. Steiger, Least absolute deviations curve fitting, SIAM J. Sci. Stat. Comput., 1 (1980) 290-301.

19 A.F Vasilyev and M.B. Pankova, Zavodskaya Laboratoriya, 9 (1972) 1076-1083 (in Russian).

20 R.E. Fuguitt and J.E. Hawkins, Rate of thermal isomerization of α-pinene in the liquid phase. J. Amer. Chem. Soc. 69 (1947) 319-325.

21 G.P.E. Box, W.G. Hunter, J.F. MacGregor and J. Erjavec, Some problems associated with the analysis of multiresponse data, Technometrics, 15 (1973) 33-51.
22 D.N. Lawley and A.E. Maxwell, Factor Analysis as a Statistical Method, Butterworth, London, 1963.
23 E.R. Malinowski and D.G. Howery, Factor Analysis in Chemistry, John Wiley, New York, 1980.
24 S. Wold, Factor and Principal Component Analysis, in D.L. Massart, A. Dijkstra and L. Kaufman (Editors), Evaluation and Optimization of Laboratory Methods and Analytical Procedures, Elsevier, Amsterdam, 1978.

Chapter 2

NONLINEAR EQUATIONS AND EXTREMUM PROBLEMS

On the basis of the previous chapter you can tell in advance the number of elimination steps or, more generally, the number of algebraic operations required for solving the system $\mathbf{Ax} = \mathbf{b}$ of linear equations. Unfortunately, there exist no similar finite procedures for solving the system of nonlinear equations of the general form

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} . \qquad (2.1)$$

Root finding in (2.1) invariably proceeds by iteration (refs. 1-3), constructing a sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots$ of approximate solutions that are expected to converge to a root $\mathbf{r}$ of (2.1). Naturally you would like to terminate the iteration when $\mathbf{x}_k$ satisfies the condition $\|\mathbf{x}_k - \mathbf{r}\| \le E$, where E is a desired error bound, but the root $\mathbf{r}$ is unknown. Some practical termination criteria you may use are $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| \le E_1$, $\|\mathbf{f}(\mathbf{x}_k)\| \le E_2$, or simply $k > IM$, where E_1 and E_2 are small parameters, and IM is an upper bound on the number of iterations. Neither of these conditions will assure that $\mathbf{x}_k$ is really close to the root $\mathbf{r}$, but will save you from useless iterations that can move $\mathbf{x}_k$ even further apart from $\mathbf{r}$ because of the accumulating roundoff errors. Since you certainly know what reasonable tolerance means for your particular problem, following the iterations on the screen is often superior to sophisticated convergence criteria.

Another class of problems requiring iteration is minimization or maximization of a nonlinear scalar valued function g which depends on one or more variables $\mathbf{x}$ (ref. 4). A value $\mathbf{r}$ of the independent variables is a local minimum point if $g(\mathbf{r}) \le g(\mathbf{x})$ for all $\mathbf{x}$ in a neighborhood of $\mathbf{r}$. Similarly, $\mathbf{r}$ is a local maximum if $g(\mathbf{r}) \ge g(\mathbf{x})$ in a neighborhood, and then $\mathbf{r}$ is a local minimum point of the function $-g(\mathbf{x})$. Therefore, we will restrict consideration to the problem

$$g(\mathbf{x}) \longrightarrow \min_{\mathbf{x}} . \qquad (2.2)$$

In some problems the possible region of independent variables is defined by equality or inequality constraints. As you have seen in Section 1.2, such constrained extremum problems are easy to solve if both the constraints and the objective function are linear. In the nonlinear case the most popular approach is to convert constrained problems into a sequence of unconstrained ones by penalizing points outside the feasible region (ref. 5). Such sophisticated methods are beyond the scope of our book. Nevertheless, our programs will also keep the estimates within a region defined by the user in order to avoid function evaluation at points where the function may not be defined at all. This simple test is sufficient if the extremum is known to be at some inner point of the feasible region.

While the number of independent variables is arbitrary in our definitions, it makes a tremendous difference in computations. Simultaneous solution of n equations and minimization in n dimensions are much more difficult than in one dimension. The main difference between one and several dimensions is that in one dimension it is possible to "bracket" a root or a local minimum point between some bracketing values, and then to tighten the interval of uncertainty. This gives rise to special algorithms, and hence the solution of a single equation and minimization in one variable will be discussed separately from the multidimensional methods.

Solutions of equations and those of extremum problems are closely related. A point $\mathbf{r}$ is the root of the equations $f(\mathbf{x}) = \mathbf{0}$ only if it minimizes the function $g = \mathbf{f}^T\mathbf{f}$. On the other hand every local extremum point of a differentiable function g satisfies the equations $\partial g(\mathbf{x})/\partial \mathbf{x} = \mathbf{0}$. Though a root is not necessarily an extremum point of g, this transformation may be advantageous in one dimension. As will be discussed the situation is, however, completely different with more than one variable.

We would like to choose methods that are robust, i.e., will converge to a solution if our initial estimate is reasonably close to it and, in addition, will converge rapidly. Apart from the one-dimensional case, where the solution can be bracketed and found in a very safe way, robustness of a method is much problem dependent. To measure how fast the convergence is we can use the local approximation

$$\|e_{k+1}\| \approx C \|e_k\|^p$$

in a small neighborhood of the solution $\mathbf{r}$, where $e_k = \mathbf{x}_k - \mathbf{r}$ is the error in the k-th iteration. The exponent p depends only on the method, which is then said to have convergence of order p . Since C is problem dependent and this analysis is local, the order p does not characterize the computational effort required to solve a particular problem. For this latter purpose one can use the number of iterations. We may need, however, to evaluate the function (and its partial derivatives, if the algorithm requires them) different times in each iteration of different methods, and hence a somewhat more realistic measure of the computational effort is the number of equivalent function evaluations.

2.1 NONLINEAR EQUATIONS IN ONE VARIABLE

2.1.1 Cardano method for cubic equations

The roots of quadratic and cubic equations are well known as algebraic expressions of the equation's coefficients, and hence this section is comletely disconnected from the rest of the chapter. Nevertheless, these simple problems are so frequently encountered that we cannot ignore their special solutions. You certainly know how to solve a quadratic equation, but we provide a routine for solving the cubic equation

$$Ax^3 + Bx^2 + Cx + D = 0 \ . \qquad (2.3)$$

Since $A \neq 0$ (otherwise we have a quadratic equation), introducing the variable $x = y - B/(3A)$, (2.3) can be reduced to the form

$$y^3 + py + q = 0 \ , \qquad (2.4)$$

where $p = (3C/A - B^2/A^2)/3$ and $q = (27D/A - 9BC/A^2 + 2B^3/A^3)/27$.

We first evaluate the discriminant $d = (p/3)^3 + (q/2)^2$. If $d \leq 0$, then the cubic equation has three (but not necessarily different) real roots. If, on the other hand, $d > 0$, then the equation has one real root and a conjugate pair of complex roots. Since you find the expressions for the roots in mathematical tables we proceed to the module.

Program module M20

```
2000 REM ****************************************************
2002 REM *         SOLUTION OF A CUBIC EQUATION             *
2004 REM *                CARDANO METHOD                    *
2006 REM ****************************************************
2008 REM INPUT:
2010 REM    A,B,C,D  COEFFICIENTS OF THE EQUATION:
2012 REM                  A*X^3+B*X^2+C*X+D=0
2014 REM OUTPUT:
2016 REM      ER     STATUS FLAG
2018 REM               0  SUCCESSFULL SOLUTION
2020 REM               1  DATA ERROR: A=0
2022 REM      NR     NUMBER OF REAL ROOTS (1 OR 3)
2024 REM IF NR=1
2026 REM      X      REAL ROOT
2028 REM    XR,XI    REAL AND IMAGINARY PART OF THE COMPLEX
2030 REM             CONJUGATE ROOTS XR+i*XI AND XR-i*XI
2032 REM IF NR=3
2034 REM   X1,X2,X3  REAL ROOTS
```

```
2036 IF A=0 THEN ER=1 :GOTO 2076
2038 ER=0 :P3=3.141593/3
2040 P0=B/A/3: P1=C/A/3-P0*P0 :P2=P0*P0*P0+(D-C*P0)/2/A
2042 IF P1<>0 THEN 2048
2044 P4=ABS(2*P2)^(1/3) : IF P2<0 THEN X1=P4-P0 ELSE X1=-P4-P0
2046  X2=X1 :X3=X1 :NR=3 :GOTO 2076
2048 DC=P2*P2+P1*P1*P1 :P=SQR(ABS(P1)) :IF P2<0 THEN P=-P
2050 P4=P2/P/P/P
2052 IF P1>=0 THEN 2070
2054 IF DC>0 THEN 2062
2056 NR=3 :FI=ATN(SQR(1-P4*P4)/P4)
2058  X1=-2*P*COS(FI/3)-P0 :X2=2*P*COS(P3-FI/3)-P0
2060  X3=2*P*COS(P3+FI/3)-P0 :GOTO 2076
2062 NR=1 :FI=LOG(P4+SQR(P4*P4-1)) :FI=EXP(FI/3)
2064  FI=(FI-1/FI)/(FI+1/FI) :P5=1-FI*FI
2066  X=-2*P/SQR(P5) :XI=P*FI*SQR(3/P5) :XR=-X/2-P0 :X=X-P0
2068  GOTO 2076
2070 NR=1 :FI=LOG(P4+SQR(P4*P4+1)) :FI=EXP(FI/3)
2072  FI=(FI-1/FI)/(FI+1/FI) :P5=1-FI*FI
2074  X=-2*P*FI/SQR(P5) :XI=P*SQR(3/P5) :XR=-X/2-P0 :X=X-P0
2076 RETURN
2078 REM ****************************************************
```

The only potential trouble is $A = 0$, which gives the return value of the status flag ER = 1 . The return value of NR is the number of real roots. If NR = 3 , the real roots will occupy the variables X1, X2 and X3 . If NR = 1 then the only real root will occupy X , whereas you will find the real and imaginary parts of the conjugate complex pair in the variables XR and XI, respectively.

Example 2.1.1 Molar volume of n-buthane from the Peng-Robinson equation of state

Find the molar volume v of n-buthane at temperature $T = 373.15$ K and pressure $P = 1.5\times10^{6}$ Pa by solving the Peng-Robinson equation of state (ref. 6)

$$P = \frac{RT}{v - b} - \frac{a(T)}{v(v + b) + b(v - b)} , \qquad (2.5)$$

where $R = 8.3144$ J/(mol K) is the universal gas constant, b and $a(T)$ are parameters of the equation of state depending on substance specific properties (and temperature). The expression for b is

$$b = 0.07780\ RT_c/P_c ,$$

where T_c is the critical temperature and P_c is the critical pressure. In addition to the two critical properties, the expression for $a(T)$ contains the actual temperature T and a third substance specific property called Pitzer's accentricity factor ω,

$$a(T) = 0.45724(R^2T_c^2/P_c)\left\{ 1 + m[1 - (T/T_c)^{0.5}]\right\}^2 ,$$

where

$$m = 0.37464 + 1.54226\omega - 0.26992\omega^2 .$$

For n-buthane the substance specific properties are (ref. 7) Tc = 425.2 K, $P_c = 3.75\times10^6$ Pa and $\omega = 0.193$.

The following main program computes b and a(T), rearranges (2.5) to the form (2.3) and calls module M20. If the equation has 3 real roots, we print only the largest, corresponding to gaseous state, and the smallest, which corresponds to liquid state. The root between them has no physical meaning.

```
100 REM -----------------------------------------------------------
102 REM EX. 2.1.1 MOLAR VOLUME BY CARDANO METHOD
104 REM MERGE M20
106 REM ---------- DATA  (R, Tc. Pc, OMEGA; TEMPERATURE AND PRESSURE)
108 RU=8.3144 :TC=425.2 :PC=3750000! :OM=.193
110 TT=373.15 :PP=1500000!
200 REM ---------- COEFFICIENTS OF THE EQUATION OF STATE
202 BE=.07780*RU*TC/PC :ME=.37464+1.54226*OM-.26992*OM^2
204 AE=.45724*(RU*TC)^2/PC*(1+ME*(1-(TT/TC)^.5))^2
206 REM ---------- COEFFICIENTS OF THE CUBIC EQUATION
208 A=PP :B=PP*BE-RU*TT :C=-3*PP*BE^2-2*RU*TT*BE+AE
210 D=PP*BE^3+RU*TT*BE^2-AE*BE
212 REM ---------- CARDANO METHOD
214 GOSUB 2000
216 V$=STRING$(50,"-")
218 LPRINT V$
220 LPRINT "NUMBER OF REAL ROOTS ................ ";NR
222 IF NR=3 THEN 228
224 LPRINT "V, m^3/mol .......................... ";X
226 GOTO 232
228 LPRINT "Vgas, m^3/mol ....................... ";X1
230 LPRINT "Vliq, m^3/mol ....................... ";X2
232 LPRINT V$
234 STOP
```

The output is as follows:

```
--------------------------------------------------
NUMBER OF REAL ROOTS ................  3
Vgas, m^3/mol .......................  1.505298E-03
Vliq, m^3/mol .......................  1.270651E-04
--------------------------------------------------
```

Further information is needed to select the thermodynamically stable state. The equilibrium vapor pressure is $P^{sat} = 1.529\times10^6$ Pa at the given temperature (ref. 9), hence we accept the root $v = 1.505298\times10^{-3}$ m^3/mol corresponding to the gaseous state. (If no experimental value is available for P^{sat} we can compute the fugacity coefficients for both states from the equation of state and select the thermodynamic state with the lower fugacity coefficient, see ref. 6).

This example illustrates that there may exist several roots even for very simple problems and we need a priori information to select the 'right' one. In iterative procedures this information is necessary for choosing an initial guess that will promote convergence to the desired root or, in one dimension, for choosing an interval that brackets it.

The possibility of several roots has two further consequences. First, you should always try to get some idea of how your function looks like, either on the basis of theoretical expectation, or by constructing a crude function plot. Second, it is advantageous to have methods that never get outside of the bracketing bounds, or never "jump" to a very far point of the region thereby avoiding divergence or convergence to a wrong root when the initial guess is sufficiently good.

2.1.2 Bisection method

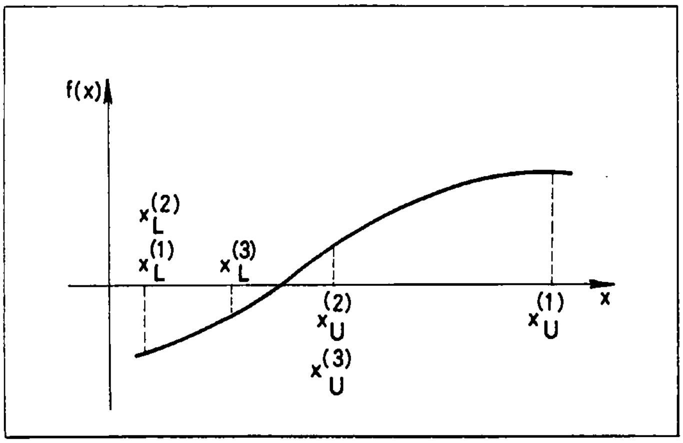


Fig. 2.1. Iterations in the bisection method

To apply this classical method we have to find a "bracketing" interval $[x_L, x_U]$ on which the continuous function $f(x)$ changes sign, thus the isolated root has odd multiplicity. The idea is very simple. We evaluate the function value f(x) at the interval's midpoint $\bar{x} = (x_L + x_U)/2$. If $f(\bar{x})f(x_L) \geq 0$, then $\bar{x}$ replaces the lower limit x_L, otherwise it will replace the upper limit x_U. Each iteration decreases the length of the interval containing the root by a factor of two. Therefore, to achieve the given tolerance EP , we need

$$IM = \log_2 \left[\frac{x_U - x_L}{EP} \right]$$

iterations. Fig. 2.1 shows three iterations, where $x_L^{(k)}$ and $x_U^{(k)}$ are the lower and upper limits in the k-th step.

The only information used in bisection is the sign of the function. The convergence is slow (of order 1), but never fails. Its disadvantages are the need for bracketing, which may be hard when two roots are very close, and the unability to find a root of odd multiplicity.

Program module M21

```
2100 REM ****************************************************
2102 REM *        SOLUTION OF A NONLINEAR EQUATION          *
2104 REM *                 BISECTION METHOD                 *
2106 REM ****************************************************
2108 REM INPUT:
2110 REM      XL    LOWER BOUND
2112 REM      XU    UPPER BOUND
2114 REM      EP    ERROR TOLERANCE ON THE ROOT
2116 REM OUTPUT:
2118 REM      ER    STATUS FLAG
2120 REM              0  SUCCESSFUL SOLUTION
2122 REM              1  NO SIGN CHANGE BETWEEN XA AND XU
2124 REM      X     ESTIMATE OF THE ROOT
2126 REM      F     FUNCTION VALUE F(X)
2128 REM USER-SUPPLIED SUBROUTINE:
2130 REM    FROM LINE 900;   X ---> F   ( FUNCTION EVALUATION )
2132 X=XL :GOSUB 900 :FL=F :X=XU :GOSUB 900 :FU=F
2134 IF FL*FU>0 THEN ER=1 :GOTO 2148
2136 IM=LOG(2+ABS(XU-XL)/EP)/LOG(2)
2138 FOR IT=1 TO IM
2140  X=(XL+XU)/2 :GOSUB 900
2142  IF F*FL>=0 THEN XL=X :FL=F ELSE XU=X :FU=F
2144 NEXT IT
2146 ER=0
2148 RETURN
2150 REM ****************************************************
```

The module returns the value ER = 1 if there is no sign change in the given interval. Otherwise it calculates the number IM of required iterations and performs IM bisections.

Example 2.1.2 Molar volume of n-buthane by bisection method

All methods in this section will be tested by solving the problem presented in Example 2.1.1 . Rearranging (2.5) we have the function

$$f(x) = \left[P + \frac{a(T)}{x(x + b) + b(x - b)}\right](x - b) - RT \ , \qquad (2.6)$$

where the solution of the equation $f(x) = 0$ is the molar volume. The simple main program we use is as follows.

```
100 REM ------------------------------------------------------------
102 REM EX. 2.1.2 MOLAR VOLUME BY BISECTION
104 REM MERGE M21
106 REM ---------- DATA  (R, Tc, Pc, OMEGA; TEMPERATURE AND PRESSURE)
108 RU=8.3144 :TC=425.2 :PC=3750000! :OM=.193
110 TT=373.15 :PP=1500000!
200 REM ---------- COEFFICIENTS OF THE EQUATION OF STATE (b,m and a)
202 BE=.0778*RU*TC/PC :ME=.37464+1.54226*OM-.26992*OM^2
204 AE=.45724*(RU*TC)^2/PC*(1+ME*(1-(TT/TC)^.5))^2
206 REM ---------- INITIAL INTERVAL AND ERROR TOLERANCE
208 XL=RU*TT/PP/2 :XU=RU*TT/PP*2 :EP=XU*.000001
210 V$=STRING$(50,"-")
212 LPRINT V$
214 GOSUB 2100
216 LPRINT
218 LPRINT "Vgas, m^3/mol ....................... ";X
220 LPRINT
222 LPRINT V$
224 STOP
900 REM ---------- FUNCTION EVALUATION
902 F=(PP+AE/(X*(X+BE)+BE*(X-BE)))*(X-BE)-RU*TT
904 LPRINT USING" IT= ###   X=#.######^^^^    F=#.#####^^^^";IT,X,F
906 RETURN
```

This is the first program in this book that needs a subroutine supplied by the user. Each program intending to call the module M21 must include BASIC statements that evaluate the function f at x. The first line of the user supplied subroutine is line 900 if only one is needed. Almost every program further in the book will require one, two or even three such subroutines, starting at lines 900, 800 and 700, respectively. Now you contribute to the program and hence it is advisable to include some extra prints in the subroutines for debugging. Since there are no local variables in BASIC, you should be careful when fitting user supplied subroutines to more complex programs. Particularly dangerous is altering values of the FOR-NEXT loop variables (in this case IT is such a variable). To minimize the threat of conflict try to distinguish your variables from ours, e.g. through the use of variable names consisting of three or more letters if your BASIC version does accept such longer names. A user supplied subroutine is always closed by a

RETURN statement.

In this example line 902 evaluates the function (2.6) and stores its value in the variable F . We print X and F to follow the iteration. The bracketing interval is chosen on the basis of a priori information. We know that in this example the compressibility factor PV/(RT) is close to one, and use the lower and upper limits $x_L = v^o/2$ and $x_U = 2v^o$, respectively, where v^o is the ideal molar volume

$$v^o = RT/P \ . \tag{2.7}$$

The error tolerance EP is set to the value EP = x_U*1E-6 , which is certainly smaller than the attainable accuracy based on the approximate equation (2.5). Due to the PRINT statement in the user supplied subroutine the program output is long, and only a few iterations are shown in Table 2.1.

Table 2.1
Steps in the bisection method

STEP	x_L, m^3/mol (sign f(x) = -1)	x_U, m^3/mol (sign f(x) = +1)	sign f($\bar{x}$)
1	0.103417E-02	0.413669E-02	+1
2	"	0.258543E-02	+1
3	"	0.180980E-02	-1
4	0.142199E-02	"	+1
5	"	0.161590E-02	+1
.			
.			
.			
15	0.150512E-02	0.150531E-02	-1
16	0.150521E-02	"	-1
17	0.150526E-02	"	-1
18	0.150528E-02	"	-1
19	0.150530E-02	"	+1
20	"	0.150530E-02	

2.1.3 False position method

Similarly to the bisection method, we need an interval $[x_L, x_U]$ that includes the root. The method is based on local linear interpolation of the function f by the straight line or chord through the points $\{x_L, f(x_L)\}$ and $\{x_U, f(x_U)\}$, as shown in Fig. 2.2. The "root" of this interpolating linear function is

$$\bar{x} = \frac{x_L f(x_U) - x_U f(x_L)}{f(x_U) - f(x_L)} \quad . \tag{2.8}$$

If $f(\bar{x})f(x_L) \geq 0$ then the new lower limit will be $\bar{x}$, otherwise $\bar{x}$ will replace the upper limit.

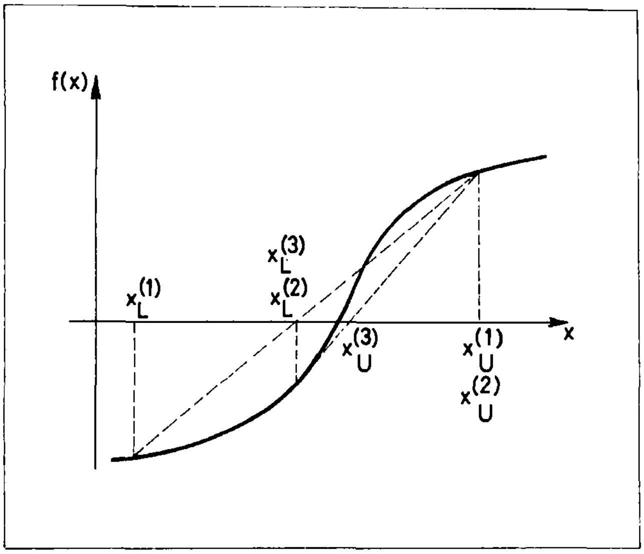


Fig. 2.2. Iterations in the false position method

We use the convergence criterion $\left|\bar{x}^{(k)} - \bar{x}^{(k-1)}\right| \leq EP$, where $\bar{x}^{(k)}$ is the estimate (2.8) in the k-th iteration. Three iterations are shown in Fig. 2.2.

The convergence is of order p , where p is slightly larger than 1. Indeed, the method usually performs better then the bisection method, while having the same robustness. Therefore, it is recommended for solving problems with little information available on the form of the function f. The only requirement is sufficient smoothness of f near the root.

Program module M22

```
2200 REM ####################################################
2202 REM #        SOLUTION OF A NONLINEAR EQUATION          #
2204 REM #              REGULA FALSI METHOD                 #
2206 REM ####################################################
2208 REM INPUT:
2210 REM      XL     LOWER BOUND
2212 REM      XU     UPPER BOUND
2214 REM      EP     ERROR TOLERANCE ON THE ROOT
2216 REM      IM     MAXIMUM NUMBER OF ITERATIONS
2218 REM OUTPUT:
2220 REM      ER     STATUS FLAG
2222 REM                0  SUCCESSFUL SOLUTION
2224 REM                1  NO SIGN CHANGE BETWEEN XL AND XU
2226 REM                2  REQUIRED ACCURACY NOT ATTAINED
2228 REM      X      ESTIMATE OF THE ROOT
2230 REM      F      FUNCTION VALUE F(X)
2232 REM USER-SUPPLIED SUBROUTINE:
2234 REM    FROM LINE 900;   X ---> F    ( FUNCTION EVALUATION )
2236 X=XL :GOSUB 900 :FL=F :X=XU :GOSUB 900 :FU=F
2238 IF FL*FU>0 THEN ER=1 :GOTO 2252
2240 FOR IT=1 TO IM
2242  XO=X :X=(XL*FU-XU*FL)/(FU-FL) :GOSUB 900
2244  IF F*FL>=0 THEN XL=X :FL=F ELSE XU=X :FU=F
2246  IF ABS(X-XO)<=EP THEN ER=0 :GOTO 2252
2248 NEXT IT
2250 ER=2
2252 RETURN
2254 REM ####################################################
```

Example 2.1.3 Molar volume by false position method

The main program is almost the same as in Example 2.1.2. The only differences are in the lines listed below.

```
102 REM EX. 2.1.3 MOLAR VOLUME BY FALSE POSITION METHOD
104 REM MERGE M22
208 XL=RU*TT/PP/2 :XU=RU*TT/PP*2 :EP=XU*.000001 :IM=30
214 GOSUB 2200
```

since we have to specify the number IM of allowed iterations and call the module M22. The iteration process is summarized in Table 2.2 , where the lower and upper limits, the inner point and the corresponding function value are shown in each iteration.

Table 2.2
Steps in the false position method

step	x_L, m^3/mol (sign f(x) = -1)	x_U, m^3/mol (sign f(x) = +1)	$\bar{x}$	f($\bar{x}$), J/mol
1	0.103417E-02	0.413669E-02	0.132899E-02	-.15706E+03
2	0.132899E-02	"	0.145394E-02	-.48040E+02
3	0.145394E-02	"	0.149163E-02	-.12955E+02
.				
.				
.				
8	0.150524E-02	"	0.150528E-02	-.14648E-01
9	0.150528E-02	"	0.150529E-02	-.39063E-02
10	0.150529E-02	"	0.150530E-02	-.12207E-02

Note that one of the limits is fixed during the iterations. This often happens with the false position method.

2.1.4 Secant method

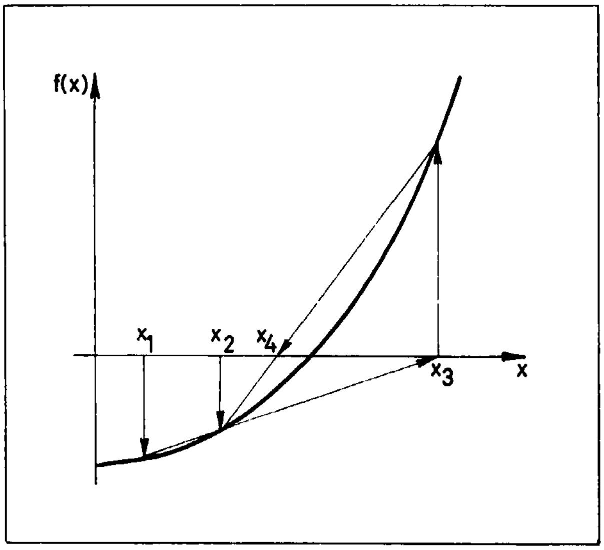


Fig. 2.3. Iterations in the secant method

The basic idea is the same as in the false position method, i.e., local linear approximation of the function. The starting interval $[x_1, x_2]$ does not, however, necessarily include the root. Then the straight line through the points $(x_1, f(x_2))$ and $(x_2, f(x_2))$ extrapolates rather than interpolates the function, and its "root"

$$\bar{x} = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)} \qquad (2.9)$$

will replace the "older" of the two previous points, thereby always retaining the most recent two estimates. (This requires an arbitrary choice in the first iteration.) If x_2 is the latest point, then x_1 is replaced by x_2 and x_2 by $\bar{x}$, as shown in Fig. 2.3. The convergence criterion is again

$$\left| \bar{x}^{(k)} - \bar{x}^{(k-1)} \right| \le EP.$$

Retaining the latest estimates for x_1 and x_2 , the slope of the line follows more closely the form of the function than in the false position method. The order of convergence can be shown to be 1.618, the "golden ratio", which we will encounter in Section 2.2.1. The root, however, is not necessarily bracketed, and the next estimate x_3 may be far away if the function value $f(x_1)$ is close to $f(x_2)$. Therefore we may run into trouble when starting the search in a region where the function is not monotonic.

Program module M23

```
2300 REM ****************************************************
2302 REM *       SOLUTION OF A NONLINEAR EQUATION          *
2304 REM *               SECANT METHOD                      *
2306 REM ****************************************************
2308 REM INPUT:
2310 REM     X1     INITIAL ESTIMATE OF THE ROOT
2312 REM     X2     SECOND INITIAL ESTIMATE OF THE ROOT
2314 REM     EP     ERROR TOLERANCE ON THE ROOT
2316 REM     IM     MAXIMUM NUMBER OF ITERATIONS
2318 REM OUTPUT:
2320 REM     ER     STATUS FLAG
2322 REM              0  SUCCESSFULL SOLUTION
2324 REM              1  REQUIRED ACCURACY NOT ATTAINED
2326 REM              2  ZERO SLOPE
2328 REM     X      ESTIMATE OF THE ROOT
2330 REM     F      FUNCTION VALUE F(X)
2332 REM USER SUPPLIED SUBROUTINE:
2334 REM   FROM LINE 900;    X ---> F   ( FUNCTION EVALUATION )
2336 X=X1 :GOSUB 900 :F1=F :X=X2 :GOSUB 900 :F2=F
2338 FOR IT=1 TO IM
2340  IF ABS(F2-F1)<1E-30 THEN ER=2 :GOTO 2352
2342  X=(X1*F2-X2*F1)/(F2-F1) :GOSUB 900
2344  IF ABS(X-X2)<=EP THEN ER=0 :GOTO 2352
2346  X1=X2 :F1=F2 :X2=X :F2=F
2348 NEXT IT
2350 ER=1
2352 RETURN
2354 REM ****************************************************
```

According to (2.9) the method breaks down if $f(x_1) = f(x_2)$ in any one of the iterations. Then the module returns the value ER = 2 .

Example 2.1.4 Molar volume by secant method

We deliberetely do not "bracket" the root and use the initial estimates $x_1 = v^o$ and $x_2 = 1.01v^o$, where v^o is the molar volume calculated from the ideal gas law (2.7). The iteration is expected to converge to the root corresponding to the gaseous state. We do not present the main program, because the deviations from the previous two main programs are only in the lines:

```
102 REM EX. 2.1.2 MOLAR VOLUME BY SECANT METHOD
104 REM MERGE M23
208 X1=RU*TT/PP :X2=1.01*RU*TT/PP :EP=X1*.000001 :IM=30
214 GOSUB 2300
```

i.e., we have to specify X1 and X2 instead of XL and XU . Results are listed in Table 2.3.

Table 2.3
Iterations in the secant method

step	x_1, m^3/mol	x_2, m^3/mol	$\bar{x}$	$f(\bar{x})$, J/mol
1	0.206835E-02	0.208903E-02	0.155446E-02	0.47540E+02
2	0.208903E-02	0.155446E-02	0.151126E-02	0.56833E+01
3	0.155446E-02	0.151126E-02	0.150539E-02	0.86914E-01
4	0.151126E-02	0.150539E-02	0.150530E-02	0.24414E-03
5	0.150539E-02	0.130530E-02	0.150530E-02	-.24414E-03

2.1.5 Newton-Raphson method

The idea is again local linear approximation, but now we use the tangent line at a current estimate x of the root. The tangent line will cross the zero at the abscissa

$$\bar{x} = x - \frac{f(x)}{f'(x)} , \qquad (2.10)$$

where $f'(x)$ is the derivative of function f at x , and we adopt $\bar{x}$ as the next estimate.

While all the previous methods use two points, the correction (2.10) is based exclusively on the local behavior of the function as shown on Fig. 2.4.

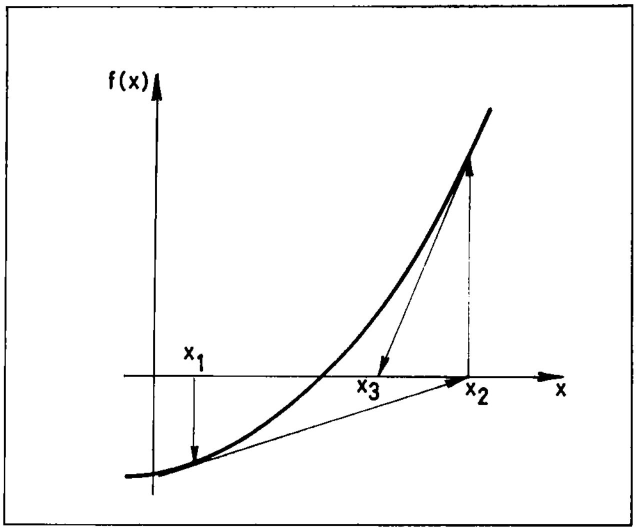


Fig. 2.4. Iterations in the Newton-Raphson method

Therefore the method has excellent convergence properties near the root (with order of convergence $p = 2$), but may result in meaningless estimates otherwise. In addition, the number of equivalent function evaluations is usually larger than in the secant method, which does not require the derivative but has almost the same convergence rate. Neither the Newton-Raphson, nor the secant method are recommended if the function f has an extremum near the root. You can easily construct pathological cases to understand this rule.

In the following module if the return value of the status flag is ER = 2, the derivative $f'(x)$ vanishes in one of the iterations, and by (2.10) the procedure breaks down.

Program module M24

```
2400 REM ####################################################
2402 REM #       SOLUTION OF A NONLINEAR EQUATION           #
2404 REM #            NEWTON-RAPHSON METHOD                  #
2406 REM ####################################################
2408 REM INPUT:
2410 REM      X      INITIAL ESTIMATE OF THE ROOT
2412 REM      EP     ERROR TOLERANCE ON THE ROOT
2414 REM      IM     MAXIMUM NUMBER OF ITERATIONS
2416 REM OUTPUT:
2418 REM      ER     STATUS FLAG
2420 REM               0  SUCCESSFUL SOLUTION
2422 REM               1  REQUIRED ACCURACY NOT ATTAINED
2424 REM               2  ZERO SLOPE
2426 REM      X      ESTIMATE OF THE ROOT
2428 REM      F      FUNCTION VALUE F(X)
2430 REM USER-SUPPLIED SUBROUTINE:
2432 REM    FROM LINE 900;   X ---> F  ( FUNCTION EVALUATION )
2434 REM    FROM LINE 800;   X ---> D  ( DERIVATIVE EVALUATION )
2436 GOSUB 900
2438 FOR IT=1 TO IM
2440  GOSUB 800
2442  IF ABS(D)<1E-30 THEN ER=2 :GOTO 2452
2444  DX=-F/D :X=X+DX :GOSUB 900
2446  IF ABS(DX)<=EP THEN ER=0 :GOTO 2452
2448 NEXT IT
2450 ER=1
2452 RETURN
2454 REM ####################################################
```

Example 2.1.5 Molar volume by Newton-Raphson method

To use the module M24 you should supply two subroutines. As in the previous methods the one starting at line 900 will evaluate the value F of the function. The second subroutine, starting at line 800, gives the current value of the derivative $f'(x)$ to the variable D . To start the iteration we need a single initial guess X . Once again we use the ideal gas volume as initial estimate. The lines different from the lines of the previous program are:

```
208 X=RU*TT/PP :EP=X*.000001 :IM=30

214 GOSUB 2400

800 REM ---------- DERIVATIVE
802 D=PP+AE/(X*(X+BE)+BE*(X-BE))-(X-BE)*AE*(2*X+2*BE)/(X*(X+BE)+BE*(X-BE))^2
804 RETURN
```

Results are shown in Table 2.4.

Table 2.4
Iterations in the Newton-Raphson method

STEP	x, m^3/mol	f(x)	$\bar{x}$, m^3/mol
0	0.2068345E-02	0.61113E+03	0.1553254E-02
1	0.1553254E-02	0.46354E+02	0.1505993E-02
2	0.1505993E-02	0.66138E+00	0.1505298E-03
3	0.1505298E-02	0.24414E-03	0.1505298E-03
4	0.1505298E-02	-.24414E-03	

A brief comparison of the different methods is given in Table 2.5. You may notice that the methods that use more information (i.e., the value of the function, not only its sign; a pair of values, not only one of them) converge more rapidly. You already know, however, that robustness is decreasing along the same line. Therefore, choosing a method you ought to consider how much is known on the form of the function and the position of its roots.

Table 2.5
Convergence behaviour of the different methods in the test example

Method	Number of iterations	Number of equivalent function evaluations	Theoretical order of convergence, p
Bisection	19	21	1
False position	10	12	>1
Secant	5	7	1.6
Newton-Raphson	4	9	2

2.1.6 Successive approximation

This method has such poor convergence properties that it is usually omitted from up-to-date textbooks on numerical analysis. We mention it, however, because it is very simple and still in use. In addition, the method can be easily extended to systems of equations where it is the basis for a number of improved techniques. The idea is writing the equation in the form

$$x = g(x) \qquad (2.11)$$

and performing the iteration

$$\bar{x} = g(x) \qquad (2.12)$$

where x and $\bar{x}$ are the old and new guesses of the root, respectively. A sufficient condition for convergence is the existence of a constant $K < 1$ and of an interval around the root on which

$$|g'(x)| \leq K , \qquad (2.13)$$

if our initial guess is also in this iterval. The steps of this procedure, also known as direct iteration, can be well followed on plots like the ones shown in Fig. 2.5. The 45° straight line helps to convert a $g(x)$ value into a new guess $\bar{x}$. You may encounter the situations of monotonic or oscillating convergence (Fig. 2.5.a and b, respectively) and monotonic or oscillating divergence (Fig. 2.5.c and d, respectively).

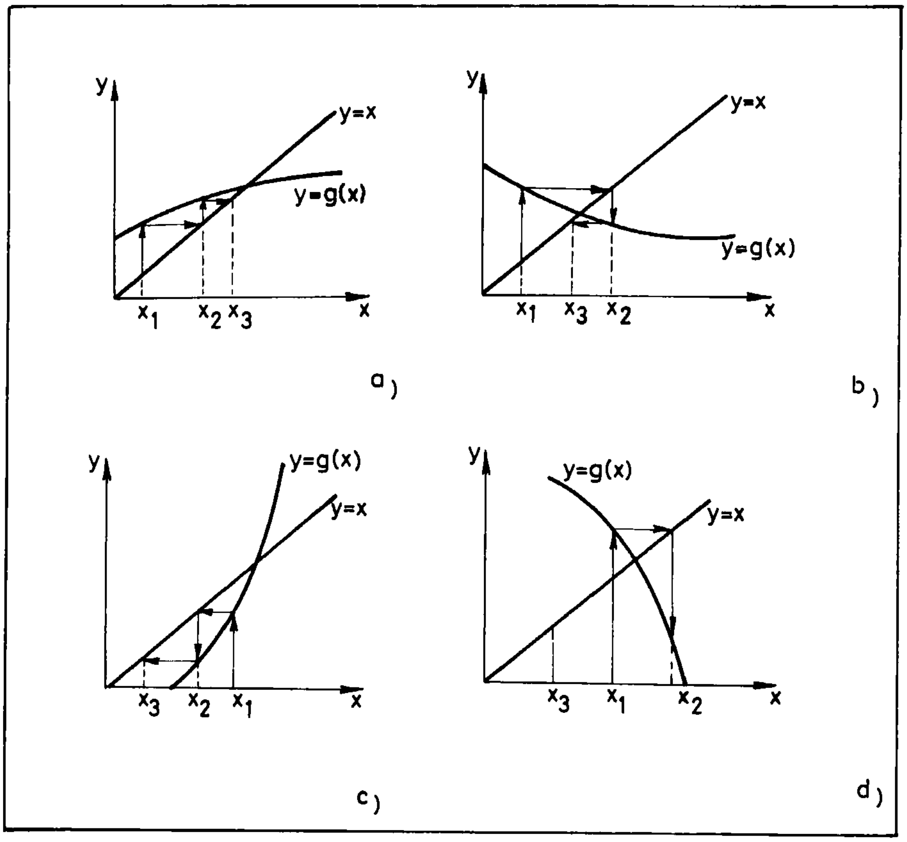


Fig. 2.5. Typical situations in successive approximation

It is more difficult to judge the properties of successive approximation if the original equation is of the form $f(x) = 0$, since it can be rearranged to the form (2.11) in many different ways, thereby significantly influencing the convergence. For example, an appropriate rearrangement results in (2.10), and hence even the Newton-Raphson method can be regarded as successive approximation.

Exercises

- Derive the iteration formulas (2.8), (2.9) and (2.10) on the basis of the geometrical ideas used in the corresponding method.

- Solve the test problem of this section at the pressure $P = 1.6\times10^6$ Pa keeping in mind that now the n-buthane is in liquid state.

- Three rearrangements of equation (2.5) to the form $x = g(x)$ are:

  $$g_1(x) = RT/P - a(x - b)/[x(x + b) + b(x - b)] + b$$

  $$g_2(x) = RT/P\{1 + a/P/[x(x + b) + b(x - b)]\}^{-1} + b$$

  $$g_3(x) = x + P(x - b) + a(x - b)/[x(x + b) + b(x - b)] - RT \ .$$

  Try to solve the test problem by successive approximation on the basis of these rearrangements. What is the reason of divergence in the case of g_3 ?

2.2 MINIMUM OF FUNCTIONS IN ONE DIMENSION

Similarly to the most robust methods of solving nonlinear equations, we start with bracketing. Assume that the interval $[x_U, x_L]$ contains a single minimum point r, i.e., the function f is decreasing up to r and increasing afterwards. Then the function is said to be unimodal on the interval $[x_L, x_U]$. This property is exploited in cut-off methods, purported to reduce the length of the interval which will, however, include the minimum point in all iterations.

The idea we use is similar to bisection, but now we need to evaluate the function at two inner points x_1 and x_2 of the interval, where $x_L < x_1 < x_2 < x_U$. If $f(x_1) \le f(x_2)$, then the minimum point is in the interval $[x_L, x_2]$, since we assumed that the function is decreasing up to the minimum point, see Fig. 2.6.a. Similarly, $f(x_1) \ge f(x_2)$ implies that the minimum point is in the interval $[x_1, x_U]$, as shown in Fig. 2.6.b. In both cases we can disregard some portion of the interval, either $(x_2, x_U]$ or $[x_L, x_1)$.

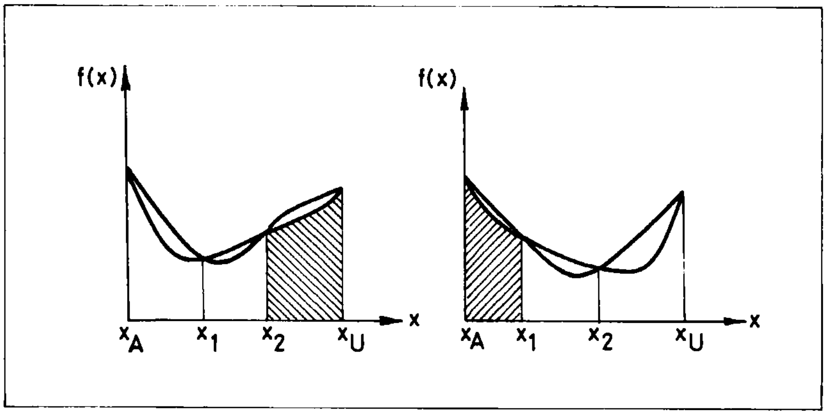


Fig. 2.6. Two situations in cut-off methods

The above discussion suggests selecting x_1 and x_2 close to the midpoint, thereby reducing the interval almost by a factor of two in one "cut". This is true in a single step. The search is, however, iterative, and there is a better strategy which involves a single function evaluation in each iteration (except the first one), while significantly reducing the bracketing interval.

2.2.1 Golden section search

We select the internal points x_1 and x_2 with the same spacing from either end, as shown in Fig. 2.7, where λ denotes the ratio of the longer segment to the total length of the uncertainty interval, i.e.,

$$\lambda = (x_2 - x_L) : (x_U - x_L) = (x_U - x_1) : (x_U - x_L).$$

The efficiency of the golden section stems from the special value of the ratio λ . We require the ratio of the larger of the two segments to the total length of the interval be the same as the ratio of the smaller to the larger segment, i.e., $\lambda/1 = (1 - \lambda)/\lambda$.

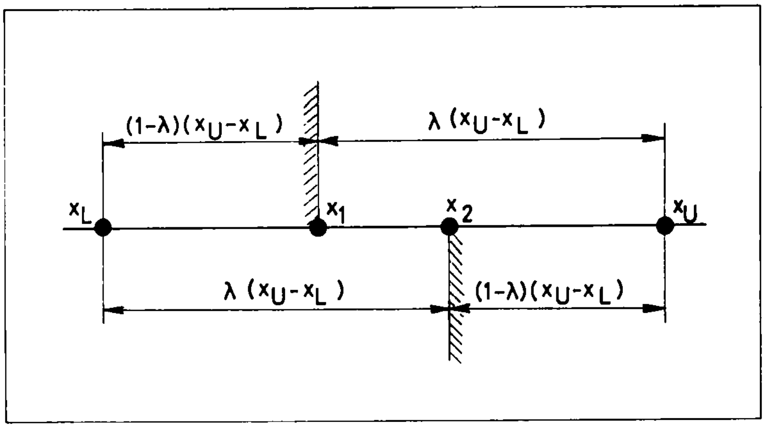


Fig. 2.7. Notations used in golden-section search derivation

The positive solution

$$\lambda = (\sqrt{5} - 1)/2 = 0.618... \qquad (2.14)$$

of this quadratic equation is the golden ratio, whose origin goes back to the ancient Greeks, but pops up in many different places in mathematics. Thus, the internal points are selected according to

$$x_1 = \lambda x_L + (1-\lambda)x_U$$
$$x_2 = (1-\lambda)x_L + \lambda x_U \; . \qquad (2.15)$$

To show why this famous ratio λ is good for us, assume that $f(x_1) > f(x_2)$ as shown in Fig. 2.8, and hence we cut off the interval $[x_L, x_1)$. Then the ratio of the remaining two segments is given by

$$\frac{x_U - x_2}{x_U - x_1} = \frac{1 - \lambda}{\lambda} = \lambda \; , \qquad (2.16)$$

where the last equality follows from the special choice of λ . Thus the reduced interval $[x_1, x_U]$ is already divided by the point x_2 in the same way as the original interval $[x_L, x_U]$ was divided by x_1. Therefore, we replace x_L and x_1 by the old value of x_1 and x_2, respectively, and need to evaluate the function only at the new point x_2, selected again by (2.15). Fig. 2.8 shows how the roles of our four points have been changed when performing this step. Similarly, for $f(x_1) < f(x_2)$ the new bracketing interval is $[x_L, x_2]$ and thus we replace x_2 and x_U by x_1 and x_2, respectively, and evaluate the function at the newly selected x_1.

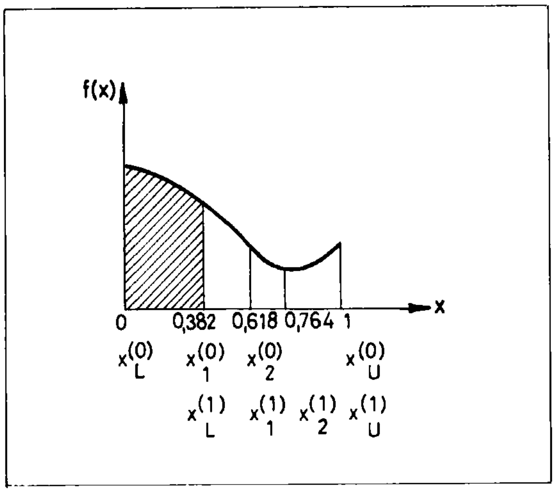


Fig. 2.8. Steps in the golden-section search

The golden section search guarantees that each new function evaluation will reduce the uncertainty interval to a length of λ times the previous interval. This is comparable to, but not as good as interval halving in the bisection method of solving a nonlinear equation. You can easily calculate that to attain an error tolerance EP we need

$$IM = -\log \left[\frac{x_U - x_L}{EP}\right] / \log \lambda$$

iterations. The following module calculates IM and performs the iterations.

Program module M25

```
2500 REM ****************************************************
2502 REM *     MINIMUM OF A FUNCTION OF ONE VARIABLE        *
2504 REM *           METHOD OF GOLDEN SECTIONS              *
2506 REM ****************************************************
2508 REM INPUT:
2510 REM     XL     LOWER BOUND
2512 REM     XU     UPPER BOUND
2514 REM     EP     ERROR TOLERANCE ON MINIMUM POINT
2516 REM OUTPUT:
2518 REM     X      ESTIMATE OF THE MINIMUM POINT
2520 REM     F      MINIMUM FUNCTION VALUE F(X)
2522 REM USER-SUPPLIED SUBROUTINE
2524 REM    FROM LINE 900;  X ---> F  ( FUNCTION EVALUATION )
2526 RL=(SQR(5)-1)/2 :RU=1-RL :RE=1/RL
2528 IM=LOG(RE*ABS(XU-XL)/EP)/LOG(RE)
2530 X1=RL*XL+RU*XU :X=X1 :GOSUB 900 :F1=F
2532 X2=RU*XL+RL*XU :X=X2 :GOSUB 900 :F2=F
2534 FOR IT=1 TO IM
2536   IF F1>F2 THEN 2542
2538   XU=X2 :X2=X1 :F2=F1 :X1=RL*XL+RU*XU :X=X1 :GOSUB 900 :F1=F
2540   GOTO 2544
2542   XL=X1 :X1=X2 :F1=F2 :X2=RU*XL+RL*XU :X=X2 :GOSUB 900 :F2=F
2544 NEXT IT
2546 RETURN
2548 REM ****************************************************
```

The module needs a user supplied routine starting at line 900 that will set the variable F to the value of the function evaluated at the actual value of X .

Example 2.2.1 Optimal drug dosing by golden section search

Consider a tablet that is taken regularly once a day. We want to find the optimal quantity of the drug (i.e., the only active ingredient) in the tablet in order to keep the drug concentration in the blood within a given therapeutic range $[c_L, c_U]$ as strictly as possible. To predict the drug concentration we use the linear compartmental model shown in Fig. 2.9, one of the most popular models in pharmacokinetics.

The model assumes that the drug enters compartment 1, representing mainly the gastrointestinal tract. The drug is then absorbed into the blood flow, represented by compartment 2. The absorption rate is $k_a q_1$, where q_1 is the current drug quantity in compartment 1. There is also a secretion or elimination process from compartment 2, with the elimination rate kq_2 , where q_2 denotes the quantity of drug in compartment 2.

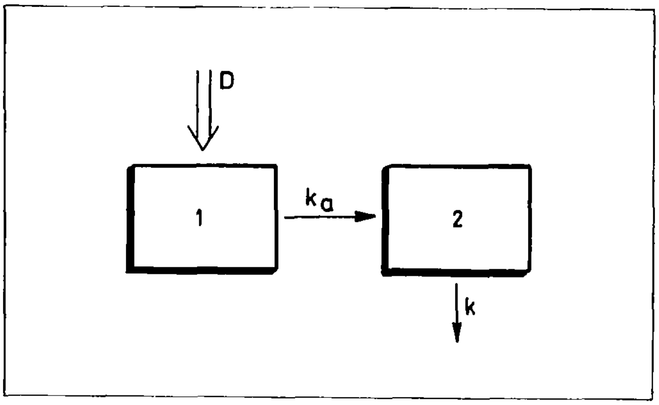


Fig. 2.9. Pharmacokinetic compartmental model
1 - gastrointestinal tract; 2 - blood flow

The compartmental model gives rise to a system of two linear differential equations whose forcing term (i.e., the drug intake) is a periodic function (ref. 9). After a transient period the solution of the differential equations is also a periodic function. This periodic solution predicts the drug concentration

$$c(t) = \frac{k_aD}{V(k_a-k)} \left[\frac{1}{1-\exp(-k\tau)} \exp(-kt) - \frac{1}{1-\exp(-k_a\tau)} \exp(-k_at) \right] \tag{2.17}$$

where D denotes the dosis, i.e., the quantity of drug in the tablet (mg); V is the distribution volume of the blood compartment; k_a is the absorption coefficient; k is the elimination rate coefficient; τ is the period, i.e., the time elapsed between two intakes of the tablet, and t is the time elapsed after the latest intake. In this example V = 10 l, k_a = 0.231 h^{-1}, k = 0.0693 h^{-1} and τ = 24 h (ref. 9).

We want to find the value of D that will keep c(t) between the values c_L = 14 mg/l and c_U = 26 mg/l as far as possible. For this purpose we minimize the objective function

$$f(D) = (1/\tau)\int_{0}^{\tau}\left[h^2{}_1(t) + h^2{}_2(t)\right]dt \qquad (2.18)$$

where

$$h_1(t) = \begin{cases} c(t) - c_U, \text{ if } c(t) > c_U \\ 0 \text{ otherwise ,} \end{cases} \qquad h_2(t) = \begin{cases} c_L - c(t), \text{ if } c(t) < c_L \\ 0 \text{ otherwise ,} \end{cases}$$

thereby more penalizing concentration values far from the therapeutic range. The area contributing to the objective function is shaded in Fig. 2.10.

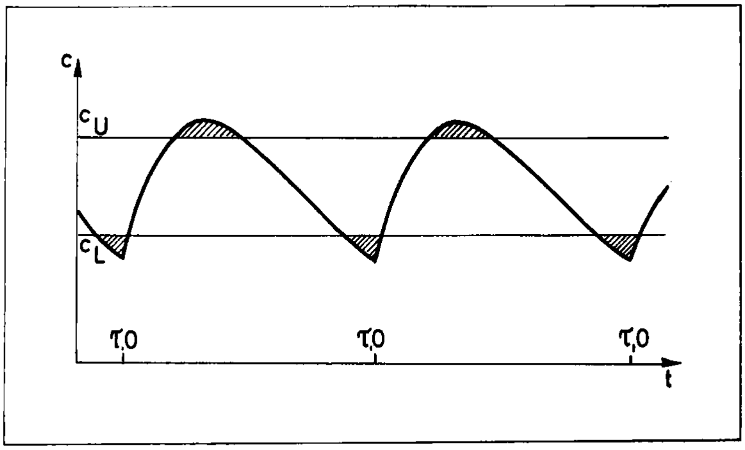


Fig. 2.10. Periodic drug concentration in blood

You are certainly aware that the compartmental model is a simplified representation of the real physicological process. Therefore, it is completely adequate to use a simplified objective function by approximating the integrals in (2.17). We divide the interval $[0, \tau]$ of integration into NW equal subintervals of length $\Delta t = \tau/NW$, and approximate $c(t)$ by its midpoint value $c_i = c[(i - 1/2)\Delta t]$. The objective function is approximated by

$$f(D) = (1/\tau)\sum_{i=1}^{NW} \Delta f_i(D) \qquad (2.19)$$

where

$$\Delta f_i(D) = \begin{cases} (c_i - c_U)^2 \Delta t & \text{if } c_i > c_U \\ (c_L - c_i)^2 \Delta t & \text{if } c_i < c_L \\ 0 \text{ otherwise .} \end{cases}$$

Since the dosis D is expected to raise the blood concentration at least to c_U at certain time points, from the approximate balance equation $D/V \approx c_U$ we have $D \approx 260$ mg . Therefore, the initial interval $[0, 1000]$ certainly includes the minimum point, which can be easily checked evaluating the function (2.19) over a course grid. The desired error tolerance is $EP = 0.1$, more than adequate in this problem. The main program we use is as follows.

```
100 REM ------------------------------------------------------------
102 REM EX. 2.2.1 OPTIMUM DOSING BY GOLDEN SECTION METHOD
104 REM MERGE M25
106 REM ---------- DATA
108 REM (VOLUME,ABSORPTION,ELIMINATION)
110 VW=10 :KA=.231 :KW=6.930001E-02
112 REM (TIME INTERVAL,LOWER AND UPPER LIMIT OF CONCENTRATION)
114 TW=24 :CL=14 :CU=26
116 REM (NUMBER OF NODES)
118 NW=48
200 REM ---------- AUXILIARY QUANTITIES
202 DT=TW/NW
204 E1=1/(1-EXP(-KW*TW)) :E2=1/(1-EXP(-KA*TW))
206 REM ---------- LOWER AND UPPER LIMIT OF DOSE, ERROR TOLERANCE
208 XL=0 :XU=1000 :EP=.1
210 REM ---------- GOLDEN SECTION MODULE
212 V$=STRING$(50,"-")
214 LPRINT V$ :LPRINT :LPRINT "GOLDEN SECTION METHOD" :LPRINT
216 GOSUB 2500
218 IF ER THEN LPRINT "STATUS FLAG:";ER :GOTO 254
220 LPRINT :LPRINT "   MINIMIZATION OF SQUARE ERRORS" :LPRINT
222 LPRINT "CYCLE LENGTH, h ......................... ";TW
224 LPRINT "NUMBER OF NODES ......................... ";NW
226 LPRINT "OPTIMUM DRUG DOSE, mg/l ................. ";INT(10*X)*.1
228 LPRINT "MINIMUM OBJECTIVE FUNCTION VALUE ........ ";F
230 LPRINT
232 LPRINT V$
234 LPRINT "TIME, h    PLASMA CONC, mg/l       REMARK"
236 LPRINT V$
238 FOR T=1 TO TW
240  Y=X*KA/VW/(KA-KW)*(E1*EXP(-KW*T)-E2*EXP(-KA*T))
242  LPRINT USING "###            ##.##             ";T,Y,
244  IF Y>CU THEN LPRINT "HIGH CONCENTRATION";
246  IF Y<CL THEN LPRINT "LOW  CONCENTRATION";
248  LPRINT
250 NEXT T
252 LPRINT V$
254 STOP
```

```
900 REM ---------- OBJECTIVE FUNCTION
902 F=0
904 FOR I=1 TO NW
906  T=(I-.5)*DT
909  Y=X*KA/VW/(KA-KW)*(E1*EXP(-KW*T)-E2*EXP(-KA*T))
910  IF Y>CU THEN F=F+(Y-CU)^2*DT
912  IF Y<CL THEN F=F+(CL-Y)^2*DT
914 NEXT I
916 LPRINT USING" IT=###  DOSE=####.##   OBJ. FUN.=#.######^^^^";IT,X,F
918 RETURN
```

The limits of the uncertainty interval in some of the iterations are shown in Table 2.6. The optimal dosis is D_{opt} = 335.4 mg , which gives the minimum value $f(D_{opt})$ = 14.44 ($mg^2 l^{-2} s$).

Applying a finer grid (NW > 48) does not alter the location of the minimum more than the desired tolerance EP = 0.1 mg . In Fig. 2.10 we have already shown the concentration of the drug following the dosis D_{opt}, taken at the beginning of each period of length τ = 24 h. According to this solution, one tablet a day does not enable us to keep drug concentration $c(t)$ within the therapeutic range for all times. We could decrease the period, i.e., τ = 20 h would be a suitable choice, but it is not a practical advice to take a tablet each 20 hours. Taking two tablets a day (i.e., with τ = 12 h), there exists an interval $[D_L, D_U]$ such that $f(D) = 0$ for all D in this interval. From physiological point of view the best choice is D_L, i.e., the least dosis that gives the desired drug concentration in blood. The golden section search module as presented here will result in this lower limit (D_L = 138.2 mg) because in line 2536 we used the relation sign ">" and not ">=" .

Table 2.6
Steps in the golden section search

step	x_L, mg	x_U, mg	relation of f_1 to f_2
1	0	1000	<
2	"	618.034	>
3	236.068	"	<
4	"	472.136	>
.			
.			
.			
18	335.275	335.555	>
19	335.382	"	<
final	"	335.489	

Although the golden section search works quite well, it is obviously not the best available for a given number of function evaluations. For example, with only two evaluations allowed it is better to choose the internal points close to the midpoint of the initial interval, as we already discussed. The idea can be extended to any a priori fixed number N of function evaluations, and gives rise to the Fibonacci search strategy, involving the famous Fibonacci numbers (ref.10). For sufficiently large N, however, the golden section search is almost as efficient as the Fibonacci search (and can be regarded as the limiting case of the latter). Comparing the function values in the inner points, both methods use little information, and their convergence is linear (i.e., of order $p = 1$). Similarly to the methods of solving a nonlinear equation we can increase the order p by constructing a local approximation of the function. While in equation solving a linear approximation did the job, now we look for a minimum, and hence the approximating function should be at least quadratic.

2.2.2 Parabolic interpolation

In this method the next estimate $\bar{x}$ is the location

$$\bar{x} = x - \frac{(x-v)^2[f(x)-f(w)] - (x-w)^2[f(x)-f(v)]}{(x-v)[f(x)-f(w)] - (x-w)[f(x)-f(v)]} \qquad (2.20)$$

of the minimum of the parabol through the last point $\{x,f(x)\}$ and two previously evaluated points $\{w, f(w)\}$ and $\{v, f(v)\}$. The method fails if the three points are on a straight line, since then the denominator is zero (i.e., the parabola has no minimum). In addition, equation (2.20) will locate the maximum rather than the minimum if the coefficient of the second order term in the interpolating parabola is negative.

To avoid these problems Brent (ref. 11) suggested a combination of the parabolic fit and the golden section bracketing technique. The main idea is to apply equation (2.20) only if (i) the next estimate falls within the most recent bracketing interval; (ii) the movement from the last estimate is less than half the step taken in the iteration before the last. Otherwise a golden section step is taken. The following module based on (ref. 12) tries to avoid function evaluation near a previously evaluated point.

Program module M26

```
2600 REM ####################################################
2602 REM #    MINIMUM OF A FUNCTION OF ONE VARIABLE         #
2604 REM #  PARABOLIC INTERPOLATION  -  BRENT'S METHOD      #
2606 REM ####################################################
2608 REM INPUT:
2610 REM      XL     LOWER BOUND
2612 REM      XU     UPPER BOUND
2614 REM      EP     ERROR TOLERANCE ON MINIMUM POINT
2616 REM      IM     MAXIMUM NUMBER OF ITERATION
```

```
2618 REM OUTPUT:
2620 REM      X      ESTIMATE OF THE MINIMUM POINT
2622 REM      F      MINIMUM FUNCTION VALUE F(X)
2624 REM      ER     STATUS FLAG
2626 REM               0
2628 REM               1  TOLERANCE NOT ATTAINED IN 'IM' ITERATIONS
2630 REM USER-SUPPLIED SUBROUTINE
2632 REM    FROM LINE 900;   X ---> F  ( FUNCTION EVALUATION )
2634 ER=0 :RL=(SQR(5)-1)/2 :DX=(XU-XL)/2 :X=(XU+XL)/2
2636 V=X :W=X :E=0: GOSUB 900 :FX=F :FV=F :FW=F
2638 REM ----- LOOP
2640 FOR IT=1 TO IM
2642  XM=(XL+XU)/2 :IF ABS(X-XM)<=2*EP-(XU-XL)/2 THEN 2696
2644  IF ABS(E)<EP THEN 2664
2646  REM ----- AUXILIARY QUANTITIES TO A PARABOLIC STEP
2648  R=(X-W)*(FX-FV) :Q=(X-V)*(FX-FW) :P=(X-V)*Q-(X-W)*R
2650  Q=2*(Q-R) :IF Q>=0 THEN P=-P ELSE Q=-Q
2652  EL=E :E=DX
2654  IF ABS(P)>=ABS(Q*EL/2) OR P<=Q*(XL-X) OR P>=Q*(XU-X) THEN 2664
2656  REM ----- PARABOLIC STEP
2658  DX=P/Q :U=X+DX
2660  IF (U-XL)<2*EP OR (XU-U)<2*EP THEN IF XM>X THEN DX=EP ELSE DX=-EP
2662  GOTO 2670
2664  REM ----- GOLDEN SECTION STEP
2666  IF X>=XM THEN E=XL-X ELSE E=XU-X
2668  DX=RL*E
2670  REM ----- FUNCTION EVALUATION
2672  IF ABS(DX)>=EP THEN U=X+DX ELSE IF DX>0 THEN U=X+EP ELSE U=X-EP
2674  X0=X :X=U :GOSUB 900 :FU=F :X=X0
2676  REM ----- NEW BRACKET AND PREVIOUS POINTS
2678  IF FU>FX THEN 2684
2680   IF U>=X THEN XL=X ELSE XU=X
2682   V=W :FV=FW :W=X :FW=FX :X=U :FX=FU :GOTO 2692
2684  IF U<X THEN XL=U ELSE XU=U
2686  IF FU>FW AND W<>X THEN 2690
2688   V=W :FV=FW :W=U :FW=FU :GOTO 2692
2690  IF FU<=FV OR V=X OR V=W THEN V=U :FV=FU
2692 NEXT IT
2694 ER=1
2696 F=FX :RETURN
2698 REM ****************************************************
```

The input to the module is similar to the one of the module M25. The only difference is that in this case the maximum number IM of iterations should be specified before calling.

Example 2.2.2 Optimum dosing by Brent's method

We solve the problem of Example 2.2.1 with the same starting interval. The main program is essentially the same except the following lines:

```
208 XL=0 :XU=1000 :EP=.1 :IM=30
214 LPRINT V$ :LPRINT :LPRINT "BRENT'S METHOD" :LPRINT
216 GOSUB 2600
```

The iteration process is summarised in Table 2.6.

Table 2.6
Steps in Brent's method

iteration	x_L, mg	x_U, mg	type of step	best estimate x	best estimate f(x)
1	0	1000	golden s.	190.983	350.169
2	"	500	golden s.	381.966	108.784
3	190.983	"	parabolic	301.111	35.696
4	"	381.966	parabolic	318.578	21.586
5	301.111	"	parabolic	324.801	17.703
6	318.578	"	golden s.*	"	"
7	"	360.131	golden s.*	"	"
8	"	346.636	parabolic	334.473	14.469
.					
.					
.					
12	334.783	335.522	parabolic	335.422	14.439
final	335.322	"		"	"

* parabolic movement would be too big compared to the movement two steps before

Parabolic interpolation is more effective than golden section search for this problem, because the function is of parabolic character in the vicinity of the minimum. To show a counterexample we slightly change the approximate objective function (2.19) and define Δf_i by

$$\Delta f_i(D) = \begin{cases} (c_i - c_U)\Delta t & \text{if } c_i > c_U \\ (c_L - c_i)\Delta t & \text{if } c_i < c_L \\ 0 \text{ otherwise} , \end{cases} \tag{2.21}$$

i.e., now we minimize the shaded area shown in Fig. 2.10. (You can easily make this change in the main program by dropping the exponent 2 from lines 910 and 912 .) In this case Brent's method needs the same number of function evaluations as the golden section search does.

As we mentioned, in one-dimensional problems it might be advantageous to solve the equation $f'(x) = 0$ instead of minimizing f. The roots of $f'(x) = 0$ are, however, not necessarily minimum points, and hence we can run into trouble without a good a priori knowledge of the form of the function. In addition, we need an expression for the derivative $f'(x)$ which is frequently not available, e.g., in the optimal dosing problem of this section.

2.3 SYSTEMS OF NONLINEAR EQUATIONS

In the following sections $\mathbf{x}$, $\mathbf{f}$ and $\mathbf{g}$ are all n-vectors, and we should slightly change our notations. The estimates of a root (or those of a minimum point) in iterations $1,2,\ldots,k$ will be denoted by $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, ..., $\mathbf{x}^{(k)}$, whereas $x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}$ will denote the vector components of the k-th estimate $\mathbf{x}^{(k)}$.

The simplest method of solving a system of nonlinear equations is the successive approximation

$$\mathbf{x}^{(k)} = \mathbf{g}(\mathbf{x}^{(k-1)}) \ , \qquad (2.22)$$

where $\mathbf{g}$ denotes the function rearranged as described in Section 2.1.6. As in one dimension, the method is slow and does not guarantee the convergence, though these properties heavily depend on the way of rearranging the equations to the form $\mathbf{x} = \mathbf{g}(\mathbf{x})$. It is, however, extraordinarily simple and hence convenient in many applications, e.g., for flowsheeting in chemical engineering (ref. 13), and hence must be taken more seriously than in one dimension. Great efforts have been devoted to improve the basic method. The simplest modified version is

$$\mathbf{x}^{(k)} = (1 - c)\mathbf{x}^{(k-1)} + c\mathbf{g}(\mathbf{x}^{(k-1)}) \ , \qquad (2.23)$$

which retains the previous estimate $x^{(k-1)}$ up to the weighting factor $(1-c)$. If $c = 1$ (i.e., the simplest direct iteration) gives rise to monotonic convergence, then we can try to increase the rate of convergence by setting $c > 1$. This simple trick is known as acceleration or overrelaxation. On the other hand a divergent or wildly oscillating iteration observed at $c = 1$ may be improved by choosing an appropriate value $0 < c < 1$, which leads to relaxed or damped iteration. The method is so simple that we do not include a program module, but suggest to write your own program and experiment with different values of c on the test example we will study in the next sections.

2.3.1 Wegstein method

A popular version of successive approximation due to Wegstein (ref. 14) can be best understood by considering the one dimensional case as shown in Fig. 2.11. Let $x^{(1)}$ and $x^{(2)}$ denote two current estimates of the root of equation $x = g(x)$. Geometrically the method consists of extending the line through $\{x^{(1)}, g(x^{(1)})\}$ and $\{x^{(2)}, g(x^{(2)})\}$ until it crosses the line $y = x$. The new estimate is then set to the abscissa $x^{(3)}$ of the cross point, replacing the oldest of the previous estimates.

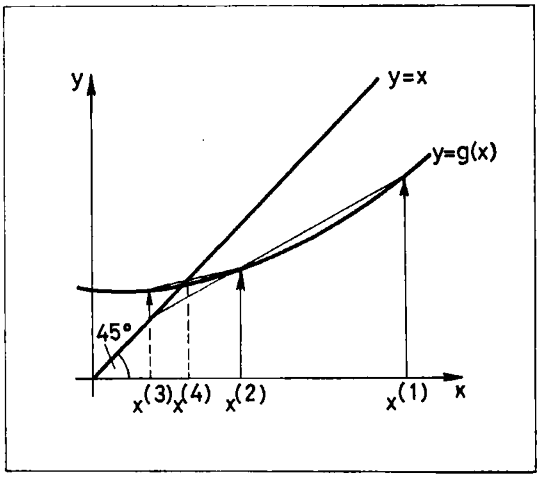


Fig. 2.11. Geometric idea of the Wegstein method

You can verify that this iteration can be described by (2.23) where the parameter c is chosen according to

$$c = \frac{x^{(2)} - x^{(1)}}{x^{(2)} - x^{(1)} - [g(x^{(2)}) - g(x^{(1)})]} . \qquad (2.24)$$

Therefore the above expression provides an automatic selection of the damping or accelerating factor c in each iteration. The idea is easy to extend to a system of equations, if each element x_i of the vector $\mathbf{x}$ is regarded to be independent of the others when using the expressions (2.23) and (2.24) . Thus the Wegstein method uses separate factors c_i for each variable:

$$x_i^{(k)} = (1 - c_i^{(k)})x_i^{(k-1)} + c_i^{(k)} g_i(\mathbf{x}^{(k-1)}) \qquad (2.25)$$

where the factor for the i-th variable is given by

$$c_i^{(k)} = \frac{x_i^{(k-1)} - x_i^{(k-2)}}{x_i^{(k-1)} - x_i^{(k-2)} - [g_i(\mathbf{x}^{(k-1)}) - g_i(\mathbf{x}^{(k-2)})]} . \qquad (2.26)$$

The geometrical idea introduced for one dimension applies only to a system of independent equations with g_i depending only on x_i . Though in the general case the method does not have a sound theoretical basis, it may perform surprisingly well.

Program module M30

```
3000 REM ####################################################
3002 REM #    SOLUTION OF SIMULTANEOUS EQUATIONS X=G(X)     #
3004 REM #                 WEGSTEIN METHOD                  #
3006 REM ####################################################
3008 REM INPUT:
3010 REM      N      PROBLEM SIZE
3012 REM     X(N)    STARTING POINT
3014 REM     D(N)    PERTURBATION OF STARTING POINT
3016 REM      EP     THRESHOLD ON NORM OF THE STEP
3018 REM      IM     MAXIMUM NUMBER OF ITERATION
3020 REM OUTPUT:
3022 REM      ER     STATUS FLAG
3024 REM               0  SUCCESSFUL SOLUTION
3026 REM               1  UNADMISSIBLE STARTING POINT
3028 REM               2  REQUIRED ACCURACY NOT ATTAINED
3030 REM     X(N)    ESTIMATE OF THE SOLUTION
3032 REM     G(N)    RHS OF EQUATIONS AT FINAL ESTIMATE
3034 REM USER-SUPPLIED SUBROUTINE:
3036 REM    FROM LINE 900;   X(.) --> G(.)  ( RHS EVALUATION )
3038 REM AUXILIARY ARRAY:
3040 REM     R(N)
3042 ER=0 :GOSUB 900 :IF ER<>0 THEN ER=1 :GOTO 3072
3044 FOR IT=1 TO IM
3046  FOR I=1 TO N :R(I)=G(I) :X(I)=X(I)+D(I) :NEXT I
3048  ER=0 :GOSUB 900 :IF ER=0 THEN 3054
3050   FOR I=1 TO N :D(I)=.95*D(I) :X(I)=X(I)-.05*D(I) :NEXT I
3052   SD=SD*.9025 :GOTO 3048
3054  IF IT>1 AND SQR(SD)<=EP THEN 3072
3056  SD=0
3058  FOR I=1 TO N
3060   C=D(I)-G(I)+R(I) :IF ABS(C)<1E-30 THEN C=SGN(C)*1E-30
3062   C=D(I)/C :D(I)=C*(G(I)-X(I)) :SD=SD+D(I)*D(I)
3064  NEXT I
3066  IF SQR(SD)<=EP THEN ER=0 :GOTO 3072
3068 NEXT IT
3070 ER=2
3072 RETURN
3074 REM ####################################################
```

To start the procedure we need a vector X of initial estimates and a vector D of initial corrections (movements), both of dimension N . During the iteration the vector D contains the current correction vector $x^{(k)} - x^{(k-1)}$. The convergence criterion is $\|D\| \le EP$. The user supplied subroutine starting at line 900 sets the vector G to the actual value of the right hand side vector computed at the current estimate X.

An important feature of the program module is checking the feasibility of the current estimate X. If X is outside the region you anticipate to contain the solution, you should set the error flag ER to a nonzero value in the subroutine. At a nonzero ER value the module will repeatedly decrease the length of the correction vector by 5% in order to keep the estimate within the feasible region. This is particularly important if the function g is not defined for all values of the variables (e.g., in the following example all variables should be positive). All the further programs of this chapter will include such test and modification of a potential new point.

Example 2.3.1 Equilibrium of chemical reactions by Wegstein method

We consider a chemical system consisting of the following species: methane (CH_4) , water (H_2O) , carbon monoxid (CO) , carbon dioxide (CO_2) , and hydrogen (H_2) . There are two linearly independent reactions among these species, e.g.,

$$CH_4 + H_2O = CO + 3H_2 \qquad (2.27)$$

and

$$CO + H_2O = CO_2 + H_2 \; . \qquad (2.28)$$

We want to find the equilibrium composition at the temperature $T = 1000$ K and pressure $P = 1.013\times10^5$ Pa if the system initially contains 2 moles of methane and 3 moles of water. For the given temperature and pressure the natural logarithms of the equilibrium constants K_1 and K_2 expressed in terms of mole fractions are known: $\log K_1 = 3.4789$ and $\log K_2 = -0.0304$. Let n_1, n_2, ..., n_5 denote the mole numbers of species CH_4, H_2O, CO, CO_2, and H_2, respectively. Then $n=\sum_{i=1}^{5} n_i$ is the total mole number. Writing the mole fractions $y_j = n_j/n$ into the equilibrium relations $K_1 = y_3y_5^3/(y_1y_2)$ and $K_2 = y_4y_5/(y_3y_2)$ and taking the logarithms of both sides we arrive at the following equations

$$\begin{aligned} &\log [n_3n_5^3/(n_1n_2n^2)] = \log K_1 \\ &\log [n_4n_5/(n_3n_2)] = \log K_2 \; . \end{aligned} \qquad (2.29)$$

As discussed in Section 1.8.1, with known initial mole numbers n_1^o through n_5^o the extents x_1 and x_2 of reactions (2.27) and (2.28), respectively, determine the current mole vector uniquely. Since x_1 measures the moles of CH_4 consumed in the first reaction and x_2 measures the moles of CO_2 produced in the second reaction, we have the stoichiometric relations:

$$n_1 = n_1^o - x_1, \quad n_2 = n_2^o - x_1 - x_2, \quad n_3 = n_3^o + x_1 - x_2, \quad n_4 = n_4^o + x_2, \quad n_5 = n_5^o + 3x_1 + x_2$$

$$n = \left[\sum_{i=1}^{5} n_i^o \right] + 2x_1 \ . \tag{2.30}$$

Using (2.30) we can write equations (2.29) in terms of x_1 and x_2 . These equations will be rearranged to the form $\mathbf{x} = \mathbf{g}(\mathbf{x})$, simply by adding x_1 to both sides of the first, and x_2 to both sides of the second:

$$\begin{aligned} x_1 &= \log\, [n_3 n_5^3/(n_1 n_2 n^2)] - \log K_1 + x_1 = g_1(x_1,x_2) \\ x_2 &= \log\, [n_4 n_5/(n_3 n_2)] - \log K_2 + x_2 = g_2(x_1,x_2). \end{aligned} \tag{2.31}$$

The following main program solves the system (2.31) .

```
100 REM ------------------------------------------------------------
102 REM EX. 2.3.1 REACTION EQUILIBRIUM BY WEGSTEIN METHOD
104 REM MERGE M30
106 REM ---------- DATA
108 DIM N0(5),NW(5),N$(5)
110 REM (NATURAL LOG K VALUES)
112 W1=3.4789 :W2=-.0304
114 REM (INITIAL MOLE NUMBERS)
116 N0(1)=2 :N0(2)=3 :N0=5
118 REM (NAMES)
120 N$(1)="methan ........"
122 N$(2)="water ........."
124 N$(3)="carbon monoxid "
126 N$(4)="carbon dioxid ."
128 N$(5)="hydrogen ......"
200 REM ---------- PROBLEM SIZE AND CONTROL PARAMETERS
202 N=2 :IM=30 :EP=.000001
204 DIM X(N),D(N),G(N),R(N)
206 REM ---------- STARTING POINT AND STARTING STEP
208 X(1)=1   :X(2)=.1
210 D(1)=.01 :D(2)=.01
212 V$=STRING$(53,"-")
214 LPRINT "WEGSTEIN METHOD" :LPRINT
216 GOSUB 3000
218 LPRINT :LPRINT V$
220 LPRINT "              INITIAL     %     EQUILIBRIUM     %"
222 F$=" ##.### ###.###     ##.###### ###.###"
224 LPRINT V$
226 FOR I=1 TO 5
228  LPRINT N$(I);
230  LPRINT USING F$;N0(I);N0(I)/N0*100,NW(I),NW(I)/NW*100
232 NEXT I
234 LPRINT V$ :LPRINT
236 STOP
```

```
900 REM ---------- G(X)
902 NW(1)=N0(1)-X(1) :NW(2)=N0(2)-X(1)-X(2)   :NW(3)=N0(3)+X(1)-X(2)
904 NW(4)=N0(4)+X(2) :NW(5)=N0(5)+3*X(1)+X(2) :NW=N0+2*X(1)
906 FOR IW=1 TO 5
908  ER=ER-(NW(IW)<=0)
910 NEXT IW
912 IF ER<>0 THEN 922
914 G(1)=LOG(NW(3)*NW(5)^3/(NW(1)*NW(2)*NW^2))-W1+X(1)
916 G(2)=LOG(NW(4)*NW(5)/(NW(3)*NW(2)))-W2+X(2)
918 LPRINT USING"IT=### x(1)=#.#####^^^^ x(2)=#.#####^^^^ ";IT,X(1),X(2);
920 LPRINT USING"g(1)=#.#####^^^^ g(2)=#.#####^^^^";G(1),G(2)
922 RETURN
```

Starting at line 900 you find the user subroutine. In this routine the mole numbers occupy the array elements NW(1), NW(2), ..., NW(5) and the scalar variable NW stores the total mole number. At the current value X(1) and X(2) of the reaction extents we first calculate the mole numbers. If any of them is negative or zero, the error flag ER is set to a nonzero value. If the mole numbers are feasible, the values computed according to (2.31) will occupy the array elements G(1) and G(2). The initial estimates are X(1) = 1 and X(2) = 0.1 , the first corrections are D(1) = D(2) = 0.01 . The following output shows some of the iterations.

```
WEGSTEIN METHOD

IT=  0 x(1)=0.10000E+01 x(2)=0.10000E+00 g(1)=-.37237E+01 g(2)=-.15773E+01
IT=  1 x(1)=0.10100E+01 x(2)=0.11000E+00 g(1)=-.36603E+01 g(2)=-.14486E+01
IT=  2 x(1)=0.18848E+01 x(2)=0.24129E+00 g(1)=0.21785E+01 g(2)=0.26207E+00
.
.
.

IT=  8 x(1)=0.18570E+01 x(2)=0.24259E+00 g(1)=0.18573E+01 g(2)=0.24274E+00
IT=  9 x(1)=0.18569E+01 x(2)=0.24258E+00 g(1)=0.18569E+01 g(2)=0.24261E+00
IT= 10 x(1)=0.18569E+01 x(2)=0.24258E+00 g(1)=0.18569E+01 g(2)=0.24259E+00
IT= 11 x(1)=0.18569E+01 x(2)=0.24258E+00 g(1)=0.18569E+01 g(2)=0.24258E+00

-----------------------------------------------------
               INITIAL      %      EQUILIBRIUM     %
-----------------------------------------------------
methan ........  2.000  40.000      0.143061    1.642
water .........  3.000  60.000      0.900486   10.334
carbon monoxid   0.000   0.000      1.614364   18.526
carbon dioxid .  0.000   0.000      0.242575    2.784
hydrogen ......  0.000   0.000      5.813392   66.714
-----------------------------------------------------
```

2.3.2 Newton-Raphson method in multidimensions

As in one dimension, the Newton-Raphson method is based on local linear approximation of the function f around the current estimate $x^{(k-1)}$. The approximating linear function is given by

$$y = f(x^{(k-1)}) + J^{(k-1)}[x^{(k)} - x^{(k-1)}] , \qquad (2.32)$$

where

$$[J^{(k-1)}]_{ij} = \partial f_i(x^{(k-1)})/\partial x_j$$

are the elements of the $n \times n$ Jacobian matrix of f at $x^{(k-1)}$. Setting $y = 0$ in (2.32) we obtain a set of linear equations for the correction $d^{(k)} = x^{(k)} - x^{(k-1)}$. The solution of this matrix equation is

$$x^{(k)} - x^{(k-1)} = [J^{(k-1)}]^{-1} f(x^{(k-1)}). \qquad (2.33)$$

Though (2.33) is the well known form of the Newton-Raphson correction formula, it is more efficient to solve the matrix equation for $d^{(k)}$ by LU decomposition and backward substitution.

As for a single equation, the convergence is of order 2, and hence the method is expected to perform very well if the elements of the Jacobian matrix are continuous functions in a neighborhood of the root and the initial guess is sufficiently good. The computational costs are, however, high, since we perform n equivalent function evaluations for constructing the Jacobian matrix in each iteration. The solution of the matrix equation is also a nontrivial task. In addition, a singular or nearly singular Jacobian matrix (2.32) gives meaningless corrections.

Program module M31

```
3100 REM ####################################################
3102 REM #   SOLUTION OF SIMULTANEOUS EQUATIONS F(X)=0     #
3104 REM #             NEWTON-RAPHSON METHOD                #
3106 REM ####################################################
3108 REM INPUT:
3110 REM     N      PROBLEM SIZE
3112 REM     X(N)   STARTING POINT
3114 REM     E1     THRESHOLD ON FUNCTION NORM
3116 REM     E2     THRESHOLD ON STEP LENGTH
3118 REM     IM     MAXIMUM NUMBER OF ITERATIONS
3120 REM OUTPUT:
3122 REM     ER     STATUS FLAG
3124 REM              0  SUCCESSFUL SOLUTION
3126 REM              1  UNADMISSIBLE STARTING POINT
3128 REM              2  SINGULAR JACOBI MATRIX
3130 REM              3  NEITHER THRESHOLD ATTAINED
3132 REM     X(N)   ESTIMATE OF THE SOLUTION
3134 REM     F(N)   FUNCTION VALUES AT THE ESTIMATE
3136 REM    A(N,N)  INVERSE OF THE JACOBI MATRIX AT THE ESTIMATE
3138 REM AUXILIARY VECTOR:
3140 REM     R(N)
3142 REM USER SUPPLIED SUBROUTINES:
3144 REM    FROM LINE 900,   X(.) --> F(.)  ( FUNCTION EVALUATION )
3146 REM    FROM LINE 800,   X(.) --> A(.,.) ( JACOBI MATRIX EVALUATION )
3148 REM MODULES CALLED: M14,M15
```

```
3150 ER=0 :GOSUB 900 :IF ER<>0 THEN ER=1 :GOTO 3182
3152 FOR IT=1 TO IM
3154  SF=0 :FOR I=1 TO N :R(I)=X(I) :SF=SF+F(I)*F(I) :NEXT I
3156  IF SQR(SF)<=E1 THEN 3182
3158  REM --------- LU DECOMPOSITION OF THE JACOBIAN MATRIX
3160  GOSUB 800 :GOSUB 1400 :IF ER=1 THEN ER=2 :GOTO 3182
3162  REM --------- BACKSUBSTITUTION
3164  FOR I=1 TO N: X(I)=-F(I) :NEXT I :GOSUB 1500
3166  SX=0 :FOR I=1 TO N :SX=SX+X(I)*X(I) :X(I)=R(I)+X(I) :NEXT I
3168  REM --------- CHECK NEW POINT
3170  ER=0 :GOSUB 900 :IF ER=0 THEN 3176
3172   FOR I=1 TO N :X(I)=.95*X(I)+.05*R(I) :NEXT I
3174   SX=SX*.9025 :GOTO 3170
3176 IF SQR(SX)<=E2 THEN 3182
3178 NEXT IT
3180 ER=3
3182 RETURN
3184 REM ***************************************************
```

Two subroutines should be supplied by the user of the module. The subroutine starting at line 900 computes the left hand sides of the equations $f(x) = 0$, and stores them in array F. The subroutine starting at line 800 evaluates the elements of the Jacobian matrix and puts them into the array A. The subroutine starting at line 900 should return the error flag value $ER \neq 0$ if the current estimate stored in array X is unfeasible. The matrix equation is solved by calling the modules M14 and M15, so that do not forget to merge these modules when using module M31. We terminate the procedure if $\|f(x^{(k)})\| \leq E1$ or $\|d^{(k)}\| \leq E2$.

Example 2.3.2 Equilibrium of chemical reactions by Newton-Raphson method

The problem is the one stated in the previous example. The equations are obtained rearranging (2.29). Since the Jacobian is always calculated after function evaluation, the subroutine starting at line 800 makes use of the computed mole numbers. We show the main program and the iterations, whereas the final results are the same as in the previous example and hence omitted from the output.

```
100 REM ------------------------------------------------------------
102 REM EX. 2.3.2 REACTION EQUILIBRIUM BY NEWTON-RAPHSON METHOD
104 REM MERGE M14,M15,M31
106 REM ---------- DATA
108 DIM N0(5),NW(5),N$(5)
110 REM (NATURAL LOG K VALUES)
112 W1=3.4789 :W2=-.0304
114 REM (INITIAL MOLE NUMBERS)
116 N0(1)=2 :N0(2)=3 :N0=5
```

```
118 REM (NAMES)
120 N$(1)="methan ........"
122 N$(2)="water ........."
124 N$(3)="carbon monoxid "
126 N$(4)="carbon dioxid ."
128 N$(5)="hydrogen ......"
200 REM ---------- PROBLEM SIZE AND CONTROL PARAMETERS
202 N=2 :IM=30 :E1=1E-08 :E2=.000001
204 DIM X(N),F(N),A(N,N)
206 REM ---------- STARTING POINT
208 X(1)=1 :X(2)=.1
210 V$=STRING$(53,"-")
212 LPRINT "NEWTON-RAPHSON METHOD" :LPRINT
214 GOSUB 3100
216 LPRINT :LPRINT V$
218 LPRINT "                 INITIAL      %      EQUILIBRIUM     %"
220 F$=" ##.### ###.###     ##.###### ###.###"
222 LPRINT V$
224 FOR I=1 TO 5
226  LPRINT N$(I);
228  LPRINT USING F$;N0(I);N0(I)/N0*100,NW(I),NW(I)/NW*100
230 NEXT I
232 LPRINT V$ :LPRINT
234 STOP
800 REM ---------- JACOBI MATRIX
802 A(1,1)=1/NW(3)+9/NW(5)+1/NW(1)+1/NW(2)-4/NW
804 A(1,2)=-1/NW(3)+3/NW(5)+1/NW(2)
806 A(2,1)=3/NW(5)-1/NW(3)+1/NW(2)
808 A(2,2)=1/NW(4)+1/NW(5)+1/NW(3)+1/NW(2)
810 RETURN
900 REM ---------- F(X)
902 NW(1)=N0(1)-X(1) :NW(2)=N0(2)-X(1)-X(2)   :NW(3)=N0(3)+X(1)-X(2)
904 NW(4)=N0(4)+X(2) :NW(5)=N0(5)+3*X(1)+X(2) :NW=N0+2*X(1)
906 FOR IW=1 TO 5
908  ER=ER-(NW(IW)<=0)
910 NEXT IW
912 IF ER<>0 THEN 922
914 F(1)=LOG(NW(3)*NW(5)^3/(NW(1)*NW(2)*NW^2))-W1
916 F(2)=LOG(NW(4)*NW(5)/(NW(3)*NW(2)))-W2
918 LPRINT USING"IT=### x(1)=#.#####^^^^ x(2)=#.#####^^^^ ";IT,X(1),X(2);
920 LPRINT USING"f(1)=#.#####^^^^ f(2)=#.#####^^^^";F(1),F(2)
922 RETURN
```

```
 NEWTON-RAPHSON METHOD

IT=  0 x(1)=0.10000E+01 x(2)=0.10000E+00 f(1)=-.47237E+01 f(2)=-.16773E+01
IT=  1 x(1)=0.19421E+01 x(2)=0.21007E+00 f(1)=0.11097E+01 f(2)=-.11628E+00
IT=  2 x(1)=0.18854E+01 x(2)=0.23661E+00 f(1)=0.29576E+00 f(2)=-.59448E-02
IT=  3 x(1)=0.18593E+01 x(2)=0.24221E+00 f(1)=0.22473E-01 f(2)=0.12591E-03
IT=  4 x(1)=0.18570E+01 x(2)=0.24257E+00 f(1)=0.13113E-03 f(2)=0.18273E-05
IT=  5 x(1)=0.18569E+01 x(2)=0.24258E+00 f(1)=-.23842E-06 f(2)=-.18626E-07
```

2.3.3 Broyden method

The Broyden method is one of the simplest quasi-Newton method. The aim of quasi Newton methods is to achieve convergence properties comparable to those of the Newton-Raphson method, but without the use of the Jacobian matrix, and with no need for solving a matrix equation in each iteration. All quasi-Newton methods are based on local linear approximation

$$\Delta f^{(k)} = B^{(k+1)}\Delta x^{(k)} , \qquad (2.34)$$

where $\Delta x^{(k)} = x^{(k)} - x^{(k-1)}$, $\Delta f^{(k)} = f(x^{(k)}) - f(x^{(k-1)})$ and $B^{(k+1)}$ can be regarded as the approximation of the Jacobian matrix. Similarly to the correction formula (2.33) of the Newton-Raphson method we can derive the correction

$$\Delta x^{(k+1)} = x^{(k+1)} - x^{(k)} = - H^{(k+1)}f(x^{(k)}), \qquad (2.35)$$

where $H^{(k+1)} = [B^{(k+1)}]^{-1}$. In one dimension the scalar $B^{(k+1)}$ is the slope of the secant and knowing two previous points we can calculate it from (2.34). In multidimensions, however, $B^{(k+1)}$ is an $n\times n$ matrix, whereas we have only n equations in (2.34). To fill the gap we need assumptions, and different assumptions result in different quasi-Newton methods, see, e.g. (ref. 15). In the so called rank 1 methods $B^{(k+1)}$ is restricted to the form

$$B^{(k+1)} = B^{(k)} + u^{(k)}[v^{(k)}]^T, \qquad (2.36)$$

where $u^{(k)}$ and $v^{(k)}$ are n-vectors. The matrix modifying the current estimate of the Jacobian is therefore obtained from a column vector multiplied by a row vector. The rank of such matrices does not exceed one which gives the name of the methods.

In the method of Broyden (ref. 16) $v^{(k)}$ is selected to be equal to $\Delta x^{(k)}$ and $u^{(k)}$ is then obtained from the n equations (2.34). The geometric idea behind selecting this $v^{(k)}$ is to leave $B^{(k+1)}$ unchanged along directions with no new information available in the k-th iteration. Indeed, for any vector z orthogonal to $\Delta x^{(k)}$ (i.e., with $\Delta x^{(k)}z^T = 0$) we get $B^{(k+1)}z = B^{(k)}z$, and hence $B^{(k+1)}$ behaves similarly to $B^{(k)}$ along these vectors.

Using the estimate $B^{(k+1)}$ updated in each iteration we do not need to evaluate the Jacobian matrix. The second improvement is avoiding the inversion of $B^{(k+1)}$ through the use of the Hausholder formula. According to this latter, the inverse of $B^{(k+1)}$ of the form (2.36) is given by

$$[B^{(k+1)}]^{-1} = [B^{(k)}]^{-1} - [B^{(k)}]^{-1}uv^T[B^{(k)}]^{-1}/(1 + v^T[B^{(k)}]^{-1}u) , \qquad (2.37)$$

where we omitted the superscript k for the vectors u and v. Therefore we can derive $H^{(k+1)} = [B^{(k+1)}]^{-1}$ directly from the previous estimate of the inverse $H^{(k)} = [B^{(k)}]^{-1}$ and the vectors u and v. In the Broyden method the particular selection of these vectors results in the updating formula

$$H^{(k+1)} = H^{(k)} - [H^{(k)}\Delta f^{(k)} - \Delta x^{(k)}][\Delta x^{(k)}]^T H^{(k)} / \left[[\Delta x^{(k)}]^T H^{(k)} \Delta f^{(k)}\right] \quad . \qquad (2.38)$$

The $(k+1)$-th iteration of the Broyden method consists of updating the inverse according to (2.38) and then performing a correction by (2.35).

The convergence properties are similar to those of the Newton-Raphson method, usually with more iterations but less equivalent function evaluations. In some cases, however, the correction vector $\Delta x^{(k)}$ gets into a subspace and remains there in all subsequent iterations. Then the method is unable to explore the whole space of the variables. This problem can be resolved by restarting the procedure at the point where it claims to have found a root (i.e., reinitialize $H^{(1)}$ to the identity matrix).

In the following program module based on (ref. 17) we need only an initial guess $x^{(o)}$, whereas $H^{(1)} = I$, the identity matrix. At the beginning, however, we perform n steps to update only $H^{(1)}$, while the estimate of the solution is left unchanged. The Broyden iteration, involving both (2.38) and (2.35) starts only after this initial updating cycle. The procedure is terminated if

$\|\Delta f^{(k)}\| \le E1$ or $\|\Delta x^{(k)}\| \le E2$.

Program module M32

```
3200 REM ####################################################
3202 REM #    SOLUTION OF SIMULTANEOUS EQUATIONS F(X)=0     #
3204 REM #                 BROYDEN METHOD                   #
3206 REM ####################################################
3208 REM INPUT:
3210 REM     N      PROBLEM SIZE
3212 REM     X(N)   STARTING POINT
3214 REM     E1     THRESHOLD ON FUNCTION NORM
3216 REM     E2     THRESHOLD ON STEP LENGTH
3218 REM     IM     MAXIMUM NUMBER OF ITERATIONS
3220 REM OUTPUT:
3222 REM     ER     STATUS FLAG
3224 REM              0 SUCCESSFUL SOLUTION
3226 REM              1 UNADMISSIBLE STARTING POINT
3228 REM              2 NEITHER THRESHOLD ATTAINED
3230 REM     X(N)   ESTIMATE OF THE SOLUTION
3232 REM     F(N)   FUNCTION VALUES AT THE FINAL ESTIMATE
3234 REM    H(N,N)  ESTIMATE OF THE INVERSE OF THE JACOBI MATRIX
3236 REM USER-SUPPLIED SUBROUTINE:
3238 REM    FROM LINE 900;   X(.) --> F(.)  ( FUNCTION EVALUATION )
3240 REM AUXILIARY ARRAY: R(3,N)
3242 REM ---------- STARTING POINT
3244 ER=0 :GOSUB 900 :IF ER<>0 THEN ER=1 :GOTO 3334
3246 REM ---------- UNIT MATRIX INTO H
3248 FOR I=1 TO N
3250  R(1,I)=0
3252  FOR J=1 TO N :H(I,J)=-(I=J) :NEXT J
3254 NEXT I
```

```
3256 REM ---------- N STEPS TO INITIALIZE H
3258 FOR K=1 TO N :R(1,K)=100*E2 :GOSUB 3292 :R(1,K)=0 :NEXT K
3260 REM ---------- ITERATION
3262 FOR IT=1 TO IM
3264  FOR I=1 TO N
3266   SA=0
3268   FOR J=1 TO N :SA=SA-H(I,J)*F(J) :NEXT J
3270   R(1,I)=SA
3272  NEXT I
3274  GOSUB 3292
3276  SA=0 :SB=0
3278  FOR I=1 TO N
3280   SA=SA+F(I)*F(I) :SB=SB+R(1,I)*R(1,I)
3282  NEXT I
3284  REM --------- CONVERGENCE
3286  IF SQR(SA)<=E1 OR SQR(SB)<=E2 THEN ER=0 :GOTO 3334
3288 NEXT IT
3290 ER=2 :GOTO 3334
3292 REM ---------- STEP OF THE BROYDEN METHOD
3294  FOR I=1 TO N :X(I)=X(I)+R(1,I) :R(3,I)=F(I) :NEXT I
3296  ER=0 :GOSUB 900 :IF ER=0 THEN 3302
3298   FOR I=1 TO N :X(I)=X(I)-.05*R(1,I) :R(1,I)=.95*R(1,I)  :NEXT I
3300   GOTO 3296
3302  FOR I=1 TO N :R(2,I)=F(I)-R(3,I) :NEXT I
3304  SA=0
3306  FOR I=1 TO N
3308   SB=0
3310   FOR J=1 TO N :SB=SB+H(I,J)*R(2,J) :NEXT J
3312   R(3,I)=SB-R(1,I) :SA=SA+SB*R(1,I)
3314  NEXT I
3316  IF SA=0 THEN 3330
3318  FOR J=1 TO N
3320   SB=0
3322   FOR I=1 TO N :SB=SB+R(1,I)*H(I,J) :NEXT I
3324   SB=SB/SA
3326   FOR I=1 TO N :H(I,J)=H(I,J)-SB*R(3,I) :NEXT I
3328  NEXT J
3330  RETURN
3332 REM ---------- END OF STEP
3334 RETURN
3336 REM **************************************************
```

We need only a single user subroutine starting at line 900, which is completely analogous to the corresponding one required by the Newton-Raphson method.

Example 2.3.3 Equilibrium of reactions by Broyden method

In order to compute the reaction equilibrium studied in the previous examples we slightly change the main program of Example 2.3.2. Lines 800-810 are omitted and the following lines are replaced:

```
104 REM MERGE M32
212 LPRINT "BROYDEN METHOD" :LPRINT
214 GOSUB 3200
```

The part of the output that shows the iterations is as follows.

```
BROYDEN METHOD

IT=  0 x(1)=0.10000E+01 x(2)=0.10000E+00 f(1)=-.47237E+01 f(2)=-.16773E+01
IT=  0 x(1)=0.10001E+01 x(2)=0.10000E+00 f(1)=-.47232E+01 f(2)=-.16772E+01
IT=  0 x(1)=0.10001E+01 x(2)=0.10010E+00 f(1)=-.47232E+01 f(2)=-.16760E+01
IT=  1 x(1)=0.19413E+01 x(2)=0.21025E+00 f(1)=0.10944E+01 f(2)=-.11599E+00
IT=  2 x(1)=0.17614E+01 x(2)=0.22388E+00 f(1)=-.79736E+00 f(2)=-.20486E+00
IT=  3 x(1)=0.18370E+01 x(2)=0.23228E+00 f(1)=-.19684E+00 f(2)=-.82558E-01
IT=  4 x(1)=0.18618E+01 x(2)=0.23865E+00 f(1)=0.44468E-01 f(2)=-.18855E-01
IT=  5 x(1)=0.18572E+01 x(2)=0.24012E+00 f(1)=-.38958E-03 f(2)=-.14622E-01
IT=  6 x(1)=0.18571E+01 x(2)=0.24149E+00 f(1)=0.96369E-03 f(2)=-.63162E-02
IT=  7 x(1)=0.18569E+01 x(2)=0.24262E+00 f(1)=-.20742E-04 f(2)=0.28936E-03
IT=  8 x(1)=0.18569E+01 x(2)=0.24258E+00 f(1)=-.95367E-06 f(2)=0.34180E-05
IT=  9 x(1)=0.18569E+01 x(2)=0.24258E+00 f(1)=-.23842E-06 f(2)=-.83819E-07
```

Table 2.8 shows the computational efforts required to solve the test problem on reaction equilibrium by five different methods. For comparison successive approximation and damped successive approximation with a damping factor $c = 0.75$ are also included.

Table 2.8
Computational effort in different methods

Method	Number of iterations	Number of equivalent function evaluations
Successive approximation	24	24
Damped iteration ($c = 0.75$)	9	9
Wegstein	11	11
Newton-Raphson	5	15
Broyden	9	11

Exercises

- Derive the formulas (2.23) and (2.24) of the Wegstein iteration from the geometrical idea.
- Consider the $n \times n$ matrices A and $B = A + uv^T$, where u and v are n-vectors and assume that A and B are nonsingular. According to the Hausholder formula, exploited in Section 2.3.3, the inverse of B is given by

  $$B^{-1} = A^{-1} - A^{-1}uv^TA^{-1}/(1 + v^TA^{-1}u) .$$

  To prove this relationship, show that $B^{-1}B = I$.
- Solve the system $Ax - b = 0$ with a square, nonsingular A by the Broyden method and test whether or not the final matrix H will satisfy the equality $H = A^{-1}$.

2.4 MINIMIZATION IN MULTIDIMENSIONS

In this section we deal with the problem of finding the minimum of a function of more than one variables.

There are three major families of algorithms for minimization:

i) direct search methods, involving the evaluation of the function $f(x)$ only;

ii) gradient methods, based on the use of the gradient vector g of the elements $g_i = \partial f(x)/\partial x_i$, in addition to the values of $f(x)$;

iii) Newton type methods that require also the Hessian matrix H of the elements $[H]_{ij} = \partial^2 f/\partial x_i \partial x_j$, in addition to the gradient and function values.

The direct methods are not very efficient in terms of the number of function evaluations, but are robust, decreasing the objective function up to some extent in most cases. Requiring only one user supplied subroutine they are easy to use.

The most traditional and simplest gradient method is the steepest descent. Its idea is moving the current estimate $x^{(k)}$ to the next one $x^{(k+1)}$ by minimizing the objective function along the line from $x^{(k)}$ in the direction of the local negative gradient $[-g(x^{(k)})]$. Thus in each iteration we solve the one dimensional minimization problem

$$f[x^{(k)} - \lambda g(x^{(k)})] \longrightarrow \min_{\lambda \geq 0} , \qquad (2.39)$$

called directional search. The entire step is then repeated from the new estimate as many times as needed. Though the method will decrease the function value in each iteration, it will perform very small steps in most cases, particularly when going down a long, narrow valley. The convergence is of order 1, and the numerical efficiency is poor because of the effort required in the directional search. Though there are considerably improved versions, e.g., the conjugate gradient methods, we will not consider them here.

The Newton method will set the next estimate $x^{(k+1)}$ to the minimum point

$$x^{(k)} = x^{(k-1)} - [H(x^{(k-1)})]^{-1} g(x^{(k-1)}) \qquad (2.40)$$

of the local quadratic approximation of the function. Comparing (2.40) and (2.33) shows that we use essentially the same correction formula for function minimization and for solving a set of nonlinear equations. In (2.40), however, the matrix H is always symmetric, and at convergence (but not necessarily in intermediate iterations) it is positive definite. The properties of the method are also retained. The convergence is of order 2, and hence is rapid near the minimum point, but may be poor far from the solution. In addition, the number of equivalent function evaluations is high because of the need for evaluating H . The Newton method finds, however, the minimum of a positive definite quadratic function in a single step, and we will exploit this advantageous property in parameter estimation.

The quasi-Newton methods estimate the matrix $C = H^{-1}$ by updating a previous guess of C in each iteration using only the gradient vector. These methods are very close to the quasi-Newton methods of solving a system of nonlinear equations. The order of convergence is between 1 and 2, and the minimum of a positive definite quadratic function is found in a finite number of steps.

The algorithms using first or second derivatives are somewhat more powerful than those using only function values, but not always enough so as to compensate for the additional function evaluations. Nevertheless, if you can compute the the derivatives select a method that will use them. Therefore, the Newton method is the best choice if you are able to differentiate the function twice and and have a good initial guess. Replacing the derivative with finite differences is more controversial. If only the gradient vector is available in analytic form, the variable metric method due to Davidon, Fletcher and Powell usually dominates the finite difference version of the Newton method. If you do not have analytic derivatives at all, it is usually better to consider a direct search. From this latter family we describe here the simplex method due to Nelder and Mead.

2.4.1 Simplex method of Nelder and Mead

A simplex is the closed geometric figure consisting, in n dimensions, of $n+1$ vertices and all their interconnecting straight line segments. In two dimensions a simplex is a triangle, not necessarily a regular one. The search procedure due to Nelder and Mead (ref.18) is based on selecting a starting simplex represented by $n+1$ vertices $x^{(1)}, x^{(2)}, \ldots, x^{(n+1)}$ and then successively improving it.

To describe the method we introduce the following concepts.

- worst point $x^{(max)}$: $f(x^{(i)}) \le f(x^{(max)})$ for $i = 1, 2, \ldots, n+1$
- best point $x^{(min)}$: $f(x^{(i)}) \ge f(x^{(min)})$ for $i = 1, 2, \ldots, n+1$
- centroid $\bar{x} = \frac{1}{n}\left[\left(\sum_{i=1}^{n+1} x^{(i)}\right) - x^{(max)}\right]$.

Notice that the centroid excludes the worst point. In one step of the search the following candidates are investigated in order to replace the worst point:

- reflection point $x^{*} = 2\bar{x} - x^{(max)}$
- expansion point $x^{**} = x^{*} + (\bar{x} - x^{(max)})$
- contraction point $x^{***} = (x^{(max)} + \bar{x})/2$.

If none of these candidates is better than the worst point, the size of the simplex is reduced leaving only the best point in place:

- reduction operation: $x^{(i)}$ <-- $(x^{(i)} + x^{(min)})/2$ for all i.

Fig. 2.12 shows the initial simplex and the candidate points in two dimensions. The method is summarized in the logic diagram based on (ref. 19) and shown in Fig. 2.13.

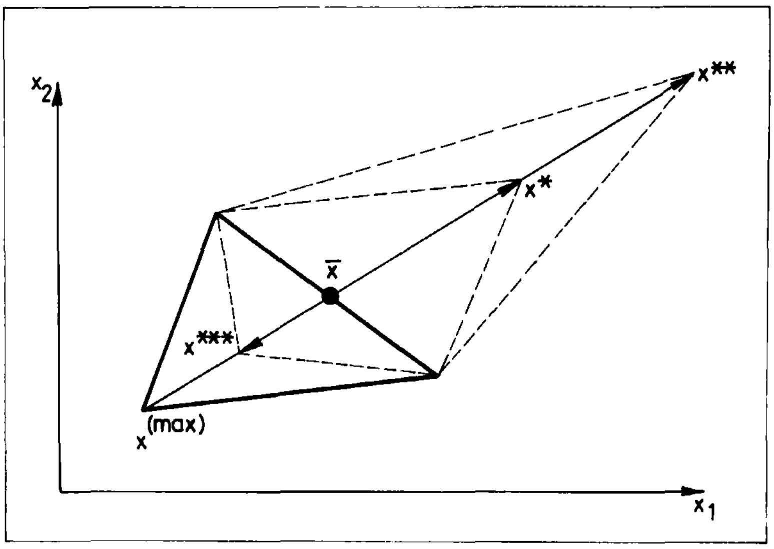


Fig. 2.12. A simplex in two dimensions

The iteration is stopped if the norm of the correction in the centroid and the distance between the best point and the centroid are both less than a small threshold EP .

The algorithm has great versatility to adopt the simplex to the local landscape of the function surface. It will elongate and take a large step if can do so, it will change direction on encountering a valley at an angle and it

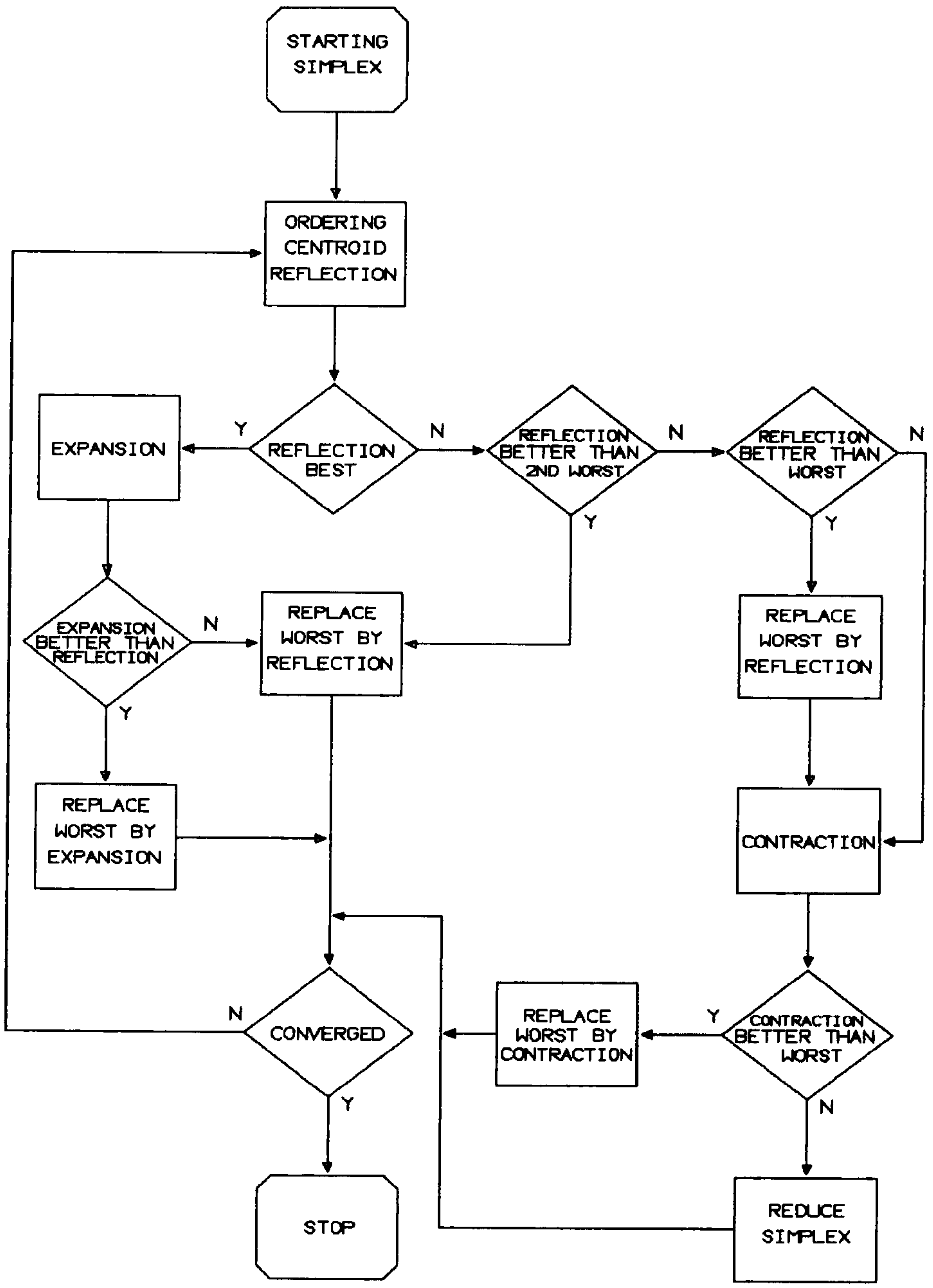

STARTING SIMPLEX
ORDERING CENTROID REFLECTION
EXPANSION
Y
REFLECTION BEST
N
REFLECTION BETTER THAN 2ND WORST
N
REFLECTION BETTER THAN WORST
N
Y
Y
EXPANSION BETTER THAN REFLECTION
N
REPLACE WORST BY REFLECTION
REPLACE WORST BY REFLECTION
Y
REPLACE WORST BY EXPANSION
CONTRACTION
N
CONVERGED
REPLACE WORST BY CONTRACTION
Y
CONTRACTION BETTER THAN WORST
Y
N
STOP
REDUCE SIMPLEX

Fig. 2.13. Logic diagram of the simplex method of Nelder and Mead

will contract in the neighbourhood of a minimum. All these steps provide us useful information on the form of the surface, though we usually have to pay the price by evaluating the function at many points. A considerable advantage of the method is that its code is very concise, as shown in the following module.

<u>Program module M34</u>

```
3400 REM ****************************************************
3402 REM * MINIMIZATION OF A FUNCTION OF SEVERAL VARIABLES *
3404 REM *              NELDER-MEAD METHOD                  *
3406 REM ****************************************************
3408 REM INPUT:
3410 REM     N       NUMBER OF VARIABLES
3412 REM   S(N+1,N)  INITIAL SIMPLEX COORDINATES (ROW BY ROW)
3414 REM     EP      THRESHOLD ON NORM OF THE CENTROID CORRECTION
3416 REM     IM      MAXIMUM NUMBER OF ITERATIONS
3418 REM OUTPUT:
3420 REM     ER      STATUS FLAG
3422 REM               0  SUCCESSFUL SEARCH
3424 REM               1  UNADMISSIBLE POINT IN INITIAL SIMPLEX
3426 REM               2  THRESHOLD NOT ATTAINED
3428 REM    X(N)     ESTIMATE OF THE MINIMUM POINT
3430 REM     F       FUNCTION VALUE AT THE FINAL ESTIMATE
3432 REM USER-SUPPLIED SUBROUTINES:
3434 REM    FROM LINE 900;  X(.) --> F    ( FUNCTION EVALUATION )
3436 REM AUXILIARY ARRAY:
3438 REM   R(3,N+1)
3440 REM ---------- INITIAL SIMPLEX EVALUATION
3442 ER=0
3444 FOR JN=1 TO N+1
3446  FOR I=1 TO N :X(I)=S(JN,I) :NEXT I
3448  GOSUB 900 :IF ER<>0 THEN ER=1 :GOTO 3562
3450  R(3,JN)=F
3452 NEXT JN
3454 REM ---------- ITERATION (BEST:KN, WORST:N1, NEXT WORST:N2)
3456 FOR IT=1 TO IM
3458  F=R(3,N+1) :FK=F :KN=N+1 :F1=F :N1=N+1 :F2=-1E+30
3460  FOR J=1 TO N
3462   F=R(3,J)
3464   IF F<FK THEN FK=F :KN=J  :GOTO 3470
3466   IF F>F2 AND F<=F1 THEN F2=F :N2=J :GOTO 3470
3468   IF F>F2 THEN F2=F1 :N2=N1: F1=F :N1=J
3470  NEXT J
3472  REM --------- CENTROID
3474  FOR I=1 TO N
3476   R(2,I)=R(1,I) :R(1,I)=0
3478   FOR J=1 TO N+1
3480    IF J<>N1 THEN R(1,I)=R(1,I)+S(J,I)/N
3482   NEXT J
3484  NEXT I
3486  REM --------- REFLECTION
3488  FOR I=1 TO N :X(I)=2*R(1,I)-S(N1,I) :NEXT I
3490  ER=0 :GOSUB 900 :IF ER<>0 THEN 3528
3492  IF F>FK THEN 3508
```

```
3494 REM --------- SUCCESSFUL STEP
3496 FOR I=1 TO N: S(N1,I)=X(I): NEXT I :R(3,N1)=F :FK=F :KN=N1
3498 REM --------- ---------- EXPANSION
3500 FOR I=1 TO N :X(I)=2*X(I)-R(1,I) :NEXT I
3502 ER=0 :GOSUB 900 :IF ER<>0 THEN 3528
3504 IF F<=FK THEN FOR I=1 TO N :S(N1,I)=X(I) :NEXT I :R(3,N1)=F
3506 GOTO 3548
3508 REM --------- NEUTRAL
3510 IF F>=F2 THEN 3514
3512 FOR I=1 TO N: S(N1,I)=X(I): NEXT I :R(3,N1)=F :GOTO 3548
3514 REM --------- UNSUCCESSFUL STEP
3516 IF F<F1 THEN FOR I=1 TO N: S(N1,I)=X(I): NEXT I :R(3,N1)=F :F1=F
3518 REM --------- ---------- CONTRACTION
3520 FOR I=1 TO N :X(I)=(R(1,I)+S(N1,I))/2 :NEXT I
3522 ER=0 :GOSUB 900 :IF ER<>0 THEN 3528
3524 IF F<FK THEN KN=N1 :FK=F
3526 IF F<F1 THEN FOR I=1 TO N: S(N1,I)=X(I): NEXT I :R(3,N1)=F :GOTO 3548
3528 REM --------- ---------- REDUCING SIMPLEX SIZE
3530 FOR J=1 TO N+1
3532  IF J<>KN THEN FOR I=1 TO N :S(J,I)=(S(J,I)+S(KN,I))/2 :NEXT I
3534 NEXT J
3536 FOR J=1 TO N+1
3538  IF J=KN THEN 3546
3540  FOR I=1 TO N :X(I)=S(J,I) :NEXT I
3542  GOSUB 900 :IF ER<>0 THEN ER=2 :GOTO 3562
3544  R(3,J)=F :IF F<FK THEN FK=F :KN=J
3546 NEXT J
3548 SX=0 :SK=0 :F=FK
3550 FOR I=1 TO N
3552  D=R(1,I)-R(2,I) :SX=SX+D*D :X(I)=S(KN,I) :D=X(I)-R(1,I) :SK=SK+D*D
3554 NEXT I
3556 IF SQR(SX)<=EP AND SQR(SK)<EP THEN 3562
3558 NEXT IT
3560 ER=2
3562 RETURN
3564 REM ########################################################
```

The function is calculated in a user routine starting at line 900. On the input you should define the N+1 vertices of the simplex. If you do not have a better idea, these can be generated by perturbing the elements of an initial guess one-by-one.

Example 2.4.1 Minimization of the Rosenbrock function by the simplex method of Nelder and Mead

The function

$$f(x) = 100(x_2 - x^2_1)^2 + (1 - x_1)^2 , \tag{2.41}$$

proposed by Rosenbrock (ref. 20), is a simple but famous test problem in nonlinear minimization, since it is far from easy to find its minimum at $x = (1,1)^T$ starting from the initial guess $x^{(o)} = (-1.2,1)^T$. In the following main program we regard this last point as one of the vertices, and generate the other two, perturbing the coordinates of $x^{(o)}$ by 0.01 in turn.

```
100 REM ------------------------------------------------------------
102 REM EX. 2.4.1 ROSENBROCK PROBLEM BY NELDER-MEAD METHOD
104 REM MERGE M34
200 REM ---------- PROBLEM SIZE
202 N=2
204 DIM X(N),S(N+1,N),R(3,N+1)
206 REM ---------- CONTROL PARAMETERS
208 EP=.00001 :IM=100
210 REM ---------- INITIAL SIMPLEX
212 X(1)=-1.2 :X(2)=.1
214 FOR J=1 TO N+1
216  FOR I=1 TO N
218   S(J,I)=X(I)-.01*(I=J)
220  NEXT I
222 NEXT J
224 V$=STRING$(60,"-")
226 LPRINT "SIMPLEX METHOD OF NELDER AND MEAD" :LPRINT
228 LPRINT V$
230 GOSUB 3400
232 LPRINT :LPRINT "MINIMUM";
234 LPRINT TAB(10);"x(1)=";X(1);TAB(25);"x(2)=";X(2);TAB(40);"F=";F
236 LPRINT :LPRINT V$ :LPRINT
238 STOP
900 REM ---------- FUNCTION EVALUATION
902 F=100*(X(2)-X(1)^2)^2+(1-X(1))^2
904 LPRINT USING"IT=### x(1)=#.######^^^^ x(2)=#.######^^^^ ";IT,X(1),X(2);
906 LPRINT USING"F=#.#####^^^^";F
908 RETURN
```

The shortened iteration history is as follows.

```
SIMPLEX METHOD OF NELDER AND MEAD

------------------------------------------------------------
IT=  0 x(1)=-.119000E+01 x(2)=0.100000E+00 F=0.17801E+03
IT=  0 x(1)=-.120000E+01 x(2)=0.110000E+00 F=0.18173E+03
IT=  0 x(1)=-.120000E+01 x(2)=0.100000E+00 F=0.18440E+03
IT=  1 x(1)=-.119000E+01 x(2)=0.110000E+00 F=0.17539E+03
IT=  1 x(1)=-.118500E+01 x(2)=0.115000E+00 F=0.17098E+03
.
.
.
IT= 50 x(1)=0.975553E+00 x(2)=0.942696E+00 F=0.87133E-02
IT= 51 x(1)=0.101253E+01 x(2)=0.101498E+01 F=0.10619E-01
IT= 51 x(1)=0.985004E+00 x(2)=0.963877E+00 F=0.42642E-02
.
.
.
IT= 83 x(1)=0.999987E+00 x(2)=0.999977E+00 F=0.67269E-09
IT= 83 x(1)=0.999998E+00 x(2)=0.999995E+00 F=0.55511E-10
IT= 84 x(1)=0.100001E+01 x(2)=0.100002E+01 F=0.88448E-10
IT= 84 x(1)=0.100000E+01 x(2)=0.100001E+01 F=0.33879E-10
IT= 85 x(1)=0.100000E+01 x(2)=0.100000E+01 F=0.53944E-10

MINIMUM  x(1)= 1.000003 x(2)= 1.000007 F= 3.387868E-11
------------------------------------------------------------
```

2.4.2 Davidon-Fletcher-Powell method

The method (also called variable metric method, ref.21) is based on the correction formula

$$x^{(k+1)} = x^{(k)} - \lambda^{(k+1)}C^{(k+1)}g(x^{(k)}) , \qquad (2.42)$$

which differs from the Newton correcton (2.40) in the use of a current estimate $C^{(k+1)}$ of the inverse Hesse matrix $[H(x^{(k)})]^{-1}$. Furthermore, the step size $\lambda^{(k+1)}$ is found by directional search, i.e., it is the solution of the one-dimensional problem

$$f[x^{(k)} - \lambda C^{(k+1)}g(x^{(k)})] \dashrightarrow \min_{\lambda \geq 0} . \qquad (2.43)$$

where $g(x^{(k)})$ is the gradient vector computed at $x^{(k)}$. At start $C^{(1)}$ is a symmetric, positive definite matrix. The usual choice is $C^{(1)} = I$, i.e., the identity matrix. It is then updated according to

$$C^{(k+1)} = C^{(k)} + \frac{\Delta x^{(k)}[\Delta x^{(k)}]^T}{[\Delta x^{(k)}]^T \Delta g^{(k)}} - \frac{[C^{(k)}\Delta g^{(k)}][C^{(k)}\Delta g^{(k)}]^T}{[\Delta g^{(k)}]^T C^{(k)} \Delta g^{(k)}} \qquad (2.44)$$

where $\Delta x^{(k)} = x^{(k)} - x^{(k-1)}$ and $\Delta g^{(k)} = g^{(k)} - g^{(k-1)}$. Comparing (2.44) to the updating formula (2.38) of the Broyden method shows the similarity of the underlying ideas. The rank of the correction matrix in (2.44) equals, however, two, and hence this algorithm is a rank 2 method. Furthermore, starting with a positive definite symmetric matrix $C^{(k)}$ it remains symmetric and positive definite in all iterations.

The following module strictly follows the algorithmic ideas of the FLEPOMIN program (ref. 22), the original implementation of the Davidon-Fletcher-Powell algorithm.

By the simple initial guess $C^{(1)} = I$, the first steps may be too large, and hence it is advisable to scale the variables by transformations bringing their value close to one. You will need two user subroutines for the module M36. The first one starts at line 900 and is the usual function evaluation. The second one, starting at line 800, computes the gradient vector and stores its elements in array G.

Program module M36

```
3600 REM ####################################################
3602 REM # MINIMIZATION OF A FUNCTION OF SEVERAL VARIABLES #
3604 REM #         DAVIDON-FLETCHER-POWELL METHOD           #
3606 REM ####################################################
3608 REM INPUT:
3610 REM      N      NUMBER OF VARIABLES
3612 REM     X(N)    STARTING POINT
3614 REM      EP     THRESHOLD ON STEP LENGTH
3616 REM      IM     MAXIMUM NUMBER OF ITERATIONS
3618 REM OUTPUT:
3620 REM      ER     STATUS FLAG
3622 REM                0  SUCCESSFUL SEARCH
3624 REM                1  UNADMISSIBLE STARTING POINT
3626 REM                2  ESTIMATE  C IS NOT POSITIVE DEFINITE
3628 REM                3  UNIDIRECTIONAL SEARCH FAILED
3630 REM                4  THRESHOLD NOT ATTAINED
3632 REM     X(N)    ESTIMATE OF THE MINIMUM POINT
3634 REM      F      FUNCTION VALUE AT THE FINAL ESTIMATE
3636 REM     G(N)    GRADIENT VECTOR AT THE FINAL ESTIMATE
3638 REM    C(N,N)   ESTIMATE OF THE INVERSE HESSIAN MATRIX
3640 REM USER-SUPPLIED SUBROUTINES:
3642 REM     FROM LINE 900;  X(.) --> F    ( FUNCTION EVALUATION )
3644 REM     FROM LINE 800;  X(.) --> G(.) ( GRADIENT VECTOR EVALUATION )
3646 REM AUXILIARY ARRAY:
3648 REM    R(3,N)
3650 ER=0 :GOSUB 900 :IF ER<>0 THEN ER=1 :GOTO 3792
3652 GOSUB 800
3654 REM ---------- INITIALIZE C TO UNIT MATRIX
3656 FOR I=1 TO N
3658  FOR J=1 TO N :C(I,J)=-(I=J) :NEXT J
3660 NEXT I
3662 REM ---------- START OF ITERATION
3664 ST=1
3666 FOR IT=1 TO IM
3668  GB=0 :GA=0
3670  FOR I=1 TO N
3672   R(2,I)=X(I) :R(3,I)=G(I) :R=0
3674   FOR J=1 TO N :R=R-C(I,J)*G(J) :NEXT J :R(1,I)=R
3676   GB=GB+G(I)*R :GA=GA+G(I)*G(I)
3678  NEXT I
3680  FO=F :IF GA=0 THEN ER=0 :GOTO 3792
3682  REM ---------- DIRECTIONAL SEARCH ALONG S
3684  FB=F
3686  IF GB>0 THEN ER=2 :GOTO 3792
3688  SP=ST
3690  REM ---------- --------------- EXTRAPOLATE
3692  FA=FB :GA=GB
3694  FOR I=1 TO N :X(I)=X(I)+SP*R(1,I) :NEXT I
3696  ER=0 :GOSUB 900 :IF ER=0 THEN 3702
3698  FOR I=1 TO N :X(I)=X(I)-SP*R(1,I) :R(1,I)=.95*R(1,I) :NEXT I
3700  GOTO 3694
3702  GOSUB 800
3704  FB=F :GB=0
3706  FOR I=1 TO N :GB=GB+G(I)*R(1,I) :NEXT I
3708  IF GB<0 AND FB<FA THEN SP=4*SP :ST=4*ST :GOTO 3684
```

```
3710  REM ---------- --------------- INTERPOLATE
3712  Z=3*(FA-FB)/SP+GA+GB
3714  W=SQR(Z*Z-GA*GB)
3716  SL=SP*(GB+W-Z)/(GB-GA+2*W)
3718  FOR I=1 TO N :X(I)=X(I)-SL*R(1,I) :NEXT I
3720  GOSUB 900 :GOSUB 800
3722  IF F<=FA*1.00001  AND F<=FB*1.00001 THEN 3742
3724  ST=ST/4
3726  IF FB>=FA THEN 3732
3728  FOR I=1 TO N :X(I)=X(I)+SL*R(1,I) :NEXT I
3730  F=FB :GOTO 3742
3732  GB=0
3734  FOR I=1 TO N :GB=GB+G(I)*R(1,I) :NEXT I
3736  IF IT<N THEN 3740
3738  IF GB<0 AND ST<.000001 THEN ER=3: GOTO 3792
3740  FB=F :SP=SP-SL :IF SP>0 THEN 3712
3742  REM ---------- END OF UNIDIRECTIONAL SEARCH
3744  GOSUB 3752
3746  IF IT>=N AND (SQR(SS)<EP OR SQR(SI)<EP) OR F>=FO THEN 3792
3748 NEXT IT
3750 ER=4 :GOTO 3792
3752 REM ---------- UPDATE C
3754  SG=0 :SS=0 :SI=0
3756  FOR I=1 TO N
3758   R(2,I)=X(I)-R(2,I) :R(3,I)=G(I)-R(3,I)
3760   SG=SG+R(2,I)*R(3,I) :SS=SS+R(1,I)*R(1,I) :SI=SI+R(2,I)*R(2,I)
3762  NEXT I
3764  GH=0
3766  FOR I=1 TO N
3768   S=0 :FOR J=1 TO N :S=S+C(I,J)*R(3,J) :NEXT J
3770   R(1,I)=S :GH=GH+S*R(3,I)
3772  NEXT I
3774  IF SG=0 OR GH=0 THEN RETURN
3776  FOR I=1 TO N
3778   FOR J=1 TO I
3780    C(I,J)=C(I,J)+R(2,I)*R(2,J)/SG-R(1,I)*R(1,J)/GH
3782    C(J,I)=C(I,J)
3784   NEXT J
3786  NEXT I
3788  RETURN
3790 REM ---------- END OF UPDATING C
3792 RETURN
3794 REM ****************************************************
```

Example 2.4.2 Minimization of the Rosenbrock function by Davidon-Fletcher-Powell method

The initial guess is the one used in the previous example. The main program and the shortened output are as follows.

```
100 REM ------------------------------------------------------------
102 REM EX. 2.4.2  ROSENBROCK PROBLEM BY DAVIDON-FLETCHER-POWELL M.
104 REM MERGE M36
200 REM ---------- PROBLEM SIZE
202 N=2
204 DIM X(N),G(N),C(N,N),R(3,N)
```

```
206 REM ---------- CONTROL PARAMETERS
208 EP=.00001 :IM=100
210 REM ---------- INITIAL POINT
212 X(1)=-1.2 :X(2)=.1
214 V$=STRING$(60,"-")
216 LPRINT "DAVIDON-FLETCHER-POWELL METHOD" :LPRINT
218 LPRINT V$
220 GOSUB 3600
222 LPRINT :LPRINT "MINIMUM";
224 LPRINT TAB(10);"x(1)=";X(1);TAB(25);"x(2)=";X(2);TAB(40);"F=";F
226 LPRINT :LPRINT V$ :LPRINT
228 IF ER<>0 THEN LPRINT "STATUS FLAG:";ER
230 STOP
800 REM ---------- GRADIENT EVALUATION
802 G(1)=-400*(X(2)-X(1)^2)*X(1)-2*(1-X(1))
804 G(2)= 200*(X(2)-X(1)^2)
806 RETURN
900 REM ---------- FUNCTION EVALUATION
902 F=100*(X(2)-X(1)^2)^2+(1-X(1))^2
904 LPRINT USING"IT=### x(1)=#.#####^^^^ x(2)=#.#####^^^^ ";IT,X(1),X(2);
906 LPRINT USING"F=#.#####^^^^";F
908 RETURN
```

```
DAVIDON-FLETCHER-POWELL METHOD

------------------------------------------------------------
IT=  0 x(1)=-.12000E+01 x(2)=0.10000E+00 F=0.18440E+03
IT=  1 x(1)=0.64640E+03 x(2)=0.26810E+03 F=0.17436E+14
IT=  1 x(1)=0.21560E+03 x(2)=0.89821E+02 F=0.21525E+12
IT=  1 x(1)=0.72002E+02 x(2)=0.30393E+02 F=0.26562E+10
IT=  1 x(1)=0.24126E+02 x(2)=0.10581E+02 F=0.32662E+08
IT=  1 x(1)=0.81463E+01 x(2)=0.39678E+01 F=0.38935E+06
IT=  1 x(1)=0.27649E+01 x(2)=0.17408E+01 F=0.34887E+04
IT=  1 x(1)=0.82338E+00 x(2)=0.93735E+00 F=0.67596E+01
IT=  2 x(1)=0.86586E+00 x(2)=0.86315E+00 F=0.13048E+01
IT=  2 x(1)=0.10358E+01 x(2)=0.56633E+00 F=0.25653E+02
IT=  2 x(1)=0.89828E+00 x(2)=0.80650E+00 F=0.10364E-01
.
.
.
IT= 12 x(1)=0.97897E+00 x(2)=0.96692E+00 F=0.77253E-02
IT= 12 x(1)=0.99973E+00 x(2)=0.99946E+00 F=0.75976E-07
IT= 13 x(1)=0.10139E+01 x(2)=0.10280E+01 F=0.19362E-03
IT= 13 x(1)=0.10000E+01 x(2)=0.10000E+01 F=0.28422E-10
IT= 14 x(1)=0.99970E+00 x(2)=0.99939E+00 F=0.11266E-06
IT= 14 x(1)=0.10000E+01 x(2)=0.10000E+01 F=0.00000E+00

MINIMUM  x(1)= 1       x(2)= 1       F= 0
------------------------------------------------------------
```

The method needed 44 function evaluations, almost four times less than the simplex method of Nelder and Mead.

We noticed that in one dimension it may be advantageous to seek the root of the equation $f'(x) = 0$ instead of the minimum of a differentiable function $f(x)$. This trick rarely gives you any good in multidimensions. First, as we emphasised, solving a system of nonlinear equations is more difficult than sliding downhill on a single surface in minimization, where you can always measure your progress. Second, you may bring in several roots that are not minimum points. These problems raise another question. If solving system of equations is so hard, why not to replace it by minimization of the function $g = \mathbf{f}^T\mathbf{f}$ in all cases? Indeed, the function g is positive semidefinite, and has a global minimum of zero exactly at all solutions of the original set of equations. Unfortunately, in multidimensions this trick does rarely work either. You must be prepared to have several local minima of the function g, and each local minimum is a trap for the minimization techniques. Therefore equation solving and minimization are completely different problems in multidimensions, in spite of their algorithmic similarities.

Exercises

- Solve the problem in Example 2.3.1 by minimization.

- Find the minimum of the Rosenbrock function by solving the nonlinear equations $\partial f(\mathbf{x})/\partial \mathbf{x} = \mathbf{0}$.

- Find the minimum of the quadratic function

  $f(\mathbf{x}) = (1/2)(\mathbf{x} - \mathbf{b})^T\mathbf{A}(\mathbf{x} - \mathbf{b})$

  by the Davidon-Fletcher-Powell method selecting $\mathbf{A}$ to be an $n \times n$ symmetric, diagonally dominant matrix with positive diagonal elements. Check if the equality $\mathbf{C} = \mathbf{A}^{-1}$ holds at convergence.

2.5 APPLICATIONS AND FURTHER PROBLEMS

2.5.1 Analytic solution of the Michaelis-Menten kinetic equation

The simplest mechanism of enzyme reactions is of the form

$$E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \underset{k_4}{\overset{k_3}{\rightleftharpoons}} E + P , \qquad (2.45)$$

where E, S, P and ES denote the enzyme, the substrate, the product and the enzyme-substrate complex, respectively (ref. 23). The reaction rates are

$$r_1 = -\frac{d[S]}{dt} = k_1[E][S] - k_2[ES]$$
$$r_2 = \frac{d[P]}{dt} = k_3[ES] - k_4[E][P], \qquad (2.46)$$

where k_1, k_2, k_3 and k_4 are the rate coefficients and $[.]$ denotes concentration of the corresponding species. As we will discuss in Section 5.4, the quasi steady state approximation for species $[ES]$ gives an excellent approximation of the global reaction rate for most enzyme reactions:

$$r = r_1 = r_2 = \frac{V_S[S]/K_S - V_P[P]/K_p}{1 + [S]/K_S + [P]/K_P}, \qquad (2.47)$$

where

$$V_S = k_3[E]_0, \qquad V_P = k_2[E]_0, \qquad K_S = (k_2 + k_3)/k_1, \qquad K_P = (k_2 + k_3)/k_4$$

are called Michaelis-Menten parameters.

Introducing the reaction extent $x = [S]_0 - [S] = [P]$, corresponding to the initial condition $[P]_0 = 0$, equation (2.47) is of the form

$$\frac{dx}{dt} = \frac{A + Bx}{C + Dx} \qquad (2.48)$$

where

$$A = V_S[S]_0/K_S, \quad B = -(V_S/K_S + V_P/K_P), \quad C = 1 + [S]_0/K_S, \quad D = 1/K_P - 1/K_P .$$

The differential equation (2.48) is separable, and by integrating the rational function on its right-hand side the solution is given by

$$t = \frac{D}{B}x + \left[\frac{C}{B} - \frac{AD}{B^2}\right] \log\left[\frac{A + Bx}{A}\right] . \qquad (2.49)$$

We want to use (2.49) to calculate the concentration $[P]$ at $t = 180$ s in the enzyme-catalysed hydrolysis of fumarate, with the initial enzyme concentration $[E]_0 = 5\times10^{-4}$ mmol/m^3 and substrate concentration $[S]_0 = 40$ mmol/m^3. The Michaelis-Menten parameters for this reaction are $V_S = 0.65$ mmol m^{-3} s^{-1}, $K_S = 3.9$ mmol m^{-3} , $V_P = 0.4$ mmol m^{-3} s^{-1} and $K_P = 10.3$ mmol m^{-3}.

By (2.48) $dx/dt = 0$ implies $A + Bx = 0$, and the equilibrium reaction coordinate $-A/B$ clearly is an upper bound on the solution. Use the methods of Section 2.1 to verify the solution $[P] = 32.268$ mmol m^{-3}.

2.5.2 Solution equilibria

In textbooks of computational chemistry you will invariably find examples calculating the $pH = -\lg\{[H^+]/(mol/l)\}$ in weak acid - strong base or strong acid - weak base solutions. Indeed, these examples are important in the study of acids, bases and of complex formation, as well as for calculating titration curves. Following (ref. 24) we consider here the aquous solution that contains a weak tribasic acid H_3A and its sodium salts NaH_2A, Na_2HA and Na_3A in known initial concentrations. The dissociation reactions and equilibrium relations are given as follows.

$$H_3 \rightleftarrows H^+ + H_2A^- \qquad K_1 = \frac{[H^+][H_2A^-]}{[H_3A]} \; ; \tag{2.50}$$

$$H_2A^- \rightleftarrows H^+ + HA^{2-} \qquad K_2 = \frac{[H^+][HA^{2-}]}{[H_2A^-]} \; ; \tag{2.51}$$

$$HA^{2-} \rightleftarrows H^+ + HA^{3-} \qquad K_3 = \frac{[H^+][A^{3-}]}{[HA^{2-}]} \; ; \tag{2.52}$$

$$H_2O \rightleftarrows H^+ + OH^- \qquad K_v = [H^+][OH^-] \; . \tag{2.53}$$

Further constraints are the mass balance equation for the total acid concentration C_A

$$C_A = [H_3A] + [H_2A^-] + [HA^{2-}] + [A^{3-}] \; , \tag{2.54}$$

and the charge balance equation

$$[H^+] + [Na^+] = [H_2A^-] + 2[HA^{2-}] + 3[A^{3-}] + [OH^-] \; . \tag{2.55}$$

From the initial conditions

$$C_A = [H_3A]_o + [NaH_2A]_o + [Na_2HA]_o + [Na_3A]_o$$

and

$$[Na^+] = [NaH_2A]_o + 2[Na_2HA]_o + 3[Na_3A]_o \; ,$$

where the initial concentrations are denotedf by subscript o . To calculate the hydrogen ion concentration we express $[H_3A]$, $[H_2A^-]$ and $[HA^{2-}]$ from the equilibrium relations (2.50), (2.51) and (2.52) using only $[A^{3-}]$ and $[H^+]$ and substitute these expressions into (2.54). As a result we obtain the expressions

$$[H_3A] = [H^+]^3C_A/D, \qquad [H_2A^-] = K_1[H^+]^2C_A/D,$$
$$[HA^{2-}] = K_1K_2[H^+]C_A/D, \qquad [HA^{3-}] = K_1K_2K_3C_A/D, \qquad (2.56)$$

where

$$D = [H^+]^3 + K_1[H^+]^2 + K_1K_2[H^+] + K_1K_2K_3.$$

Substituting (2.56) into the charge balance equation and using (2.53) to eliminate $[OH^-]$ we obtain the fith-order polynomial equation in $[H^+]$:

$$[H^+]^5 + a_1[H^+]^4 + a_2[H^+]^3 + a_3[H^+]^2 + a_4[H^+] + a_5 = 0 , \qquad (2.57)$$

where

$$a_1 = K_1 + [Na^+]$$

$$a_2 = K_1(K_2 + [Na^+] - C_A) - K_V$$

$$a_3 = K_1[K_2(K_3 + [Na^+] - 2C_A) - K_V]$$

$$a_4 = K_1K_2[K_3([Na^+] - 3C_A) - K_V]$$

$$a_5 = -K_1K_2K_3K_V .$$

We want to calculate the equilibrium pH of the solution if its initial composition is given by $[H_3PO_4]_o = 1$ mol/l and $[Na_3PO_4]_o = 1$ mol/l . The ten based logarithms of the dissociation constants are:

$\lg[K_1/(\mathrm{mol/l})] = -2.15$, $\lg[K_2/(\mathrm{mol/l})] = -7.21$, $\lg[K_1/(\mathrm{mol/l})] = -12.36$

and $K_V = 10^{-14}\ \mathrm{mol}^2\ \mathrm{l}^{-2}$.

Equation (2.57) has five, real or complex roots. From chemistry, however, we know a good starting guess for the pH (it is slightly above 7). Using this information we can easily find the solution applying any of the methods of Section 2.1. For extreme initial concentrations it might be necessary to scale the variable, e.g., by introducing $x = 10^7[H^+]$. We note that there are special methods for finding all roots of a polynomial equation, see e.g., (ref. 12), but then you should "polish" the selected real root for higher accuracy, e.g., by Newton-Raphson iteration. With the a priori information available in scientific applications, you rarely need such special methods.

Exercise

- It is interesting to try to solve the original system of equations (2.50 - 2.55) in six unknowns using one of the methods of Section 2.3. The computational effort is certainly higher and you should select the starting values of the variables very carefully.

2.5.3 Liquid-liquid equilibrium calculation

The determination of the equilibrium composition in two contacting liquid phases has great significance for extraction process design. To obtain a system of equations we need a thermodynamic model for the excess Gibbs free energy $\Delta G^E/(RT)$. We chose the 3-suffix Margules equation (refs. 25-26) expressing the excess Gibbs free energy of a C component liquid as a function of the mole fractions $z_1, z_2, \ldots, z_C$:

$$\frac{\Delta G^E}{RT} = \sum_{k=1}^{C} \sum_{\substack{l=1 \\ l \neq k}}^{C} (z_k)^2 z_l A_{lk} + \sum_{k=1}^{C} \sum_{l=k+1}^{C} \sum_{m=l+1}^{C} z_k z_l z_m A^*_{klm} \qquad (2.58)$$

where

$$A^*_{klm} = (A_{kl} + A_{km} + A_{lk} + A_{lm} + A_{mk} + A_{ml})/2$$

and the table of coefficients A_{kl}, which can be determined from infinite dilution activity data, is supposed to be known. The activity coefficient γ_i of the i-th component can be computed from the thermodynamic relation

$$\log \gamma_i = \frac{\partial}{\partial n_i}\left[\frac{n\Delta G^E}{RT}\right] = \frac{\partial}{\partial z_i}\left[\frac{\Delta G^E}{RT}\right] - 2\left[\frac{\Delta G^E}{RT}\right] \qquad (2.59)$$

where n_i is the mole number of the i-th component, n is the total mole number and the second equality holds only because (2.58) is a cubic expression of the mole fractions.

Let us denote by superscript R the raffinate phase and by superscript E the extract phase. In equilibrium the distribution ratio $z^E_i/z^R_i = K_i$ can be calculated from the activity coefficients (or rather from their logarithms) as

$$K_i = \gamma^R_i/\gamma^E_i = \exp(\log \gamma^R_i - \log \gamma^E_i) . \qquad (2.60)$$

Equations (2.58-2.60) form the thermodynamic base for liquid equilibrium calculations.

Suppose we add 6.6 mol Furfural (1) to the mixture of 0.2 mol n-Heptane (2) and 0.8 mol Cyclohexane (3). We want to determine the composition of the extract phase rich in Furfural and of the raffinate phase poor in Furfural. The Margules coefficients (ref. 26) are shown in Table 2.9.

Table 2.9
A_{ij} coefficients for the Furfural - n-Heptane - Cyclohexane system

i\j	1	2	3
1	-	3.16892	3.0975
2	3.1252	-	0
3	2.3399	0	-

We have eight unknowns: x_1 - the raffinate in moles, x_2 - the extract in moles, x_3 - the mole fraction of Furfural in the raffinate phase, x_4 - the mole fraction of n-Heptane in the raffinate phase, x_5 - the mole fraction of Cyclohexane in the raffinate phase; x_6, x_7 and x_8 - the mole fractions for the extract phase in the same order. The eight equations are as follows.
Overall material balance

$$x_1 + x_2 - 7.6 = 0 \; ; \tag{2.61}$$

mole fraction summation for the raffinate phase

$$x_3 + x_4 + x_5 - 1 = 0 \; ; \tag{2.62}$$

material balances for each component

$$\begin{aligned} &x_1x_3 + x_2x_6 - 6.6 = 0 \\ &x_1x_4 + x_2x_7 - 0.2 = 0 \\ &x_1x_5 + x_2x_8 - 0.8 = 0 \; ; \end{aligned} \tag{2.63}$$

and equilibrium conditions for each component

$$\begin{aligned} &x_6 - K_1x_3 = 0 \\ &x_7 - K_2x_4 = 0 \\ &x_8 - K_3x_5 = 0 \; . \end{aligned} \tag{2.64}$$

In the following main program equations (2.61-2.64) are solved using the Broyden method. The distribution coefficients are computed from equations (2.58 - 2.60) written for $C = 3$ components. The starting values used are very simple, for the extract phase we start from poor Furfural and for the raffinate phase from the original n-Heptane - Cyclohexane mixture. Negative mole numbers and mole fractions are not allowed.

```
100 REM -----------------------------------------------------------
102 REM EX. 2.5.3 LIQUID-LIQUID EQUILIBRIUM BY BROYDEN METHOD
104 REM MERGE M32
106 REM ---------- DATA
108 REM  MARGULES COEFFICIENTS (FURFURAL, N-HEPTANE, CYCLOHEXANE)
110 A12=3.16892 :A13=3.0975 :A21=3.1252 :A31=2.3393 :A23=0 :A32=0
112 A123=(A12+A21+A13+A31+A23+A32)/2
200 REM ---------- PROBLEM SIZE AND CONTROL PARAMETERS
202 N=8 :IM=30 :E1=.000001 :E2=.000001
204 DIM X(N),F(N),H(N,N),R(3,N)
```

```
206 REM ---------- STARTING VALUES
208 REM  RAFFINATE (MOL) AND EXTRACT (MOL)
210 X(1)=1 :X(2)=6.6
212 REM  RAFFINATE MOLE FRACTIONS AND EXTRACT MOLE FRACTIONS
214 X(3)=0 :X(4)=.2 :X(5)=.8 :X(6)=1 :X(7)=0 :X(8)=0
216 REM ---------- CALL MODULE
218 V$=STRING$(53,"-")
220 LPRINT "BROYDEN METHOD" :LPRINT
222 GOSUB 3200
224 LPRINT :LPRINT V$
226 LPRINT ,"RAFFINATE                  EXTRACT"
228 F$=   "MOLES      #.##### (##.###%)          #.##### (##.###%)"
230 LPRINT USING F$; X(1), X(1)/(X(1)+X(2))*100,X(2),X(2)/(X(1)+X(2))*100
234 LPRINT USING F$; 100*X(3),100*X(6)
236 F$=   "N-HEPTANE      ##.###%                       ##.###%"
238 LPRINT USING F$; 100*X(4),100*X(7)
240 F$=   "CYCLOHEXAN     ##.###%                       ##.###%"
242 LPRINT USING F$; 100*X(5),100*X(8)
244 LPRINT V$
246 STOP
900 REM ---------- F(X)
902 ER=0 :FOR IE=1 TO N: ER=ER-(X(IE)<0):NEXT IE :IF ER>0 THEN RETURN
904 REM DISTRIBUTION COEFFICIENTS FROM LOG ACTIVITY COEFFICIENTS
906 Z1=X(3) :Z2=X(4) :Z3=X(5) :GOSUB 950 :K1=L1 :K2=L2 :K3=L3
908 Z1=X(6) :Z2=X(7) :Z3=X(8) :GOSUB 950
910 K1=EXP(K1-L1) :K2=EXP(K2-L2) :K3=EXP(K3-L3)
912 REM EQUATIONS
914 F(1)=X(1)+X(2)-7.6             :REM MASS BALANCE
916 F(2)=X(3)+X(4)+X(5)-1          :REM MOLE FRACTION SUMMATION FOR RAFFINATE
918 F(3)=X(1)*X(3)+X(2)*X(6)-6.6 :REM FURFURAL MASS BALANCE
920 F(4)=X(1)*X(4)+X(2)*X(7)-.2  :REM N-HEPTANE MASS BALANCE
922 F(5)=X(1)*X(5)+X(2)*X(8)-.8  :REM CYCLOHEXANE MASS BALANCE
924 F(6)=X(6)-K1*X(3)              :REM EQUILIBRIUM
926 F(7)=X(7)-K2*X(4)              :REM      "
928 F(8)=X(8)-K3*X(5)              :REM      "
930 LPRINT "IT=";IT;TAB(10)"R=";X(1)TAB(32)"MOLE FR:";X(3);X(4);X(5)
932 LPRINT TAB(10)         "E=";X(2)TAB(32)"        ";X(6);X(7);X(8)
934 FOR KF=1 TO 8:LPRINT ,"F("KF")=";F(KF):NEXT KF :LPRINT
938 RETURN
950 REM ---------- LOG ACTIVITY COEFFICIENTS FROM 3-SUFFIX MARGULES EQUATION
952 DG=Z1^2*(Z2*A21+Z3*A31)+Z2^2*(Z1*A12+Z3*A32)+Z3^2*(Z1*A13+Z2*A23)
954 DG=DG+Z1*Z2*Z3*A123
956 L1=2*Z1*(Z2*A21+Z3*A31)+Z2^2*A12+Z3^2*A13+Z2*Z3*A123-2*DG
958 L2=2*Z2*(Z1*A12+Z3*A32)+Z1^2*A21+Z3^2*A23+Z1*Z3*A123-2*DG
960 L3=2*Z3*(Z1*A13+Z2*A23)+Z1^2*A31+Z2^2*A32+Z1*Z2*A123-2*DG
962 RETURN
```

After 17 iterations we arrive at the equilibrium compositions.

```
-----------------------------------------------------
             RAFFINATE                 EXTRACT
MOLES     0.10066 ( 1.324%)        7.49934 (98.676%)
FURFURAL      6.896%                  87.915%
N-HEPTANE    27.688%                   2.295%
CYCLOHEXAN   65.416%                   9.790%
-----------------------------------------------------
```

Calculation of other types of phase equilibria, e.g., vapor-liquid equilibrium, are based on similar principles. If the same thermodynamic model is used for both phases, the method (whatever sophisticated it is) can easily converge to identical compositions in the two phases. This is called trivial solution and the best way to avoid it is to start from physically reasonable guesses of the compositions.

2.5.4 Minimization subject to linear equality constraints: chemical equilibrium composition in gas mixtures

If the n variable function $f(\mathbf{x})$ should be minimized subject to the $m < n$ independent linear constraints

$$\mathbf{A}\mathbf{x} = \mathbf{b} \ , \tag{2.65}$$

where $\mathbf{A}$ is a given $m \times n$ matrix and $\mathbf{b}$ is a given m-vector, then the method of Lagrange multipliers might be useful. We introduce the $n+m$ variable function

$$L(\mathbf{x};\boldsymbol{\lambda}) = f(\mathbf{x}) + (\mathbf{A}\mathbf{x}-\mathbf{b})^T\boldsymbol{\lambda} \ , \tag{2.66}$$

where $\boldsymbol{\lambda}$ is the m-vector of Lagrange multipliers. At the constrained minimum point the Lagrange function has zero partial derivatives with respect to all its variables, i.e.,

$$\mathbf{f}_x(\mathbf{x}) + \mathbf{A}^T\boldsymbol{\lambda} = \mathbf{0} \ , \tag{2.67}$$

$$\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \ , \tag{2.68}$$

where $\mathbf{f}_x(\mathbf{x})$ is the gradient vector of the objective function f. The set of equations (2.67-2.68) is solved by the Newton-Raphson method, linearizing (2.67) around a feasible initial guess $\mathbf{x}^o$ satisfying (2.68). Then the corrections $\Delta\mathbf{x}$ together with the multipliers $\boldsymbol{\lambda}$ can be obtained from the $n + m$ linear equations

$$\mathbf{f}_x(\mathbf{x}^o) + \mathbf{F}_{xx}(\mathbf{x}^o)\Delta\mathbf{x} + \mathbf{A}^T\boldsymbol{\lambda} = \mathbf{0} \ , \tag{2.69}$$

$$\mathbf{A}\Delta\mathbf{x} = \mathbf{0} \ , \tag{2.70}$$

where $\mathbf{F}_{xx}(\mathbf{x}^o)$ is the Hessian matrix of the objective function f computed at the point $\mathbf{x}^o$. The usual procedure of solving (2.69-2.70) is to add the corrections $\Delta\mathbf{x}$ to the initial guess $\mathbf{x}^o$ and repeat the iteration until convergence. An important property of the algorithm is that any new point

$$\mathbf{x} = \mathbf{x}^o + \xi\Delta\mathbf{x} \tag{2.71}$$

is also a feasible solution in the sense of (2.65), whatever the scalar ξ is. Consequently, we can make a reduced correction with $\xi < 1$, if the calculated correction is not acceptable.

Equations (2.69 - 2.70) can be solved by standard LU decomposition and backward substitution, but very often we can reduce the computational effort considerably by making use of the special structure of the equations. In the chemical equilibrium problem these ideas lead to a very concise algorithm.

If the temperature T and pressure P of a closed system are kept constant, the total Gibbs free energy is minimum in equilibrium. For an ideal gas mixture of NS species the Gibbs free energy is given by

$$G = \sum_{i=1}^{NS} n_i \left[g_i^o(T) + RT \log (P/P_u) + RT \log \left[n_i \Big/ \sum_{j=1}^{NS} n_j \right] \right] \tag{2.72}$$

where R is the universal gas constant, n_i is the number of moles of the i-th species, $g_i^o(T)$ is the standard molar Gibbs free energy of the i-th species corresponding to temperature T and standard pressure P_u. As discussed in Section 1.8.1, the mole numbers must satisfy the atom conservation constraints

$$\mathbf{An} = \mathbf{b} \ , \tag{2.73}$$

where **A** is the atom matrix with NA rows and NS columns, and **b** is the NA vector of the initial elemental abundances.

Knowing the standard molar Gibbs free energy values and giving an initial mole number vector the determination of the equilibrium composition consists of minimizing (2.72) subject to the linear constraints (2.73). The direct application of (2.69-2.70), however, would be rather complicated. In the RAND method (ref. 27) a function $f(x_1,x_2,\dots,x_{NS+1})$ of NS + 1 variables is minimized instead of the function $G(n_1,n_2,\dots,n_{NS})$ of NS variables, f being defined as

$$f(x_1,x_2,\dots,x_{NS+1}) = \sum_{i=1}^{NS} x_i f_i(x_1,x_2,\dots,x_{NS+1}) \ , \tag{2.74}$$

where

$$f_i(x_1,x_2,\dots,x_{NS+1}) = c_i + \log x_i - \log x_{NS+1}$$

$$c_i = [\ g_i^o(T) + RT \log (P/P_u)\ ] \ / \ (RT) \ ,$$

and the relations between the mole numbers n_i and the new variables x_i are given by

$$x_i = n_i, \quad (i = 1, 2, \ldots, NS) \; ; \qquad x_{NS+1} = \sum_{j=1}^{NS} n_j \; .$$

The minimum of (2.72) subject to (2.73) can also be found minimizing (2.74) subject to an extended set of constraints:

$$\begin{array}{lllll}
a_{11}x_1 & + a_{12}x_2 & + \ldots + a_{1,NS}x_{NS} & & = b_1 \\
a_{21}x_1 & + a_{22}x_2 & + \ldots + a_{2,NS}x_{NS} & & = b_2 \\
\cdot & & & & \\
\cdot & & & & \\
\cdot & & & & \\
a_{NA,1}x_1 & + a_{NA,2}x_2 & + \ldots + a_{NA,NS}x_{NS} & & = b_{NA} \\
x_1 & + x_2 & + \ldots + x_{NS} & - x_{NS+1} & = 0 \; .
\end{array} \qquad (2.75)$$

Due to the simple structure of the function f, its first and second partial derivatives are easy to compute

$$[f_x]_i = f_i \qquad (i=1,2,\ldots NS)$$

$$[f_x]_{NS+1} = -\left\{ \sum_{j=1}^{NS} x_j \right\} / x_{NS+1} = -1$$

$$[F_{xx}]_{ij} = 0 \qquad (i \neq j, \quad i,j \leq NS)$$

$$[F_{xx}]_{ii} = 1/x_i \qquad (i \leq NS)$$

$$[F_{xx}]_{NS+1,i} = [F_{xx}]_{i,NS+1} = -1/x_{NS+1} \qquad (i \leq NS)$$

$$[F_{xx}]_{NS+1,NS+1} = \left\{ \sum_{j=1}^{NS} x_j \right\} / x^2_{NS+1} = 1/x_{NS+1} \; .$$

With the above derivatives equations (2.69) are given by

$$f_i + \Delta x_i/x_i - \sum_{j=1}^{NA} \lambda_j a_{ji} - \lambda_{NA+1} = 0 \qquad (i=1,2,\ldots NS) \qquad (2.76)$$

$$\lambda_{NA+1} - \Delta x_{NS+1}/x_{NS+1} = 0 \; . \qquad (2.77)$$

From (2.76) and (2.77) the corrections Δx_1, Δx_2, ..., Δx_{NS+1} can be expressed as

$$\Delta x_i = x_i(-f_i + \lambda_{NA+1} + \sum_{j=1}^{NA} \lambda_j a_{ji}) \; , \qquad (i=1,2,\ldots NS) \qquad (2.78)$$

and

$$\Delta x_{NS+1} = x_{NS+1}\lambda_{NA+1} \ . \qquad (2.79)$$

Substituting (2.78(and (2.79) into the actual form of (2.70) we obtain the set of NA+1 linear equations in NA+1 unknowns with the coefficient matrix and right hand side vector as follows.

$$\begin{array}{llllll}
\lambda_1 & \lambda_2 & \cdots & \lambda_{NA} & \lambda_{NA+1} & \text{Right h. s.} \\
\hline
\sum_{i=1}^{NS} a_{1i}a_{1i}x_i & \sum_{i=1}^{NS} a_{1i}a_{2i}x_i & \cdots & \sum_{i=1}^{NS} a_{1i}a_{NA,i}x_i & \sum_{i=1}^{NS} a_{1i}x_i & \sum_{i=1}^{NS} a_{1i}f_ix_i \\
\sum_{i=1}^{NS} a_{2i}a_{1i}x_i & \sum_{i=1}^{NS} a_{2i}a_{2i}x_i & \cdots & \sum_{i=1}^{NS} a_{2i}a_{NA,i}x_i & \sum_{i=1}^{NS} a_{2i}x_i & \sum_{i=1}^{NS} a_{2i}f_ix_i \\
\vdots & & & & & \\
\sum_{i=1}^{NS} a_{NA,i}a_{1i}x_i & \sum_{i=1}^{NS} a_{NA,i}a_{2i}x_i & \cdots & \sum_{i=1}^{NS} a_{NA,i}a_{NA,i}x_i & \sum_{i=1}^{NS} a_{NA,i}x_i & \sum_{i=1}^{NS} a_{NA,i}f_ix_i \\
\sum_{i=1}^{NS} a_{1i}x_i & \sum_{i=1}^{NS} a_{2i}x_i & \cdots & \sum_{i=1}^{NS} a_{NA,i}x_i & 0 & \sum_{i=1}^{NS} f_ix_i \\
\hline
\end{array}$$

The solution of this matrix equation is used to compute the corrections (2.78-2.79). If the correction vector results in one or more zero or negative mole numbers, equation (2.71) is applied with ξ selected to give maximum 95% reduction in any mole number in one iteration step.

Notice that the number NA of atoms is usually small compared to the number NS of species, and hence the RAND algorithm is very effective in terms of computational effort. The rank of the atom matrix, however, must be equal to the number NA of atoms. At this point it is interesting to remark that instead of the atom matrix we can use a virtual atom matrix, i.e., the matrix of reaction invariant coefficients if the atom matrix is not available or we are interested in a restricted equilibrium. For details see Section 1.8.1.

The following main program is an implementation of the RAND algorithm. Its use is illustrated on the example of hydrazin combustion at $T = 3500$ K and $P = 5.17\times10^6$ Pa (ref. 27). The elemental abundances of hydrogen, nitrogen and oxygen satisfy the ratio H : N = H : O = 2 : 1 . The species present at equilibrium in physically meaningful quantities are listed in Table 2.10. The reduced molar Gibbs free energies c_i and the initial mole numbers of the species are also shown in the table. Since the total mole number is arbitrary, only the ratios of the initial elemental abundances are of interest when specifying the problem.

Table 2.10
Data for hydrazin combustion equilibrium calculation

No. i	Name	Formula	Reduced Gibbs free energy, c_i (-)	initial n^o_i (mol)
1	Hydrogen atom	H	-6.089	2
2	Hydrogen	H_2	-17.164	0
3	Water	H_2O	-34.054	0
4	Nitrogen atom	N	-5.914	1
5	Nitrogen	N_2	-24.721	0
6	NH radical	NH	-14.986	0
7	Nitrogen monoxid	NO	-24.1	0
8	Oxygen atom	O	-10.708	1
9	Oxygen	O_2	-26.662	0
10	Hydroxil radical	OH	-22.179	0

The input is accepted in chemical notation. The atom matrix is constructed in the "formula interpreter" section in lines 214-248 . Strictly speaking the function we minimize is not defined if any one of the mole numbers is zero. Since the vector of initial mole numbers serves also as an initial guess of the solution and it often contains zero elements, we add a small quantity to each n^o_i and correct for this bias after the first iteration.

```
100 REM ------------------------------------------------------------
101 REM EX. 2.5.4 CHEMICAL EQUILIBRIUM OF GASEOUS MIXTURES
102 REM MERGE M14,M15
104 REM INPUT DATA STRUCTURE:
106 REM        NS - NUMBER OF SPECIES
108 REM        FOR EACH SPECIES
110 REM             1) NAME
112 REM             2) FORMULA (e.g. Na2O - note second letter is lower case)
114 REM             3) (MOLAR GIBBS FREE ENERGY) / ( R * T )
116 REM                AT THE TEMPERATURE AND PRESSURE OF THE MIXTURE
118 REM             4) INITIAL NUMBER OF MOLES
```

```
120 REM ---------- DATA
122 DATA 10
124 DATA "H atom",      H,   -6.089, 2
126 DATA "hydrogen",    H2, -17.164, 0
128 DATA "water",       H2O,-34.054, 0
130 DATA "N atom",      N,   -5.914, 1
132 DATA "nitrogen",    N2, -24.721, 0
134 DATA "NH radical",  NH, -14.986, 0
136 DATA "N monoxid",   NO, -24.1,   0
138 DATA "O atom",      O,  -10.708, 1
140 DATA "oxygen",      O2, -26.662, 0
142 DATA "hydroxil",    OH, -22.179, 0
200 REM ---------- READ DATA
202 READ NS
204 DIM M(NS,10),C(NS),Z(NS),X(11),Y(NS+1),A(11,11),N$(NS),A$(10),K$(NS)
206 FOR I=1 TO NS
208  READ N$(I),K$(I),C(I),Z(I)
210 NEXT I
212 EP=.00001 :IM=20
214 REM ---------- FORMULA INTERPRETER
216 NA=0 :Z=0
218 FOR I=1 TO NS
220  L=LEN(K$(I)) :K=1
222  A$=MID$(K$(I),K,1)
224  C$="" :IF K=L THEN 234
226   B$=MID$(K$(I),K+1,1) :IF B$>="a" AND B$<="z" THEN A$=A$+B$ :K=K+1
228   IF K=L THEN 234
230   D$=MID$(K$(I),K+1,1) :IF D$>="A" AND D$<="Z" THEN 234
232   C$=C$+D$ :K=K+1 :GOTO 228
234  IF C$="" THEN C$="1"
236  FOR J=1 TO NA
238   IF A$(J)=A$ THEN 242
240  NEXT J :NA=NA+1 :A$(NA)=A$ :J=NA
242  M(I,J)=VAL(C$)
244  IF K<L THEN K=K+1 :GOTO 222
246  Z=Z+Z(I) :Y(I)=Z(I)
248 NEXT I
250 REM ---------- RAND ALGORITHM
252 Y=Z :E=Y*.0001
254 FOR I=1 TO  NS :Y(I)=Y(I)+E :NEXT I :Y=Y+E*NS
256 N=NA+1
258 REM ---------- START ITERATION
260 FOR IT=1 TO IM
262  FOR I=1 TO NS :F(I)=C(I)+LOG(Y(I)/Y) :NEXT I
264  FOR I=1 TO NA
266   FOR J=1 TO I
268     A=0
270     FOR K=1 TO NS :A=A+M(K,I)*M(K,J)*Y(K) :NEXT K
272     A(I,J)=A :A(J,I)=A
274   NEXT J
276   X=0 :FOR  K=1 TO NS :X=X+M(K,I)*F(K)*Y(K) :NEXT K :X(I)=X
278  NEXT I
280  FOR J=1 TO NA
282   A=0
284   FOR K=1 TO NS :A=A+M(K,J)*Y(K) :NEXT K
286   A(N,J)=A :A(J,N)=A
288  NEXT J
290  A(N,N)=0 :X=0
292  FOR K=1 TO NS :X=X+F(K)*Y(K) :NEXT K :X(N)=X
```

```
294  REM ---------- SOLVE SYSTEM OF LINEAR EQUATIONS
296  GOSUB 1400 :IF ER>0 THEN ER=2 :GOTO 346
298  GOSUB 1500
300  REM ---------- COMPUTE STEP
302  FOR I=1 TO NS
304   A=X(N) :FOR K=1 TO NA :A=A+X(K)*M(I,K) :NEXT K
306   D(I)=Y(I)*(A-F(I))
308  NEXT I
310  DT=Y*X(N)
312  REM ---------- SET XI TO ASSURE FEASIBILITY
314  XI=1
316  FOR I=1 TO NS
318   IF D(I)<0 AND XI*D(I)/Y(I)<-.95   THEN XI=-.95*Y(I)/D(I)
320  NEXT I
322  REM ---------- NEW VECTOR OF MOLE NUMBERS
324  D=0
326  FOR I=1 TO NS :Y(I)=Y(I)+XI*D(I) :D=D+D(I)*D(I) :NEXT I
328  Y=Y+XI*DT
330  IF IT>1 THEN 340
332  REM ---------- IF FIRST ITERATION THEN CORRECT
334  FOR I=1 TO NS
336   IF Y(I)>E THEN Y(I)=Y(I)-E :Y=Y-E
338  NEXT I
340  IF SQR(D)<=EP THEN ER=0  :GOTO 346
342 NEXT IT
344 ER=1
346 REM ----------- PRINT RESULTS
348 IF ER=1 THEN LPRINT "REQUIRED ACCURACY NOT ATTAINED"
350 IF ER=2 THEN LPRINT "RANK OF MATRIX IS LESS THAN NUMBER OF ATOMS"
352 LPRINT :LPRINT
354 V$=STRING$(66,"-") :F$="#.#####^^^^ "
356 LPRINT V$
358 LPRINT " I    NAME     FORMULA      C(I)      INITIAL   EQUILIBRIUM   %"
360 LPRINT V$
362 FOR I=1 TO NS
364  LPRINT I;TAB(5);N$(I);TAB(16);K$(I);TAB(24)" ";
366  LPRINT USING F$;C(I),Z(I),Y(I),:LPRINT USING "###.##";100*Y(I)/Y
368 NEXT I
370 LPRINT V$
372 LPRINT "SUM";TAB(36)" ";:LPRINT USING F$;Z,Y,:LPRINT "100.00"
374 LPRINT :LPRINT
376 V$=STRING$(33,"-")
378 LPRINT V$
380 LPRINT " I  ATOM      LAGRANGE MULTIPLIER"
382 LPRINT V$
384 FOR I=1 TO NA :LPRINT I;TAB(5);A$(I);TAB(15);X(I) :NEXT I
386 LPRINT V$
388 LPRINT :LPRINT
390 STOP
```

After ten iterations the convergence criterion is satisfied and the following results are printed:

```
-----------------------------------------------------------------
 I   NAME      FORMULA     C(I)       INITIAL   EQUILIBRIUM   %
-----------------------------------------------------------------
 1 H atom      H        -.60890E+01 0.20000E+01 0.40669E-01   2.48
 2 hydrogen    H2       -.17164E+02 0.00000E+00 0.14774E+00   9.02
 3 water       H2O      -.34054E+02 0.00000E+00 0.78315E+00  47.80
 4 N atom      N        -.59140E+01 0.10000E+01 0.14142E-02   0.09
 5 nitrogen    N2       -.24721E+02 0.00000E+00 0.48525E+00  29.62
 6 NH radical  NH       -.14986E+02 0.00000E+00 0.69318E-03   0.04
 7 N monoxid   NO       -.24100E+02 0.00000E+00 0.27399E-01   1.67
 8 O atom      O        -.10708E+02 0.10000E+01 0.17947E-01   1.10
 9 oxygen      O2       -.26662E+02 0.00000E+00 0.37312E-01   2.28
10 hydroxil    OH       -.22179E+02 0.00000E+00 0.96870E-01   5.91
-----------------------------------------------------------------
SUM                                 0.40000E+01 0.16385E+01 100.00


--------------------------------
 I  ATOM      LAGRANGE MULTIPLIER
--------------------------------
 1  H         -9.785044
 2  O         -15.22209
 3  N         -12.96893
--------------------------------
```

At convergence the Lagrange multipliers have some physical meaning similar to the "shadow prices" discussed in Section 1.2. The interested reader may consult (refs. 28-29) where nonideality, treatment of condensed phases, numerical difficulties and other problems are also discussed. Handbooks like (ref. 30) contain the necessary standard Gibbs free energy data for a great number of substances.

REFERENCES

1 J.M Ortega and W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.

2 J. Stoer and R. Bulirsch, Introduction to Numerical Analysis, Springer Verlag, New York, 1980.

3 J.F. Traub, Iterative Methods for the Solution of Equations, Chelsea Publ. Corp., New York, 1982.

4 D. Himmelblau, Applied Nonlinear Programming, McGraw-Hill, New York, 1972.

5 A.V. Fiacco and McGormick, Nonlinear Sequential Unconstrained Minimization Techniques, John Wiley, New York, 1968.

6 D. Peng and D.B. Robinson, A new two-constant equation of state, Ind. Eng. Chem., Fundam., 15 (1976) 59-64.

7 R.C. Reid, J.M. Prausnitz and T.K. Sherwood, The Properties of Gases and Liquids, 3rd ed., McGraw-Hill, New York, 1977.

8 N.B. Vargaftik, Handbook of Thermophysical Data of Gases and Liquids, Nauka, Moscow, 1972. (in Russian)

9 M.S. Gibaldi and D. Perrier, Pharmacokinetics, Marcel Dekker, New York, 1975.

10 B. Carnahan and J.O. Wilkes, Digital Computing and Numerical Methods, John Wiley, New York, 1973.

11 R.P. Brent, Algorithms for Minimization without Derivatives, Prentice Hall, Englewood Cliffs, N.J., 1973.
12 W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, Numerical Recipes: The Art of Scientific Computing, Cambridge University Press, Cambridge, 1986.
13 A.W. Westerberg, H.P. Hutchinson, R.L. Motard and P. Winter, Process Flowsheeting, University Press, Cambridge, 1979.
14 J.M. Wegstein, Accelerating convergence of iterative processes. Communications of the ACM 1, No.6 (1958) 6-9.
15 D.A.H. Jacobs (Editor), The State of the Art in Numerical Analysis, Academic Press, London, 1977.
16 C.G. Broyden, A class of methods for solving nonlinear simultaneous equations, Math. Comp. 19 (1965) 577.
17 C.G. Broyden, A new method of solving solving simultaneous nonlinear equations, Comput J. 19 (1969) 94-99.
18 J.A. Nelder and R. Mead, A simplex method for function minimization, Computer J., 7 (1964) 308-313.
19 T.E. Shoup, A Practical Guide to Computer Methods for Engineers, Prentice-Hall, Englewood Cliffs, N.J., 1979.
20 H.H. Rosenbrock, An automatic method for finding the greatest or least value of a function, Computer J. 3 (1960) 175-184.
21 R. Fletcher and M.J.D. Powell, A rapidly convergent decent method for Minimization, Computer J. 6 (1963) 163-168.
22 R. Fletcher, Certification of Algorithm 251, Communications of the ACM,
23 R.A. Alberty and F. Daniels, Physical Chemistry, 5th ed. John Wiley, New York, 1980.
24 K.J. Johnson, Numerical Methods in Chemistry, Marcel Dekker, New York, 1980.
25 D.N. Hanson, J.H. Duffin and G.F Sommerville, Computation of Multistage Separation Processes, Reinhold, New York, 1962.
26 E. Kehat and B. Ghitis, Simulation of an extraction column, Computers and Chemical Engineering, 5 (1981) 171-180.
27 W.B. White, S.M. Johnson and G.B. Dantzig, Chemical equilibrium in complex mixtures, J. of Chemical Physics, 28 (1958) 751-755.
28 W.R. Smith and I. Missen, Chemical Reaction Equilibrium Analysis, John Wiley, New York, 1982.
29 M. Uchida, MPEC2: A code for multi-phase chemical equilibria, Computers and Chemistry, 11 (1987) 19-24.
30 V.P. Glushko (Editor), Thermodynamic Properties of Individual Substances, (in 4 volumes) Nauka, Moscow, 1978-1982 (in Russian)

Chapter 3

PARAMETER ESTIMATION

The most immediate goal of scientific or industrial experimentation is to find relationships among manipulated and observed variables, or to validate such relationships coming from some underlying theory. A mathematical description almost invariably involves estimating the values of some unknown parameters to best match the available body of experimental observations.

The simplest mathematical description or model of a system is the function

$$y = f(\mathbf{x},\mathbf{p}) \; , \qquad (3.1)$$

assumed to predict the dependent variable y in terms of the independent variables $\mathbf{x} = (x_1,x_2,\ldots,x_{nx})^T$ and unknown parameters $\mathbf{p} = (p_1,p_2,\ldots,p_{nx})^T$. To begin with a relatively simple problem we will assume that the independent variables can be manipulated or observed error-free, and only the dependent variable y is corrupted by measurement errors. Thus the outcome of the i-th experiment is given by the vector $(x_{i1}, x_{i2}, \ldots, x_{i,nx}, \tilde{y}_i)$, where

$$\tilde{y}_i = f(\mathbf{x}_i,\mathbf{p}) + \epsilon_i \; . \qquad (3.2)$$

Our basic assumption is that the response function $f(\mathbf{x},\mathbf{p})$ is a correct one and the random quantity ϵ_i represents the measurement error. It is then meaningful to ask what the true value $\mathbf{p}$ of the parameters is, though by the imprecise nature of measurements we can never hope to determine it with absolute certainty. However, having a set $\{ (x_{i1}, x_{i2}, \ldots, x_{i,nx}, \tilde{y}_i) ;$ $i = 1,2,\ldots,nm \}$ of observations and assuming some statistical properties of the errors, it is reasonable to seek parameter estimates that yield not only a good fit to the data, but on the average comes firmly close to the true values, and do not vary excessively from one set of experiments to the next.

Parameter estimation is rooted in several scientific areas with their own preferences and approaches. While linear estimation theory is a nice chapter of mathematical statistics (refs. 1-3), practical considerations are equally important in nonlinear parameter estimation. As emphasised by Bard (ref. 4), in spite of its statistical basis, nonlinear estimation is mainly a variety of computational algorithms which perform well on a class of problems but may fail on some others. In addition, most statistical tests and estimates of variability are formulated for linear models, and in the nonlinear case most often the best we can do is to apply these linear results as approximations. Furthermore, in practice no parameter estimation problem can be solved automatically in one go even with a fairly good numerical algorithm available. As you will see in this chapter, one usually needs additional assumptions, good knowledge of underlying processes, or simply common sense, and thus we end up with a typical problem of scientific computing rather than that of mathematical statistics.

In spite of the variety of approaches and methods, it is relatively easy to formulate the common steps of solving an estimation problem, as we do in the remainder of this section.

Response function selection

The form of the response function to be fitted depends on the goal of modeling, and the amount of available theoretical and experimental information. If we simply want to avoid interpolation in extensive tables or to store and use less numerical data, the model may be a convenient class of functions such as polynomials. In many applications, however, the model is based on theoretical relationships that govern the system, and its parameters have some well defined physical meaning. A model coming from the underlying theory is, however, not necessarily the best response function in parameter estimation, since the limited amount of data may be insufficient to find the parameters with any reasonable accuracy. In such cases simplified models may be preferable, and with the problem of simplifying a nonlinear model we leave the relatively safe waters of mathematical statistics at once.

Selection of error structure and estimation criterion

For a model of the form (3.2) it is natural to choose parameter values that minimize some norm of the errors. The first norm that comes to mind is the sum of squares

$$Q(\mathbf{p}) = \sum_{i=1}^{nm} [\tilde{y}_i - f(x_i,\mathbf{p})]^2 w_i \tag{3.3}$$

where the w's are *a priori* fixed weighting coefficients measuring the importance of particular observations in the sum.

Other error norms have been considered in Sections 1.8.2 and 1.8.3. Why the least squares method is the most popular? Where does it come from? If it is good at least for a well defined class of problems, why to experiment with other estimation criteria? We try to answer these questions in turn.

Without information on the errors any error norm is as good as the others. Thus, to explain the popularity of the least squares method we have to make a number of assumptions. In particular, for model (3.2) we assume that

(i) the independent variables x are error-free;

(ii) the error ϵ_i is independent of x_i;

(iii) ϵ_i has zero mean, i.e., $E\{\epsilon_i\} = 0$;

(iv) the errors ϵ_i and ϵ_j, $i \neq j$, are independent;

(v) the variance $D^2\{\epsilon_i\} = \sigma_i^2$ of ϵ_i is known, at least up to a common scalar factor in all variances; and

(vi) ϵ_i is a normally distributed random variable.

Assumptions (i) and (ii) justify the model in the form (3.2), with an additive error as the only random variable. By (iii) we assume that the model is correct and there are no systematic measurement errors, i.e., $E\{\tilde{y}_i\} = f(x_i,p)$ for the true value p of the parameters. The role of other assumptions will be clarified later. At this moment the most important message, coming from mathematical statistics, is as follows. If assumptions (i) through (iii) are satisfied, the model (3.2) is linear in the parameters, and we select the weighting coefficients according to $w_i = \sigma^2/\sigma^2_i$, where σ is a (possibly unknown) scalar, then the vector $\hat{p}$ of least squares estimate has very satisfying statistical properties. First, $\hat{p}$ is unbiased, thus $E\{\hat{p}\} = p$, the true parameter vector. Second, $\hat{p}$ has the least variance among all unbiased estimates (ref. 1). While for a nonlinear function of the parameters these properties can be shown only assymptotically, i.e., increasing the number of experiments beyond bound, the method produces acceptable estimates in many situations (ref. 4).

While the least squares estimator appeared several centuries ago as an independent method giving good results under certain assumptions, we have to dig deeper into mathematical statistics to see its roots, and, in particular, its limitations. This general subject is the maximum likelihood principle, one of the basic concepts of mathematical statistics. The principle is simple: select the parameters such that the occurence of the observed values $\tilde{y}_1$, ..., $\tilde{y}_{nm}$ be the most likely among all the possible outcomes of the experiment. But how this can be done? It is very important that given a value of the parameters one can compute the probability of occurence of a particular data set, if the error distribution function is known. There is only a small trick: since the $\tilde{y}$'s take on continuous values, this probability is always zero unless we consider an interval $\sigma_i\Delta$ around each observation. So we always assume such intervals when talking about probabilities. According to assumptions (iv), (v) and (vi), our data points $\tilde{y}_i$ are independently random and distributed as a normal (Gaussian) distribution around the true $f(x_i,p)$ with the standard deviation σ_i. Then the probability of obtaining the data set $\tilde{y}_1, \ldots, \tilde{y}_{nm}$ (recall the intervals $\sigma_i\Delta$ around them!) is the product of the probabilities of each point,

$$p\left\{ \left|\frac{\tilde{y}_i - f(x_i,p)}{\sigma_i}\right| \le \Delta, \quad i = 1, \ldots, nm \right\} =$$

$$= (2\pi)^{-nm/2} \prod_{i=1}^{nm} \left\{\sigma_i^{-1}\exp\left[-\frac{1}{2} \left(\frac{\tilde{y}_i - f(x_i,p)}{\sigma_i}\right)^2 \right] \Delta \right\}. \qquad (3.4)$$

Maximizing (3.4) is equivalent to minimizing its negative logarithm. Furthermore, since Δ is constant and the σ_i's are known, minimizing this equation is equivalent to minimizing (3.3) with $w_i = \sigma^2/\sigma_i^2$, where the particular value of σ^2 clearly does not affect the location of the minimum.

Though the maximum likelihood principle is not less intuitive than the least squares method itself, it enables the statisticans to derive estimation criteria for any known distribution, and to generally prove that the estimates have nice properties such as asymptotic unbiasedness (ref. 1). In particular, the method of least absolute deviations introduced in Section 1.8.2 is also a maximum likelihood estimator assuming a different distribution for the error.

Since the final form of a maximum likelihood estimator depends on the assumed error distribution, we partially answered the question why there are different criteria in use, but we have to go further. Maximum likelihood estimates are only guaranteed to have their expected properties if the error distribution behind the sample is the one assumed in the derivation of the method, but in many cases are relatively insensitive to deviations. Since the error distribution is known only in rare circumstances, this property of robustness is very desirable. The least squares method is relatively robust, and hence its use is not restricted to normally distributed errors. Thus, we can drop condition (vi) when talking about the least squares method, though then it is no more associated with the maximum likelihood principle. There exist, however, more robust criteria that are superior for errors with distributions significantly deviating from the normal one, as we will discuss in Section 3.10.1.

Up to this point we relaxed only assumption (vi), now we try to do the same with the others, except (iii). This latter is necessary, since a nonzero mean $\bar{\epsilon}_i = E\{\epsilon_i\}$ is undistinguishable from the response $f(x_i,p)$. We can relax the other assumptions, but then the least squares method no more applies. In particular, one can drop (iv) and (v), and estimate the covariance matrix (or part of it) simultaneously with the model parameters. This means introducing additional parameters, and hence the problem is clearly more difficult to solve. Nevertheless, observing several variables simultaneously, the assumption of independent errors is frequently unfeasible. A possible treatment of the problem will be considered in Section 3.6. In another class of applications we cannot neglect the error in the independent variables of (3.1), and hence give up assumption (i), estimating the expected value of all variables simultaneously with estimating the parameters. As you will see in Section 3.8, the treatment of such error-in-variables models differs considerably from that of the model (3.2).

While you will use the least squares method in most cases, do not forget that selecting an estimation criterion you make assumptions on the error structure, even without a real desire to be involved with this problem. Therefore, it is better to be explicit on this issue, for the sake of consistency in the further steps of the estimation.

Parameter estimation

In a strict sense parameter estimation is the procedure of computing the estimates by localizing the extremum point of an objective function. A further advantage of the least squares method is that this step is well supported by efficient numerical techniques. Its use is particularly simple if the response function (3.1) is linear in the parameters, since then the estimates are found by linear regression without the inherent iteration in nonlinear optimization problems.

Goodness-of-fit

The validity of parameter estimation clearly depends on the validity of the assumptions on the form of the response function and the error distribution. The simplest way to check these assumptions is to inspect the residuals

$$r_i = \tilde{y}_i - f(x_i,\hat{p}) \tag{3.5}$$

computed at the estimates $\hat{\mathbf{p}}$. If the residuals are large, or of such a nonrandom structure, that they cannot be ascribed to random observation errors, then this constitutes strong grounds for rejecting the assumed model or the error structure. More generally, the method of testing the goodness-of-fit in a particular problem depends on the assumptions you made in the estimation stage.

Interpretation of the estimates

It is not enough to compute the estimates $\hat{p}$ of the parameters, we must also investigate their reliability and precision. Computed from the random variables $\tilde{y}_i$, the estimate is a random vector itself and hence can be completely characterized only by its distribution function. Some important statistical properties of $\hat{p}$ (e.g., its covariance matrix) can, however, be estimated on the basis of the assumed error structure. We can answer also questions such as "what are the chances that the estimate is off by no more than 1%?", i.e., to compute some confidence regions. It should be, however, emphasised that most statistical tests and estimates of variability apply only approximately to nonlinear models, and even for linear models they are exact only if the measurement errors do indeed follow whatever distribution was assumed for them. Nevertheless, even the approximate results are particularly useful if the parameters have physical significance.

Simulation

Even with powerful computer programs at hand, the solution of estimation problems is usually far from simple. A convenient way to eliminate computational errors and to study the effects of statistical assumptions is to solve first a problem with known true parameter values, involving data generated at some nominal parameter vector. Initially it is advisable to investigate with error-free data, then to add errors of the assumed structure. The simulation usually requires normally distributed random variables. Random numbers R that approximately are from a normal distribution with zero mean and unit variance can be obtained by

$$R = \sum_{i=1}^{12} U_i - 6 , \qquad (3.5)$$

where the U's are random numbers, uniformly distributed in the interval [0,1] and readily supplied by an internal function of most BASIC dialects.

3.1 FITTING A STRAIGHT LINE BY WEIGHTED LINEAR REGRESSION

The most frequent estimation problem is to find the parameters a and b of the linear function $y = ax + b$ in order to fit the line to the observations $\{ (x_i, \tilde{y}_i);\ i = 1,2,\ldots,n \}$, where

$$\tilde{y}_i = ax_i + b + \epsilon_i \ . \tag{3.6}$$

Assuming conditions (i) through (v) we will minimize the least squares objective function

$$Q(a,b) = \sum_{i=1}^{n} [\ \tilde{y}_i - ax_i - b\ ]^2 w_i \tag{3.7}$$

where the w's are fixed weighting coefficients. If the errors are normally distributed, then with $w_i = \sigma^2/\sigma_i^2$ (3.7) corresponds to the maximum likelihood objective function. Therefore it is advantageous to chose the weights on this basis, if estimates of the error variances σ_i^2 are available. The value of σ^2 clearly does not affect the location of the minimum, and hence it suffices to know (or, in practice to assume) the relative error variances in advance.

Equations $\partial Q(a,b)/\partial a = 0$ and $\partial Q(a,b)/\partial b = 0$ are linear in the parameters. Solving them simultaneously we obtain the least squares estimates

$$\hat{a} = \frac{\Sigma w_i \tilde{y}_i (x_i - \bar{x}_w)}{\Sigma w_i (x_i - \bar{x}_w)^2} \tag{3.8a}$$

and

$$\hat{b} = \bar{y}_w - \hat{a}\bar{x}_w \ , \tag{3.8b}$$

where the summation goes from 1 to n , and the weighted means $\bar{y}_w$ and $\bar{x}_w$ are defined by

$$\bar{y}_w = \frac{\Sigma w_i \tilde{y}_i}{\Sigma w_i} \ , \quad \bar{x}_w = \frac{\Sigma w_i x_i}{\Sigma w_i} \ . \tag{3.9}$$

The estimates yield the regression line

$$\hat{y} = \hat{a}x + \hat{b} \ . \tag{3.10}$$

The goodness of fit can be measured by the weighted residual sum of squares $Q(\hat{a},\hat{b})$.

If the errors have the same variance, we can take $\sigma = \sigma_i$ without knowing the real value of σ by putting $w_i = 1$ for all i. This case is called unweighted least squares , and the quantity

$$s^2 = \frac{Q(\hat{a},\hat{b})}{n-2} \tag{3.11}$$

is an unbiased estimate of σ^2 (see e.g., ref. 5). The square root of s^2 is called standard residual error or simply standard error.

With unequal variances we cannot speak of an "overall standard error". In that case s^2 computed by (3.11) yields an unbiased estimate of the constant σ^2 in the weighting coefficients. Therefore, $s_i^2 = s^2/w_i$ is an unbiased estimate of the error variance σ_i^2. If we have a different independent estimate of the same variance, for example computed from the replicates at the value x_i of the independent variable, then our assumptions can be checked by an F-test, involving the ratio of the two estimates, see e.g. Himmelblau (ref. 5). Though this is the best way to measure the goodness-of-fit, it requires additional information (i.e., replicates), not always available.

Under the conditions (i) through (v) the least square estimates are unbiased in the linear case. Thus $E(\hat{a}) = a$, and the variance $D^2(\hat{a})$ is

$$D^2(\hat{a}) = E\{ \left[a - E(\hat{a})\right]^2 \} = E\{ (a - \hat{a})^2 \} . \tag{3.12}$$

From the last expression $D^2(\hat{a})$ can actually be computed, since replacing $\tilde{y}_i$ by the error-free variable y_i in (3.8a) we would obtain the true parameter a as the estimate. Therefore, we set this expression and (3.8a) into (3.12), and compute the expectation. Since $E(\hat{y}_i - y_i) = \sigma_i^2$, which can be estimated by $s_i^2 = s^2/w_i$, after some algebraic manipulation we have the estimate

$$s_a^2 = s^2 \frac{1}{\Sigma w_i(x_i-\bar{x}_w)^2} , \tag{3.13}$$

for the variance $D^2(\hat{a})$. Similarly, we obtain the estimate

$$s_b^2 = s^2 \left[\frac{1}{\Sigma w_i} - \frac{(\bar{x}_w)^2}{\Sigma w_i(x_i-\bar{x}_w)^2} \right] \tag{3.14}$$

for the variance $D^2(\hat{b})$.

According to (3.8a) $\hat{a}$ is a linear combination of the observations $\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{nm}$. Therefore, normally distributed observations result in normally distributed estimates. Then the quantity defined by $t = (\hat{a} - a)/s_a$ has t-distribution (also called Student distribution) with n-2 degrees of freedom (ref. 5). The intervals, that contain the true parameters with α% probability, called α% confidence intervals, are given by

$$\begin{aligned} \hat{a} - s_a t_{p,n-2} \leqq\ & a \leqq \hat{a} + s_a t_{p,n-2} \\ \hat{b} - s_b t_{p,n-2} \leqq\ & b \leqq \hat{b} - s_b t_{p,n-2} \end{aligned} \tag{3.15}$$

where $p = 1 - \alpha/100$, and $t_{p,n-2}$ is the tabular value of the t - distribution with n-2 degrees of freedom at the probability p . The following program module computes and prints the quantities discussed.

Program module M40

```
4000 REM ****************************************************
4002 REM *  FITTING A STRAIGHT LINE BY LINEAR REGRESSION  *
4004 REM ****************************************************
4006 REM INPUT:
4008 REM     N      NUMBER OF SAMPLE POINTS
4010 REM     X(N)   OBSERVATIONS OF INDEPENDENT VARIABLE X
4012 REM     Y(N)   OBSERVATIONS OF DEPENDENT VARIABLE Y
4014 REM     WI     IDENTIFIER OF WEIGHTING OPTIONS
4016 REM              0  IDENTICAL WEIGHTS ( W(I)=1 )
4018 REM              1  RELATIVE WEIGHTS ( W(I)=1/Y(I)^2 )
4020 REM              2  USER-SPECIFIED WEIGHTS FURTHER GIVEN IN
4022 REM     W(N)
4024 REM OUTPUT:
4026 REM     A      SLOPE
4028 REM     B      Y-INTERCEPT
4030 REM     ...    AND FURTHER PRINTED RESULTS
4032 REM MODULE CALLED: M41
4034 XW=0 :YW=0 :WW=0
4036 FOR I=1 TO N
4038  IF WI=0 THEN W(I)=1 ELSE IF WI=1 THEN W(I)=1/Y(I)^2
4040  XW=XW+W(I)*X(I) :YW=YW+W(I)*Y(I) :WW=WW+W(I)
4042 NEXT I
4044 XW=XW/WW :YW=YW/WW : D=0
4046 FOR I=1 TO N : D=D+W(I)*(X(I)-XW)^2 :A=A+W(I)*Y(I)*(X(I)-XW) :NEXT I
4048 A=A/D :B=YW-A*XW
4050 S2=0 :FOR I=1 TO N :DE=Y(I)-A*X(I)-B :S2=S2+W(I)*DE*DE :NEXT I
4052 NF=N-2 :S2=S2/NF :SA=SQR(S2/D) :SB=SQR(S2*(1/WW+XW^2/D))
4054 GOSUB 4100
```

```
4056 REM ---------- PRINT RESULTS
4058 V$=STRING$(70,"-") :F$="#.#####^^^^   "
4060 LPRINT TAB(20)"LEAST SQUARES FIT OF LINE Y=A*X+B" :LPRINT :LPRINT
4062 LPRINT V$
4064 LPRINT " I","X MEAS","Y MEAS","Y COMP","RESIDUAL" :LPRINT V$
4066 FOR I=1 TO N
4068  Y=A*X(I)+B :DE=Y(I)-Y :LPRINT I, :LPRINT USING F$;X(I),Y(I),Y,DE
4070 NEXT I :LPRINT V$ :LPRINT :IF WI>0 THEN LPRINT "(WEIGHTED)"
4072 LPRINT " RESIDUAL SUM OF SQUARES ............... ";S2*NF
4074 IF WI=0 THEN LPRINT " STANDARD RESIDUAL ERROR ............... ";SQR(S2)
4076 IF WI>0 THEN LPRINT " ESTIMATED SIGMA FACTOR IN WEIGHTS ..... ";SQR(S2)
4078 LPRINT " DEGREES OF FREEDOM .................... ";NF
4080 LPRINT " CRITICAL T-VALUE AT 95 % CONF. LEVEL .. ";T
4082 LPRINT :LPRINT V$
4084 LPRINT "PARAMETER ","ESTIMATE","STNRD.ERROR","LOWER BOUND","UPPER BOUND"
4086 LPRINT V$
4088 LPRINT "    A", :LPRINT USING F$;A,SA,A-T*SA,A+T*SA
4090 LPRINT "    B", :LPRINT USING F$;B,SB,B-T*SB,B+T*SB
4092 LPRINT V$ :LPRINT
4094 RETURN
4096 REM ****************************************************
```

The module offers three weighting options. If no weighting is used, $w_i = 1$ is set for all i by the module. If relative weighting is used, the module computes the weights $w_i = 1 / \tilde{y}_i^2$, thus this option is not recommended if any observed variable is near to zero. If you choose the third weighting option then you should supply the weights in the vector W(N). No error flag is implied, although errors may occur if the number of points is less than 3, the x_i values are all the same or some of the weights are negative.

The tabular value of the t-distribution, required to find the confidence intervals (3.15), is obtained by calling the following auxiliary module.

Program module M41

```
4100 REM ****************************************************
4102 REM *    CRITICAL T-VALUE AT 95 % CONFIDENCE LEVEL     *
4104 REM ****************************************************
4106 REM INPUT:
4108 REM      NF     DEGREES OF FREEDOM
4110 REM OUTPUT:
4112 REM      T      CRITICAL T-VALUE
4114 IF NF>20 THEN 4126
4116 T= -(NF=1)*12.71 -(NF=2)*4.3   -(NF=3)*3.18 -(NF=4)*2.78  -(NF=5)*2.57
4118 T=T-(NF=6)*2.45  -(NF=7)*2.37  -(NF=8)*2.31 -(NF=9)*2.26  -(NF=10)*2.23
4120 T=T-(NF=11)*2.2  -(NF=12)*2.18 -(NF=13)*2.16-(NF=14)*2.15 -(NF=15)*2.13
4122 T=T-(NF=16)*2.12 -(NF=17)*2.11 -(NF=18)*2.1 -(NF=19)*2.09 -(NF=20)*2.09
4124 GOTO 4134
4126 IF NF<31 THEN AT=12.3:BT=(LOG(AT)-LOG(8.2))/10*(20-NF) :GOTO 4132
4128 IF NF<61 THEN AT=8.2 :BT=(LOG(AT)-LOG(4))/30*(30-NF)   :GOTO 4132
4130 AT=3.9 :BT=(LOG(AT)-LOG(2))/60*(60-NF)
4132 T=INT(196.5+AT*EXP(BT))/100
4134 RETURN
4136 REM ****************************************************
```

The only goal of this simple module is to return the t-value found in statistical tables. Thus, the module could be based on DATA and READ statements instead of the expressions above, but the present form is more convenient to use if the module is called several times.

Example 3.1 Fitting a straight line by least squares method

Table 1.1 lists nicotine and tar concentrations found in different sorts of cigarettes. As discussed in Section 1.8.2, one has reason to assume a simple linear relationship between the two quantities. First we assume that the error variance is constant, and solve the unweighted least squares problem by the following main program.

```
100 REM ------------------------------------------------------------
102 REM EX. 3.1. FITTING A REGRESSION LINE
104 REM MERGE M40,M41
106 REM ---------- DATA
108 REM  (N)
110 DATA 10
112 REM  ( X ,  Y )
114 DATA  8.3, 0.32
116 DATA 12.3, 0.46
118 DATA 18.8, 1.10
120 DATA 22.9, 1.34
122 DATA 23.1, 1.26
124 DATA 24.0, 1.44
126 DATA 27.3, 1.42
128 DATA 30.0, 1.96
130 DATA 35.9, 2.23
132 DATA 41.6, 2.20
200 REM ---------- READ DATA
202 READ N
204 DIM X(N),Y(N),W(N)
206 FOR I=1 TO N :READ X(I),Y(I) :NEXT I
208 REM ---------- FIT A STRAIGHT LINE WITH NO WEIGHTING
210 WI=0 :GOSUB 4000
212 STOP
```

It is interesting to compare the following output printed by the module with the results of Examples 1.8.2 and 1.8.3.

```
              LEAST SQUARES FIT OF LINE Y=A*X+B


------------------------------------------------------------------------
I         X MEAS        Y MEAS        Y COMP        RESIDUAL
------------------------------------------------------------------------
1         0.83000E+01   0.32000E+00   0.35851E+00   -.38515E-01
2         0.12300E+02   0.46000E+00   0.61025E+00   -.15025E+00
3         0.18800E+02   0.11000E+01   0.10193E+01   0.80685E-01
4         0.22900E+02   0.13400E+01   0.12773E+01   0.62659E-01
5         0.23100E+02   0.12600E+01   0.12899E+01   -.29928E-01
6         0.24000E+02   0.14400E+01   0.13466E+01   0.93432E-01
7         0.27300E+02   0.14200E+01   0.15542E+01   -.13425E+00
8         0.30000E+02   0.19600E+01   0.17242E+01   0.23583E+00
9         0.35900E+02   0.22300E+01   0.20955E+01   0.13453E+00
10        0.41600E+02   0.22000E+01   0.24542E+01   -.25419E+00
------------------------------------------------------------------------

 RESIDUAL SUM OF SQUARES ............... .2004705
 STANDARD RESIDUAL ERROR ............... .1582998
 DEGREES OF FREEDOM .................... 8
 CRITICAL T-VALUE AT 95 % CONF. LEVEL .. 2.31

------------------------------------------------------------------------
PARAMETER   ESTIMATE      STNRD.ERROR   LOWER BOUND   UPPER BOUND
------------------------------------------------------------------------
    A       0.62933E-01   0.52507E-02   0.50804E-01   0.75062E-01
    B       -.16383E+00   0.13765E+00   -.48180E+00   0.15413E+00
------------------------------------------------------------------------
```

Though the variances are unknown, considering the small residuals the fit can be intuitively judged acceptable. This is supported by the lack of trend in the sequence of the residuals. The slope $\hat{a}$ is more reliable than the intercept $\hat{b}$. In fact the latter estimate heavily depends on the estimation criterion, as shown in Sections 1.8.2 and 1.8.3. The relations among the different methods we used to solve this problem will be discussed in Section 3.10.1.

<u>Exercises</u>

□ Solve the regression problem with relative weighting (use option WI=1). Compare the two sequences of residuals.

□ Since tar concentrations are also corrupted by measurement errors, and since we do not know which variable is more reliable, it is equally meaningful to fit the inverse model $x = Ay + B$ to the data, thereby regarding the nicotine concentration as independent variable. Show that the two regression lines differ, thus $\hat{a} \neq 1/\hat{A}$ and $\hat{b} \neq -\hat{B}/\hat{A}$. This problem will be further discussed in Section 3.8.

3.2 MULTIVARIABLE LINEAR REGRESSION

Extending the methods of the previous section we first fit the linear model

$$y = p_1x_1 + p_2x_2 + \cdots + p_{nx}x_{nx} \tag{3.16}$$

to the set $\{ (x_{i1}, x_{i2}, \ldots, x_{i,nx}, \tilde{y}_i) ; \quad i = 1,2,\ldots,nm \}$ of observations, where

$$\tilde{y}_i = p_1x_{i1} + p_2x_{i2} + \cdots + p_{nx}x_{i,nx} + \epsilon_i \ . \tag{3.17}$$

As in Section 3.1, we assume that the errors are of zero mean and independent, with the variances

$$D^2\{\epsilon_i\} = \sigma_i^2 = \sigma^2/w_i \ , \tag{3.18}$$

where the weighting coefficients w_i are known. The least squares objective function is

$$Q(p) = \sum_{i=1}^{nm} [\ \tilde{y}_i - p_1x_{i1} - p_2x_{i2} - \cdots - p_{nx}x_{i,nx}\ ]^2 w_i \ . \tag{3.19}$$

Introducing the notations

$$\tilde{\mathbf{Y}} = \begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \vdots \\ \tilde{y}_{nm} \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1,nx} \\ x_{21} & x_{22} & \cdots & x_{2,nx} \\ \vdots & & & \\ x_{nm,1} & x_{nm,2} & \cdots & x_{nm,nx} \end{bmatrix}, \quad \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_{nm} \end{bmatrix} \tag{3.20}$$

expressions (3.17) and (3.19) are reduced to

$$\tilde{\mathbf{Y}} = \mathbf{X}\mathbf{p} + \epsilon \tag{3.21}$$

and

$$Q(p) = (\tilde{\mathbf{Y}} - \mathbf{X}\mathbf{p})^T\mathbf{W}(\tilde{\mathbf{Y}} - \mathbf{X}\mathbf{p}) \ , \tag{3.22}$$

respectively, where $\mathbf{W}$ is an $nm \times nm$ diagonal matrix with diagonal entries $w_1, w_2, \ldots, w_{nm}$. Solving the simultaneous linear equations $\partial Q(p)/\partial p_i = 0$; $i = 1,2,\ldots,nx$, gives the least squares estimates

$$\hat{p} = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{Y} . \tag{3.23}$$

The goodness-of-fit is again measured in terms of the residual sum of squares $Q(\hat{p})$ and the variance

$$s^2 = \frac{Q(\hat{p})}{nm - nx} \tag{3.24}$$

of the residuals. As in the previous section, s^2 is an estimate of the constant σ^2 in the weights, and hence $s_i^2 = s^2/w_i$ is an unbiased estimate of the error variance σ_i^2 for all i. Having another estimate $\tilde{s}_i^2$ of the same variance (e.g. from replicates), the F-test involving the ratio $F = s_i^2 / \tilde{s}_i^2$ can be used to check our assumptions. In practice, however, such independent estimates $\tilde{s}_i^2$ are available in rare circumstances, and the goodness-of-fit is usually assessed by studying the sequence $r_i = \tilde{y}_i - \hat{p}_1x_{i1} - \hat{p}_2x_{i2} - \ldots$ $\ldots - \hat{p}_{i,nx}x_{nx}$, $i = 1,2,\ldots,nm$, of residuals. While many diagnosis methods are in use, the basic idea is that in case of a satisfactory fit the observations should be randomly distibruted around the regression hyperplane

$$\hat{y} = \hat{p}_1x_1 + \hat{p}_2x_2 + \ldots + \hat{p}_{nx}x_{nx} . \tag{3.25}$$

Simple but useful diagnosis tools are the residual plots discussed by Wood (ref. 6) . If the residuals are of highly nonrandom structure, at least one of the assumptions is questionable. This nonrandomness implies that the elements of the residual sequence $r_1, r_2, \ldots, r_{nm}$ are correlated. A measure of this serial correlation is the D-statistics proposed by Durbin and Wattson (ref. 7), and computed according to

$$D = \sum_{i=2}^{nm} (r_i - r_{i-1})^2 / \sum_{i=2}^{nm} r_i^2 . \tag{3.26}$$

Too large or too small values of (3.26) indicate nonrandomness in the residual sequence. The critical values of D are tabulated in many textbooks (see, e.g., refs. 5) for $nm > 15$. Assymptotically (i.e., for $nm \geq 100$), the fit is acceptable at 95% confidence level if $1.7 \leq D \leq 2.3$, and this interval is larger for smaller samples. Unfortunately the value of D statistics depends on the particular order of the observations, which is arbitrary in many cases. Thus you should be careful with D statistics in multivariable regression.

The most important variability measure of the estimate $\hat{\mathbf{p}}$ is its covariance matrix defined by

$$\mathrm{cov}\{ \hat{\mathbf{p}} \} = E\{ (\hat{\mathbf{p}} - \mathbf{p})(\hat{\mathbf{p}} - \mathbf{p})^T \}. \tag{3.27}$$

This definition already takes into account that in the linear case the least square estimates are unbiased, thus $E\{ \hat{\mathbf{p}} \} = \mathbf{p}$. Let $\mathbf{Y}' = (y_1, y_2, \ldots, y_{nm})^T$ denote the vector of the "true" dependent variables in the sample points, then replacing $\mathbf{Y}$ by $\mathbf{Y}'$ in (3.23) we obtain the true parameters $\mathbf{p}$ as estimates. Using this expression for $\mathbf{p}$ and (3.23) for $\hat{\mathbf{p}}$, the definition (3.27) gives

$$\mathrm{cov}\{ \hat{\mathbf{p}} \} = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\, E\{\epsilon\epsilon^T\}\, \mathbf{W}(\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1} , \tag{3.28}$$

where $\epsilon = \mathbf{Y} - \mathbf{Y}'$. The factor $E\{\epsilon\epsilon^T\}$ in (3.28) is the covariance matrix of the measurement errors, and according to (3.18) it is given by

$$\mathrm{cov}\{\epsilon\epsilon\} = E\{\epsilon\epsilon^T\} = \sigma^2\mathbf{W}^{-1} . \tag{3.29}$$

Using (3.29) and taking into account that s^2 is the estimate of σ^2, (3.28) yields the expression

$$\mathbf{C}_p = s^2 \, (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1} \tag{3.30}$$

to estimate the covariance matrix of $\hat{\mathbf{p}}$. According to the definition (3.27) of the covariance matrix, the diagonal entries of $\mathbf{C}_p$ estimate the variances of individual parameters, and we can also evaluate confidence intervals for them, similarly to (3.15). The only difference is that now there are $nm - nx$ degrees of freedom.

The statistical dependence between the estimates $\hat{p}_i$ and $\hat{p}_j$ is expressed in term of the correlation coefficients r_{ij}, forming the correlation matrix of the estimates

$$[\mathbf{R}_p]_{ij} = r_{ij} = [\, \mathbf{C}_p]_{ij} \,/\, ([\, \mathbf{C}_p\,]_{ii}[\, \mathbf{C}_p\,]_{jj})^{1/2} . \tag{3.31}$$

If the estimates are strongly correlated then they are far from being independent and it is better to evaluate their joint confidence region instead of individual confidence intervals. As shown e.g., by Bard (ref. 4), the quantity $(p - \hat{p}) C_p^{-1} (p - \hat{p})$ follows χ^2 distribution with nx degrees of freedom, and hence the region of the parameter space defined by

$$(p - \hat{p}) C_p^{-1} (p - \hat{p}) \le \chi^2_{p,nx} \qquad (3.32)$$

contains the true parameter vector in α% of all possible data samples. In (3.32) $\chi^2_{p,nx}$ is the tabular value of the χ^2 distribution with nx degrees of freedom at the probability $p = 1 - \alpha/100$. The α% confidence region (3.32) is a hyperellipsoid in the nx-dimensional space around the estimate $\hat{p}$. As shown in Fig. 3.1, the confidence region may include parameter values that are not at all close to the actual estimate $\hat{p}$, whereas the individual confidence limits usually underestimate this uncertainty and do not reflect the dependences among the parameters.

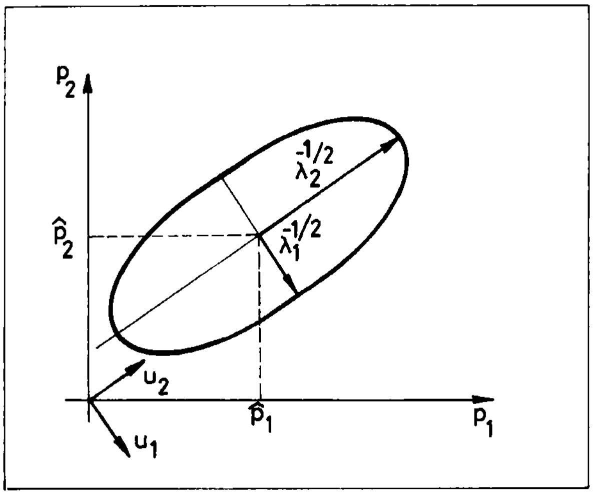


Fig. 3.1. Confidence region of the parameter estimates

In the multivariate linear regression module M42 first we normalize the matrix X^TWX to a correlation-type matrix by a transformation similar to (3.31) in order to somewhat decrease the numerical errors. This transformation is equivalent to a scaling of the parameters, i.e., the unknown variables of the normal equations. With an ill-conditioned X^TWX , however, the estimates are strongly influenced by small perturbations in the observations vector $\tilde{Y}$. This is a frequent problem in parameter estimation, and we use the eigenvalue-eigenvector decomposition of the normalized X^TWX in order to detect it. Interpretation of the results of this procedure will be detailed in Section 3.5.

The three weighting options of the module are similar to the ones of the module M40. With no weighting or with relative weighting the array W containing the diagonal entries of the weighting matrix is generated automatically. This array should be evaluated in the main program only if the option of user specified weights is used.

The parameter RP among the input data is the ridge parameter that will be exploited in Section 3.5. In normal regression problems RP = 0 should be used.

Program module M42

```
4200 REM ####################################################
4202 REM #        MULTIVARIABLE LINEAR REGRESSION          #
4204 REM #             WEIGHTED LEAST SQUARES              #
4206 REM ####################################################
4208 REM INPUT:
4210 REM     NM      NUMBER OF SAMPLE POINTS
4212 REM     NX      NUMBER OF INDEPENDENT VARIABLES
4214 REM   X(NM,NX)  TABLE OF INDEPENDENT VARIABLES
4216 REM    Y(NM)    OBSERVATIONS OF DEPENDENT VARIABLE
4218 REM     WI      IDENTIFIER OF WEIGHTING OPTIONS
4220 REM                0  IDENTICAL WEIGHTS ( W(I)=1 )
4222 REM                1  RELATIVE WEIGHTS ( W(I)=1/Y(I)^2 )
4224 REM                2  USER-SPECIFIED WEIGHTS
4226 REM                   GIVEN BY FURTHER INPUT AS
4228 REM    W(NM)    VECTOR OF WEIGHTS (ONLY FOR WI=2)
4230 REM     RP      RIDGE PARAMETER (ZERO FOR ORDINARY LEAST SQUARES)
4232 REM OUTPUT:
4234 REM     ER      STATUS FLAG
4236 REM                0  REGRESSION COMPLETED
4238 REM                1  SINGULAR COVARIANCE MATRIX
4240 REM    P(NX)    REGRESSION COEFFICIENTS IN THE EQUATION
4242 REM                   Y = P1#X1 + P2#X2 + ... + Pnx#Xnx
4244 REM    .....    (FURTHER RESULTS ARE PRINTED IN THE MODULE)
4246 REM AUXILIARY ARRAYS:
4248 REM  A(NX,NX),C(NX,NX),U(NX,NX),D(NX)
4250 REM MODULES CALLED: M16,M18,M41
4252 IF WI=0 THEN FOR K=1 TO NM :W(K)=1 :NEXT K :GOTO 4260
4254 IF WI=2 THEN 4260
4256 FOR K=1 TO NM :Y=ABS(Y(K)) :IF Y<1E-15 THEN Y=1E-15
4258  W(K)=1/Y/Y :NEXT K
```

```
4260 REM ---------- COMPUTE X'WX AND WX'Y
4262 FOR I=1 TO NX
4264  P(I)=0 :FOR J=1 TO I :C(I,J)=0 :NEXT J
4266 NEXT I
4268 FOR K=1 TO NM
4270  FOR I=1 TO NX
4272   FOR J=1 TO I: C(I,J)=C(I,J)+W(K)*X(K,I)*X(K,J) :NEXT J
4274   P(I)=P(I)+W(K)*X(K,I)*Y(K)
4276  NEXT I
4278 NEXT K
4280 REM ---------- COVARIANCE MATRIX
4282 TR=1E-30 :FOR I=1 TO NX :C(I,0)=C(I,I) :NEXT I
4284 FOR I=1 TO NX
4286  IF C(I,0)<=TR THEN C(I,0)=1 ELSE C(I,0)=SQR(C(I,0))
4288 NEXT I
4290 FOR I=1 TO NX :FOR J=1 TO I
4292  C(I,J)=C(I,J)/C(I,0)/C(J,0)
4294 NEXT J :NEXT I
4296 REM ---------- RIDGE STEP
4298 FOR I=1 TO NX :FOR J=1 TO I
4300  C(I,J)=C(I,J)-RP*(I=J)
4302 NEXT J: NEXT I
4304 REM ---------- PRINCIPAL COMPONENT ANALYSIS OF THE COVARIANCE MATRIX
4306 N=NX
4308 FOR I=1 TO N :FOR J=1 TO I :A(I,J)=C(I,J) :NEXT J :NEXT I
4310 GOSUB 1800
4312 REM ---------- MATRIX INVERSION
4314 FOR I=1 TO N :FOR J=1 TO I :A(I,J)=C(I,J) :NEXT J :NEXT I
4316 GOSUB 1600 :IF ER=1 THEN 4358
4318 REM ---------- COMPUTE PARAMETER ESTIMATES
4320 FOR I=1 TO NX
4322  D=0 :FOR J=1 TO NX :D=D+A(I,J)/C(J,0)*P(J) :NEXT J :D(I)=D
4324 NEXT I
4326 FOR I=1 TO NX :P(I)=D(I)/C(I,0) :NEXT I
4328 REM ---------- WEIGHTED SUM OF SQUARES AND DURBIN-WATTSON STATISTICS
4330 FOR K=1 TO NM
4332  DE=D :D=Y(K) :FOR I=1 TO NX :D=D-P(I)*X(K,I) :NEXT I
4334  S2=S2+W(K)*D*D
4336  DN=DN+D*D :IF K>1 THEN DS=DS+(D-DE)*(D-DE)
4338 NEXT K
4340 NF=NM-NX :SE=SQR(S2/NF)
4342 IF DN<1E-30 THEN DS=2 ELSE DS=DS/DN
4344 REM ---------- STANDARD ERRORS AND CORRELATION MATRIX OF ESTIMATES
4346 FOR I=1 TO NX
4348  D(I)=SQR(S2/NF*A(I,I)/C(I,0)/C(I,0)) :C(0,I)=SQR(A(I,I))
4350 NEXT I
4352 FOR I=1 TO NX :FOR J=1 TO NX
4354  C(I,J)=A(I,J)/C(0,I)/C(0,J)
4356 NEXT J:NEXT I
4358 REM ---------- PRINT RESULTS
4360 V$=STRING$(70,"-") :F$="#.#####^^^^   " :F1$="#.###      "
4362 LPRINT TAB(20);"MULTIVARIABLE LINEAR REGRESSION"
4364 LPRINT TAB(25);"METHOD OF LEAST SQUARES"
4366 LPRINT :LPRINT :LPRINT
4368 LPRINT "NUMBER OF INDEPENDENT VARIABLES ..... ";NX
4370 LPRINT "NUMBER OF SAMPLE POINTS ............. ";NM
4372 IF RP<>0 THEN LPRINT "RIDGE PARAMETER ..................... ";RP
```

```
4374 LPRINT :LPRINT
4376 LPRINT "PRINCIPAL COMPONENT ANALYSIS OF THE CORRELATION MATRIX"
4378 LPRINT :LPRINT "EIGENVALUE";
4380 FOR I=1 TO NX :LPRINT TAB(11*I+3);" X(";I;") "; :  NEXT I :LPRINT :LPRINT
4382 FOR I=1 TO NX
4384  LPRINT USING F$;U(0,I);
4386  FOR J=1 TO NX :LPRINT USING F1$; U(J,I); :NEXT J :LPRINT
4388 NEXT I
4390 LPRINT :LPRINT
4392 IF ER<>1 THEN 4398
4394 LPRINT " SINGULAR COVARIANCE MATRIX OF INDEPENDENT VARIABLES"
4396 GOTO 4452
4398 LPRINT V$
4400 LPRINT " I"," Y MEAS"," WEIGHT"," Y COMP"," RESIDUAL" :LPRINT V$
4402 FOR K=1 TO NM
4404  Y=0 :FOR I=1 TO NX :Y=Y+P(I)*X(K,I) :NEXT I
4406 D=Y(K)-Y :LPRINT K, :LPRINT USING F$;Y(K),W(K),Y,D
4408 NEXT K :LPRINT V$ :LPRINT
4410 IF WI=0 THEN LPRINT "SUM OF SQUARES ...................... ";S2
4412 IF WI>0 THEN LPRINT "WEIGHTED SUM OF SQUARES ............. ";S2
4414 LPRINT "DEGREES OF FREEDOM .................. ";NF
4416 IF WI=0 THEN LPRINT "STANDARD ERROR ...................... ";SE
4418 IF WI>0 THEN LPRINT "SIGMA FACTOR IN THE WEIGHTS ......... ";SE
4420 LPRINT "DURBIN-WATSON D-STATISTICS .......... ";DS
4422 GOSUB 4100
4424 LPRINT "CRITICAL T-VALUE AT 95 % CONF. LEVEL  ";T
4426 LPRINT :LPRINT V$
4428 LPRINT "PARAMETER","ESTIMATE","ST.ERROR","LOWER BOUND","UPPER BOUND"
4430 LPRINT V$
4432 FOR I=1 TO NX
4434  LPRINT " P(";I;") ", :LPRINT USING F$;P(I),D(I),P(I)-T*D(I),P(I)+T*D(I)
4436 NEXT I
4438 LPRINT V$ :LPRINT
4440 LPRINT "CORRELATION MATRIX OF PARAMETERS" :LPRINT
4442 FOR I=1 TO NX :LPRINT TAB(11*I+3);" P(";I;") "; :NEXT I :LPRINT :LPRINT
4444 FOR I=1 TO NX
4446  LPRINT "P(";I;") ",
4448  FOR J=1 TO I :LPRINT USING F1$;C(I,J); :NEXT J :LPRINT
4450 NEXT I :LPRINT :LPRINT
4452 RETURN
4454 REM ****************************************************
```

When computing the estimates (3.23), the matrix $\mathbf{X}^T\mathbf{W}\mathbf{X}$ is already normalized, with unit entries in its diagonal. The modul M16 performs the inversion and returns the status flag ER = 1 if this step is not successful, i.e., the problem cannot be solved.

Example 3.2 Decomposing the rate constant of an acid-catalysed reaction

The hydrolysis of o-aceticacid-ethylester, described by

$CH_3C(OC_2H_5)_3 + H_2O \longrightarrow CH_3COOC_2H_5 + 2C_2H_5OH$

is a typical acid-catalysed reaction. As shown by Schwetlick (ref. 8), in the presence of the weak acid $NO_2C_6H_4OH$ and at constant ionic strength the rate constant k of the reaction can be decomposed as

$$k = k_o + k_H[H^+] + k_{HA}[HA] , \tag{3.33}$$

where k_o is the rate constant of the uncatalysed reaction, whereas k_H and k_{HA} are catalysis constants that measure the influence of the hydrogen ion concentration $[H^+]$ and that of the undissociated acid concentration $[HA]$, respectively. In our case HA is $NO_2C_6H_4OH$. Table 3.1, originally published in (ref. 9), lists the rate constants observed at different values of $[H^+]$ and $[HA]$. Column 3 of the table will be used only in a forthcoming investigation of Section 3.5.

Table 3.1
Rate constant of an acid-catalysed reaction

Experimental conditions			$k\times10^4$
$[H^+]\times10^9$ mol/l	$[HA]\times10^3$ mol/l	$[HA^-]\times10^3$ mol/l	1/s
4.8	2.42	2.42	1.21
4.8	5.66	5.66	1.20
4.8	16.00	16.00	1.35
4.8	21.21	20.20	1.44
6.5	3.84	2.84	1.54
6.5	10.25	7.56	1.61
6.5	18.30	13.50	1.77
10.2	3.10	1.45	2.37
10.2	10.30	4.83	2.47
10.2	30.90	14.50	2.84

We present a simple main program to estimate the parameters k_o, k_H and k_{HA} by the unweighted least squares method. The program can be used for solving other linear regression problems if altering the DATA statements appropriately. The first DATA line specifies the sample size and the number of independent variables. The observations are listed in separate DATA lines, where the first number is the dependent variable. The second number equals 1 and will result in the constant term k_o of the model (3.33). This is followed by the values of $[H^+]$ and $[HA]$.

```
100 REM -----------------------------------------------------------
102 REM EX. 3.2. MULTIVARIABLE LINEAR REGRESSION - ACID CATALYSIS
104 REM MERGE M16,M18,M41,M42
106 REM (NUMBER OF SAMPLE POINTS AND NUMBER OF INDEP. VARIABLES)
108 DATA 10,3
110 REM (DEPENDENT VARIABLE AND INDEPENDENT VARIABLES)
112 DATA   1.21E-4,   1,    4.8E-9,    0.00242
114 DATA   1.20E-4,   1,    4.8E-9,    0.00566
116 DATA   1.35E-4,   1,    4.8E-9,    0.01600
118 DATA   1.44E-4,   1,    4.8E-9,    0.02121
120 DATA   1.54E-4,   1,    6.5E-9,    0.00384
122 DATA   1.61E-4,   1,    6.5E-9,    0.01025
124 DATA   1.77E-4,   1,    6.5E-9,    0.01830
126 DATA   2.37E-4,   1,   10.2E-9,    0.00310
128 DATA   2.47E-4,   1,   10.2E-9,    0.01030
130 DATA   2.84E-4,   1,   10.2E-9,    0.03090
200 REM ---------- READ DATA
202 READ NM,NX
204 DIM X(NM,NX),Y(NM),W(NM),P(NX)
206 DIM A(NX,NX),C(NX,NX),U(NX,NX),D(NX)
208 FOR I=1 TO NM
210  READ Y(I)
212  FOR J=1 TO NX :READ X(I,J) :NEXT J
214 NEXT I
216 REM ---------- CALL MODULE (NO WEIGHTING AND NO RIDGE)
218 WI=0 :RP=0
220 GOSUB 4200
222 STOP
```

The first part of the output contains the principal component analysis of the correlation matrix discussed later in Section 3.5. In addition to the residuals, goodness-of-fit, parameter estimates and bounds, the Durbin-Wattson D statistics is also printed by the module.

```
                    MULTIVARIABLE LINEAR REGRESSION
                        METHOD OF LEAST SQUARES



NUMBER OF INDEPENDENT VARIABLES .....  3
NUMBER OF SAMPLE POINTS .............  10


PRINCIPAL COMPONENT ANALYSIS OF THE CORRELATION MATRIX

EIGENVALUE    X( 1 )      X( 2 )      X( 3 )

0.27101E+01   0.589       0.588       0.554
0.24102E+00   -.374       -.410       0.832
0.48887E-01   -.716       0.698       0.022
```

```
----------------------------------------------------------------------
 I          Y MEAS       WEIGHT       Y COMP       RESIDUAL
----------------------------------------------------------------------
 1          0.12100E-03  0.10000E+01  0.11480E-03  0.62011E-05
 2          0.12000E-03  0.10000E+01  0.11993E-03  0.70540E-07
 3          0.13500E-03  0.10000E+01  0.13630E-03  -.13030E-05
 4          0.14400E-03  0.10000E+01  0.14455E-03  -.55316E-06
 5          0.15400E-03  0.10000E+01  0.15513E-03  -.11318E-05
 6          0.16100E-03  0.10000E+01  0.16528E-03  -.42821E-05
 7          0.17700E-03  0.10000E+01  0.17803E-03  -.10294E-05
 8          0.23700E-03  0.10000E+01  0.23685E-03  0.15071E-06
 9          0.24700E-03  0.10000E+01  0.24825E-03  -.12506E-05
 10         0.28400E-03  0.10000E+01  0.28087E-03  0.31289E-05
----------------------------------------------------------------------

SUM OF SQUARES ......................  7.251703E-11
DEGREES OF FREEDOM ..................  7
STANDARD ERROR ......................  3.21863E-06
DURBIN-WATSON D-STATISTICS ..........  1.150208
CRITICAL T-VALUE AT 95 % CONF. LEVEL   2.37

----------------------------------------------------------------------
PARAMETER     ESTIMATE     ST.ERROR     LOWER BOUND  UPPER BOUND
----------------------------------------------------------------------
 P( 1 )       0.34346E-05  0.34061E-05  -.46378E-05  0.11507E-04
 P( 2 )       0.22403E+05  0.45859E+03  0.21316E+05  0.23489E+05
 P( 3 )       0.15835E-02  0.11693E-03  0.13064E-02  0.18606E-02
----------------------------------------------------------------------

CORRELATION MATRIX OF PARAMETERS

              P( 1 )     P( 2 )     P( 3 )

P( 1 )        1.000
P( 2 )        -.861      1.000
P( 3 )        -.257      -.173      1.000
```

The standard error is about 2%, which is certainly not larger than the error in the observed rate coefficients. Therefore, the fit is acceptable in spite of some nonrandomness in the sequence of residuals. This conclusion is supported by the acceptable value of D-statistics, athough with only 10 data points we cannot use this test rigorously.

Though the confidence intervals of the parameters are reasonably small, the interval for k_o includes the value $k_o = 0$, and hence at the given significance level we cannot reject the hypothesis $k_o = 0$. Indeed, fitting a simplified model $k = k_H[H^+] + k_{HA}[HA]$ to the data yields the standard error $s = 3.22\times10^{-6}$, so that the goodness-of-fit is practically unchanged. Dropping the constant term is supported by an F-test at any reasonable significance level. On the other hand, a model containing even more than three terms might seem to be natural from a chemist's point of view. We will return to this question in Section 3.5.1.

Exercises

- □ Apply (3.23) and (3.30) to the model $y = p_1x + p_2$. Compare the resulting expressions with the corresponding expressions of Section 3.1.
- □ Discuss the relation between transforming the matrix $\mathbf{X}^T\mathbf{WX}$ into a correlation type matrix and scaling of the parameters.
- □ Solve Example 3.2 with the simplified model $k = k_H[H^+] + k_{HA}[HA]$ without weighting, then, in turn, apply relative weighting and user specified weights $w_i = 1/\tilde{k}_i$ (also called Poisson weighting).
- □ Fit a parabol $y = p_1 + p_2x + p_3x^2$ to the data of Examples 1.8.2, 1.8.3 and 3.1 using the program module M42. (See Section 3.9 for a more straightforward solution of this problem.)

3.3 NONLINEAR LEAST SQUARES

In this Section we estimate the parameters of the nonlinear vector valued function

$$y = f(x,p) \tag{3.34}$$

given by ny functions as

$$\begin{aligned} y_1 &= f_1(x,p) \\ &\vdots \\ y_{ny} &= f_{ny}(x,p) \ . \end{aligned}$$

The model is fitted to the observations $\{ (x_{i1}, \ldots, x_{i,nx}; \tilde{y}_{i1}, \ldots, \tilde{y}_{i,ny}),$ $i = 1,\ldots,nm \}$. Let $\epsilon_i = (\epsilon_{i1}, \ldots, \epsilon_{i,ny})^T$ denote the error vector in the i-th observation. We assume that the ny×ny covariance matrices of the error vectors are known, at least up to a constant factor. The nm weighting matrices of dimensions ny×ny are selected according to

$$\mathrm{cov}\{ \epsilon_i \} = \sigma^2 \mathbf{W}_i^{-1} , \tag{3.36}$$

where σ^2 is a (possibly unknown) scalar multiplier. Note that nondiagonal $\mathbf{W}_i$ matrices are also allowed. The least squares objective function

$$Q(p) = \sum_{i=1}^{nm} [\tilde{y}_i - f(x_i,p)]^T W_i [\tilde{y}_i - f(x_i,p)] \qquad (3.37)$$

is in agreement with the maximum likelihood principle.

For the sake of simplicity we introduce the notations

$$\tilde{Y} = \begin{bmatrix} \tilde{y}_1 \\ . \\ . \\ . \\ \tilde{y}_{nm} \end{bmatrix}, \quad F(p) = \begin{bmatrix} f(x_1,p) \\ . \\ . \\ . \\ f(x_{nm},p) \end{bmatrix}, \quad W = \begin{bmatrix} W_1 & & & \\ & . & & \\ & & . & \\ & & & . \\ & & & W_{nm} \end{bmatrix}, \qquad (3.38)$$

therby reducing the objective function to the form

$$Q(p) = [\tilde{Y} - F(p)]^T W [\tilde{Y}_i - F(p)] \ . \qquad (3.39)$$

The minimum of (3.39) can be localized by the methods discussed in Section 2.4. As shown in many comparative studies (see, e.g., refs. 10-12), apart from some special cases (ref. 13) the most efficient algorithms to minimize sum-of-squares objective functions are the various versions of the Gauss-Newton method. The method is based on the local linear approximation

$$F(p) \cong F(p^{(o)}) + J(p^{(o)})[p - p^{(o)}] \qquad (3.40)$$

of the function F around the initial estimate $p^{(o)}$ of the parameters. The (nm×ny)×np Jacobian matrix J of F is defined by

$$J(p) = \begin{bmatrix} \frac{\partial f(x_1 \ ,p)}{\partial p_1} & . \ . \ . & \frac{\partial f(x_1 \ ,p)}{\partial p_{np}} \\ \frac{\partial f(x_2 \ ,p)}{\partial p_1} & . \ . \ . & \frac{\partial f(x_2 \ ,p)}{\partial p_{np}} \\ \vdots & & \\ \frac{\partial f(x_{nm},p)}{\partial p_1} & . \ . \ . & \frac{\partial f(x_{nm},p)}{\partial p_{np}} \end{bmatrix} . \qquad (3.41)$$

Setting (3.40) into (3.39) yields the quadratic approximation

$$\tilde{Q}(p) = \left[\ \tilde{Y} - F - J(p - p^{(o)}) \ \right]^T W \ \left[\ \tilde{Y} - F - J(p - p^{(o)}) \ \right] \qquad (3.42)$$

of the objective function, where the argument $p^{(o)}$ of F and J is dropped for notational simplicity. The next estimate $p^{(1)}$ is then the minimum point of the quadratic function (3.42), which is easy to find. Indeed, regarding $\Delta p = p - p^{(o)}$ as the unknown parameter vector, minimization of (3.42) is equivalent to a linear regression problem with the vector of dependent variables $\tilde{Y} - F$ and the matrix of independent variables **J**. The solution to this problem is $\Delta p = [J^TWJ]^{-1}J^TW\,[\tilde{Y} - F]$. Repeated application of this idea yields the Gauss-Newton iteration

$$p^{(k+1)} = p^{(k)} + [J^TWJ]^{-1}J^TW\,[\tilde{Y} - F]\ , \qquad (3.43)$$

where J and F are computed at $p^{(k)}$. Similarly to the quasi Newton optimization methods, the Gauss-Newton algorithm offers quadratic convergence close to the minimum. Further apart, however, the step size is frequently inflated, particularly when $[J^TWJ]$ is nearly singular. Then $p^{(k+1)}$ might be a worse approximation to the minimum of (3.39) than $p^{(k)}$ itself. The goal of the famous Levenberg-Marquardt modification (refs. 14-15) of the Gauss-Newton algorithm is to overcome this disadvantage through the iteration

$$p^{(k+1)} = p^{(k)} + [J^TWJ + \lambda^{(k+1)}I]^{-1}J^TW\,[\tilde{Y} - F]\ , \qquad (3.44)$$

where I is the np×np unit matrix and the nonnegative scalar $\lambda^{(k+1)}$ is the Marquardt parameter. With λ sufficiently large, the additional term moderates the length of the step and forces its direction toward the negative gradient of the objective function. A variety of rules has been proposed for selecting the Marquardt parameter in subsequent iterations (refs. 5,12). In a convergent iteration most of the methods decrease its value, thereby returning to the Gauss-Newton procedure.

Analogously to the linear case, the goodness-of-fit is measured in terms of the residual sum of squares $Q(\hat{p})$ and the residual variance (or sigma square) s^2 , defined by (3.24) with the degrees (nm×ny − np) of freedom in the denominator. Interpretation of estimates is based on the observation that each iteration of the Gauss-Newton algorithm is equivalent to solving a linear regression problem. Replacing the matrix **X** in (3.30) by the Jacobian $J(\hat{p})$, corresponding to the linear approximation of the response function F in a neighborhood of $\hat{p}$, the covariance matrix of estimates is approximated by

$$C_p = s^2\,[\,J^T(\hat{p})\,W\,J(\hat{p})\,]^{-1}\ . \qquad (3.45)$$

Based on the same linear approximation, the confidence region is described by (3.32) as in the linear case. This is an approximate relationship, and may considerably differ from the exact confidence region given by

$Q(p) - Q(\hat{p}) \le X^2$, where X^2 depends on the probability level, (see, e.g., Bard, ref. 5). The exact confidence region has little practical value for np > 2, since it is very difficult to compute, whereas the local linear approximation (3.32) will be very useful.

The following simple tricks improve the efficiency of the Gauss-Newton-Marquardt algorithm implemented in the module M45.

(i) The parameters are normalized. In the (k+1)-th iteration the minimum is localized in the space of the parameters defined by $\beta_j = p_j/p_j^{(k)}$. Therefore, the initial guess is $\beta_j = 1$ in every iteration, and the entries of the Jacobian matrix are

$$\frac{\partial f_l(x_i,\beta)}{\partial \beta_j} = \frac{\partial f_l(x_i,p)}{\partial p_j} p_j^{(k)} . \qquad (3.46)$$

In spite of the definition of β_j, according to (3.46) we never divide by p_j. Thus you can choose the initial estimate $p_j = 0$, but then the j-th parameter remains zero during the iterations.

(ii) The cross product matrix $[J^T(\beta)WJ(\beta)]$ is further normalized to a correlation type matrix before inversion. At this point we leave a diagonal entry unchanged if it is less than a threshold selected relatively to the trace of the matrix. The idea behind this trick is to allow the additional term $\lambda^{(k+1)}I$ to eliminate the possible near singularity of the matrix to be inverted.

(iii) The above normalization enables us to use simple rules for selecting the Marquardt parameter.

Initially $\lambda^{(o)} = 0.01$, whereas in subsequent iterations

$\lambda^{(k+1)} = 0.1\lambda^{(k)}$ if $Q(p^{(k+1)}) < Q(p^{(k)})$, and

$\lambda^{(k+1)} = 10\lambda^{(k)}$ otherwise.

(iv) The sign of the parameters are usually known from physical considerations. Restricting $\beta_j \ge 0$ we keep the sign of the starting estimate of the parameters.

The termination conditions are $\|\Delta\beta_j^{(k)}\| \le EP$ or $k > IM$, where EP is the selected lower bound on the relative step size, and IM is the maximum number of iterations.

Program module M45

```
4500 REM ****************************************************
4502 REM * WEIGHTED LEAST SQUARES ESTIMATION OF PARAMETERS *
4504 REM *       IN MULTIVARIABLE NONLINEAR MODELS         *
4506 REM *       GAUSS - NEWTON - MARQUARDT METHOD         *
4508 REM ****************************************************
4510 REM INPUT:
4512 REM     NM      NUMBER OF SAMPLE POINTS
4514 REM     NX      NUMBER OF INDEPENDENT VARIABLES
4516 REM     NY      NUMBER OF DEPENDENT VARIABLES
4518 REM     NP      NUMBER OF PARAMETERS
4520 REM   T(NM,NX)  TABLE OF INDEPENDENT VARIABLES
4522 REM   V(NM,NY)  TABLE OF DEPENDENT VARIABLES
4524 REM     WI      IDENTIFIER OF WEIGHTING OPTIONS
4526 REM              0  IDENTICAL WEIGHTS ( W(I,I)=1, W(I,J)=0 )
4528 REM              1  RELATIVE WEIGHTS ( W(I,I)=1/V(M,I)^2, W(I,J)=0 )
4530 REM              2  USER-SPECIFIED WEIGHTS GIVEN BY FURTHER INPUT AS
4532 REM   W(NY,NY)  MATRIX OF WEIGHTING COEFFICIENTS ( ONLY FOR WI=2 )
4534 REM              3  WEIGHTS COMPUTED FOR SAMPLE POINT M IN USER
4536 REM                 SUPPLIED SUBROUTINE STARTING AT LINE 800
4538 REM    P(NP)    INITIAL PARAMETER ESTIMATES
4540 REM     EP      THRESHOLD ON RELATIVE STEP LENGTH
4542 REM     IM      MAXIMUM NUMBER OF ITERATIONS
4544 REM OUTPUT:
4546 REM     ER      STATUS FLAG
4548 REM               0  SUCCESSFUL ESTIMATION
4550 REM               1  REQUIRED THRESHOLD NOT ATTAINED
4552 REM    P(NP)    PARAMETER ESTIMATES
4554 REM    .....    FURTHER RESULTS ARE PRINTED IN THE MODULE
4556 REM USER-SUPPLIED SUBROUTINES:
4558 REM   FROM LINE 900:
4560 REM             X(1,...,nx) AND P(1,...,np)  -->  Y(1,...,ny)
4562 REM                    ( RESPONSE FUNCTION EVALUATION )
4564 REM   FROM LINE 800:
4566 REM             M --> W(1,...,ny;1,...,ny)
4568 REM             ( COMPUTE ACTUAL WEIGHTS FOR SAMPLE M
4570 REM               CALLED ONLY IF WI=3 )
4572 REM AUXILIARY ARRAYS:
4574 REM  A(NP,NP),C(NP,NP),U(NP,NP),B(NP),D(NP),G(NY,NP)
4576 REM MODULES CALLED: M16,M18,M41
4578 IF WI<>0 THEN 4582
4580 FOR I=1 TO NY :FOR J=1 TO NY :W(I,J)=-(I=J) :NEXT J :NEXT I
4582 REM ---------- STARTING VALUE OF MARQUARDT'S LAMDA IS 0.01
4584 PM=.01 :EI=0 :ES=0
4586 REM ---------- SUM OF SQUARES
4588 GOSUB 4760
4590 REM ---------- START OF ITERATION
4592 LPRINT :LPRINT "STARTING POINT";TAB(25);"SUM SQ=";F :LPRINT
4594 FOR K=1 TO NP :LPRINT TAB(25);"P(";K;")=";P(K) :NEXT K
4596 FOR IT=1 TO IM
4598  FOR K=1 TO NP :U(K,0)=P(K) :NEXT K :FR=F
4600  REM ---------- COMPUTE T'WT AND WT'Y
4602  FOR K=1 TO NP :B(K)=0 :FOR L=1 TO K :C(K,L)=0 :NEXT L :NEXT K
4604  FOR M=1 TO NM
4606   FOR I=1 TO NX :X(I)=T(M,I) :NEXT I
4608   IF WI=1 THEN GOSUB 4784
4610   IF WI=3 THEN ER=0 :GOSUB 800 :IF ER>0 THEN 4932
4612   GOSUB 4792
```

```
4614   FOR K=1 TO NP
4616    FOR L=1 TO K
4618     A=0
4620     FOR I=1 TO NY:FOR J=1 TO NY
4622      A=A+W(I,J)*G(I,L)*G(J,K)*P(L)*P(K)
4624     NEXT J :NEXT I :C(K,L)=C(K,L)+A
4626    NEXT L
4628    A=0
4630    FOR I=1 TO NY:FOR J=1 TO NY
4632     A=A+W(I,J)*G(J,K)*(V(M,I)-Y(I))*P(K)
4634    NEXT J :NEXT I :B(K)=B(K)+A
4636   NEXT K
4638  NEXT M
4640  REM ---------- NORMALIZE
4642  TR=0 :FOR I=1 TO NP :C(I,0)=C(I,I) :TR=TR+C(I,I) :NEXT I
4644  TR=TR/NP/1000
4646  FOR I=1 TO NP
4648   IF C(I,0)<=TR THEN C(I,0)=1 ELSE C(I,0)=SQR(C(I,0))
4650  NEXT I
4652  FOR I=1 TO NP :FOR J=1 TO I
4654   U(I,J)=C(I,J) :C(I,J)=C(I,J)/C(I,0)/C(J,0)
4656  NEXT J :NEXT I
4658  REM ---------- MARQUARDT'S COMPROMISE
4660  FOR I=1 TO NP
4662   FOR J=1 TO I-1 :A(I,J)=C(I,J) :NEXT J
4664   A(I,I)=C(I,I)+PM
4666  NEXT I
4668  REM ---------- MATRIX INVERSION
4670  ER=0 :N=NP :GOSUB 1600 :IF ER=1 THEN 4718
4672  REM ---------- COMPUTE STEP
4674  FOR I=1 TO NP
4676   D=0 :FOR J=1 TO NP :D=D+A(I,J)/C(J,0)*B(J) :NEXT J :D(I)=D/C(I,0)
4678  NEXT I
4680  REM ---------- CHECK SIGN AND REDUCE STEP IF NEEDED
4682  SL=0 :XI=1
4684  FOR I=1 TO NP
4686   IF XI*D(I)<=-.95 THEN XI=-.95/D(I)
4688   SL=SL+D(I)*D(I)
4690  NEXT I :SL=SQR(SL)*XI
4692  REM ---------- NEW ESTIMATES
4694  FOR I=1 TO NP :P(I)=U(I,0)*(1+XI*D(I)) :NEXT I
4696  GOSUB 4760
4698  REM ---------- PRINT ITERATION STEP
4700  F$="#.#^^^^" :LPRINT
4702  LPRINT "IT=";IT;TAB(10);"PM="; :LPRINT USING F$;PM;
4704  LPRINT TAB(25);"SUM SQ=";F;TAB(50);"SL=";SL :LPRINT
4706  IF F>=FR THEN 4710
4708  FOR K=1 TO NP :LPRINT TAB(25);"P(";K;")=";P(K) :NEXT K
4710  REM ---------- END OF PRINT
4712  IF SL<=EP THEN EI=0 :GOTO 4726
4714  REM ---------- MARQUARDT'S PARAMETER
4716  IF F<FR THEN 4720
4718  PM=10*PM :GOTO 4658
4720  PM=PM/10 :IF PM<.000001 THEN PM=.000001
4722 NEXT IT
4724 EI=1
4726 IF F<FR THEN 4730
4728 F=FR :FOR I=1 TO NP :P(I)=U(I,0) :NEXT I
```

```
4730 REM ---------- STANDARD ERROR AND CORRELATION MATRIX OF PARAMETERS
4732 NF=NM*NY-NP :SE=SQR(F/NF)
4734 FOR I=1 TO NP :FOR J=1 TO I :A(I,J)=C(I,J) :NEXT J:NEXT I
4736 GOSUB 1600 :IF ER=1 THEN ES=1 :GOTO 4752
4738 FOR I=1 TO NP
4740  B(I)=SQR(F/NF*A(I,I)/C(I,0)/C(I,0))
4742  C(0,I)=SQR(A(I,I))
4744 NEXT I
4746 FOR I=1 TO NP :FOR J=1 TO  NP
4748  C(I,J)=A(I,J)/C(0,I)/C(0,J)
4750 NEXT J:NEXT I
4752 REM ---------- PRINCIPAL COMPONENT ANALYSIS
4754 FOR I=1 TO NP :FOR J=1 TO I :A(I,J)=U(I,J) :NEXT J :NEXT I
4756 N=NP :GOSUB 1800
4758 GOTO 4810
4760 REM ---------- COMPUTE SSQ
4762 F=0
4764 FOR M=1 TO NM
4766  FOR I=1 TO NX :X(I)=T(M,I) :NEXT I
4768  IF WI=1 THEN GOSUB 4784
4770  IF WI=3 THEN GOSUB 800
4772  GOSUB 900
4774  FOR I=1 TO NY :FOR J=1 TO NY
4776   F=F+W(I,J)*(V(M,I)-Y(I))*(V(M,J)-Y(J))
4778  NEXT J :NEXT I
4780 NEXT M
4782 RETURN
4784 REM ---------- RELATIVE WEIGHTS
4786 FOR I=1 TO NY :Y=ABS(V(M,I)) :IF Y<1E-15 THEN Y=1E-15
4788  W(I,I)=1/Y/Y :NEXT I
4790 RETURN
4792 REM ---------- COMPUTE JACOBI MATRIX G(NY,NP) AND RESPONSE Y(NY)
4794 FOR J=1 TO NP
4796  DE=.001*ABS(P(J))+1E-10 :P(J)=P(J)+DE :GOSUB 900
4798  FOR I=1 TO NY :G(I,J)=Y(I)/DE :NEXT I
4800  P(J)=P(J)-DE :D(J)=DE
4802 NEXT J
4804 GOSUB 900
4806 FOR I=1 TO NY :FOR J=1 TO NP :G(I,J)=G(I,J)-Y(I)/D(J) :NEXT J: NEXT I
4808 RETURN
4810 REM ---------  PRINT RESULTS
4812 LPRINT :LPRINT
4814 LPRINT TAB(15);"WEIGHTED LEAST SQUARES PARAMETER ESTIMATION"
4816 LPRINT TAB(21);"IN MULTIVARIABLE NONLINEAR MODELS"
4818 LPRINT TAB(21);"GAUSS - NEWTON - MARQUARDT METHOD"
4820 LPRINT :LPRINT :LPRINT
4822 LPRINT "NUMBER OF INDEPENDENT VARIABLES ..... ";NX
4824 LPRINT "NUMBER OF DEPENDENT VARIABLES ....... ";NY
4826 LPRINT "NUMBER OF PARAMETERS ................ ";NP
4828 LPRINT "NUMBER OF SAMPLE POINTS ............. ";NM
4830 LPRINT "OPTION OF WEIGHTING ................. ";WI;
4832 IF WI=0 THEN LPRINT "(IDENTICAL WEIGHTS)"
4834 IF WI=1 THEN LPRINT "(RELATIVE WEIGHTS)"
4836 IF WI=2 THEN LPRINT "(USER DEFINED WEIGHTS, INDEPENDENT ON THE SAMPLE"
4838 IF WI=3 THEN LPRINT "(USER DEFINED WEIGHTS, DEPENDENT ON THE SAMPLE"
4840 F$="#.#####^^^^   " :F1$="#.###      ":LPRINT :LPRINT
4842 LPRINT "PRINCIPAL COMPONENT ANALYSIS OF NORMED CROSS PRODUCT MATRIX"
```

```
4844 LPRINT :LPRINT "EIGENVALUE";
4846 FOR I=1 TO NP :LPRINT TAB(11*I+3);" P(";I;") "; :  NEXT I :LPRINT :LPRINT
4848 FOR I=1 TO NP
4850  LPRINT USING F$;U(0,I),
4852  FOR J=1 TO NP :LPRINT USING F1$; U(J,I); :NEXT J :LPRINT
4854 NEXT I
4856 LPRINT :LPRINT
4858 V$=STRING$(70,"-") :V1$=STRING$(55,"-")
4860 IF EI=1 THEN LPRINT " REQUIRED THRESHOLD NOT ATTAINED" :LPRINT :LPRINT
4862 IF ES=1 THEN LPRINT " SINGULAR CROSS PRODUCT MATRIX" :LPRINT :LPRINT
4864 FOR I=1 TO NY
4866  LPRINT :IF NY>1 THEN LPRINT "RESPONSE FUNCTION";I
4868 LPRINT V1$ :LPRINT "SAMPLE No"," Y MEAS"," Y COMP"," RESIDUAL" :LPRINT V1$
4870  FOR M=1 TO NM
4872   FOR J=1 TO NX :X(J)=T(M,J) :NEXT J
4874   GOSUB 900
4876   LPRINT M, :LPRINT USING F$;V(M,I),Y(I),V(M,I)-Y(I)
4878  NEXT M :LPRINT V1$
4880 NEXT I :LPRINT :LPRINT
4882 IF WI=0 THEN LPRINT "SUM OF SQUARES ...................... ";F
4884 IF WI>0 THEN LPRINT "WEIGHTED SUM OF SQUARES ............. ";F
4886 LPRINT "DEGREES OF FREEDOM .................. ";NF
4888 IF WI=0 THEN LPRINT "STANDARD ERROR ...................... ";SE
4890 IF WI>0 THEN LPRINT "SIGMA FACTOR IN THE WEIGHTS ......... ";SE
4892 GOSUB 4100
4894 LPRINT "CRITICAL T-VALUE AT 95 % CONF. LEVEL  ";T
4896 LPRINT :LPRINT V$
4898 LPRINT "PARAMETER", "ESTIMATE",
4900 IF ES=0 THEN LPRINT "ST. ERROR","LOWER BOUND","UPPER BOUND";
4902 LPRINT :LPRINT V$
4904 FOR I=1 TO NP
4906  LPRINT " P(";I;") ", :LPRINT USING F$;P(I),
4908  PB=ABS(B(I)*P(I))
4910  IF ES=0 THEN LPRINT USING F$;PB,P(I)-T*PB,P(I)+T*PB;
4912  LPRINT
4914 NEXT I
4916 LPRINT V$ :LPRINT
4918 IF ES=1 THEN ER=1 :GOTO 4932
4920 LPRINT "CORRELATION MATRIX OF PARAMETERS" :LPRINT
4922 FOR I=1 TO NP :LPRINT TAB(11*I+3);" P(";I;") "; :NEXT I :LPRINT :LPRINT
4924 FOR I=1 TO NP
4926  LPRINT "P(";I;") ",
4928  FOR J=1 TO I :LPRINT USING F1$;C(I,J); :NEXT J :LPRINT
4930 NEXT I :LPRINT :LPRINT
4932 RETURN
4934 REM ****************************************************
```

The role of the input data NM, NX, NY and NP is obvious from the text and the remark lines, but the array T(NM,NX) of independent variables deserves some explanation. Each line of the array should contain all information that enables us to compute the value of the dependent variables for a sample point at the current values of the parameters. Therefore, the module transfers the appropriate row of T(NM,NX) into the vector X(NX) for further use in the user supplied subroutine. This subroutine starting at line 900 computes the independent variables Y(NY) at the current parameters P(NP) and independent variables X(NX). If the model consists only of one response function, then NY = 1 and only Y(1) is evaluated in the user subroutine. The observed values of the dependent variables are stored in the array V(NM,NY). If there is only one response function, this array consists of one column.

There are four weighting options. No weighting (WI = 0) and relative weighting (WI = 1) are easy to use, because the weights are generated automatically. You should remember, however, that relative weighting is not recommended if any observed value is near to zero. With the option WI = 2 you should provide an NY×NY matrix of weights in the array W(NY,NY). The same weighting matrix will be then used in all sample points.

You may also wish to use different weighting matrices for different observations. For this purpose the weighting option WI = 3 is provided. To use this option you must supply a second subroutine starting at line 800 , where you have access to the index M of the current sample point. The task of the second routine is to compute the NY×NY weighting matrix for the current sample point and to place it into the array W.

Selecting the initial estimates of the parameters P(NP) you should keep in mind that their signs remain unchanged during the iterations. For a first try it is reasonable to set a low limit on the number of iterations, say IM = 5 , and to use a moderate value, say 0.01 or 0.001 , for EP.

The subroutine between lines 4792 - 4808 provides divided difference approximation of the appropriate segment of the Jacobian matrix, stored in the array G(NY,NP). In some applications the efficiency of the minimization can be considerably increased replacing this general purpose routine by analytical derivatives for the particular model. In that case, however, Y(NY) should be also updated here.

Example 3.3 Fitting a nonlinear rate expression

Rational functions are frequently encountered as rate expressions of catalytic reactions. In addition, the function

$$y = p_1 + x_1/(p_2x_2 + p_3x_3) \qquad (3.48)$$

is a popular test problem for comparing parameter estimation procedures (refs. 10,12). In this case we have only one response function, three independent variables and three parameters. Line 110 of the following main program specifies these values, together with the number NM = 15 of observations.

The 15 DATA lines starting at line 114 correspond to the 15 observation points. The values of the dependent variable and of the independent variables can be easily reconstructed from the listing. Since NY = 1 , the subroutine starting at line 900 computes the single value Y(1). Selecting the unweighted option WI = 0 we do not need the second user subroutine. The starting estimate of the parameters is given in line 220.

```
100 REM ------------------------------------------------------------
102 REM EX. 3.3. NONLINEAR LSQ PARAMETER ESTIMATION - BARD EXAMPLE
104 REM MERGE M16,M18,M41,M45
106 REM ---------- DATA
108 REM (NM,  NY, NX, NP)
110 DATA 15,   1,  3,  3
112 REM    ( Y,  X1, X2, X3 )
114 DATA   0.14,  1, 15,  1
116 DATA   0.18,  2, 14,  2
118 DATA   0.22,  3, 13,  3
120 DATA   0.25,  4, 12,  4
122 DATA   0.29,  5, 11,  5
124 DATA   0.32,  6, 10,  6
126 DATA   0.35,  7,  9,  7
128 DATA   0.39,  8,  8,  8
130 DATA   0.37,  9,  7,  7
132 DATA   0.58, 10,  6,  6
134 DATA   0.73, 11,  5,  5
136 DATA   0.96, 12,  4,  4
138 DATA   1.34, 13,  3,  3
140 DATA   2.10, 14,  2,  2
142 DATA   4.39, 15,  1,  1
200 REM ---------- READ DATA
202 READ NM,NY,NX,NP
204 DIM T(NM,NX),V(NM,NY),P(NP),X(NX),Y(NY),W(NY,NY)
206 DIM A(NP,NP),C(NP,NP),U(NP,NP),B(NP),D(NP),G(NY,NP)
208 FOR I=1 TO NM
210  FOR J=1 TO NY :READ V(I,J) :NEXT J
212  FOR J=1 TO NX :READ T(I,J) :NEXT J
214 NEXT I
216 REM ---------- CALL NONLINEAR LSQ ESTIMATION MODULE
218 WI=0 :EP=.0001  :IM=20
220 P(1)=1 :P(2)=1 :P(3)=1
222 GOSUB 4500
224 STOP
900 REM ---------- FUNCTION EVALUATION
902 Y(1)=P(1)+X(1)/(P(2)*X(2)+P(3)*X(3))
904 RETURN
```

According to the following output, the module needed six iterations to find the minimum of the objective function. The value of the Marquardt parameter PM, i.e., $\lambda^{(k)}$ is gradually decrased. In iterations 5 and 6 several attempts with different Marquardt parameters are necessary to improve the objective function. In less cooperative estimation problems the module frequently needs to increase the Marquardt parameter. The current value of the sum of squares, i.e., the objective function and the relative step length SL are also printed in every iteration.

If a less conservative termination criterion, say EP = 0.001 were used, the procedure would be stopped after the 5-th iteration as seen from the value of SL.

```
STARTING POINT       SUM SQ= 41.6817

                     P( 1 )= 1
                     P( 2 )= 1
                     P( 3 )= 1

IT= 1    PM=0.1E-01  SUM SQ= 1.345128        SL= 1.079673

                     P( 1 )= .1061849
                     P( 2 )= 1.42408
                     P( 3 )= 1.43237

IT= 2    PM=0.1E-02  SUM SQ= 3.852356E-02    SL= .3596549

                     P( 1 )= 9.080309E-02
                     P( 2 )= 1.47196
                     P( 3 )= 1.90143

IT= 3    PM=0.1E-03  SUM SQ= 8.241143E-03    SL= .326665

                     P( 1 )= 8.347398E-02
                     P( 2 )= 1.144983
                     P( 3 )= 2.330202

IT= 4    PM=0.1E-04  SUM SQ= 8.214884E-03    SL= 1.660105E-02

                     P( 1 )= 8.244001E-02
                     P( 2 )= 1.133951
                     P( 3 )= 2.342825

IT= 5    PM=0.1E-05  SUM SQ= 8.214884E-03    SL= 7.513525E-04


IT= 5    PM=0.1E-04  SUM SQ= 8.214876E-03    SL= 7.476011E-04

                     P( 1 )= 8.241451E-02
                     P( 2 )= 1.133249
                     P( 3 )= 2.343488

IT= 6    PM=0.1E-05  SUM SQ= 8.214885E-03    SL= 2.55171E-04


IT= 6    PM=0.1E-04  SUM SQ= 8.214888E-03    SL= 2.539624E-04


IT= 6    PM=0.1E-03  SUM SQ= 8.214876E-03    SL= 2.426527E-04


IT= 6    PM=0.1E-02  SUM SQ= 8.214886E-03    SL= 1.711297E-04


IT= 6    PM=0.1E-01  SUM SQ= 8.214867E-03    SL= 6.640381E-05

                     P( 1 )= 8.240981E-02
                     P( 2 )= 1.133213
                     P( 3 )= 2.343516
```

```
              WEIGHTED LEAST SQUARES PARAMETER ESTIMATION
                   IN MULTIVARIABLE NONLINEAR MODELS
                   GAUSS - NEWTON - MARQUARDT METHOD


NUMBER OF INDEPENDENT VARIABLES .....  3
NUMBER OF DEPENDENT VARIABLES .......  1
NUMBER OF PARAMETERS ................  3
NUMBER OF SAMPLE POINTS .............  15
OPTION OF WEIGHTING .................  0 (IDENTICAL WEIGHTS)


PRINCIPAL COMPONENT ANALYSIS OF NORMED CROSS PRODUCT MATRIX

EIGENVALUE    P( 1 )     P( 2 )     P( 3 )

0.14589E+02   -.048      0.437      0.898
0.79533E-01   0.923      -.325      0.207
0.65923E-02   0.383      0.839      -.388


-------------------------------------------------------
SAMPLE No      Y MEAS        Y COMP        RESIDUAL
-------------------------------------------------------
 1            0.14000E+00   0.13411E+00   0.58885E-02
 2            0.18000E+00   0.17972E+00   0.27615E-03
 3            0.22000E+00   0.22026E+00   -.26277E-03
 4            0.25000E+00   0.25653E+00   -.65301E-02
 5            0.29000E+00   0.28917E+00   0.83274E-03
 6            0.32000E+00   0.31869E+00   0.13067E-02
 7            0.35000E+00   0.34553E+00   0.44672E-02
 8            0.39000E+00   0.37004E+00   0.19964E-01
 9            0.37000E+00   0.45222E+00   -.82215E-01
 10           0.58000E+00   0.56179E+00   0.18212E-01
 11           0.73000E+00   0.71519E+00   0.14812E-01
 12           0.96000E+00   0.94529E+00   0.14710E-01
 13           0.13400E+01   0.13288E+01   0.11208E-01
 14           0.21000E+01   0.20958E+01   0.42033E-02
 15           0.43900E+01   0.43968E+01   -.68097E-02
-------------------------------------------------------

SUM OF SQUARES ......................  8.214867E-03
DEGREES OF FREEDOM ..................  12
STANDARD ERROR ......................  2.616433E-02
CRITICAL T-VALUE AT 95 % CONF. LEVEL   2.18

----------------------------------------------------------------------
PARAMETER     ESTIMATE      ST. ERROR     LOWER BOUND   UPPER BOUND
----------------------------------------------------------------------
 P( 1 )       0.82410E-01   0.12369E-01   0.55446E-01   0.10937E+00
 P( 2 )       0.11332E+01   0.30815E+00   0.46145E+00   0.18050E+01
 P( 3 )       0.23435E+01   0.29666E+00   0.16968E+01   0.29902E+01
----------------------------------------------------------------------

CORRELATION MATRIX OF PARAMETERS

              P( 1 )     P( 2 )     P( 3 )

P( 1 )        1.000
P( 2 )        0.753      1.000
P( 3 )        -.724      -.997      1.000
```

Most part of the output is similar to the output of the linear regression module. The eigenvalues and eigenvectors refer to the matrix $[J^T(\beta)WJ(\beta)]$. We will discuss in Section 3.5 how to use this information.

Exercises

- Show that increasing the Marquardt parameter moves the correction vector Δp toward the direction of the negative gradient of the objective function while the length of the correction vector decreases.

- The Hessian matrix of the quadratic approximation (3.42) of the objective function equals $\tilde{H} = 2J^TWJ$. Compare this with the true Hessian matrix of the objective function (3.39). Show that the Gauss-Newton method can be interpreted as a quasi-Newton method of minimization that neglects a certain term in the Hessian. Can you justify this approximation if the residuals are small?

- Rerun Example 3.3 with different starting estimates. Does the number of iterations depend heavily on the starting estimate in this problem?

3.4 LINEARIZATION, WEIGHTING AND REPARAMETERIZATION

Though module M45 is an efficient tool, fitting a nonlinear model to data usually requires considerable computational efforts, and without a good initial guess even the convergence is questionable. Therefore, a transformation replacing the problem with a linear regression one is of great practical value. A well known example is the Arrhenius dependence

$$k = A\exp[-E/(RT)] \qquad (3.49)$$

of the chemical kinetics rate coefficient k on the temperature T, where $R = 8.3144$ J/(mol K) is the universal gas constant, and the preexponential factor A and the activation energy E are the unknown parameters. These parameters are almost invariably determined by fitting the line

$$y = ax + b \; ; \quad \text{with} \quad y = \log(k) \text{ and } x = -1/T, \qquad (3.50)$$

where $E/R = a$ and $\log(A) = b$. A number of simple functions are linearizable by suitable transformations (see e.g., ref. 5) with particularly many applications in the kinetics of enzyme reactions (ref. 16) and catalytic processes (ref. 17).

Fitting the expressions (3.49) and (3.50) to experimental data we obtain, however, somewhat different estimates, since the transformation distorts the error distribution, and the original assumptions do not more apply.

In this section we show how the deviations stemming from linearization can be compensated by selecting suitable weighting coefficients. The observartions are of the form

$$\tilde{y}_i = y_i + \epsilon_i , \qquad (3.51)$$

where $y_i = f(x_i,\mathbf{p})$, and $D^2\{ \epsilon_i \} = \sigma_i^2$. Instead of fitting $y = f(\mathbf{x},\mathbf{p})$ to the data $\tilde{y}_i$ we rather fit the transformed model $y' = g[f(\mathbf{x},\mathbf{p})]$ to the transformed data $\tilde{y}_i' = g[\tilde{y}_i]$, where $g[\ ]$ is the linearizing transformation, and

$$\tilde{y}_i' = g[\tilde{y}_i] + \epsilon_i' . \qquad (3.52)$$

To find the variance of ϵ_i' note that by (3.51)

$$\epsilon_i' = g[\tilde{y}_i] - g[\tilde{y}_i - \epsilon_i] , \qquad (3.53)$$

where $g[\tilde{y}_i - \epsilon_i] \approx g[\tilde{y}_i] - g'[\tilde{y}_i]\epsilon_i$ from the linear approximation and $g' = dg/dy$. Therefore, from (3.53) $\epsilon_i' \approx g'[\tilde{y}_i]\epsilon_i$ and

$$D^2\{ \epsilon_i' \} \approx (g'[\tilde{y}_i]\sigma_i)^2 . \qquad (3.54)$$

Thus, fitting the transformed model to the data $g[\tilde{y}_i]$ the original assumptions are better retained through the use of the weighting coefficients $w_i = \sigma^2/(g'[\tilde{y}_i]\sigma_i)^2$, where σ^2 is an arbitrary positive constant.

Example 3.4 Estimation of Arrhenius parameters by weighted linear regression

Table 3.2 lists the rate coefficient of the reaction

$CH_3I + C_2H_5ONa \text{ ---> } CH_3OC_2H_5 + NaI$

at 6 different temperatures (ref. 18). First we assume that k is observed with constant error variance. Equation (3.49) is fitted to the data using nonlinear least squares with weighting coefficients $w_i = 1$. In addition to the nonlinear fit we estimate the parameters from the logarithmic model (3.50).

Table 3.2
Observed temperature dependence of rate coefficient

T, K	273.15	279.15	285.15	291.15	297.15	303.15
$k\times10^5$, 1/(mol s)	5.6	11.8	24.5	48.8	100	208

Fitting (3.50) we first use the weights $w_i = 1$, and then $w_i = \tilde{k}_i^2$ following from (3.54) . This is done by the module M40 with options WI = 0 and WI = 2, respectively. Table 3.3 lists the estimates and the 95% confidence intervals that are not symmetric, due to the exponential back transformation.

Table 3.3
Estimates and 95% confidence intervals of the Arrhenius parameters

	Nonlinear estimation		Linear estimation	
	$w_i = 1$	$w_i = 1/\tilde{k}_i^2$	$w_i = 1$	$w_i = \tilde{k}_i^2$
$A\times10^{-12}$, 1/(mol s)	3.42	0.317	0.325	3.10
	(-2.2, 9.1)	(-0.12, 0.75)	(0.09, 1.2)	(0.61, 16)
$E\times10^{-4}$, J/mol	8.83	8.25	8.25	8.81
	(8.4, 9.2)	(7.9, 8.6)	(7.9, 8.6)	(8.4, 9.2)

As seen from the first and last columns of the table, the appropriate weighting considerably reduces the deviations between the results of linear and nonlinear estimations.

The table also shows that the nonlinear fit gives a very large confidence interval for the parameter A, an inherent problem in fitting the Arrhenius expression (3.49) directly. While the extremely large confidence interval is an overestimation stemming from the local linear approximation of the model, it still reveals a real problem. As discussed in the previous section, the Gauss-Newton method involves a sequence of quadratic approximations of the objective function. Each of such approximations is a long valley along the coordinate axis corresponding to A, and its minimum is rather difficult to localize with reasonable accuracy. This problem, reconsidered in the next section, increases the significance of the simple linear estimation through logarithmic transformation.

The "observed" rate constants are, in fact, derived from other measurable quantities, and according to chemists the assumption of constant relative variances (i.e., σ_i^2 is proportional to $\tilde{k}_i^2$) is usually closer to the reality than that of constant variances. Assuming such error structure one chooses the weighting coefficients $w_i = \tilde{k}_i^{-2}$ when fitting (3.49) directly, and hence unit weights $w_i = 1$ in the linear regression involving (3.50). These considerations and the corresponding results shown in the second and third columns of Table 3.3 justify the use of unweighted linear regression for estimating Arrhenius parameters.

Unfortunately, many transformations purported to linearize the model also interchange the role of dependent and independent variables. Important examples are the various linearization transformations of the simple steady-state Michaelis-Menten model

$$r = V \frac{[S]}{K + [S]} \tag{3.55}$$

of the enzyme reaction studied in Section 2.5.1, where $[S]$ denotes the concentration of the substrate and r is the rate of the reaction. To estimate the Michaelis-Menten parameters V and K from the data $\{ ([S_i], \tilde{r}_i); i = 1,2,\ldots,nm \}$, one can fit, for example, the following linear functions (ref. 19):

$$r = - K \frac{r}{[S]} + V \qquad \text{(Eadie--Hofstee)}, \tag{3.56}$$

$$\frac{[S]}{r} = \frac{1}{V} [S] + \frac{K}{V} \qquad \text{(Hanes)}, \tag{3.57}$$

$$\frac{r}{[S]} = - \frac{1}{K} r + \frac{V}{K} \qquad \text{(Scatchard)} \tag{3.58}$$

$$\frac{1}{r} = \frac{K}{V} \frac{1}{[S]} + \frac{1}{V} \qquad \text{(Lineweaver--Burk)} \tag{3.59}$$

These classical methods are still popular. Since the error in the observed reaction rate $\tilde{r}_i$ is usually much larger than the error in the substrate concentration $[S_i]$, assumption (i) of the least squares method is approximately satisfied when fitting (3.55) directly. This assumption is, however, clearly violated in models (3.56 - 3.58), where the error corrupted r appears also on the right hand side. Therefore, the use of most linearized models should be restricted to determining a good initial guess for the nonlinear parameter estimation (ref. 20).

Linearization by transformation and rearrangement of the variables is not the only way to reduce computational efforts in nonlinear estimation. A faster convergence can be expected if the nonlinear character of the model is decreased by manipulating the parameters. Bates and Watts (ref. 21) proposed a measure of nonlinearity and found that the major part of nonlinearity was due to the particular parameterization in many models. In such cases nonlinear parameter transformations may considerably improve the efficiency of the search algorithm. While the literature provides a number of interesting applications (refs. 22), model reparameterization is somewhat a kind of art owing to the lack of systematic approaches.

Exercises

□ Fit the models (3.55) through (3.59) to the data listed in Table 3.4 (ref. 19) by the modules M45 and M40. Compare the estimates and the confidence intervals.

Table 3.4
Initial substrate concentrations and rates for an enzyme reaction

$[S]\times 10^3$, mol/l	$r\times 10^5$, mol/(l s)	$[S]\times 10^3$, mol/l	$r\times 10^5$, mol/(l s)
50	1.967	10	0.717
40	1.723	8	0.537
30	1.517	5	0.300
20	1.150	3	0.243
15	0.967	1	0.103

□ The Weibull growth model $y = a - b \exp(- c\, x^d)$ is frequently used in biological and agricultural applications. According to the investigations in (ref. 22), the nonlinearity of this model is considerably reduced if fitted in one of the reparameterized forms

(i) $y = p_1 - p_2 \exp[- \exp(-p_3)\, x^{p_4}]$

with $a = p_1$, $b = p_2$, $c = \exp(-p_3)$ and $d = p_4$, or

(ii) $y = \exp(p_1) - \exp[p_2 - \exp(-p_3)\, x^{p_4}]$

with $a = \exp(p_1)$, $b = \exp(p_2)$, $c = \exp(-p_3)$ and $d = p_4$.

Select values of the independent variable from the interval [0, 100]. Generate error-free data with nominal parameters $a = 70$, $b = 60$, $c = 0.0002$ and $d = 2$. Investigate the convergence behavior of the module M45 for the original and for the two reparametrized models. Use several sets of starting parameter values, paying attention to the relations between the original and the newly introduced parameters.

3.5 ILL-CONDITIONED ESTIMATION PROBLEMS

To obtain the estimate (3.23) in a multivariate linear regression problem we solve a set of linear equations. According to Section 1.7, the estimate $\hat{p}$ is sensitive to small perturbations of the observation vector $\tilde{\mathbf{Y}}$ if the matrix $\mathbf{X}^T\mathbf{W}\mathbf{X}$ is ill-conditioned, i.e., its condition number is large. The condition number of this matrix is the ratio of its largest eigenvalue λ_1 to its smallest eigenvalue λ_{nx}. In the program module M42 the matrix $\mathbf{X}^T\mathbf{W}\mathbf{X}$ is transformed to a correlation type matrix. The sum of the eigenvalues of this matrix is nx and the largest eigenvalue is always near to one. You can easily recognize an ill-conditioned regression problem looking at the smallest eigenvalue λ_{nx} of the correlation matrix. If λ_{nx} is less than, say, 10^{-5} then the results should be treated with caution.

Now we analyze a little deeper the effect of a small eigenvalue. By (3.30) and (3.32) the joint confidence region of the parameters at a given confidence level is a hyperellipsoid

$$[\Delta p]^T[\mathbf{X}^T\mathbf{W}\mathbf{X}][\Delta p] \le \text{const} , \tag{3.60}$$

where $\Delta p = p - \hat{p}$. In the basis of the eigenvectors $u_1, u_2, \ldots, u_{nx}$ of $\mathbf{X}^T\mathbf{W}\mathbf{X}$ the left hand side of (3.60) reduces to canonical form, and the confidence ellipsoid is given by

$$\sum_{i=1}^{nx} \lambda_i(\Delta f_i)^2 \le \text{const} , \tag{3.61}$$

where $\lambda_1, \lambda_2, \ldots, \lambda_{nx}$ are the eigenvalues of $\mathbf{X}^T\mathbf{W}\mathbf{X}$ and $\Delta f_i = [u_i]^T\Delta p$ denotes the i-th principal component. From (3.61) follows that the principal axes of the ellipsoid are along the eigenvectors, and the length of the axis along u_i is proportional to $\lambda_i^{-1/2}$. If λ_i is small, the ellipsoid is elongated along u_i and we get almost the same goodness-of-fit at parameter values that are far apart. Furthermore, the mean square error, i.e., the expected distance between the estimate $\hat{p}$ and the true parameter vector p satisfies the inequality (ref. 23)

$$E\{ [p - \hat{p}]^T[p - \hat{p}] \} = \text{trace } [\mathbf{X}^T\mathbf{W}\mathbf{X}]^{-1} = \sigma^2 \sum_{i=1}^{nx} \frac{1}{\lambda_i} > \frac{\sigma^2}{\lambda_{nx}} \approx \frac{s^2}{\lambda_{nx}} . \tag{3.62}$$

Thus, with a nearly zero eigenvalue of the covariance matrix of the independent variables the estimates tend to be inflated and the results are meaningless. Therefore, in nearly singular estimation problems reducing the mean square error of the estimates is of first importance. Since the least squares estimator gives minimum variance only in the class of unbiased estimators, we rather give up unbiasedness.

3.5.1 Ridge regression

The simplest and most popular biased estimator is due to Hoerl and Kennard (ref. 23), estimating the unknown parameters by

$$\hat{p}(\lambda) = [\ \mathbf{X}^T\mathbf{W}\mathbf{X} \ + \lambda I\ ]^{-1}\ \mathbf{X}^T\mathbf{W}\tilde{\mathbf{Y}} \tag{3.63}$$

instead of equation (3.23) of ordinary least squares. The scalar λ is called the ridge parameter. As in the Marquardt modification of the Gauss-Newton method, the additional term λI increases the smallest eigenvalue of the matrix to be inverted. The role of the ridge parameter differs, however, considerably from that of the Marquardt parameter. We usually fix the ridge patrameter at some positive value that hopefully gives a smaller square error than $\lambda = 0$, whereas the Marquardt parameter can be considered as a technical tool used only during the iteration and not affecting the final result. Unfortunately, selecting an appropriate ridge parameter is far from simple. Very often we rather vary the ridge parameter and plot the ridge estimates (3.63) at different values of λ . The plot reveals possible instability of some parameters. Since $\mathbf{X}^T\mathbf{W}\mathbf{X}$ is normalized to a correlation matrix in the module M42, the ridge parameter is usually varied between 0 and 1.

You may notice that the ridge regression is a straightforward statistical counterpart of the regularization methods discussed in Section 1.7.

Example 3.5.1 Analysis of the rate coefficient of an acid-catalysed reaction by ridge regression

We assume that the reaction considered in Example 3.2 is not only acid-catalysed but also basis-catalysed. Then its rate coefficient is of the form

$$k = k_0 + k_H[H^+] + k_{HA}[HA] + k_{OH}[OH^-] + k_A[A^-] \ . \tag{3.64}$$

In this system $[A^-] = [NO_2C_6H_4O^-]$. Table 3.1 includes the data we need, since the concentration $[OH^-]$ can easily be obtained from the ionic product $[H^+][OH^-] = 10^{-14}$ $(mol/l)^2$ of the water. Fitting (3.64) to the data of Table 3.1 we have the following results:

```
                    MULTIVARIABLE LINEAR REGRESSION
                       METHOD OF LEAST SQUARES


NUMBER OF INDEPENDENT VARIABLES .....  5
NUMBER OF SAMPLE POINTS ............  10


PRINCIPAL COMPONENT ANALYSIS OF THE CORRELATION MATRIX

EIGENVALUE    X( 1 )     X( 2 )     X( 3 )     X( 4 )     X( 5 )

0.43760E+01   0.464      0.441      0.441      0.448      0.442
0.40586E+00   -.370      -.407      0.547      -.301      0.555
0.20395E+00   0.048      -.627      -.343      0.647      0.263
0.13648E-01   -.026      0.302      -.624      -.302      0.654
0.58193E-03   -.803      0.395      0.002      0.447      -.006


----------------------------------------------------------------------
 I            Y MEAS        WEIGHT        Y COMP        RESIDUAL
----------------------------------------------------------------------
 1          0.12100E-03  0.10000E+01  0.11847E-03  0.25275E-05
 2          0.12000E-03  0.10000E+01  0.12260E-03  -.26002E-05
 3          0.13500E-03  0.10000E+01  0.13577E-03  -.77304E-06
 4          0.14400E-03  0.10000E+01  0.14328E-03  0.72010E-06
 5          0.15400E-03  0.10000E+01  0.15359E-03  0.40521E-06
 6          0.16100E-03  0.10000E+01  0.16322E-03  -.22158E-05
 7          0.17700E-03  0.10000E+01  0.17529E-03  0.17123E-05
 8          0.23700E-03  0.10000E+01  0.23584E-03  0.11570E-05
 9          0.24700E-03  0.10000E+01  0.24830E-03  -.13041E-05
 10         0.28400E-03  0.10000E+01  0.28396E-03  0.43015E-07
----------------------------------------------------------------------

SUM OF SQUARES ......................  2.531233E-11
DEGREES OF FREEDOM ..................  5
STANDARD ERROR ......................  2.249992E-06
DURBIN-WATSON D-STATISTICS ..........  2.466769
CRITICAL T-VALUE AT 95 % CONF. LEVEL   2.57

----------------------------------------------------------------------
PARAMETER     ESTIMATE      ST.ERROR      LOWER BOUND   UPPER BOUND
----------------------------------------------------------------------
 P( 1 )       -.45465E-04   0.23677E-04   -.10632E-03   0.15386E-04
 P( 2 )       0.25216E+05   0.16272E+04   0.21034E+05   0.29398E+05
 P( 3 )       0.21348E-02   0.25814E-03   0.14714E-02   0.27982E-02
 P( 4 )       0.19113E+02   0.80678E+01   -.16212E+01   0.39848E+02
 P( 5 )       -.86085E-03   0.37287E-03   -.18191E-02   0.97436E-04
----------------------------------------------------------------------
```

Since the model (3.64) includes all terms of the model (3.33), and this latter gave good fit to the same data in Example 3.2, it is not surprising that the fit is excellent. While it is not always the case, the standard residual error is even slightly decreased by extending the model. Although according to the F-test this decrease is not significant, the improved fit is revealed by the better value of the D-statistics. We obtain, however, negative and hence physically meaningless parameters for $p_1 = k_0$ and $p_5 = k_A$.

The smallest eigenvalue of the correlation matrix is 5.8×10^{-4}. From a strictly numerical point of view the matrix to be inverted is not ill-conditioned. From a statistical point of view, however, this eigenvalue is too small. Indeed, X^TWX is in normalized form, each of its diagonal entry being 1. Such a normalized matrix with nearly orthogonal columns would have eigenvalues close to 1, and the obtained much smaller eigenvalue reveals near linear dependency among the columns.

We use the ridge option of the module M42, i.e., input parameter RP , to construct a ridge plot shown in Fig. 3.2. In this plot the ratios

$\alpha_i = \hat{p}_i(\lambda) / |\hat{p}_i(0)|$ are shown as functions of the ridge parameter.

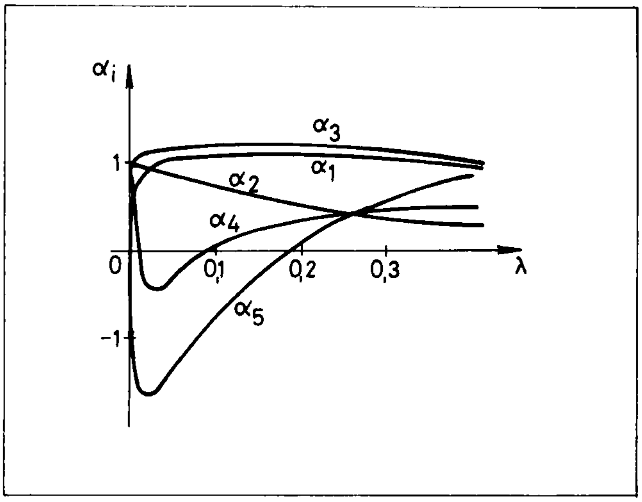


Fig. 3.2. Relative change of the estimates as a function of the ridge parameter

A small increase of λ heavily affects the ridge estimates of $p_4 = k_{OH}$ and $p_5 = k_A$ and even their signs are changed. These estimates are not stable. At some small value $\lambda > 0$ we have $\hat{k}_{OH} \approx 0$ and $\hat{k}_A \approx 0$. The estimate $p_3 = k_{HA}$ is almost constant, whereas $p_2 = k_H$ moderately decreases. That latter is a normal behavior even for an orthogonal matrix X^TWX , thus the estimate of k_H is also stable. The estimate $p_1 = \hat{k}_0$ changes in an interesting way. While $\hat{p}_1(0)$ is negative, it changes sign at a small $\lambda > 0$ and remains almost constant further on. Thus this parameter estimate is eventually stable, but can be neglected because of its small value. Our analysis supports the assumption that the reaction is acid-catalysed with the only essential parameters k_H and k_{HA} considered in model (3.64).

The information revealed by ridge plots as the one shown in Fig. 3.2 can be better understood noting that the ridge estimate (3.63) is the solution of the minimization problem:

$\|p\|$ --> min , subject to the constraint $Q(p) - Q(p(0)) = C$,

where $C > 0$ is an increasing function of λ (ref. 23) . Therefore, an elongated confidence ellipsoid results in wildly changing ridge estimates for some of the parameters, whereas other parameters remain stable.

3.5.2 Overparameterized nonlinear models

On the basis of its constrained minimization interpretation the ridge regression technique can be extended to nonlinear models, but the construction of ridge plots requires considerable computing effort. Therefore, ridge regression is rarely used in nonlinear estimation, though near singularity is an even more inherent problem than in the linear case. In fact, the small eigenvalues of the cross product matrix $\mathbf{X}^T\mathbf{WX}$ of a linear model can be increased by appropriate experiment design (see Section 3.10.2), the eigenvalues of the matrix $\mathbf{J}^T(\boldsymbol{\beta})\mathbf{WJ}(\boldsymbol{\beta})$ of a nonlinear model depend, however, also on the form of the response function and the actual parameter values. Therefore, the possibilities of eliminating near singularity by experiment design are usually quite restricted in the nonlinear case. For example, the partial derivatives of the Arrhenius function (3.49) are $\partial k/\partial A = \exp[-E/(RT)]$ and $\partial k/\partial E = - A \exp[-E/(RT)]/(RT)$, and the columns of the Jacobian matrix are nearly collinear if the rate constants are observed over a relatively small temperature intervall, as usually restricted by the experimental techniques. In such cases the model might be overparameterized (see, e.g., ref. 24) in spite of its apparent simplicity.

Overparameterization and frequently its sources are revealed by an eigenvalue-eigenvector analysis. In the module M45 the matrix $\mathbf{J}^T(\boldsymbol{\beta})\mathbf{WJ}(\boldsymbol{\beta})$ is investigated. We call it normalized cross product matrix, because the partial derivatives are computed with respect to the normalized parameters

$\beta_j = p_j/p_j^{(k)}$. In contrast to the linear case, this matrix is not normalized further to a correlation type matrix before the principal component analysis. Its eigenvalues and eigenvectors are of considerable help when interpreting the results. For example, at most 10% relative mean error in the parameters implies the inequality $E\{ [\beta - \hat{\beta}]^T[\beta - \hat{\beta}] \} < 0.01$. Due to the use of normalized parameters in $J^T(\beta)WJ(\beta)$ and according to (3.62) , this can be attained if the smallest eigenvalue satisfies the inequality $\lambda_{np} > 100\ \sigma^2 \approx 100\ s^2$, where s^2 is the estimate of the squared sigma factor in the weights. As usual, we consider the estimation problem nearly singular if the smallest eigenvalue is below this limit.

Another advantage of the normalized parameters is that the eigenvectors corresponding to small eigenvalues frequently reveal the form of nonlinear dependences among the estimates. For this interpretation we introduce the parameters $\alpha_j = \log [\ p_j\ ]$. It is important to note that at $\mathbf{p} = \mathbf{p}^{(k)}$ we have $\partial f/\partial\beta_j = \partial f/\partial\alpha_j = (\partial f/\partial p_j)p_j^{(k)}$, and hence the two parameter transformations we introduced locally give the same Jacobian matrix. Furthermore, we exploit the canonical form

$$\tilde{Q}(\ \alpha\ ) - \tilde{Q}(\ \hat{\alpha}\ ) = \sum_{i=1}^{np} \lambda_i(\Delta f_i)^2 \qquad (3.65)$$

of the quadratic approximation (3.42), where $\hat{\alpha}_j = \log [\ \hat{p}_j\ ]$ and $\Delta f_i = u_i^T[\ \alpha - \hat{\alpha}\ ]$. Moving from the point $\hat{\alpha}$ along the eigenvector u_i by a step of unit length implies $(\Delta f_i)^2 = 1$, $(\Delta f_j)^2 = 0$ for i≠j, and hence $\tilde{Q}(\ \alpha\ ) - \tilde{Q}(\ \hat{\alpha}\ ) \approx \lambda_i$. Assume that $\lambda_i \approx 0$, and the corresponding eigenvector is $u_i = [0.707,\ 0.707,\ 0,\ \ldots,\ 0]^T$. Then selecting $\Delta\alpha_1 = \Delta\alpha_2$ we move along u_i, and $\tilde{Q}(\ \alpha\ ) - \tilde{Q}(\ \hat{\alpha}\ ) \approx 0$. The line $\Delta\alpha_1 = \Delta\alpha_2$ in the space of the α's corresponds to the curve $\log[\ p_1/p_2\ ] = \log[\ \hat{p}_1/\hat{p}_2\ ]$, i.e., p_1/p_2 = const., in the space of the original parameters. Thus, keeping the ratio p_1/p_2 fixed, the objective function value remains almost unchanged. In other words the objective function depends only on the ratio p_1/p_2 , and does not depend on the individual parameters p_1 and p_2 separately. Similarly, the eigenvector $u_i = [0.707,\ -0.707,\ 0,\ \ldots,\ 0]^T$ corresponding to a nearly zero eigenvalue λ_i reveals that the objective function depends only on the product p_1p_2. It is even simpler to interpret a unit vector corresponding to a nearly zero eigenvalue. Then the parameter corresponding to the coefficient 1 in the eigenvector cannot be identified. The analysis can also be extended to find relationships among several parameters, and is particularly useful in chemical kinetics (ref. 25-26).

Exercise

□ At a fixed value $V = 0.035\times10^{-3}$ mol/(l s) and several values of K between 10^{-3} mol/l and 0.1 mol/l , compute the error-free rates from the model (3.55) at the substrate concentrations listed in Table 3.4. Perform principal component analysis (using the module M45 and setting IM = 1) of the normalized cross product matrix. Find a value K_1 of K such that for $K < K_1$ only the parameter V can be estimated with reasonable accuracy. Similarly, find K_2 such that if $K > K_2$ then a reasonable estimate can be obtained only for the ratio V/K.

3.6 MULTIRESPONSE ESTIMATION

In Section 3.3 we allowed the errors in the observations $\tilde{y}_{i1}$, $\tilde{y}_{i2}$, ..., $\tilde{y}_{i,ny}$ to be correlated, but apart from a scalar factor σ^2 their covariance matrix was assumed to be known. The multiresponse estimation method proposed by Box and Draper (ref. 27) does not require this strong assumption. The method is based on the maximum likelihood principle, and involves the minimization of the objective function

$$Q(p) = \det[\ V(p)\ ] \tag{3.36}$$

where

$$[V(p)]_{ij} = \sum_{k=1}^{nm} [\ \tilde{y}_{ki} - f_i(x_k,p)\ ]\ [\ \tilde{y}_{kj} - f_j(x_k,p)\ ] \tag{3.67}$$

is the $ny\times ny$ empirical covariance matrix computed at the actual parameter vector p . Notice that the errors in different sample points are still assumed to be uncorrelated. The determinant criterion (3.66) is equivalent to the unweighted least squares method if only one dependent variable is observed in every sample point. For the multiresponse case it is, at least from a theoretical point of view, a more general estimator than the least squares.

Unfortunately, there are some technical difficulties associated with the determinant criterion (ref. 28). Minimizing the determinant (3.66) is not a trivial task. In addition, the method obviously does not apply if $\det[\ V(p)\ ]$ is zero or nearly zero for all parameter values. This is the case if there exist affine linear relationships among the responses y_1, y_2, ..., y_{ny}, as we discussed in Section 1.8.7. To overcome this problem the principal component analysis of the observations is applied before the estimation step.

Example 3.6 Comparison of the determinant criterion with least squares

Return to the example of Box et al. (ref. 29) we started to discuss in Section 1.8.7. The thermal isomerization be described by the mechanism shown in Figure 3.3.

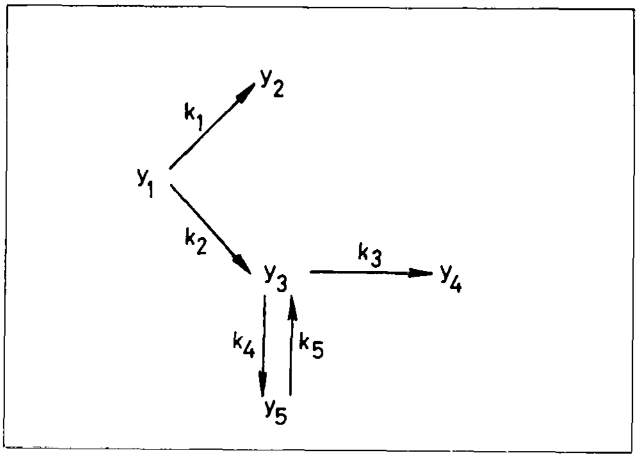


Fig 3.3. Mechanism of the thermal isomerization of α-pinene

Assuming first order reactions, the mechanism gives rise to a set of first order differential equations. The following solution of the equations gives the component concentrations y_1, y_2, ..., y_5 as function of the reaction time t:

$$y_1 = y_{10} \exp[-\Phi t]$$

$$y_2 = \frac{k_1 y_{10}}{\Phi} (1-\exp[-\Phi t])$$

$$y_3 = c_1 \exp[-\Phi t] + c_2 \exp[\beta t] + c_3 \exp[\gamma t]$$

$$y_4 = k_3 \left[\frac{c_1}{\Phi} (1-\exp[-\Phi t]) + \frac{c_2}{\beta} (\exp[\beta t]-1) + \frac{c_3}{\gamma} (\exp[\gamma t]-1) \right]$$

$$y_5 = k_4 \left[\frac{c_1}{k_5-\Phi} (1-\exp[-\Phi t]) + \frac{c_2}{k_5+\beta} (\exp[\beta t]-1) + \frac{c_3}{k_5+\gamma} (\exp[\gamma t]-1) \right] ,$$

where $y_{10} = 100\%$ is the initial concentration of the first component (α-pinene); k_1, k_2, k_3, k_4 and k_5 are the unknown rate coefficients, and

$$\Phi = k_1 + k_2$$

$$\alpha = k_3 + k_4 + k_5$$

$$\beta = \left[-\alpha + (\alpha^2 - 4k_3k_5)^{1/2}\right] / 2$$

$$\gamma = \left[-\alpha - (\alpha^2 - 4k_3k_5)^{1/2}\right] / 2$$

$$c_1 = \frac{k_2y_{10}(k_5-\Phi)}{(\Phi+\beta)(\Phi+\gamma)}, \quad c_2 = \frac{k_2y_{10}(k_5+\beta)}{(\Phi+\beta)(\beta-\gamma)} \quad \text{and} \quad c_3 = \frac{k_2y_{10}(k_5+\gamma)}{(\Phi+\gamma)(\gamma-\beta)} .$$

The observed concentrations have been listed in Table 1.3. Let us first fit the above response function to the data by the least sqares method with the weighting matrices $\mathbf{W}_i = \mathbf{I}$, i.e., without weighting. Module M45 results in the estimates shown in the first row of of Table 3.5.

Table 3.5
Estimated rate coefficients

Method	Rate coefficient×10^5, 1/min				
	k_1	k_2	k_3	k_4	k_5
Least squares, 5 responses	5.93	2.96	2.05	27.5	4.00
Box-Draper, 3 principal component	5.95	2.84	0.43	31.3	5.74
Least squares, weighted	5.95	2.87	0.51	29.6	5.16

As found in Section 1.8.7, there were two affin linear dependences among the data, classified as exact ones. Therefore, Box et al. (ref. 29) considered the principal components corresponding to the three largest eigenvalues as response functions when minimizing the objective function (3.66). By virtue of the eigenvectors derived in Section 1.8.7, these principal components are:

$$y_1^* = 0.8087y_1 - 0.5404y_2 - 0.0127y_3 - 0.0241y_4 - 0.2307y_5$$

$$y_2^* = 0.0568y_1 - 0.2236y_2 - 0.6122y_3 + 0.0375y_4 + 0.7562y_5 \qquad (3.68)$$

$$y_3^* = -0.2957y_1 - 0.6108y_2 + 0.6402y_3 - 0.0100y_4 + 0.3599y_5 .$$

The linear transformation (3.68) should obviously be applied both to the observed concentrations and to the computed ones.

Based on the analytical expression for the derivative of det[V(p)] , Bates and Watts (ref. 30) recently proposed a Gauss-Newton type procedure for minimizing the objective function (3.66). We use here, however, the simplex method of Nelder and Mead (module M34) which is certainly less efficient but does not require further programming. The determinant is evaluated by the module M14. After 95 iterations we obtain the results shown in the second row of Table 3.5, in good agreement with the estimates of Box et al. (ref. 29).

Comparing the first and second rows of Table 3.5 we could conclude that the least squares and the determinant criterion yield significantly deviating estimates. This conclusion is, however, not completely true. We repeat the estimation by the least squares method, but considering the three principal components (3.68) as responses. This can alternatively done retaining the original model with five responses, but introducing the weighting matrix with elements

$$w_{ij} = \sum_{k=1}^{3} u_{ik}u_{jk} ,$$

where u_{ik} is the i-th element of the k-th eigenvector computed in Section 1.8.7. Then the nondiagonal weighting matrix

$$W = \begin{bmatrix} 0.745 & -0.269 & -0.234 & -0.016 & -0.250 \\ -0.269 & 0.715 & -0.247 & 0.018 & -0.264 \\ -0.234 & -0.247 & 0.785 & -0.008 & -0.230 \\ -0.016 & 0.018 & -0.008 & 0.001 & 0.005 \\ -0.250 & -0.264 & -0.230 & 0.005 & 0.755 \end{bmatrix}$$

is used in the module M45, exploiting the weighting option WI = 2 . The result of the estimation is shown in the third row of Table 3.5.

Considering the recent general dislike of statisticians toward the application of the least squares method to multiresponse problems, it is surprising to see that having eliminated the linear dependences from the data, the least squares method gives very similar estimates to the determinant criterion. Thus, in this famous example a preliminary screening of the data is more important than the choice of the estimation criterion. To put it more simply, the analysis of linear dependences revealed that y_4 had not been measured but assumed, though its values significantly influenced the estimate of k_3, in accordance with the reaction mechanism shown in Fig. 3.3. Using three principal components we practically dropped these "measurements", and obtained an improved value of k_3, almost independently of the estimation criterion.

3.7 EQUILIBRATING BALANCE EQUATIONS

The problem to be solved here primarily comes from chemical engineering where one simultaneously observes several variables that are expected to satisfy a number of balance equations such as stoichiometric relations. Due to measurement errors the observed values obviously do not fulfill this expectation. Let $x_1, x_2, \ldots, x_{nv}$ denote these variables observed in a single sample point that gives the data $\{ \tilde{x}_i = x_i + \epsilon_i ;\ i=1,nv \}$. Assuming that the covariance matrix $\mathrm{cov}\{\epsilon\} = \mathbf{V}$ of the error vector ϵ is diagonal and known, we would like to find the values $\mathbf{x}$ that minimize the quadratic form

$$[\mathbf{x} - \tilde{\mathbf{x}}]^T \mathbf{V}^{-1} [\mathbf{x} - \tilde{\mathbf{x}}] \tag{3.69}$$

and, at the same time, satisfy the set

$$\mathbf{Wx} - \mathbf{b} = \mathbf{0} \tag{3.70}$$

of nb linear balance equations. Since we do not have unknown parameters, and observe the variables only once, this problem differs from the ones studied in the previous sections. Nevertheless, the same estimation technique is used and the results will be useful for parameter estimation in the next section.

Introducing the correction vector $\mathbf{c} = \mathbf{x} - \tilde{\mathbf{x}}$ and the equation error vector $\mathbf{f} = \mathbf{W}\tilde{\mathbf{x}} - \mathbf{b}$, according to (3.69) and (3.70) we minimize the objective function

$$Q(\mathbf{c}) = \mathbf{c}^T \mathbf{V}^{-1} \mathbf{c} \tag{3.71}$$

subject to the constraints

$$\mathbf{Wc} + \mathbf{f} = \mathbf{0} . \tag{3.72}$$

A similar constrained optimization problem has been solved in Section 2.5.4 by the method of Lagrange multipliers. Using the same method we look for the stationary point of the Lagrange function

$$L(\mathbf{c},\lambda) = \mathbf{c}^T \mathbf{V}^{-1} \mathbf{c} + \lambda^T [\mathbf{Wc} + \mathbf{f}] \tag{3.73}$$

where λ denotes the nb-vector of Lagrange multipliers. At the stationary point the partial derivatives of the function (3.73) vanish

$$\frac{\partial L}{\partial \mathbf{c}} = 2 \mathbf{V}^{-1} \mathbf{c} + \mathbf{W}^T \lambda = \mathbf{0} , \tag{3.74}$$

$$\frac{\partial L}{\partial \lambda} = \mathbf{Wc} + \mathbf{f} = \mathbf{0} . \tag{3.75}$$

By (3.74) $c = -(1/2)VW^T\lambda$. Introducing this value into (3.75) gives $\lambda = 2[WVW]^{-1}f$, and using (3.74) again, we obtain the optimal correction

$$\hat{c} = -VW^T[WVW^T]^{-1}f \ . \tag{3.76}$$

Thus the problem can be analytically solved, similarly to linear regression. At the correction (3.76) the objective function (3.71) takes the value

$$q^2 = f^T[WVW^T]^{-1}f \ , \tag{3.77}$$

called error measure in the literature of balance equilibration.

The error measure is invariant under rescaling the equations (3.70) and even under replacing the original equations by their independent linear combinations. This latter may be necessary if the matrix WVW^T is singular and hence the inverse in (3.77) is not defined. Since V is a diagonal matrix with nonzero diagonal entries, this case reveals that the balance equations are not linearly independent. The problem can be resolved by considering a maximum linearly independent subset of the balance equations.

Since q^2 is distributed as χ^2 , the measurements can be accepted if

$$q^2 \le \chi^2_{\alpha,nb} \ , \tag{3.78}$$

where the right hand side is the tabular value of the χ^2 distribution with nb degrees of freedom at the significance level α, usually at $\alpha = 0.05$. Unfortunately, the error variances should be exactly known for this test. If the error variances are known only up to a scalar factor, then the correction vector is still correctly given by (3.76) , but the inequality (3.78) is of no value.

Nevertheless, if (3.78) is known to be violated, a further issue is to find the variable that is primarily responsible for the violation. The ratio of the absolute value of the correction to the corresponding standard deviation provides some information but may be misleading (ref. 31) . The analysis proposed by Almásy and Sztanó (ref. 32) is based on geometric ideas. If exactly one observation is corrupted by gross error then the corresponding column of matrix W and the vector f of equation errors are nearly collinear. Useful measures of collinearity are $\gamma_i = \cos \alpha_i$, where α_i is the angle between f and the i-th column of W . The variable suspected to be corrupted significantly is then the one corresponding to the largest Almásy indicator $|\gamma_i|$. The γ_i values are invariant under scaling of the balance equations (ref. 32).

Program module M50

```
5000 REM ####################################################
5002 REM #    EQUILIBRATING LINEAR BALANCE EQUATIONS BY     #
5004 REM #    LEAST SQUARES METHOD AND OUTLIER ANALYSIS     #
5006 REM ####################################################
5008 REM INPUT:
5010 REM     NB      NUMBER OF BALANCE EQUATIONS
5012 REM     NV      NUMBER OF VARIABLES
5014 REM  W(NB,NV)   MATRIX OF COEFFICIENTS IN EQUATIONS
5016 REM    B(NB)    RIGHT HAND SIDE OF EQUATIONS
5018 REM    X(NV)    OBSERVATIONS
5020 REM    V(NV)   VARIANCES (SQUARED ERRORS) OF VARIABLES
5022 REM OUTPUT:
5024 REM     ER      STATUS FLAG
5026 REM                0  SUCCESSFUL EQUILIBRATION
5028 REM                1  LINEARLY DEPENDENT EQUATIONS
5030 REM    F(NB)    EQUATION ERRORS BEFORE EQUILIBRATING
5032 REM    C(NV)    CORRECTIONS OF VARIABLES
5034 REM     Q2      WEIGHTED SUM OF SQUARES OF CORRECTIONS
5036 REM    G(NV)    VECTOR OF ALMASY INDICATORS
5038 REM     CH      CHI SQUARE AT 0.05 SIGNIFICANCE LEVEL (IF NB<=10)
5040 REM AUXILIARY ARRAYS:
5042 REM   A(NB,NB),T(NV,NB)
5044 REM MODULE CALLED: M16
5046 REM ---------- T=VW'
5048 FOR I=1 TO NV :FOR J=1 TO NB
5050  T(I,J)=W(J,I)*V(I)
5052 NEXT J :NEXT I
5054 REM ---------- A=WVW'
5056 FOR I=1 TO NB :FOR J=1 TO I
5058  A=0 :FOR K=1 TO NV :A=A+W(I,K)*T(K,J) :NEXT K :A(I,J)=A
5060 NEXT J :NEXT I
5062 REM ---------- A=(WVW')^-1
5064 N=NB :GOSUB 1600 :IF ER=1 THEN 5128
5066 REM ---------- F
5068 FOR I=1 TO NB
5070  F=-B(I) :FOR K=1 TO NV :F=F+W(I,K)*X(K) :NEXT K :F(I)=F
5072 NEXT I
5074 REM ---------- (WVW')^-1*F
5076 FOR I=1 TO NB
5078  T=0 :FOR K=1 TO NB :T=T+A(I,K)*F(K) :NEXT K :T(0,I)=T
5080 NEXT I
5082 REM ---------- COMPUTE CORRECTIONS
5084 FOR I=1 TO NV
5086  C=0 :FOR K=1 TO NB :C=C-T(I,K)*T(0,K) :NEXT K :C(I)=C
5088 NEXT I
5090 REM ---------- SUM OF SQUARES
5092 Q2=0 :FOR I=1 TO NB :Q2=Q2+F(I)*T(0,I) :NEXT I :Q=SQR(Q2)
5094 REM ---------- T=W'(WVW')^-1
5096 FOR I=1 TO NV :FOR J=1 TO NB
5098  T=0 :FOR K=1 TO NB :T=T+W(K,I)*A(K,J) :NEXT K :T(I,J)=T
5100 NEXT J :NEXT I
5102 REM ---------- G(I)=(1/q)(DIAG[W'(WVW')^-1*W]^-0.5)*W'(WVW')^-1*F
5104 FOR I=1 TO NV
5106  D=0 :E=0
5108  FOR K=1 TO NB
5110   D=D+T(I,K)*W(K,I) :E=E+T(I,K)*F(K)
5112  NEXT K
```

```
5114 G(I)=1/Q/SQR(D)*E
5116 NEXT I
5118 REM ---------- CHI SQUARE
5120 CH=0 :IF NB>10 THEN 5126
5122 CH= -(NB=1)*3.84-(NB=2)*5.99-(NB=3)*7.81-(NB=4)*9.49-(NB=5)*11.1
5124 CH=CH-(NB=6)*12.6-(NB=7)*14.1-(NB=8)*15.5-(NB=9)*16.9-(NB=10)*18.3
5126 ER=0
5128 RETURN
5130 REM ***************************************************
```

Since the covariance matrix is diagonal, a vector denoted by V is used to store the variances. If the number of balances does not exceed ten, the module also computes the tabular value of the chi square distribution at significance level $\alpha = 0.05$ and degrees of freedom nb. The return value ER = 1 of the status flag indicates that the rows of the matrix **W** are linearly dependent, and hence you should drop at least one of the balance equations. If the source of linear dependence is not clear then the module M10 can help to uncover it.

Example 3.7 Equilibrating linear balance equations

The four variables x_1, x_2, x_3, and x_4 describing a process are expected to satisfy the balance equations (ref. 31):

$$0.1x_1 + 0.6x_2 + - 0.2x_3 - 0.7x_4 = 0$$

$$0.8x_1 + 0.1x_2 + - 0.2x_3 - 0.1x_4 = 0$$

$$0.1x_1 + 0.3x_2 + - 0.6x_3 - 0.2x_4 = 0 .$$

The observations and error variances are shown in Table 3.6.

Table 3.6
Observed values and variances for the balance equilibration problem

variable	measured	variance
x_1	0.1858	0.000209
x_2	4.7935	0.0025
x_3	1.2295	0.000576
x_4	3.8800	0.04

The main program and the results are as follows:

```
100 REM ------------------------------------------------------------
102 REM EX. 3.7. EQUILIBRATING LINEAR BALANCES
104 REM MERGE M16,M50
106 REM ---------- DATA
108 REM  (NUMBER OF BALANCES, NUMBER OF VARIABLES)
110 DATA  3, 4
112 REM  ( BALANCES )
114 DATA .1,.6,-.2,-.7,=,0
116 DATA .8,.1,-.2,-.1,=,0
118 DATA .1,.3,-.6,-.2,=,0
120 REM (MEASURED VALUE, VARIANCE)
122 DATA  .1858,.000289
124 DATA 4.7935,.0025
126 DATA 1.2295,.000576
128 DATA 3.8800,.04
130 REM ---------- READ DATA
132 READ NB,NV
134 DIM W(NB,NV),X(NV),V(NV),B(NB),F(NB),C(NV),G(NV),A(NB,NB),T(NV,NB)
136 FOR I=1 TO NB
138  FOR J=1 TO NV :READ W(I,J) :NEXT J
140  READ A$,B(I)
142 NEXT I
144 REM ---------- CALL MODULE
146 FOR I=1 TO NV :READ X(I),V(I) :NEXT I
148 GOSUB 5000
150 REM ---------- PRINT RESULTS
152 IF ER=0 THEN 158
154  LPRINT "LINEARLY DEPENDENT EQUATIONS, STATUS FLAG:";ER
156  GOTO 178
158 LPRINT
160 LPRINT "WEIGHTED SUM OF SQUARES  (ERROR MEASURE)";Q2
162 IF NB<=10 THEN LPRINT "CHI SQUARE AT 0.05 SIGNIFICANCE LEVEL   ";CH
164 LPRINT :V$=STRING$(53,"-") :F$="#.#####^^^^    " :LPRINT V$
166 LPRINT "VARIABLE   MEASURED       CORRECTED      ALMASY-GAMMA"
168 LPRINT V$
170 FOR I=1 TO NV
172  LPRINT I;TAB(11)" ";:LPRINT USING F$;X(I),X(I)+C(I),G(I)
174 NEXT I
176 LPRINT V$ :LPRINT
178 STOP
```

```
WEIGHTED SUM OF SQUARES  (ERROR MEASURE) 8.454745
CHI SQUARE AT 0.05 SIGNIFICANCE LEVEL    7.81

-----------------------------------------------------
VARIABLE   MEASURED       CORRECTED      ALMASY-GAMMA
-----------------------------------------------------
 1         0.18580E+00    0.16757E+00    0.37032E+00
 2         0.47935E+01    0.48594E+01    -.94129E+00
 3         0.12295E+01    0.11730E+01    0.90238E+00
 4         0.38800E+01    0.38540E+01    0.45320E-01
-----------------------------------------------------
```

Since the error measure is greater than the chi square value, the measurements are not acceptable. According to the Almásy indicators, the variable x_2 is most likely to be corrupted by gross error.

Now we proceed to the problem of equilibrating nonlinear balance equations of the form

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \,. \tag{3.79}$$

These equations are considered as constraints when minimizing the objective function (3.69). The basic idea is similar to that of the Gauss-Newton algorithm. Let $\hat{\mathbf{x}}$ denote an estimate of $\mathbf{x}$. We linearize the function (3.79) around $\hat{\mathbf{x}}$, and define the equation error in terms of this linear approximation by

$$\mathbf{f} = \mathbf{f}(\hat{\mathbf{x}}) + \mathbf{J}(\hat{\mathbf{x}})[\tilde{\mathbf{x}} - \hat{\mathbf{x}}] \tag{3.80}$$

where $\mathbf{J}(\hat{\mathbf{x}})$ denotes the Jacobian matrix of $\mathbf{f}(\mathbf{x})$ evaluated at $\hat{\mathbf{x}}$. Keeping $\hat{\mathbf{x}}$ temporarily fixed, we have a linear equilibration problem with the equation error vector (3.80) and coefficient matrix $\mathbf{W} = \mathbf{J}(\hat{\mathbf{x}})$, whose solution is the correction vector

$$\hat{\mathbf{c}} = - \mathbf{V}\mathbf{J}^T(\hat{\mathbf{x}}) \, [\mathbf{J}(\hat{\mathbf{x}})\mathbf{V}\mathbf{J}^T(\hat{\mathbf{x}})]^{-1} \, [\, \mathbf{f}(\hat{\mathbf{x}}) + \mathbf{J}(\hat{\mathbf{x}})[\tilde{\mathbf{x}} - \hat{\mathbf{x}}] \,] \,. \tag{3.81}$$

The nonlinear problem is solved by repeating such linear steps. Starting with the initial estimate $\hat{\mathbf{x}}^{(o)} = \tilde{\mathbf{x}}$, equation (3.81) gives the correction $\hat{\mathbf{c}}^{(o)}$ and the new estimate of the corrected variables $\hat{\mathbf{x}}^{(1)} = \tilde{\mathbf{x}} + \hat{\mathbf{c}}^{(o)}$. The procedure is repeated with the estimates $\hat{\mathbf{x}}^{(1)}$, $\hat{\mathbf{x}}^{(2)}$, ... to satisfy some termination condition. The resulting value of $\hat{\mathbf{x}}$ is a fixed point of the iteration. Substituting (3.80) into (3.72) the following equation is obtained for $\hat{\mathbf{x}}$:

$$\mathbf{J}(\hat{\mathbf{x}})[\hat{\mathbf{x}} - \tilde{\mathbf{x}}] + \mathbf{f}(\hat{\mathbf{x}}) + \mathbf{J}(\hat{\mathbf{x}})[\tilde{\mathbf{x}} - \hat{\mathbf{x}}] = \mathbf{0} \,. \tag{3.82}$$

Thus the corrected variables indeed satisfy (3.79) at convergence.

Since the corrections are now known, the error measure can be computed from (3.69). The same value can be obtained from (3.77) using the equation error defined by (3.80), i.e.,

$$q^2 = [\, \mathbf{f}(\hat{\mathbf{x}}) + \mathbf{J}(\hat{\mathbf{x}})[\tilde{\mathbf{x}} - \hat{\mathbf{x}}] \,]^T \, [\mathbf{J}(\hat{\mathbf{x}})\mathbf{V}\mathbf{J}^T(\hat{\mathbf{x}})]^{-1} \, [\, \mathbf{f}(\hat{\mathbf{x}}) + \mathbf{J}(\hat{\mathbf{x}})[\tilde{\mathbf{x}} - \hat{\mathbf{x}}] \,] \,. \tag{3.83}$$

This expression might seem to be complicated, but it will play an important role in the next section.

3.8 FITTING ERROR-IN-VARIABLES MODELS

If the assumption of neglecting errors in independent variables cannot be justified, there is no statistical distinction between dependent and independent variables. Then we rather use the vector $z = (z_1,z_2,\ldots,z_{nz})^T$ to denote the variables of the model written in the more general implicit form

$$f(z,p) = \mathbf{0} \ . \tag{3.84}$$

The model consists of nk equations and contains np unknown parameters p to be estimated from nm observations. The outcome of the i-th observation is the data vector $\tilde{z}_i = (\tilde{z}_{i1},\tilde{z}_{i2},\ldots,\tilde{z}_{i,nz})^T$ where $\tilde{z}_{ij} = z_{ij} + \epsilon_{ij}$. Thus we allow for some error ϵ_{ij} in all variables.

We assume that errors in different observations are uncorrelated. Although errors in the i-th observation can be correlated, their covariance matrix V_i is assumed to be known, i.e.,

$$E\{ \epsilon_i \} = \mathbf{0}, \quad E\{ \epsilon_i\epsilon_i^T \} = V_i \quad \text{and} \quad E\{ \epsilon_i\epsilon_j^T \} = \mathbf{0} \ \text{ if } i \neq j \ . \tag{3.85}$$

In order to obtain the parameter estimates $\hat{p}$ and the corrected variables $\hat{z}_i = (\hat{z}_{i1}, \hat{z}_{i2}, \ldots, \hat{z}_{i,nz})^T$, i = 1,...,nm, the error norm function

$$Q(\hat{z}_1,\hat{z}_2,\ldots,\hat{z}_{nm} \ ; \ p) = \sum_{i=1}^{nm} [\tilde{z}_i - \hat{z}_i]^T V_i^{-1} [\tilde{z}_i - \hat{z}_i] \tag{3.86}$$

is minimized with respect to $\hat{z}_i$'s and p , subject to the constraints

$$f(\hat{z}_i; \ p) = \mathbf{0} \ , \quad i = 1, 2, \ldots, nm \ . \tag{3.87}$$

The above criterion can be derived from the maximum likelihood principle (refs. 33-34).

Having a well defined minimization problem, we can proceed to its solution. At any fixed p minimization of (3.86) subject to (3.87) is equivalent to solving nm nonlinear balance equilibration problems of the form

$$\begin{aligned} &Q_i(\hat{z}_i) = [\tilde{z}_i - \hat{z}_i]^T V_i^{-1} [\tilde{z}_i - \hat{z}_i] \longrightarrow \min \ , \\ &f(\hat{z}_i; \ p) = \mathbf{0} \ . \end{aligned} \tag{3.88}$$

Solving the nonlinear balance equilibration problems (3.88) and computing the error measures from (3.83) we obtain

$$Q(p) = \sum_{i=1}^{nm} [\ f(\hat{z}_i,p) + J(\hat{z}_i,p)[\tilde{z}_i - \hat{z}_i]\ ]^T \times$$
$$\times [J(\hat{z}_i,p)VJ^T(\hat{z}_i,p)]^{-1} \times [\ f(\hat{z}_i,p) + J(\hat{z}_i,p)[\tilde{z}_i - \hat{z}_i]. \qquad (3.89)$$

where $J(\hat{z}_i,p)$ is the Jacobian matrix of $f(\hat{z}_i;\ p)$ with respect to the variables $\hat{z}_i$. With optimally corrected variables the objective function (3.86) takes the new form (3.89) supplying more explicit information on how the objective function changes if p is varied. We should bear in mind that $\hat{z}_i$ depends on p. Thus, minimizing (3.89) with respect to p at fixed corrected variables $\hat{z}_i$ will not take us to the solution of the whole problem in one go. Patino-Leal and Reilly (refs. 35-36) suggested to take a minimization step with the objective function (3.89), then to correct the variables $\hat{z}_i$ again, and to continue the iteration. The following algorithm is based on their ideas with some modifications (ref. 37). Let j denote the actual number of iteration.

(i) At $j = 0$ select an initial guess $p^{(0)}$ and let $\hat{z}_i^{(0)} = \tilde{z}_i$.

(ii) Starting from the estimate $p^{(j)}$ find the minimum $p^{(j+1)}$ of the function (3.89) at fixed $\hat{z}_i = \hat{z}_i^{(j)}$. If $j > 0$ and $\|p^{(j+1)}-p^{(j)}\| \le EP$, then finish, otherwise proceed to step (iii).

(iii) At fixed $p^{(j+1)}$ perform balance equilibration for each $i = 1,2,...,nm$, through the use of the iteration

$$\hat{z}_i^{(new)} = \hat{z}_i^{(old)} - V_iJ^T[J_iV_iJ^T]^{-1}[\ f + J[\tilde{z}_i - \hat{z}_i^{(old)}]\ ], \qquad (3.90)$$

where the Jacobian J and the function f are computed at $\hat{z}_i^{(old)}$ and $\hat{p}^{(j+1)}$. Denote by $\hat{z}_i^{(j+1)}$ the result of repeating the iteration (3.90) until convergence.

(iv) Replace j by $j + 1$ and return to step (ii).

Computationaly the most demanding task is locating the minimum of the function (3.89) at step (ii). Since the Gauss-Newton-Marquardt algorithm is a robust and efficient way of solving the nonlinear least squares problem discussed in Section 3.3, we would like to extend it to error-in-variables models. First we show, however, that this extension is not obvious, and the apparently simplest approach does not work.

With the weighting matrices $\mathbf{W}_i = [\ \mathbf{J}_i\mathbf{V}\mathbf{J}_i^T\ ]^{-1}$ the objective function (3.89) reminds that of the least squares method given by (3.37). This apparent similarity suggests the following iterative reweighting strategy: compute the weighting matrices $\mathbf{W}_i$ at some estimate $\mathbf{p}^{(j)}$, solve the corresponding weighted least squares problem for $\mathbf{p}^{(j+1)}$, and continue until convergence. Unfortunately, this idea is erroneous, as it can be readily shown by considering the simple example of fitting the straight line

$$y - ax - b = 0 \tag{3.91}$$

to the set $\{(\tilde{y}_i,\tilde{x}_i),\ i = 1,2,\ldots,nm\}$ of observations, where both variables are subject to error. For simplicity assume constant variances, i.e., the covariance matrix is given as

$$\mathbf{V}_i = \mathbf{V} = \begin{bmatrix} \sigma_y^2 & 0 \\ 0 & \sigma_x^2 \end{bmatrix}. \tag{3.92}$$

In our simple case the Jacobian is a row vector $\mathbf{J}_i$=[1; −a], and hence the objective function (3.89) takes the form

$$Q(a,b) = \sum_{i=1}^{nm} \frac{(\tilde{y}_i-a\tilde{x}_i-b)^2}{\sigma_y^2+a^2\sigma_x^2}\ . \tag{3.93}$$

According to the iterative reweighting we fix the weighting coefficient $(\sigma_y^2 + a^2\sigma_x^2)^{-1}$ in every iteration, thus the strategy results in the unweighted linear regression coefficients $\hat{a}$ and $\hat{b}$, whatever the actual variances σ_y^2 and σ_x^2 are. The correct solution of this problem should, however, depend on the ratio $\lambda = \sigma_y^2/\sigma_x^2$. Indeed, the limiting values $\lambda \to 0$ and $\lambda \to \infty$ result in the two regression lines, with the role of dependent and independent variables interchanged. As illustrated in Section 3.1, these two straight lines are definitely different. The iterative reweighting is unable to give this expected result, and its convergence does not guarantee that (3.93) has been minimized.

The pitfall of iterative reweighting stems from the fact that parameter-dependent matrices $[\mathbf{J}_i\mathbf{V}_i\mathbf{J}_i^T]^{-1}$ cannot simply be considered as weighting matrices. We can give, however, a true sum-of-squares structure

$$Q(a,b) = \sum_{i=1}^{nm} [\tilde{s}_i - g_i(\tilde{y}_i,\tilde{x}_i,a,b)]^2 \tag{3.94}$$

to the objective function (3.93) by introducing the response function

$$g_i(\tilde{y}_i,\tilde{x}_i,a,b) = \frac{(\tilde{y}_i-a\tilde{x}_i-b)^2}{(\sigma_y^2+a^2\sigma_x^2)^{1/2}} \qquad (3.95)$$

and the "observed" responses $\tilde{s}_i \equiv 0$ for all $i = 1,2,\ldots,nm$. Since minimization of (3.4) is now equivalent to solving a nonlinear unweighted least squares problem, the Gauss-Newton-Marquardt procedure applies. We note that for this simple illustrative problem we do not really need the iteration procedure, since there exist explicit expressions for the error-in-variables estimates of a and b, see (ref. 1). The idea of incorporating the induced weights into the response function is, however, generally applicable and requires the decomposition

$$[\ J_i(p)VJ_i^T(p)\ ]^{-1} = Q_i^T(p)Q_i(p)\ , \qquad (3.96)$$

thereby transforming the objective function (3.89) to the unweighted sum of squares form

$$Q(p) = \sum_{i=1}^{nm} \Big[\ \tilde{s}_i - Q_i(p)[\ f_i(p) + J_i(p)[\tilde{z}_i - \hat{z}_i]\ ]\ \Big]^T\times$$
$$\times\Big[\ \tilde{s}_i - Q_i(p)[\ f_i(p) + J_i(p)[\tilde{z}_i - \hat{z}_i]\ ]\ \Big] \qquad (3.97)$$

where $\hat{s}_i \equiv 0$ for all $i = 1,2,\ldots,nm$. Since in step (ii) of the error-in-variables algorithm $\hat{z}_i$ is fixed, we omitted it from the arguments of Q_i, f_i and J_i. When minimizing (3.97) we can use a nonlinear least squares algorithm with the nk-dimensional virtual response function defined by

$$Q_i(p)[\ f_i(p) + J_i(p)[\tilde{z}_i - \hat{z}_i]\ ] \qquad (3.98)$$

for the i-th observation.

If the problem is multifunctional, i.e., nk > 1 , then the decomposition (3.96) is not unique. It is advisible to use the Cholesky decomposition $[\ J_iV_iJ_i^T\ ] = L_iL_i^T$ where L_i is a lower triangular matrix. Then $Q_i = L_i^{-1}$ is a suitable matrix satisfying (3.96). Efficient algorithms for obtaining L_i and then Q_i can be found in the book of Wilkinson and Reinsch (ref. 38). Lines 5404 through 5436 of the following module are based on their algorithmic ideas.

The organization of the module is somewhat tricky in order to make use of the nonlinear least squares module M45. Indeed, the module M52 is essentially a server subroutine for the module M45.

Program module M52

```
5200 REM ****************************************************
5202 REM *    FITTING AN ERROR-IN-VARIABLES MODEL          *
5204 REM *            OF THE FORM F(Z,P)=0                 *
5206 REM *   MODIFIED PATINO-LEAL - REILLY  METHOD         *
5208 REM ****************************************************
5210 REM INPUT:
5212 REM    NM      NUMBER OF SAMPLE POINTS
5214 REM    NZ      NUMBER OF VARIABLES
5216 REM    NK      NUMBER OF EQUATIONS
5218 REM    NP      NUMBER OF PARAMETERS
5220 REM T(NM,1...NZ)  TABLE OF OBSERVATIONS
5222 REM   R(NZ)    VARIANCES OF VARIABLES
5224 REM   P(NP)    INITIAL PARAMETER ESTIMATES
5226 REM    EP      THRESHOLD ON RELATIVE STEP LENGTH OF PARAMETERS
5228 REM    EZ      THRESHOLD ON STEP LENGTH OF VARIABLES
5230 REM    IM      MAXIMUM NUMBER OF ITERATIONS
5232 REM OUTPUT:
5234 REM    ER      STATUS FLAG
5236 REM               0  SUCCESSFUL ESTIMATION
5238 REM               1  THRESHOLDS NOT ATTAINED
5240 REM               2  MATRIX Fz'*R*Fz NOT POSITIVE DEFINITE
5242 REM                  ( LOCALLY DEPENDENT EQUATIONS )
5244 REM   P(NP)    PARAMETER ESTIMATES
5246 REM T(NM,NZ+1...2*NZ)  CORRECTED VARIABLES
5248 REM   .....    FURTHER RESULTS PRINTED IN THE MODULE
5250 REM USER-SUPPLIED SUBROUTINES:
5252 REM  FROM LINE 900; OBLIGATORY STATEMENTS ARE
5254 REM            GOSUB 5398 :RETURN
5256 REM  FROM LINE 700:
5258 REM     Z(1...nz),P(1...np) ---> F(1...nk)
5260 REM               ( FUNCTION VALUE EVALUATION )
5262 REM  FROM LINE 600:
5264 REM            Z(1...nz),P(1...np) ---> E(1...nk,1...nz)
5266 REM            ( PARTIAL DERIVATIVES OF F WITH RESPECT TO Z )
5268 REM AUXILIARY ARRAYS:
5270 REM  A(NP,NP),C(NP,NP),U(NP,NP),X(2*NZ),Y(NK),B(NP),D(NP),S(NP),G(NK,NP)
5272 REM  V(NM,NK),Q(NK,NK),H(NK,NK),W(NK,NK)
5274 REM MODULES CALLED: M16,M18,M41,M45
5276 NX=NZ+NZ :NY=NK :WI=0
5278 REM ---------- INITIAL ESTIMATE OF VARIABLES
5280 FOR M=1 TO NM
5282  FOR I=1 TO NZ :T(M,NZ+I)=T(M,I) :NEXT I
5284  FOR I=1 TO NY :V(M,I)=0 :NEXT I
5286 NEXT M
5288 FOR IG=1 TO IM
5290  LPRINT :LPRINT TAB(15);"********** NONLINEAR LSQ ESTIMATION ";
5292  LPRINT "NUMBER";IG;" **********": LPRINT
5294  FOR I=1 TO NP :S(I)=P(I) :NEXT I :GOSUB 4500
5296  IF ER>0 THEN 5448
5298  LPRINT :LPRINT TAB(15);"********** CORRECTED VARIABLES **********"
5300  LPRINT :LPRINT
5302  FOR M=1 TO NM
5304   FOR IT=1 TO IM
5306 ZE=0
5308 FOR I=1 TO NX :X(I)=T(M,I) :NEXT I
5310 GOSUB 5370
```

```
5312 FOR I=1 TO NK
5314  Y=0 :FOR J=1 TO NK :Y=Y+A(I,J)*F(J) :NEXT J :Y(I)=Y
5316 NEXT I
5318 FOR I=1 TO NZ
5320  Z=X(I) :FOR J=1 TO NK :Z=Z-R(I)*E(J,I)*Y(J) :NEXT J
5322  D=Z-T(M,NZ+I) :T(M,NZ+I)=Z :ZE=ZE+D*D
5324 NEXT I
5326    IF SQR(ZE)<=EZ THEN 5332
5328   NEXT IT
5330   ER=1
5332   REM --------- PRINT VARIABLES
5334   IF M>1 THEN 5342
5336   LPRINT V$
5338   LPRINT " MEAS";TAB( 7);" I";TAB(11);"Z(I) MEAS";TAB(26);"Z(I) CORR";
5340   LPRINT TAB(40);"EQUATION ERROR AFTER CORRECTION" :LPRINT V$
5342   FOR I=1 TO NZ
5344    IF I=1 THEN LPRINT M;
5346    LPRINT TAB( 7);I; :LPRINT USING F$;X(I),Z(I)
5348   NEXT I
5350   GOSUB 700
5352   FOR K=1 TO NK :LPRINT TAB(45);"F(";K;")=";F(K) :NEXT K
5354  NEXT M
5356  LPRINT V$ :LPRINT
5358  IF ER=1 THEN 5446
5360  REM ---------- TERMINATION CONDITION
5362  PE=0 :FOR I=1 TO NP :PE=PE+(P(I)-S(I))^2/S(I)^2 :NEXT I
5364  IF SQR(PE)<=EP THEN ER=0 :GOTO 5448
5366 NEXT IG
5368 ER=1 :GOTO 5448
5370 REM ---------- A=(Fz'*R*Fz)^-1
5372 GOSUB 5376 :N=NK :GOSUB 1600 :IF ER=1 THEN ER=2
5374 RETURN
5376 REM ---------- A=Fz'*R*Fz AND F=F+Fz*(Z-X)
5378 FOR I0=1 TO NZ :Z(I0)=X(NZ+I0) :NEXT I0
5380 GOSUB 600
5382 FOR I0=1 TO NK :FOR J0=1 TO I0
5384  A=0 :FOR K0=1 TO NZ :A=A+R(K0)*E(I0,K0)*E(J0,K0) :NEXT K0 :A(I0,J0)=A
5386 NEXT J0 :NEXT I0
5388 GOSUB 700
5390 FOR I0=1 TO NK
5392  A=F(I0) :FOR J0=1 TO NZ :A=A+E(I0,J0)*(X(J0)-Z(J0)) :NEXT J0 :F(I0)=A
5394 NEXT I0
5396 RETURN
5398 REM ---------- RESPONSE FUNCTION
5400 GOSUB 5376 :IF NK>1 THEN 5404
5402 IF A(1,1)=0 THEN ER=2 :GOTO 5446 ELSE Q(1,1)=SQR(1/A(1,1)) :GOTO 5438
5404 REM ---------- ---------- DECOMPOSE A INTO H*H' BY CHOLESKY METHOD
5406 FOR I0=1 TO NK
5408  FOR J0=1 TO I0-1
5410   A=A(I0,J0) :FOR K0=1 TO J0-1 :A=A-H(I0,K0)*H(J0,K0) :NEXT K0
5412   H(I0,J0)=A/H(J0,J0)
5414  NEXT J0
5416  A=A(I0,I0) :FOR K0=1 TO I0-1 :A=A-H(I0,K0)^2 :NEXT K0
5418  IF A<=0 THEN ER=2 :GOTO 5446 ELSE H(I0,I0)=SQR(A)
5420 NEXT I0
5422 REM ---------- ---------- FIND  Q'= H^(-1)
5424 FOR I0=1 TO NK
5426  Q(I0,I0)=1/H(I0,I0)
```

```
5428  FOR J0=I0+1 TO NK
5430   A=0 :FOR K0=1 TO J0-1:A=A-H(J0,K0)*Q(K0,I0) :NEXT K0
5432   Q(J0,I0)=A/H(J0,J0)
5434  NEXT J0
5436 NEXT I0
5438 REM ---------- ---------- COMPUTE Y = Q*F
5440 FOR I0=1 TO NK
5442  Y=0 :FOR J0=1 TO I0 :Y=Y+Q(I0,J0)*F(J0) :NEXT J0  :Y(I0)=Y
5444 NEXT I0
5446 RETURN
5448 REM ---------- END OF MODULE
5450 IF ER=1 THEN LPRINT "REQUIRED THRESHOLD NOT ATTAINED"
5452 IF ER=2 THEN LPRINT "LOCALLY DEPENDENT EQUATIONS"
5454 LPRINT :LPRINT TAB(15);"********** END OF ERROR-IN-VARIABLES ";
5456 LPRINT "ESTIMATION **********" :LPRINT
5458 RETURN
5460 REM ******************************************************
```

The module M45 of the nonlinear least squares method expects a user routine, starting at line 900 and computing the values of the response function. In the error-in-variables algorithm the virtual response function is the nk vector (3.98). To free the user from unnecessary programming, we provide a subroutine starting at line 5398 that computes (3.98). Therefore, the subroutine at line 900 now consists of the single statements: "GOSUB 5398 :RETURN". There are, however, two subroutines left to you. One of them starts at line 700 , and evaluates the function $f(z,p)$. The other subroutine starts at line 600 , and evaluates the partial derivatives of functions f with respect to z . The result is an $nk \times nz$ matrix stored in the two dimensional array E .

We assume that the covariance matrix of the errors is independent of the observations and, for the sake of simplicity, is diagonal. Since the array V is already used, the error variances are stored in the vector R .

The return value ER = 2 of the status flag indicates that the functional relationships (3.85) are linearly dependent, i.e., at least one of the equations can be omitted.

Example 3.8 Radiographic calibration by error-in-variables method

In radiographic investigations the image of an object is distorted if the X-rays strike the photographic plate at an oblique angle. In order to calibrate the distortion a spherical ball is investigated. The image is an ellipse with centre $(p_1;\ p_2)$ and further parameters p_3, p_4 and p_5 as described by the equation

$$[z_1-p_1 \quad z_2-p_2] \begin{bmatrix} p_3 & p_4 \\ p_4 & p_5 \end{bmatrix} \begin{bmatrix} z_1-p_1 \\ z_2-p_2 \end{bmatrix} - 1 = 0 .$$

The above model is fitted to 20 observed pairs of coordinates $(\tilde{z}_{i1}; \tilde{z}_{i2})$ by Reilly and Patino-Leal (ref. 35) with the assumption that the errors are normally distributed and independent with variances $\sigma_1^2 = 0.0001$ and $\sigma_2^2 = 0.0001$. The following main program contains the observed coordinates in the DATA lines 114 - 152 . The initial estimates of the parameters are given in line 220. Termination criteria for the parameters (EP) and for the equalibrated variables (EZ) are given in line 218.

```
100 REM ------------------------------------------------------------
102 REM EX. 3.8. ERROR-IN-VARIABLES PARAMETER ESTIMATION - CALIBRATION
104 REM MERGE M16,M18,M41,M45,M52
106 REM ---------- DATA
108 REM (NUMBER OF SAMPLE POINTS)
110 DATA 20
112 REM   (Z1,    Z2 )
114 DATA  0.50, -0.12
116 DATA  1.20, -0.60
118 DATA  1.60, -1.00
120 DATA  1.86, -1.40
122 DATA  2.12, -2.54
124 DATA  2.36, -3.36
126 DATA  2.44, -4.00
128 DATA  2.36, -4.75
130 DATA  2.06, -5.25
132 DATA  1.74, -5.64
134 DATA  1.34, -5.97
136 DATA  0.90, -6.32
138 DATA -0.28, -6.44
140 DATA -0.78, -6.44
142 DATA -1.36, -6.41
144 DATA -1.90, -6.25
146 DATA -2.50, -5.88
148 DATA -2.88, -5.50
150 DATA -3.18, -5.24
152 DATA -3.44, -4.86
200 REM ---------- READ DATA
202 READ NM
204 NZ=2 :NK=1 :NP=5 :IM=20
206 DIM T(NM,2*NZ),V(NM,NK),R(NZ),P(NP),Z(NZ),X(2*NZ),Y(NK),F(NK)
208 DIM E(NK,NZ),A(NP,NP),C(NP,NP),U(NP,NP),B(NP),D(NP),S(NP)
210 DIM G(NK,NP),O(NK,NK)
212 FOR I=1 TO NM :READ T(I,1),T(I,2) :NEXT I
214 R(1)=.0001 :R(2)=.0001
216 REM ---------- ITERATION CONTROL
218 EP=.001 :EZ=.001 :IM=20
220 P(1)=-.57 :P(2)=-3.4 :P(3)=.1 :P(4)=.00057 :P(5)=.082
222 GOSUB 5200
224 IF ER<>0 THEN LPRINT "STATUS FLAG:";ER
226 STOP
600 REM ---------- PARTIAL DERIVATIVES WITH RESPECT TO Z
602 W1=Z(1)-P(1) :W2=Z(2)-P(2)
604 E(1,1)=2*(P(3)*W1+P(4)*W2) :E(1,2)=2*(P(5)*W2+P(4)*W1)
606 RETURN
```

```
700 REM ---------- FUNCTION EVALUATION
702 W1=Z(1)-P(1) :W2=Z(2)-P(2)
704 F(1)=W1*W1*P(3)+W2*W2*P(5)+2*W1*W2*P(4)-1
706 RETURN
900 REM ---------- OBLIGATORY STATEMENT
902 GOSUB 5398
904 RETURN
```

The detailed output produced by the program is rather long. This problem needs three repetitions of the algorithmic steps (ii-iv) , so that the module M45 is called three times. We omit most of the output associated with the first two calls. Nonlinear balance equilibration is also carried out three times, but only the results of the final equilibration are shown here.

```
              ********** NONLINEAR LSQ ESTIMATION NUMBER 1  **********


STARTING POINT          SUM SQ= 10502.64

                        P( 1 )=-.57
                        P( 2 )=-3.4
                        P( 3 )= .1
                        P( 4 )= .00057
                        P( 5 )= .082

. . .

IT= 4    PM=0.1E-04     SUM SQ= 893.6689          SL= 2.584833E-03

                        P( 1 )=-1.008047
                        P( 2 )=-2.923785
                        P( 3 )= 8.744357E-02
                        P( 4 )= 1.646901E-02
                        P( 5 )= 7.961292E-02

IT= 5    PM=0.1E-05     SUM SQ= 893.6689          SL= 5.460908E-05

. . .

              ********** NONLINEAR LSQ ESTIMATION NUMBER 2  **********

. . .

IT= 3    PM=0.1E-3      SUM SQ= 882.4754         SL= 2.366362E-04

. . .

              ********** NONLINEAR LSQ ESTIMATION NUMBER 3  **********
. . .

SUM OF SQUARES ......................  882.4716
DEGREES OF FREEDOM ..................  15
STANDARD ERROR ......................  7.670165
CRITICAL T-VALUE AT 95 % CONF. LEVEL   2.13
```

```
----------------------------------------------------------------------
PARAMETER     ESTIMATE      ST. ERROR     LOWER BOUND   UPPER BOUND
----------------------------------------------------------------------
 P( 1 )       -.99959E+00   0.11134E+00   -.12367E+01   -.76245E+00
 P( 2 )       -.29308E+01   0.10976E+00   -.31646E+01   -.26970E+01
 P( 3 )       0.87566E-01   0.41098E-02   0.78813E-01   0.96320E-01
 P( 4 )       0.16235E-01   0.27473E-02   0.10383E-01   0.22087E-01
 P( 5 )       0.79747E-01   0.34953E-02   0.72302E-01   0.87192E-01
----------------------------------------------------------------------

. . .
              ********** CORRECTED VARIABLES **********

----------------------------------------------------------------------
MEAS  I  Z(I) MEAS      Z(I) CORR     EQUATION ERROR AFTER CORRECTION
----------------------------------------------------------------------
 1     1 0.50000E+00   0.53418E+00
       2 -.12000E+00   -.72187E-01     F( 1 )= 4.112721E-05
 2     1 0.12000E+01   0.11735E+01
       2 -.60000E+00   -.62550E+00     F( 1 )= 8.344651E-07
 3     1 0.16000E+01   0.15353E+01
       2 -.10000E+01   -.10490E+01     F( 1 )=-1.716614E-05
 4     1 0.18600E+01   0.17997E+01
       2 -.14000E+01   -.14368E+01     F( 1 )=-2.652407E-05
 5     1 0.21200E+01   0.22738E+01
       2 -.25400E+01   -.24939E+01     F( 1 )=-3.045797E-05
 6     1 0.23600E+01   0.24349E+01
       2 -.33600E+01   -.33544E+01     F( 1 )=-1.883507E-05
 7     1 0.24400E+01   0.24265E+01
       2 -.40000E+01   -.39986E+01     F( 1 )=-5.900860E-06
 8     1 0.23600E+01   0.22679E+01
       2 -.47500E+01   -.47180E+01     F( 1 )= 8.940697E-06
 9     1 0.20600E+01   0.20305E+01
       2 -.52500E+01   -.52326E+01     F( 1 )= 1.740456E-05
 10    1 0.17400E+01   0.17378E+01
       2 -.56400E+01   -.56381E+01     F( 1 )= 2.121925E-05
 11    1 0.13400E+01   0.13576E+01
       2 -.59700E+01   -.59932E+01     F( 1 )= 2.074242E-05
 12    1 0.90000E+00   0.88154E+00
       2 -.63200E+01   -.62804E+01     F( 1 )= 1.490116E-05
 13    1 -.28000E+00   -.27835E+00
       2 -.64400E+01   -.65403E+01     F( 1 )=-6.67572E-06
 14    1 -.78000E+00   -.78959E+00
       2 -.64400E+01   -.65081E+01     F( 1 )=-1.490116E-05
 15    1 -.13600E+01   -.13513E+01
       2 -.64100E+01   -.63818E+01     F( 1 )=-2.110005E-05
 16    1 -.19000E+01   -.18675E+01
       2 -.62500E+01   -.61810E+01     F( 1 )=-2.169609E-05
 17    1 -.25000E+01   -.24688E+01
       2 -.58800E+01   -.58347E+01     F( 1 )=-1.376867E-05
 18    1 -.28800E+01   -.28875E+01
       2 -.55000E+01   -.55085E+01     F( 1 )=-7.748604E-07
 19    1 -.31800E+01   -.31743E+01
       2 -.52400E+01   -.52345E+01     F( 1 )= 1.28746E-05
 20    1 -.34400E+01   -.34760E+01
       2 -.48600E+01   -.48884E+01     F( 1 )= 3.314018E-05
----------------------------------------------------------------------

          ********** END OF ERROR-IN-VARIABLES ESTIMATION **********
```

As a byproduct, we obtain confidence intervals for the parameters and corrected values for the measured variables. The equation errors after correction are all negligibly small, showing that the balance equilibration has been done properly. The resulting fit is shown in Fig. 3.4.

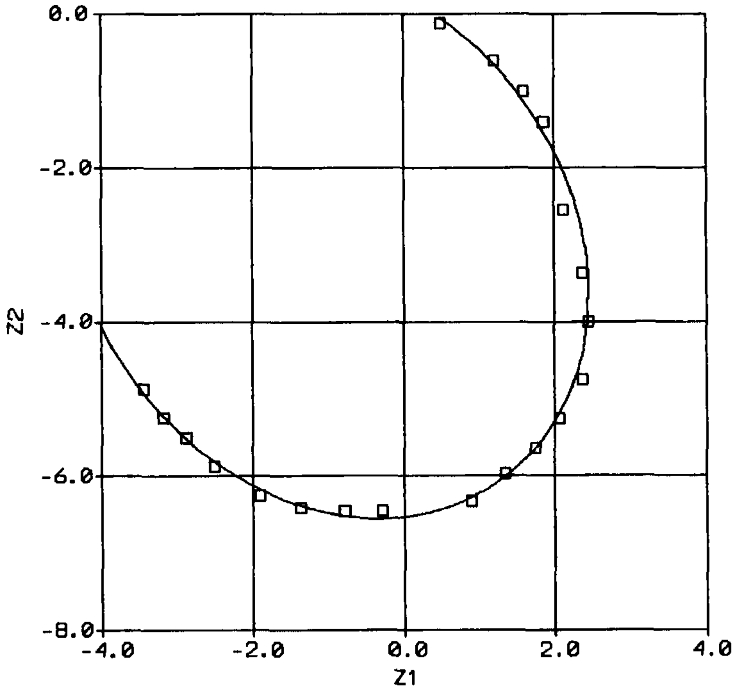


Fig. 3.4. Observed image (points) and fitted curve (continuous) in the radiographic calibration problem

Exercise

□ Assuming $\sigma_x^2/\sigma_y^2 = 0.01$ fit an error-in-variables straight line to the data listed in Table 1.1 of Section 1.8.2. Show that the slope is between the two limiting values, obtained by regressing y on x and vica versa in Section 3.1.

3.9 FITTING ORTHOGONAL POLYNOMIALS

You can use multivariable linear regression to fit a polynomial

$$y = \sum_{i=0}^{n} a_i x^i \tag{3.99}$$

to the set $\{ (x_j, \tilde{y}_j);\ j = 1,2,\ldots,np \}$ of points. The i-th row of the observation matrix $\mathbf{X}$ in (3.20) is then $(1, x_j, x_j^2, \ldots, x_j^n)$. Even for a polynomial of moderately high degree, however, the resulting cross-product matrix $\mathbf{X}^T\mathbf{X}$ has a large condition number, and the problem is ill-conditioned. This difficulty can be avoided by estimating the parameters $s_0, s_1, \ldots, s_n$ of the function

$$y = \sum_{i=0}^{n} s_i P_i(x) \ , \tag{3.100}$$

where $P_0, P_1, \ldots, P_n$ are polynomials, orthogonal on the given set of grid points. To define this property introduce the notation

$$< P_k(x),P_l(x) > = \sum_{j=1}^{np} P_k(x_j)P_l(x_j) \ .$$

According to Forsythe (ref. 39), the polynomials P_k and P_l are orthogonal over the grid points $(x_1, x_2, \ldots, x_{np})$, if $< P_k(x),P_l(x) > = 0$. By the orthogonality of the polynomials P_i, the cross product matrix of the linear regression problem associated with the model (3.100) is diagonal and hence very easy to invert. A further advantage is that increasing the degree of the polynomial from n to $n+1$, the previous estimates $\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_n$ remain unchanged.

The Forsythe polynomials are defined by the recursive relationships

$$P_{-1}(x) = 0 \ , \quad P_0(x) = 1 \ , \quad P_{i+1}(x) = (x-\alpha_{i+1})P_i(x) - \beta_i P_{i-1}(x) \ ,$$

where

$$\alpha_{i+1} = \sum_{j=1}^{np} x_j \,[P_i(x_j)]^2 \ , \quad \beta_i = \omega_{ii}/\omega_{i-1,i-1} \quad \text{and} \quad \omega_{ii} = < P_i(x),P_i(x) > \ .$$

The least squares estimate of the parameters $s_0, s_1, \ldots, s_n$ in (3.100) are simply obtained by

$$\hat{s}_i = \omega_i/\omega_{ii} \ , \quad \text{where} \quad \omega_i = < \tilde{Y},P_i(x) > \ .$$

Rearranging the polynomial (3.100) to the canonical form (3.99) gives the estimates for the coefficients $a_0, a_1, \ldots, a_n$. The following module based on (ref. 40) fits polynomials of degree $n = 0, 1, \ldots,$ ND to the set of NP points, where ND < NP. If the x_j values are not all different or numerical errors are likely to corrupt the results, the module automatically decreases the maximum degree ND, and sets the status flag ER = 1.

Program module M52

```
5500 REM ****************************************************
5502 REM *              POLYNOMIAL REGRESSION               *
5504 REM *      USING FORSYTHE ORTHOGONAL POLYNOMIALS       *
5506 REM ****************************************************
5508 REM INPUT:
5510 REM     NP      NUMBER OF SAMPLE POINTS
5512 REM    X(NP)    VALUES OF INDEPENDENT VARIABLE
5514 REM    Y(NP)    OBSERVATIONS OF DEPENDENT VARIABLE
5516 REM     ND      MAXIMUM DEGREE OF THE POLYNOMIAL ( ND<NP )
5518 REM OUTPUT:
5520 REM     ER      ERROR FLAG
5522 REM                0  SUCCESSFUL REGRESSION
5524 REM                1  SPECIFIED ND IS TOO LARGE
5526 REM                   ( IN THIS CASE A FURTHER OUTPUT IS
5528 REM     ND             ACTUAL MAXIMUM DEGREE )
5530 REM   C(J,I)    I-TH COEFFICIENT IN THE J-TH ORDER POLYNOMIAL
5532 REM             ( Y = SUM [C(J,I)*X^I]    I=0...J )
5534 REM   C(NP,J)   RESIDUAL SUM OF SQUARES FOR THE J-TH POLYNOMIAL
5536 REM REMARK:     MINIMUM SIZE OF ARRAY C IS NP*NP
5538 REM ---------- GENERATE VALUES OF FORSYTHE POLYNOMIALS
5540 FOR I=1 TO NP :C(0,I)=1 :NEXT I :C(1,0)=NP :BE=0
5542 FOR J=1 TO ND
5544 ER=0 :IF ND>NP-1 THEN ER=1 :ND=NP-1
5546  AL=0 :FOR I=1 TO NP :AL=AL+X(I)*C(J-1,I)*C(J-1,I) :NEXT I
5548  AL=AL/C(J,0) :C(NP,J)=AL
5550  FOR I=1 TO NP
5552   C(J,I)=(X(I)-AL)*C(J-1,I)
5554   IF BE<>0 THEN C(J,I)=C(J,I)-BE*C(J-2,I)
5556  NEXT I
5558  SM=0 :FOR I=1 TO NP :SM=SM+C(J,I)*C(J,I) :NEXT I
5560  C(J+1,0)=SM :BE=SM/C(J,0)
5562  IF SM<=.000001*C(J,0) THEN ER=1 :ND=J-1 :GOTO 5566
5564 NEXT J
5566 REM ---------- WEIGHTING COEFFICIENTS OF POLYNOMIALS
5568 SM=0 :FOR I=1 TO NP :SM=SM+Y(I) :NEXT I
5570 C(0,0)=1
5572 FOR I=1 TO NP-1 :C(0,I)=0 :NEXT I
5574 C(0,NP)=SM/NP :BE=0
5576 FOR J=1 TO ND
5578  SM=0 :FOR I=1 TO NP :SM=SM+Y(I)*C(J,I) :NEXT I
5580  AL=C(NP,J) :BU=C(J+1,0)/C(J,0)
5582  C(J,0)=-AL*C(J-1,0) :IF BE<>0 THEN C(J,0)=C(J,0)-BE*C(J-2,0)
5584  FOR I=1 TO ND
5586   C(J,I)=C(J-1,I-1)-AL*C(J-1,I)
5588   IF BE<>0 THEN C(J,I)=C(J,I)-BE*C(J-2,I)
5590  NEXT I
5592  FOR I=J+1 TO NP-1 :C(J,I)=0 :NEXT I
5594  C(J,NP)=SM/C(J+1,0) :BE=BU
5596 NEXT J
```

```
5598 REM ---------- CANONICAL POLYNOMIALS AND SUM OF SQUARES
5600 C(0,0)=C(0,NP)*C(0,0) :C(0,NP)=0 :SM=0 :Y=C(0,0)
5602 FOR I=1 TO NP :SM=SM+(Y(I)-Y)*(Y(I)-Y) :NEXT I :C(NP,0)=SM
5604 FOR J=1 TO ND
5606  SM=C(J,NP)
5608  FOR I=0 TO J :C(J,I)=SM*C(J,I)+C(J-1,I) :NEXT I :C(J,NP)=0
5610  SM=0
5612  FOR I=1 TO NP
5614   Y=C(J,J) :FOR K=J-1 TO 0 STEP -1 :Y=Y*X(I)+C(J,K) :NEXT K
5616   SM=SM+(Y(I)-Y)*(Y(I)-Y)
5618  NEXT I
5620  C(NP,J)=SM
5622  IF SM>=C(NP,J-1) THEN ND=J-1:GOTO 5626
5624 NEXT J
5626 RETURN
5628 REM ********************************************************
```

Example 3.9 Polynomial regression through Forsythe orthogonalization

The DATA statements of the following program include 12 data pairs $(x_i, \tilde{y}_i)$, where x is the temperature (K) and y is the equilibrium vapor pressure (bar, 1 bar = 10^5 Pa) of liquid oxygen (ref. 41).

We attempt to fit least squares polynomials of degree 0 through 11 , describing the vapor pressure as a function of temperature.

```
100 REM ------------------------------------------------------------
102 REM EX. 3.9. POLYNOMIAL REGRESSION
104 REM USING FORSYTHE ORTHOGONAL POLYNOMIALS
106 REM MERGE M55
108 REM ---------- DATA
110 REM (NUMBER OF POINTS AND MAXIMUM DEGREE)
112 DATA 12,11
114 REM (X(I)-temp     Y(I)-press)
116 DATA  54.35,        0.001500
118 DATA  60,           0.007317
120 DATA  70,           0.06236
122 DATA  80,           0.3003
124 DATA  90,           0.9943
126 DATA 100,           2.546
128 DATA 110,           5.443
130 DATA 120,          10.21
132 DATA 130,          17.44
134 DATA 140,          27.82
136 DATA 150,          42.23
138 DATA 154.77,       50.87
200 REM ---------- READ DATA AND CALL MODULE
202 READ NP,ND
204 DIM X(NP),Y(NP),C(NP,NP)
206 FOR I=1 TO NP :READ X(I),Y(I) :NEXT I
208 GOSUB 5500
```

```
210 REM ---------- PRINT RESULTS
212 IF ER THEN LPRINT "ER=1 : MAX. ADMISSIBLE DEGREE IS";ND :LPRINT
214 FOR J=0 TO ND
216  LPRINT "DEGREE:";J ,"RESIDUAL SUM OF SQUARES:";C(NP,J)
218  LPRINT USING "Y(X)= #.#####^^^^"; C(J,0)
220  FOR I=1 TO J
222   LPRINT USING "      #.#####^^^^ * X^##";C(J,I),I
224  NEXT I
226  LPRINT
228 NEXT J
230 STOP
```

The output begins with a warning message:

```
ER=1 : MAX. ADMISSIBLE DEGREE IS 7

DEGREE: 0     RESIDUAL SUM OF SQUARES: 3512.318
Y(X)= 0.13160E+02

DEGREE: 1     RESIDUAL SUM OF SQUARES: 807.5648
Y(X)= -.34171E+02
      0.45109E+00 * X^ 1

DEGREE: 2     RESIDUAL SUM OF SQUARES: 54.10914
Y(X)= 0.51196E+02
      -.13611E+01 * X^ 1
      0.86470E-02 * X^ 2

DEGREE: 3     RESIDUAL SUM OF SQUARES: .5436249
Y(X)= -.34631E+02
      0.14353E+01 * X^ 1
      -.19677E-01 * X^ 2
      0.90203E-04 * X^ 3

DEGREE: 4     RESIDUAL SUM OF SQUARES: 6.469505E-03
Y(X)= -.98362E+00
      -.32470E-01 * X^ 1
      0.32005E-02 * X^ 2
      -.61456E-04 * X^ 3
      0.36254E-06 * X^ 4

DEGREE: 5     RESIDUAL SUM OF SQUARES: 4.128297E-03
Y(X)= -.96379E+01
      0.43970E+00 * X^ 1
      -.67383E-02 * X^ 2
      0.39624E-04 * X^ 3
      -.13536E-06 * X^ 4
      0.95280E-09 * X^ 5

DEGREE: 6     RESIDUAL SUM OF SQUARES: 1.109194E-04
Y(X)= 0.33366E+02
      -.23770E+01 * X^ 1
      0.67954E-01 * X^ 2
      -.98769E-03 * X^ 3
      0.76043E-05 * X^ 4
      -.29378E-07 * X^ 5
      0.48386E-10 * X^ 6
```

```
DEGREE: 7    RESIDUAL SUM OF SQUARES: 8.767201E-06
Y(X)= 0.78537E+01
      -.42636E+00 * X^ 1
      0.55539E-02 * X^ 2
      0.95423E-04 * X^ 3
      -.34214E-05 * X^ 4
      0.36510E-07 * X^ 5
      -.16588E-09 * X^ 6
      0.29283E-12 * X^ 7
```

Polynomials of higher degree can be fitted in this case only if double precision is used for the computations.

Exercises

- Insert the following line into the program to repeat the computations in double precision:
  99 DEFDBL A-H,O-Z
  Compare the residual sum of squares obtained in single and in double precision.

- Since the vapor pressure changes over several orders of magnitude, it is more reasonable to fit polynomials to the logarithm of the vapor pressure. Repeat the computations inserting a logarithmic transformation for y. Show that for a given order of polynomial the maximum relative error of the vapor pressure is considerable lower for the logarithmized model.

- Try to fit polynomials of degree 3 through 7 to the data using the module M42 . Discuss the advantages of orthogonal polynomials in view of your experiences.

3.10 APPLICATIONS AND FURTHER PROBLEMS

3.10.1 On different criteria for fitting a straight line

You have now several estimators to fit the line $y = ax + b$ to the points $(\tilde{y}_i, x_i)$: the method of least squares (Section 3.1), the method of least absolute deviations (Section 1.8.2) and the minimax method (Section 1.8.3). Which one to use in a particular case? To answer this question consider first the problem of outliers., i.e., observations with gross errors. The presence of outliers is, unfortunately, not rare in large samples. Since in its objective function these large deviations are squared, the least squares estimates are clearly more sensitive to the outliers than the method of least absolute deviations. The least squares is a maximum likelihood estimator so far the error distribution is normal. In a normal distribution the probability of outliers is vanishingly small, and hence their presence signifies deviation from the assumed normality. Therefore, if the error distribution is suspected to be "flat", i.e., the probabilitiy of large errors is higher than expected in a normal distribution then the more robust least absolute deviations criterion is preferable.

In practice the error distribution is usually unknown, and the choice can be made on the basis of the empirical curtosis of the residuals defined by

$$k = n \left[\Sigma r_i^4\right] / \left[\Sigma r_i^2\right]^2, \qquad (3.101)$$

where the r_i's are the residuals from a least squares fit, and the summation goes from 1 to the number of sample points. According to (ref. 42), in case of a large curtosis, $k > 3.8$, the sum of absolute deviations is better to use. The other extreme case is indicated by a low curtosis, $k < 2.1$, when the error distribution is possibly "sharper" than the normal. In this case the minimax criterion is a good choice.

Exercise

□ Select the suitable criterion for the nicotine - tar data investigated in Sections 1.8.2, 1.8.3 and 3.1. Inspecting the shadow prices in the minimax estimation omit the most suspectible point and repeat the estimations by the different methods. Discuss the sensitivity of the various estimates with respect to omitting this point.

3.10.2 Design of experiments for parameter estimation

The best known application of experiment design is to find the extremum of a quantity depending on further variables by observing its value at appropriately selected points (refs. 43-46). In this section, however, consideration is restricted to design methods, purported to increase the reliability of estimates when fitting a model to observations.

A k - point design is described by the design matrix $\mathbf{X}_k$, consisting of k rows. The i-th row of the matrix specify the values of the the independent variables to be selected in the i-th experiment. Depending on the linearity or nonlinearity of the model, the design matrix affects the covariance matrix $\mathbf{C}_p$ of the estimates according to the expressions (3.30) and (3.45), respectively. The covariance matrix, in turn, determines the joint confidence region (3.32) of the parameters. Our goal is to obtain a confidence region as small as possible. The size of the ellipsoid (3.32) can be measured in different ways, and these give rise to various optimality concepts listed in Table 3.7.

Table 3.7
Optimality criteria in experiment design

Optimality concept	Criterion
D	det[C_p] ---> min
A	trace[C_p] ---> min
E	λ_{min}[C_p] ---> max

According to Table 3.7, a D - optimal design **X** minimizes the volume of the confidence ellipsoid. The mean square length of the axes is minimized in A - optimal design, whereas E - optimality means the minimum length of the longest axis. In the case of a nonlinear response function the Jacobian matrix (3.41), and hence also the approximate covariance matrix (3.45) depend on the parameter values, in addition to the design X_k. Thus optimality of a design is defined at some fixed parameter vector.

To obtain a meaningful extremum problem the number of experiments k and the set of feasible vectors of the independent variables T are fixed. In most cases T is defined by inequalities $x^L \le x_i \le x^U$, i = 1,2,...,k. Though introducing penalty functions such constrained extremum problems can be solved by the methods and modules described in Section 2.4, this direct approach is usually very inefficient. In fact, experiment design is not easy. The dimensionality of the extremum problem is high, the extrema are partly on the boundaries of the feasible region T, and since the objective functions are symmetric in the vectors x_1, x_2, ..., x_k , you have to face the difficult problem of multiple maxima (ref. 44).

In practice it is more efficient to adopt a less ambitious approach of "polishing" a starting design X_k iteratively, increasing the value of the objective function in each iteration and thereby determining a nearly optimal design. A useful algorithm is to drop one point of the current design and add an optimally selected new point x_k to the remaining design X_{k-1}. This inner iteration is repeated for each point of the design in turn. Then the procedure can be restarted updating the first point again. The convergence rate might be disappointing, but high accuracy is not necessary because of the inherent approximations.

Example 3.10.2 Approximate D - optimal design for estimating Michaelis-Menten parameters

Starting with the substrate values $x_i = [S_i]$ in Table 3.4, we construct a nearly D - optimal design to estimate the parameters of the response function (3.55). Since we have 10 measurements in the starting design, we fix $k = 10$. The feasible region is given by $x^L = 0$ and $x^U = 5\times10^{-2}$ mol/l. The nominal parameter values, necessary to be selected a priori, are $V = 4\times10^{-2}$ mol/(l s) and $K = 4\times10^{-2}$ mol/l. Constant error variance is assumed.

In every inner iteration step the objective function

$$Q(x) = \det\{ J_{k-1}^T J_{k-1} + j(x;V,K)j^T(x;V,K) \} \qquad (3.102)$$

is minimized subject to the constraint $x^L \le x \le x^U$, where the Jacobian corresponding to the remaining experiment design X_{k-1}, denoted by J_{k-1}, does not depend on x , and j is the column vector of partial derivatives at x .

Evaluating (3.102) over a course grid we can find at most two local maxima. Therefore, the program designed to solve this problem first divides the interval $[x^L,x^U]$, each of the two subintervals bracketing one of the maxima. On each interval the single maximum is localized by module M25, and the larger one is selected for the new point of the design. As shown in Table 3.8, the first three points are immediately replaced by x^U. In the next 5 inner iterations, however, the global maximum is located at inner points of the feasible interval. Finally, (3.102) takes its maximum value again on the upper end when replacing the last 2 points. The design obtained in the first outer iteration (i.e., after updating all the 10 points) remains almost unchanged subsequently, with the inner points approaching to a single value. The resulting design decreases the volume of the (approximate) confidence ellipsoid of the parameters by a factor of 2 with respect to the starting design.

Table 3.8
Outer iterations of the experiment design procedure

Outer	Design points $x_i\times10^3$, mol/l										$Q\times10^8$
iteration	1	2	3	4	5	6	7	8	9	10	
0	1.00	3.00	5.00	8.00	10.00	15.00	20.00	30.00	40.00	50.00	2.3606
1	50.00	50.00	50.00	14.74	14.60	14.63	14.75	15.13	50.00	50.00	4.5854
2	50.00	50.00	50.00	15.42	15.42	15.34	15.33	15.38	50.00	50.00	4.5939
.											
.											
.											
6	50.00	50.00	50.00	15.38	15.39	15.38	15.38	15.38	50.00	50.00	4.5939

In this example the approximate design consists only of two different points with replicates. Restricting the number of points to $k = 2$ right at the beginning, the problem can be solved by hand calculations yielding the same result. You should not draw, however, overly general conclusion from this fact, since the number of different points in a D - optimal design can exceed the number of the parameters. Nevertheless, the optimal design normally involves a relatively small number of different points, and the corresponding observations are hardly suitable for validating the model. Thus the methods of this section apply only when the form of the response function is no more questionable. The need for replicates is a disadvantage also in kinetic analysis, where in a single experimental run the variables can be sampled at points that are not too close. Such additional constraints, however, can be incorporated into the design procedures (see, e.g., refs. 47-48).

Exercise

- Repeat the design procedure of Example 3.10.2 assuming constant relative variances.

3.10.3 Selecting the order in a family of homologous models

In Example 3.5.1 we used ridge regression to confirm that the simpler model (3.33) is preferable to (3.64), though the latter gives slightly better fit. Such model selection problems are faced in many applications, particularly when considering a homologous family of candidate models. For example, in polynomial regression we should select a degree n. A similar problem, discussed in Chapter 5, is to select the order n of a linear differential equation when identifying a pharmacokinetic model.

Example 3.5 has certainly convinced you that the best fitting model is not necessarily the one to chose. In fact, it may be overparameterized with respect to the available data, leading to inflated or even meaningless estimates of the parameters. In addition, a too complex model usually gives unsatisfactory predictions, even slightly apart from the observed values of independent variables. Which model should be then adopted? The simplest rule is that model complexity (i.e., its degree or order) should be increased only while the residual variance is significantly decreasing. This can be tested comparing the residual variances of different models by the F-criterion. This test is not "sharp" enough and frequently suggests a too complex model. A number of criteria has been proposed that take the number of parameters into account more explicitely (for reviews see e.g., refs. 49-50). The most popular one is the Akaike's Information Criterion (ref. 51), suggesting to choose the model for which the quantity

$$AIC = -2 \log \text{(maximium likelihood)} + 2\,np \qquad (3.103)$$

takes its minimum value, where np is the number of the parameters. If the assumptions (i)-(vi) of the least squares method are valid, minimizing (3.103) is equivalent to minimizing the simple expression

$$AIC' = Q(\hat{p};np)/\sigma^2 + 2\,np\ , \qquad (3.104)$$

where $Q(\hat{p};np)$ is the minimum value of the weighted sum of squares with weighting coefficients $w_i = \sigma^2/\sigma_i^2$, found for the model containing np parameters. In practice σ^2 is replaced by its estimate s^2. At this point it is advantageous to use a common s^2, not depending on the number of parameters of the particular model. Obviously, the a priori choice of s^2 significantly affects the outcome of the test.

<u>Exercise</u>

□ Select the degree of the polynomial describing the logarithmic vapor pressure of oxygen as a function of the temperature (see Example 3.9).
Suppose the vapor pressure is exact to three digits and give an estimate s^2 for the logarithms. Apply (3.104) replacing σ^2 with s^2.

3.10.4 <u>Error-in-variables estimation of van Laar parameters from vapor-liquid equilibrium data</u>

At low pressures the following equations are valid for a binary vapor-liquid mixture:

$$\begin{aligned} y_1 p &= \gamma_1 x_1 p_1^{o}(T) \\ (1-y_1)p &= \gamma_2 (1-x_1) p_2^{o}(T)\ , \end{aligned} \qquad (3.104)$$

where

x_1	mole fraction of component 1 in the liquid phase
y_1	mole fraction of component 1 in the vapor phase
p	pressure
T	temperature
$p_i^{o}(T)$	equilibrium vapor pressure of pure component i
γ_i	activity coefficient of component i.

The functions $p_i^{o}(T)$ are supposed to be known exactly, given by the Antoine equation:

$$\log p_i^{o}(T/K)/Pa = A_i - B_i/[T/K + C_i]\ .$$

A popular model to describe the activity coefficients is the van Laar equation

$$\log \gamma_1 = \frac{A}{RT}\left[1 + \frac{A}{B}\frac{x_1}{x_2}\right]^{-2}$$

$$\log \gamma_2 = \frac{B}{RT}\left[1 + \frac{B}{A}\frac{x_2}{x_1}\right]^{-2},$$

where $R = 8.3144$ J/(mol K) is the universal gas constant, A and B are the van Laar parameters, characteristic for the given pair of components.

Estimate the van Laar parameters of methanol (1) and 1,2-dichloro-ethane (2) from equilibria data obtained at $T = 323.15$ K and shown in Table 3.9 if the Antoine parameters for these components are (ref. 52): $A_1 = 23.0843$, $B_1 = 3626.55$, $C_1 = -34.29$ and $A_2 = 21.0692$, $B_2 = 2927.17$, $C_2 = -50.22$.

Table 3.9
Binary vapor-liquid equilibrium data

Measurement	$100x_1$	$100y_1$	$p\times10^{-5}$, Pa
1	30	59.1	0.6450
2	40	60.2	0.6575
3	50	61.2	0.6665
4	70	65.7	0.6685
5	90	81.4	0.6262

The two functional relations stemming from (3.104) take the form

$$F_1(x_1,y_1,T,p) = \\ = \exp\{ \frac{A}{RT}\left[1 + \frac{A}{B}\frac{x_1}{x_2}\right]^{-2} \}x_1\exp\{A_1 - B_1/[T + C_1]\} - y_1p = 0$$

$$F_2(x_1,y_1,T,p) = \\ = \exp\{ \frac{B}{RT}\left[1 + \frac{B}{A}\frac{x_2}{x_1}\right]^{-2} \}(1-x_1)\exp\{A_2 - B_2/[T + C_2]\} - (1-y_1)p = 0$$

The standard errors we assume are $\sigma_x = 0.005$, $\sigma_y = 0.015$, $\sigma_p = 100$ Pa and $\sigma_T = 0.1$ K , based on the reasonable accuracy of vapor-liquid equilibria measurements.

The module M52 is used to solve the error-in-variables estimation problem. The main program contains the starting estimates of the unknown parameters $A = B = RT$ in line 230. The subroutine starting at line 700 computes the current values of the two functional relations. The partial derivatives with respect to the observed variables are computed in lines 600-622.

```
100 REM ------------------------------------------------------------
102 REM EX. 3.10.4  VAN LAAR PARAMETERS (ERROR-IN-VARIABLES METHOD)
104 REM MERGE M16,M18,M41,M45,M52
106 REM ---------- DATA
108 REM ( NM )
110 DATA  5
112 REM  ( X1     Y1        P/PA    T/K )
114 DATA  0.30,  0.591,    .6450E5, 323.15
116 DATA  0.40,  0.602,    .6575E5, 323.15
118 DATA  0.50,  0.612,    .6665E5, 323.15
120 DATA  0.70,  0.657,    .6685E5, 323.15
122 DATA  0.90,  0.814,    .6262E5, 323.15
200 REM ---------- READ DATA
202 READ NM
204 NZ=4 :NK=2 :NP=2 :IM=20
206 DIM T(NM,2*NZ),V(NM,NK),R(NZ),P(NP),Z(NZ),X(2*NZ),Y(NK),F(NK)
208 DIM E(NK,NZ),A(NP,NP),C(NP,NP),U(NP,NP),B(NP),D(NP),S(NP)
210 DIM G(NK,NP),Q(NK,NK)
212 FOR I=1 TO NM
214  FOR J=1 TO NZ :READ T(I,J) :NEXT J
216 NEXT I
218 AN1=23.4803 :BN1=3626.55 :CN1=-34.29 :REM ANTOINE PARAMETERS
220 AN2=21.0692 :BN2=2927.17 :CN2=-50.22 :REM    "
222 RU=8.3144                             :REM GAS CONSTANT
224 R(1)=(.005)^2 :R(2)=(.015)^2 :R(3)=(100)^2 :R(4)=(.1)^2 :REM VARIANCES
226 REM ---------- ITERATION CONTROL PARAMETERS AND INITIAL GUESS
228 EP=.001 :EZ=.001 :IM=20
230 P(1)=RU*323.15 :P(2)=RU*323.15
232 GOSUB 5200
234 IF ER<>0 THEN LPRINT "STATUS FLAG:";ER
236 STOP
600 REM ---------- JACOBIAN MATRIX OF F WITH RESPECT TO Z
602 AA=P(1) :BB=P(2) :PT=Z(3) :T=Z(4)
604 X1=Z(1) :X2=1-X1 :Y1=Z(2) :Y2=1-Y1
606 P1=EXP(AN1-BN1/(T+CN1)) :P2=EXP(AN2-BN2/(T+CN2))
608 S1=AA/RU/T/(1+AA/BB*X1/X2)^2
610 S2=BB/RU/T/(1+BB/AA*X2/X1)^2
612 G1=EXP(S1) :G2=EXP(S2)
614 E(1,1)= G1*P1-2*G1*X1*P1*AA/RU/T*AA/BB/X2^2/(1+X1/X2*AA/BB)^3
616 E(1,2)=-PT :E(1,3)=-Y1 :E(1,4)=-X1*P1*G1*S1/T+G1*X1*P1*BN1/(T+CN1)^2
618 E(2,1)=-G2*P2+2*G2*X2*P2*BB/RU/T*BB/AA/X1^2/(1+X2/X1*BB/AA)^3
620 E(2,2)= PT :E(2,3)=-Y2 :E(2,4)=-X2*P2*G2*S2/T+G2*X2*P2*BN2/(T+CN2)^2
622 RETURN
700 REM ---------- FUNCTION EVALUATION
702 AA=P(1) :BB=P(2) :PT=Z(3) :T=Z(4)
704 X1=Z(1) :X2=1-X1 :Y1=Z(2) :Y2=1-Y1
706 P1=EXP(AN1-BN1/(T+CN1)) :P2=EXP(AN2-BN2/(T+CN2))
708 S1=AA/RU/T/(1+AA/BB*X1/X2)^2
710 S2=BB/RU/T/(1+BB/AA*X2/X1)^2
712 G1=EXP(S1) :G2=EXP(S2)
714 F(1)=G1*X1*P1-Y1*PT
716 F(2)=G2*X2*P2-Y2*PT
718 RETURN
900 REM ---------- OBLIGATORY STATEMENT
902 GOSUB 5398
904 RETURN
```

After two outer iterations the following results are obtained.

```
----------------------------------------------------------------------
PARAMETER     ESTIMATE      ST. ERROR     LOWER BOUND   UPPER BOUND
----------------------------------------------------------------------
 P( 1 )       0.51359E+04   0.10477E+03   0.48939E+04   0.53779E+04
 P( 2 )       0.43207E+04   0.52420E+02   0.41996E+04   0.44418E+04
----------------------------------------------------------------------


----------------------------------------------------------------------
MEAS  I  Z(I) MEAS      Z(I) CORR     EQUATION ERROR AFTER CORRECTION
----------------------------------------------------------------------
 1     1 0.30000E+00   0.29879E+00
       2 0.59100E+00   0.59596E+00
       3 0.64500E+05   0.64525E+05
       4 0.32315E+03   0.32309E+03       F( 1 )=-.015625
                                         F( 2 )=-9.765625E-03
 2     1 0.40000E+00   0.39967E+00
       2 0.60200E+00   0.61214E+00
       3 0.65750E+05   0.65761E+05
       4 0.32315E+03   0.32312E+03       F( 1 )=-.046875
                                         F( 2 )=-2.734375E-02
 3     1 0.50000E+00   0.50018E+00
       2 0.61200E+00   0.62400E+00
       3 0.66650E+05   0.66618E+05
       4 0.32315E+03   0.32324E+03       F( 1 )=-.015625
                                         F( 2 )=-1.367188E-02
 4     1 0.70000E+00   0.69947E+00
       2 0.65700E+00   0.66676E+00
       3 0.66850E+05   0.66835E+05
       4 0.32315E+03   0.32319E+03       F( 1 )=-3.90625E-03
                                         F( 2 )=-3.90625E-03
 5     1 0.90000E+00   0.90060E+00
       2 0.81400E+00   0.81040E+00
       3 0.62620E+05   0.62621E+05
       4 0.32315E+03   0.32315E+03       F( 1 )=-5.078125E-02
                                         F( 2 )=-4.882813E-03
----------------------------------------------------------------------
```

The van Laar parameters $A = 5135.9$ J/mol and $B = 4320.7$ J/mol yield a good fit. The observed variables are only slightly corrected to satisfy the model equations. The quantity "equation error after correction" is expressed in Pascals, hence the above values are negligible small.

You can meet almost all the difficulties of parameter estimation when evaluating vapor-liquid equilibria data (implicit functional relations among several variables, corrupted by measurement errors that are likely to be correlated (see, e.g., ref. 53).

REFERENCES

1 M.G. Kendall and A. Stuart, The Advanced Theory of Statistics. vol. 2, Inference and Relationship, 3rd ed., Griffith, London, 1973.

2 N.R. Draper and H. Smith, Applied Regression Analysis. 2nd ed., John Wiley, 1981.

3 R.R. Hocking, Developments in linear regression methodology: 1959-1982, Technometrics, 25 (1983) 219-230.
4 Y. Bard, Nonlinear Parameter Estimation. Academic Press, New York, 1974.
5 D.M. Himmelblau, Process Analysis by Statistical Methods, John Wiley, New York, 1970.
6 F.S. Wood, The use of individual effects and residuals in fitting equations to data. Technometrics, 15 (1973) 667-696.
7 J.Durbin and G.S. Wattson, Testing for serial correlations in least squares regression, I., Biometrika, 37 (1950) 409-428.
8 K. Schwetlick, Kinetische Methoden zur Untersuchung von Reaktionmechanismen. VEB Deutscher Verlag der Wissenschaften, Berlin, 1974.
9 J.N. Brönsted and W.F.K. Wynne-Jones, Trans. Faraday Soc. 25 (1929) 59.
10 Y, Bard, Comparison of gradient methods for the solution of nonlinear parameter estimation problems, SIAM J. Numer. Anal. 7 (1970) 157-186.
11 D.M. Himmelblau, Applied Nonlinear Programming, McGraw-Hill, New York, 1972.
12 J. Garcia-Peña, S.P. Azen and R.N. Bergman, On a modification of Marquardt's compromise: Rationale and applications. Appl. Math. Computing, 12 (198 1-17.
13 L. Nazareth, Some recent approaches to solving large residual nonlinear least squares problems. SIAM Rev., 22 (1980) 1-11.
14 K. Levenberg, A method for the solution of certain nonlinear problems in least squares, Quart. Appl. Math., 2 (1944) 164-168.
15 D.W. Marquardt, An algorithm for least squares estimation of non-linear parameters. SIAM J. Appl. Math., 11 (1963) 431-441.
16 R.A. Alberty and F. Daniels, Physical Chemistry, 5th ed., John Wiley, New York,1980.
17 G.F. Froment and K.B. Bischoff, Chemical Reactor Analysis and Design, John Wiley, New York, 1979.
18 W.B.S. Newling and C.N. Hinshelwood, The kinetics of the acid and alkaline hydrolysis of esters, J. Chem. Soc, 23 (1936) 1357-1364.
19 M.J.C. Crabbe, An enzyme kinetics program for desk-top computers, Comput. Biol. Med. 4 (1982) 263-283.
20 D. Garfinkel and K.A. Fegley, Fitting physiological models to data, Am. J. Physiol. 246 (1984) R641-R650.
21 D.M. Bates and D.C. Watts, Relative curvature measures of nonlinearity, J. R. Statist. Soc., Ser. B 42 (1980) 1-25.
22 D.A. Ratkowsky, Nonlinear Regression Modeling, Marcel Dekker, New Yorrk 1983.
23 A.E. Hoerl and R.W. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, Technometrics, 12 (1970) 55-67.
24 R.I. Jennrich and P.F. Sampson, Application of stepwise regression to nonlinear estimation. Technometrics, 10 (1968) 63-72.
25 S.Vajda, P. Valkó and T. Turányi, Principal component analysis of kinetic models, Int. J. Chem. Kinet. 17 (1985) 55-81.
26 S. Vajda and T. Turányi, Principal component analysis for reducing the Edelson-Field-Noyes model of Belousov-Zhabotinsky reaction, J. Phys. Chem. 90 (1986) 1664.
27 G.E.P. Box and N.R. Draper, The Bayesian estimation of common parameters from several responses. Biometrika, 52 (1965) 355-365.
28 D.D. McLean, D.J. Pritchard, D.W. Bacon and J. Downie, Singularities in multiresponse modelling, Technometrics, 21 (1979) 291-298.
29 G.P.E. Box, W.G. Hunter, J.F. MacGregor and J. Erjavec, Some problems associated with the analysis of multiresponse data, Technometrics, 15 (1973) 33-51.
30 D.M Bates and D.G. Watts, A generalized Gauss-Newton procedure for multi-response parameter estimation, SIAM J. Sci. and Stat. Comp., 8 (1987) 49-55.
31 D.L. Ripps, Adjustment of experimental data. Chem. Engng. Prog. Symp. Ser. 61 (1965) 8-13.

32 G.A. Almásy and T. Sztanó, Checking and correction of measurements on the basis of linear system models. Probl. Control. Inf. Theory, 4 (1975) 59-69.
33 W.D. Deming, Statistical Adjustment of Data, John Wiley, New York, 1943.
34 H.I. Britt and R.H. Luecke, The estimation of parameters in nonlinear implicit models. Technometrics, 15 (1973) 233-241.
35 P.M. Reilly and H. Patino-Leal, A Bayesian study of the error-in-variables model. Technometrics, 23 (1981) 221-227.
36 H. Patino-Leal and P.M. Reilly, Statistical estimation of parameters in vapor-liquid equilibrium. AIChE J. 28 (1982) 580-587.
37 P. Valkó and S. Vajda, An extended Marquardt-type procedure for fitting error-in-variables models. Comput. Chem. Engng. 11 (1987) 37-43.
38 J.H. Wilkinson and C. Reinsch, Handbook for Automatic Computation, Vol. II. Linear Algebra, Springer Verlag, New York, 1971.
39 G.E. Forsythe, Generation and use of orthogonal polynomials for data-fitting with a digital computer. J. SIAM, 5 (1957) 74-88.
40 D.B. Marsland, Data fitting by orthogonal polynomials, in. CACHE Thermodynamics (ed. R.V. Jelinek), Sterling Swift, Manchaca TX, 1971.
41 N.B. Vargaftik, Handbook of Thermophysical Data of Gases and Liquids, Nauka, Moscow, 1972 (in Russian)
42 J.E. Gentile, W.J. Kennedy and V.A. Sposito, On least absolute values estimation, Comm. Statistics, A6, (1977) 838-845.
43 J.C. Kiefer, Design of experiments, in. Collected Papers III, Springer, New York, 1985.
44 V.V. Nalimov and N.A. Chernova, Statistical methods of design of extremal experiments, Nauka, Moscow, 1965 (in Russian)
45 G.K. Krug, Yu.A. Sosulin and V.A. Fatuev, Design of experiments in identification and extrapolation, Nauka, Moscow, 1977.
46 B. Kanyár, Parameter sensitivity analysis for designing of experiments in kinetics, Acta Biochim. Biophys. Acad. Sci. Hung. 12 (1978) 24-28.
47 D.Z. D'Argenio, Optimal sampling times for pharmacokinetic experiments, J. Pharmacokinet. Biopharm. 9 (1981) 739-755.
48 J.J DiStefano III, Optimized blood sampling protocols and sequential design of kinetic experiments, Am. J. Physiol. 240 (1981) R259-R265.
49 I.J. Leontaritis and S.A. Billings, Model selection and validation methods for non-linear systems, Int. J. Contr. 45 (1987) 311-341.
50 E.M. Landau and J.J. DiStefano III, Multiexponential, multicompartmental and noncompartmental modeling II, Data analysis and statistical considerations, Am. J. Physiol. 246 (1984) R665-R677.
51 H. Akaike, A new look at statistical model identification, IEEE Trans. Aut. Control, AC-19 (1974) 716-723.
52 R.C. Reid, J.M. Prausnitz and T.K. Sherwood, The Properties of Gases and Liquids, 3rd ed. McGraw-Hill, New York, 1977.
53 S. Kemény, J. Manczinger, S. Skold-Jorgensen and K. Tóth, Reduction of thermodynamic data by means of the multiresponse maximum likelihood principle, AIChE J. 28 (1982) 21-30.

Chapter 4

SIGNAL PROCESSING

Many experiments result in a sequence $\{ (x_i,y_i),\ i = 1, 2, \ldots, m \}$ of data pairs. As in the previous chapter, we assume that there exists a functional relationship $y = f(x)$ between the two variables, and hence refer to $(x_1, \ldots, x_m)$ and $(y_1, \ldots, y_m)$ as grid points and function values, respectively. The form of this function is, however, often unknown. In other cases it may be deduced from physical principles, but is too complex for meaningful parameter estimation, with many parameters of no particular interest. In both situations we wish to predict some properties of f directly from the observations (x_i,y_i). The most important quantities to estimate are as follows:

i) the function value $f(x)$ between two grid points (interpolation);

ii) the derivative $f'(x)$ (numerical differentiation); and

iii) the integral $\int_a^b f(x)dx$, where the limits a and b satisfy the inequalities $x_1 \le a < b \le x_m$ (numerical integration).

The important application of numerical differentiation is locating the extrema or inflection points of the curve. Finding the area under the curve involves numerical integration.

Since $f(x)$ is known only at the grid points, to solve these problems we must connect the data by some plausible interpolating function. Its form should be sufficiently general so as to be able to approximate large classes of functions, but simple enough to deal with. By far the most common among such functions are polynomials. If we use all data pairs simultaneously, the interpolation is called global. In many cases, however, local interpolation is a better choice, considering only $n < m$ grid points around the point x of interest. Local linear and quadratic interpolation (i.e., $n = 2$ and $n = 3$, respectively) are the most familiar procedures. When the interpolating function has been selected, numerical differentiation and integration are straightforward. For example, with local linear interpolation shown in Fig. 4.1, the estimate of the derivative is $(y_{i+1}-y_i)/(x_{i+1}-x_i)$ at all $x_i \le x \le x_{i+1}$, whereas $\int_{x_i}^{x_{i+1}} f(x)dx \approx (y_{i+1} + y_i)/(x_{i\ +\ 1}-x_i)/2$ by the well known trapezium rule.

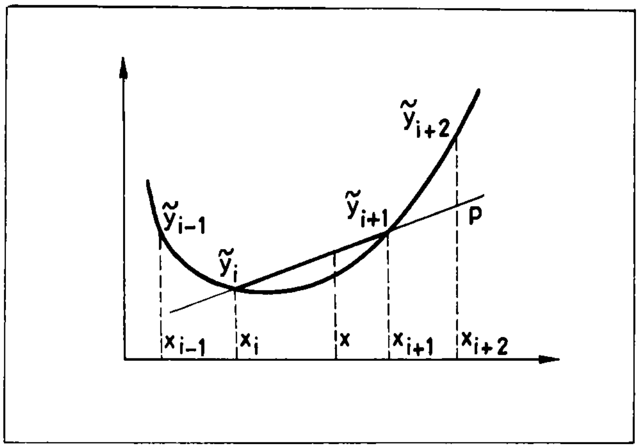


Fig. 4.1. Local linear interpolation

Interpolation assumes that the data are error-free. In many cases, however, we must assume that the observed sequence is $\{ (x_i,\tilde{y}_i),\ i = 1, 2, \ldots, m \}$, where $\tilde{y}_i = y_i + \epsilon_i$, and the errors ϵ_i are not negligible. Then it is more appropriate to look for a "smoothing" function that fits the data, but does not necessarily interpolate them. Since this function is expected to estimate the error-free function values y_i , the procedure is also called filtering. To choose a meaningful smoothing function one needs further assumptions on the error structure. In some cases the emphasis is on the magnitude of the error variance, (or the signal-to-noise ratio), assumed to be known. In other cases our assumptions rather concern the time behavior of the noise process, for instance we suppose that the noise varies much faster (or much slower) than the useful signal. Similarly to interpolation, smoothing may be global (e.g., least squares fit of a polynomial of degree $n < m - 1$ to all points) or local (e.g., fitting a quadratic to the 5 points nearest to x of interest). Differentiation of smoothing functions yields formulas less sensitive to measurement errors than the formulas of numerical differentiation derived from interpolating functions. Integration automatically removes some noise, and hence smoothing functions are rarely used in such applications.

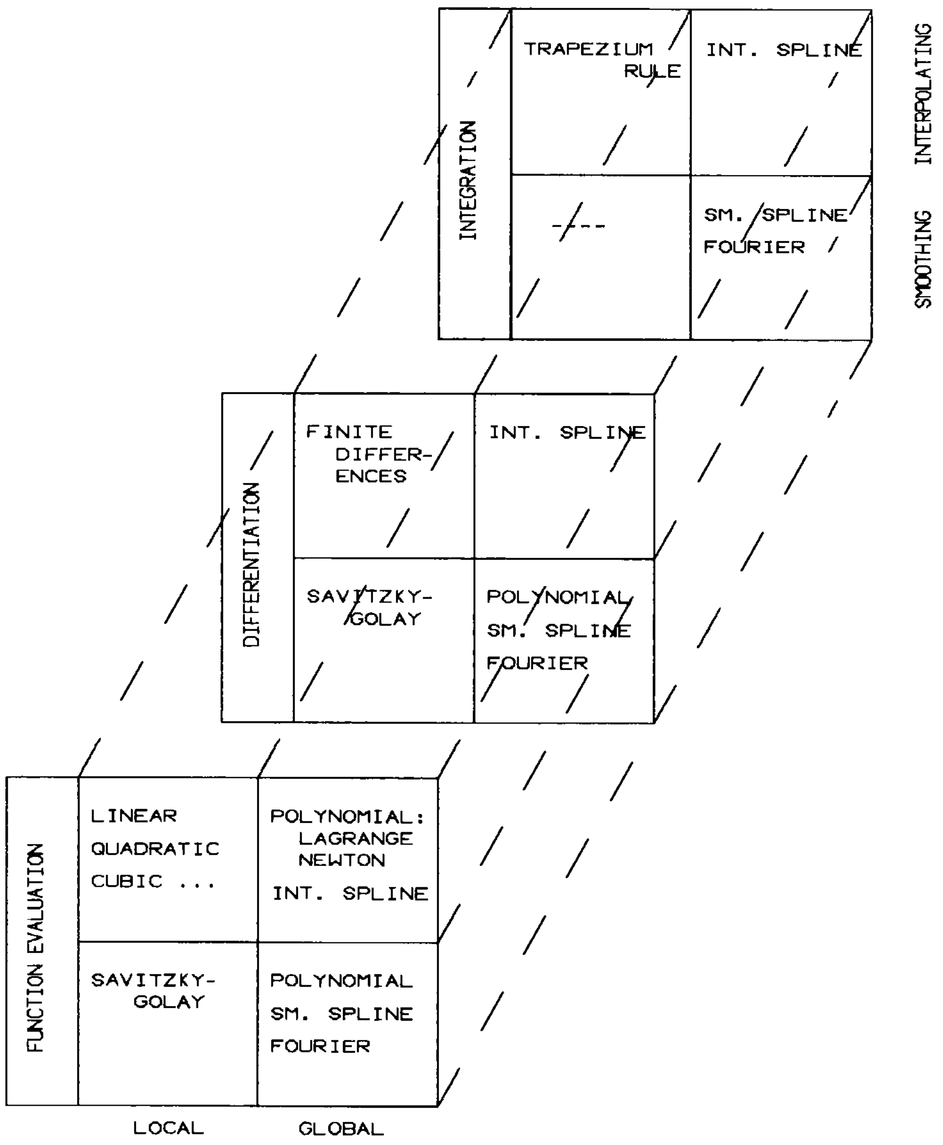


Fig. 4.2. Classification of signal processing methods

Each signal processing method discussed here involves some function which is either interpolating or smoothing, and is either local or global approximation of the data. This results in the two-way classification of the methods shown in Figure 4.2, where the quadrants of each card list methods of the same family for the particular application.

Signal processing may also involve parameter estimation methods (e.g., resolution of a spectral curve into the sum of Gaussian functions). Even in such cases, however, we may need non-parametric methods to approximate the position, height and half-width of the peaks, used as initial estimates in the parameter estimation procedure.

In this chapter we restrict consideration to non-recursive signal processing. A good introduction into recursive filtering can be found in the book of Bozic (ref. 1). Another interesting field not discussed here is to modify conventional analytical methods to produce signals, whose direct human interpretation is no longer necessary and possible (e.g., correlation chromatography). The interested reader may consult the review paper (ref. 2).

As shown in Fig. 4.2, we have several methods to solve any particular problem. The choice primarly depends on the sample size, and hence we introduce the following classification:

i) small samples (5-15 points);
ii) medium samples (10-100 points); and
iii) large samples (from 50 points).

Small samples are practically error-free in most cases (e.g., data in thermodynamical tables), but given over an irregular mesh. On the other hand, large samples almost invariably represent the "raw" output of a measuring device, or are obtained by sampling a continuous signal. In this class the grid points are equidistant that may simplify data processing. In medium samples we often have some assumption on the signal-to-noise ratio while in large samples the spectral properties of the noise process are more or less known.

There is an extensive mathematical literature devoted to interpolation, function approximation, numerical differentiation and integration (refs. 3-5), but many methods are not particularly useful for signal processing. For example, there is a large variety of efficient methods of integrating numerically a function that can be computed at any desired point. In signal processing, however, the data are a priori given, and the class of applicable methods is considerably restricted. In addition, many classical formulas are very simple so that discussing them we include only three modules.

More attention will be given to two families of very general methods. The first is based on the use of spline functions, and is going to replace many classical procedures for interpolation, smoothing, numerical differentiation and integration. The second family contains the Fourier transform spectral methods, and it has such an extensive list of potential applications that we can discuss only some of the most basic ones.

4.1 CLASSICAL METHODS

4.1.1 Interpolation

In global polynomial interpolation we fit the polynomial

$$p_{m-1}(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \dots + a_0 \qquad (4.1)$$

to the points $\{ (x_i,y_i),\ i = 1, 2, \dots, m \}$ by solving the set of linear equations

$$p_{m-1}(x_i) = y_i,\ i = 1, 2, \dots, m\ . \qquad (4.2)$$

If the grid points are distinct, the solution $a_0, a_1, \dots a_{m-1}$ of (4.2) is unique. The corresponding polynomial can be given in several explicit forms different from the canonical form (4.1). For instance, it can be computed as a linear combination

$$p_{m-1}(x) = \sum_{j=2}^{m} y_j L_j(x) \qquad (4.3)$$

of the Lagrange base polynomials defined by

$$L_j(x) = \frac{\prod_{i\neq j} (x-x_i)}{\prod_{i\neq j} (x_j-x_i)}\ . \qquad (4.4)$$

The classical Lagrange formula is not efficient numerically. One can derive more efficient, but otherwise naturally equivalent interpolation formulas by introducing finite differences. The first order divided differences are defined by

$$f(x_i,x_{i-1}) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}\ ,\ i=2,\dots m, \qquad (4.5)$$

where $f(x_i) = y_i$. Similarly the $(k+1)$-th order divided differences are defined recursively by

$$f(x_{k+1},\dots,x_2,x_1) = \frac{f(x_{k+1},\dots,x_2) - f(x_k,\dots,x_1)}{x_{k+1} - x_1} \qquad (4.6)$$

in terms of the k-th order divided differences. The simplest interpolating formulas based on divided differences go back to Newton, and involve polynomials of the form

$$p_{m-1}(x) = A_m + A_{m-1}(x-x_m) + A_{m-2}(x-x_m)(x-x_{m-1}) + \dots + A_1(x-x_m)\dots(x-x_1), \qquad (4.7)$$

where the coefficients A_k (not to be confused with the coefficients a_k in representation 4.1) are explicitly given by

$$A_m = f(x_m) \ , \quad A_{m-1} = f(x_m,x_{m-1}), \ \dots, \quad A_1 = f(x_m,x_{m-1},\dots x_1) \qquad (4.8)$$

in terms of the divided differences. To evaluate (4.7) it is useful to write it in a slightly modified form

$$p_{m-1}(x) = A_m + (x-x_m)(A_{m-1} + (x-x_{m-1})(A_{m-2} + \dots +(x-x_2)A_1)\dots) \qquad (4.9)$$

requiring only $m - 1$ multiplications.

Program module M60

```
6000 REM ****************************************************
6002 REM * NEWTON INTERPOLATION: COMPUTATION OF POLYNOMIAL *
6004 REM *      COEFFICIENTS AND INTERPOLATED VALUES       *
6006 REM ****************************************************
6008 REM INPUT:
6010 REM      M     NUMBER OF GRID POINTS
6012 REM     Z(M)   GRID POINTS
6014 REM     F(M)   FUNCTION VALUES AT GRID POINTS
6016 REM      X     POINT WHERE FUNCTION VALUE IS REQUIRED
6018 REM      FC    IDENTIFIER OF FIRST CALL
6020 REM            <>0    - FIRST INTERPOLATION
6022 REM             =0    - REPEATED INTERPOLATION
6024 REM OUTPUT:
6026 REM     F(M)   COEFFICIENTS OF THE INTERPOLATING POLYNOMIAL
6028 REM            F=F(M)+(X-Z(M))*(F(M-1)+(X-Z(M-1)*( ... F(1) ))
6030 REM      F     INTERPOLATED FUNCTION VALUE AT X
6032 IF FC=0 THEN 6048
6034 REM ---------- COEFFICIENTS
6036 FOR J=1 TO M-1
6038  FOR I=1 TO M-J
6040   F(I)=(F(I+1)-F(I))/(Z(I+J)-Z(I))
6042  NEXT I
6044 NEXT J
6046 REM ---------- INTERPOLATED VALUE
6048 F=F(1)
6050 FOR I=2 TO M :F=F*(X-Z(I))+F(I) :NEXT I
6052 FC=0 :RETURN
6054 REM ****************************************************
```

There are two operations performed in the module. First, it determines the coefficients A_k in the expression (4.7). Second, it calculates the polynomial at the specified X. Both operations are performed if the first call flag FC has a nonzero value on the input. The coefficients will be stored in the place of the function values, and the module sets the value FC = 0 . In a second (and in any subsequent) call with the same data but with a different X the coefficients are not recomputed.

Example 4.1.1 Determination of enthalpy by Newton interpolation

Each DATA line of the following main program gives a temperature T and a corresponding molar enthalpy value $H^o(T) - H^o(0)$ of SiF_4 in gaseous state (ref. 6). The units are K and kJ/mol, respectively. We find the molar enthalpy at the temperature T = 298.15 K.

```
100 REM ------------------------------------------------------------
102 REM EX. 4.1.1 NEWTON INTERPOLATION
104 REM MERGE M60
106 REM ---------- DATA
108 REM (NUMBER OF POINTS)
110 DATA 9
112 REM (T,K  H-H0,kJ/mol)
114 DATA  200, 8.722
116 DATA  300, 15.492
118 DATA  400, 23.367
120 DATA  500, 32.026
122 DATA  600, 41.225
124 DATA  700, 50.799
126 DATA  800, 60.637
128 DATA  900, 70.666
130 DATA 1000, 80.836
200 REM ---------- READ DATA
202 READ M
204 DIM Z(M),F(M)
206 FOR I=1 TO M
208  READ Z(I),F(I)
210 NEXT I
212 REM ---------- CALL INTERPOLATION MODULE
214 FC=1 :X=298.15
216 LPRINT "NEWTON INTERPOLATION, NUMBER OF POINTS:";M
218 GOSUB 6000
220 V$=STRING$(45,"-")
222 LPRINT V$
224 LPRINT USING "T=###.## K       H(T)-H(0)=##.### kJ/mol";X,F
226 LPRINT V$
228 STOP
```

The output of the program is as follows.

```
NEWTON INTERPOLATION, NUMBER OF POINTS: 9
---------------------------------------------
T=298.15 K        H(T)-H(0)=15.356 kJ/mol
---------------------------------------------
```

Global polynomial interpolation is restricted to small samples of fairly good data. If there are many grid points, the resulting higher order polynomial tends to oscillate wildly between the tabulated values as shown in Fig. 4.3.

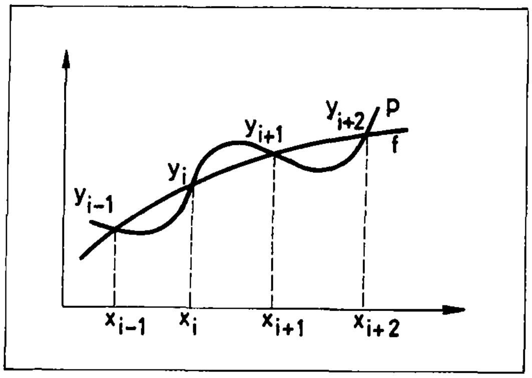


Fig. 4.3. An interpolating polynomial p oscillating around the "true" function f .

This oscillation may have no relation at all to the behavior of the "true" function. Therefore, we cannot recommend global interpolation except for small samples. In large samples interpolation is rarely needed. For medium size samples low order local interpolation considering 3 - 6 nearest neighbors of the point x of interest does the job in most cases. The most popular method is local cubic interpolation in the Aitken form programmed in the following module.

Program module M61

```
6100 REM ****************************************************
6102 REM *        LOCAL  CUBIC  INTERPOLATION               *
6104 REM ****************************************************
6106 REM INPUT:
6108 REM     M      NUMBER OF GRID POINTS
6110 REM     Z(M)   GRID POINTS
6112 REM     F(M)   FUNCTION VALUES AT GRID POINTS
6114 REM     X      GIVEN POINT
6116 REM OUTPUT:
6118 REM     F      INTERPOLATED FUNCTION VALUE
6120 FOR K=4 TO M-1
6122  IF Z(K-1)>X THEN 6128
6124 NEXT K
6126 K=M
6128 F3=F(K-3) :F2=F(K-2) :F1=F(K-1) :F=F(K)
6130 D3=Z(K-3)-X :D2=Z(K-2)-X :D1=Z(K-1)-X :D=Z(K)-X
6132 F2=(F3*D2-F2*D3)/(D2-D3)
6134 F1=(F3*D1-F1*D3)/(D1-D3)
6136 F =(F3*D -F *D3)/(D -D3)
6138 F1=(F2*D1-F1*D2)/(D1-D2)
6140 F =(F2*D -F *D2)/(D -D2)
6142 F =(F1*D -F *D1)/(D -D1)
6144 RETURN
6146 REM ****************************************************
```

The module selects the four nearest neighbors of X and evaluates the cubic interpolating polynomial.

Evaluation of a function outside the range of the grid points is called extrapolation. While extrapolation is based on the same ideas as interpolation, it is much more hazardous and should be avoided whenever possible.

4.1.2 Smoothing

Smoothing of noisy data is justified if the sampling frequency is sufficiently high, and hence the sample contains information for adjusting the observed function values by some kind of averaging. Then the smoothing function enables us to evaluate function values at both the grid points and between them as well. Global least squares polynomial fit is the most traditional method of smoothing for small and medium samples. Orthogonal polynomials and the program module M55 are useful to carry out such calculations.

For large samples global polynomial smoothing is either not sufficiently flexible (if the selected degree is low) or faces the same problems as in the case of interpolation (if the selected degree is high). Local smoothing usually gives better results. This involves least squares fit of polynomials of degree $n < 2k$ to the $2k + 1$ points (x_{-k},y_{-k}), ... (x_0,y_0), ... (x_k,y_k), where $2k$ is sufficiently smaller than m and x_o is a selected grid point of interest. The fit is particularly simple if the grid points are equidistant and the aim is to obtain a corrected value $\bar{y}_o$ at x_o. Then this estimate can be computed as the linear combination

$$\bar{y}_o = \frac{1}{F} \sum_{i=-k}^{k} c_i \tilde{y}_i \qquad (4.10)$$

of the considered $2k + 1$ function values. The coefficients c_i and the denominator F of the formulas (4.10) have been compiled by Savitzky and Golay (refs. 7-6) for several values of k and n.

Table 4.1 shows the Savitzky - Golay coefficients obtained by fitting a quadratic or cubic (these two yield identical coefficients for $\bar{y}_o$) to 5, 7, 9 and 11 points. The way to select the number of points is discussed in (ref. 9). The use of too many points is hazardous, since increasing the "extent of smoothing" such fits can distort also the useful signals. Therefore, the most popular formula involves only 5 points and the cubic

$$p_3(x) = a_3x^3 + a_2x^2 + a_1x + a_o \ . \qquad (4.11)$$

Derivation of the coefficients in (4.10) for this case is very simple. If h is the distance between the grid points denoted by $(-2h,\tilde{y}_{-2})$, $(-h,\tilde{y}_{-1})$, $(0,\tilde{y}_o)$, $(h,\tilde{y}_1)$ and $(2h,\tilde{y}_2)$, then the observation matrix $\mathbf{X}$ and observation vector $\tilde{\mathbf{Y}}$ introduced in Section 3.2 are given by

$$\mathbf{X} = \begin{bmatrix} -8h^3 & 4h^2 & -2h & 1 \\ -h^3 & h^2 & -h & 1 \\ 0 & 0 & 0 & 1 \\ h^3 & h^2 & h & 1 \\ 8h^3 & 4h^2 & 2h & 1 \end{bmatrix}, \quad \tilde{\mathbf{Y}} = \begin{bmatrix} \tilde{y}_{-2} \\ \tilde{y}_{-1} \\ \tilde{y}_o \\ \tilde{y}_1 \\ \tilde{y}_2 \end{bmatrix} . \qquad (4.12)$$

By (3.23) the least squares estimates of the coefficients in (4.11) are

$$\begin{aligned} a_o &= (-3\tilde{y}_{-2} + 12\tilde{y}_{-1} + 17\tilde{y}_o + 12\tilde{y}_1 - 3\tilde{y}_2) / 35 \\ a_1 &= (\tilde{y}_{-2} - 8\tilde{y}_{-1} + 8\tilde{y}_1 - \tilde{y}_2) / (12h) \\ a_2 &= (2\tilde{y}_{-2} - \tilde{y}_{-1} - 2\tilde{y}_o - \tilde{y}_1 + 2\tilde{y}_2) / (14h^2) \\ a_3 &= (-\tilde{y}_{-2} + 2\tilde{y}_{-1} - 2\tilde{y}_1 + \tilde{y}_2) / (12h^3) \ . \end{aligned} \qquad (4.13)$$

Since $p_3(0) = a_0$, the first expression of (4.13) is the Savitzky - Golay formula we were looking for.

Table 4.1
Coefficients for local quadratic or cubic smoothing by Savitzky and Golay

Number of points 2k + 1	Grid point weights, c_i											Denominator F
	x_{-5}	x_{-4}	x_{-3}	x_{-2}	x_{-1}	x_0	x_1	x_2	x_3	x_4	x_5	
5				-3	12	17	12	-3				35
7			-2	3	6	7	6	3	-2			21
9		-21	14	39	54	59	54	39	14	-21		231
11	-36	9	44	69	84	89	84	69	44	9	-36	429

In addition to their simplicity the Savitzky - Golay formulas are well suited to real time filtering. While more advanced methods such as smoothing by spline functions or by Fourier techniques assume the knowledge of the entire sample, to apply (4.10) we have to wait only for further k points. If k is once fixed the extent of smoothing can be increased by applying the procedure several times. If the sampling frequency is high, it may be sufficient to pick up each (2k+1)-th point and smooth only these ones using their nearest neighbors.

4.1.3 Differentiation

The derivatives of the unknown function are estimated by the derivatives of an interpolating or smoothing function fitted to the given set of data. Global interpolating polynomials wildly oscillating between grid points are not suitable for estimating the derivatives. As shown in Fig. 4.3, we may expect particularly bad estimates at the grid points where the polynomial crosses the "true" curve.

The familiar formulas of numerical differentiation are the derivatives of local interpolating polynomials. All such formulas give bad estimates if there are errors in the data. To illustrate this point consider the case of linear interpolation where the divided difference $(\tilde{y}_{i+1} - \tilde{y}_i)/(x_{i+1} - x_i)$ estimates the derivative at $x_i < x < x_{i+1}$. Let $D^2\{\tilde{y}_i\} = \sigma^2$ denote the variance of the measurement errors. We are usually more concerned with the relative errors $\sigma/\tilde{y}_i$, the inverse of the signal-to-noise ratio. If the errors are independent, then $D^2\{\tilde{y}_{i+1} - \tilde{y}_i\} = 2\sigma^2$ and hence the relative error in the slope is given by $(\sqrt{2}\ \sigma) / (\tilde{y}_{i+1} - \tilde{y}_i)$. Since usually $\left|\tilde{y}_{i+1} - \tilde{y}_i\right| << \left|\tilde{y}_i\right|$, the relative error of the slope may be much larger than the relative error in the data. Notice that this error is additional to the one introduced when approximating the function by a straight line.

It follows that the formulas of numerical differentiation do not apply to noisy sequence of data. Formulas based on the differentiation of local smoothing polynomials perform somewhat better. These are also of the form (4.10) if the derivatives are required only at the grid points. For example, the derivative of (4.11) is $p'_3(x) = 3a_3x^2 + 2a_2x + a_1$. Therefore, $p'_3(0) = a_1$, where a_1 is given by (4.13) as a linear combination of the function values. The coefficients of the formulas of smoothing differentiation based on the fit of a cubic (ref. 7) are shown in Table 4.2. To obtain correct numerical values you should multiply the denominator by the distance h as shown in (4.13).

Table 4.2
Coefficients for local cubic smoothing differentiation by Savitzky and Golay

Number of points	Grid point weights, c_i											Denominator F
2k + 1	x_{-5}	x_{-4}	x_{-3}	x_{-2}	x_{-1}	x_0	x_1	x_2	x_3	x_4	x_5	
5				1	-8	0	8	-1				12
7			22	-67	-58	0	58	67	-22			252
9		86	-142	-193	-126	0	126	193	142	-86		1188
11	300	-294	-532	-503	-296	0	296	503	532	294	-300	5148

Although numerical differentiation is considered as a routine step in signal processing, our discussion tries to emphasize that its results heavily depend on the choice of the interpolating or smoothing function. Different methods may lead to much deviating estimates. Nevertheless, from frequently sampled data we may be able to locate extrema or inflection points by numerical differentiation, since zero-crossing of the first or second derivatives is somewhat more reliable than their values.

The next module is based on the five point Savitzky - Golay formulas listed in Tables 4.1 and 4.2. It returns both the smoothed function values and the estimates of the derivative. The formulas are extended also to the four outermost points of the sample, where (4.10) does not directly apply.

Program module M62

```
6200 REM **************************************************
6202 REM *  5-POINT CUBIC SMOOTHING BY SAVITZKY AND GOLAY  *
6204 REM **************************************************
6206 REM INPUT:
6208 REM      N      NUMBER OF GRID POINTS
6210 REM     F(N)    FUNCTION VALUES AT GRID POINTS
6212 REM OUTPUT:
6214 REM    S(1,N)   S(0,I) SMOOTHED FUNCTION VALUES
6216 REM             S(1,I) SMOOTHED FIRST DERIVATIVES
6218 REM REMARK:     END POINTS ARE ALSO PROCESSED
6220  S(0,1)=(207*F(1)+12*F(2)-18*F(3)+12*F(4)-3*F(5))/210
6222  S(0,2)=(2*F(1)+27*F(2)+12*F(3)-8*F(4)+2*F(5))/35
6224 FOR I=3 TO N-2
6226  S(0,I)=(-3*F(I-2)+12*F(I-1)+17*F(I)+12*F(I+1)-3*F(I+2))/35
6228 NEXT I
6230  S(0,N)=(207*F(N)+12*F(N-1)-18*F(N-2)+12*F(N-3)-3*F(N-4))/210
6232  S(0,N-1)=(2*F(N)+27*F(N-1)+12*F(N-2) -8*F(N-3)+2*F(N-4))/35
6234  S(1,1)=(-125*F(1)+136*F(2)+48*F(3)-88*F(4)+29*F(5))/84
6236  S(1,2)=( -57*F(1)  -3*F(2)+36*F(3)+39*F(4)-15*F(5))/126
6238 FOR I=3 TO N-2
6240  S(1,I)=(F(I-2)-8*F(I-1)+8*F(I+1)-F(I+2))/12
6242 NEXT I
6244  S(1,N)=  (125*F(N)-136*F(N-1)-48*F(N-2)+88*F(N-3)-29*F(N-4))/84
6246  S(1,N-1)=( 57*F(N)  +3*F(N-1)-36*F(N-2)-39*F(N-3)+15*F(N-4))/126
6248 RETURN
6250 REM **************************************************
```

Note that the the grid points are not specified on the input. The derivative is numerically correct if the distance h between the grid points is 1. Otherwise, to obtain the derivative at the I-th point you must divide S(1,I) by the distance.

Example 4.1.3 Detection of end points in potentiometric titration by the method of Savitzky and Golay

In potentiometric titration a voltage is obtained from an electrode that is sensitive to an ionic species such as H_3O^+ , i.e., the pH of the solution in this case. We will consider the titration of the mixture of a strong acid (HCl) and a weak acid (CH_3COOH) with NaOH (ref. 10). As 2 ml volumes of the base are given to the acidic solution, the pH increases and when one of the acids is neutralized the pH changes very rapidly by a small addition of NaOH . We want to find these maximum points of the first derivative of the titration curve. In the following main program the DATA lines contain 32 data pairs, each consisting of the volume of the added NaOH in ml and the measured pH.

First we call the module to obtain the first derivative. Then this derivative is placed into the array F , and by repeatedly calling the module we obtain the estimate of the second derivative.

```
100 REM ------------------------------------------------------------
102 REM EX. 4.1.3  SMOOTHED DERIVATIVES BY SAVITZKY AND GOLAY
104 REM MERGE M62
106 REM ---------- DATA
108 REM  (NUMBER OF POINTS)
110 DATA 32
112 REM  (V,ml; pH)
114 DATA  2.4, 2.642,  2.6, 2.706,  2.8, 2.786,  3.0, 2.877
116 DATA  3.2, 2.986,  3.4, 3.126,  3.6, 3.295,  3.8, 3.480
118 DATA  4.0, 3.659,  4.2, 3.816,  4.4, 3.952,  4.6, 4.074
120 DATA  4.8, 4.183,  5.0, 4.285,  5.2, 4.384,  5.4, 4.480
122 DATA  5.6, 4.579,  5.8, 4.682,  6.0, 4.791,  6.2, 4.908
124 DATA  6.4, 5.045,  6.6, 5.211,  6.8, 5.444,  7.0, 5.859
126 DATA  7.2, 8.617,  7.4, 9.747,  7.6,10.134,  7.8,10.348
128 DATA  8.0,10.491,  8.2,10.604,  8.4,10.692,  8.6,10.766
200 REM ---------- READ DATA
202 READ N
204 DIM Z(N),F(N),S(1,N),A1(N),A2(N),A3(N),A4(N)
206 FOR I=1 TO N
208  READ Z(I),F(I) :A1(I)=F(I)
210 NEXT I
212 REM ---------- SMOOTH TWICE
214 DZ=Z(2)-Z(1)
216 GOSUB 6200
218 FOR I=1 TO N :A2(I)=S(0,I) :A3(I)=S(1,I)/DZ :F(I)=A3(I) :NEXT I
220 GOSUB 6200
222 FOR I=1 TO N :A4(I)=S(1,I)/DZ :NEXT I
224 REM
226 REM ---------- PRINT RESULTS
228 V$=STRING$(45,"-") :LPRINT V$
230 LPRINT "V,ml      pH    smoothed pH   first   second"
232 LPRINT "                              derivative    "
234 LPRINT V$
236 V1$=   " #.## ##.###      ##.### ###.### ###.###
238 FOR I=1 TO N
240 LPRINT USING V1$;Z(I),A1(I),A2(I),A3(I),A4(I)
242 NEXT I
244 LPRINT V$
246 STOP
```

The program gives the output as follows.

```
---------------------------------------------
V,ml     pH    smoothed pH   first   second
                              derivative
---------------------------------------------
 2.40   2.642       2.642    0.292    0.425
 2.60   2.706       2.707    0.357    0.299
 2.80   2.786       2.785    0.427    0.313
 3.00   2.877       2.876    0.492    0.462
 3.20   2.986       2.987    0.618    0.760
 3.40   3.126       3.127    0.779    0.758
 3.60   3.295       3.296    0.900    0.395
 3.80   3.480       3.479    0.926   -0.157
 4.00   3.659       3.658    0.846   -0.548
 4.20   3.816       3.815    0.729   -0.536
 4.40   3.952       3.953    0.642   -0.381
 4.60   4.074        4.074    0.575   -0.299
 4.80   4.183        4.183    0.523   -0.181
 5.00   4.285        4.285    0.501   -0.090
 5.20   4.384        4.383    0.485   -0.046
 5.40   4.480        4.480    0.485    0.051
 5.60   4.579        4.579    0.504    0.115
 5.80   4.682        4.682    0.528    0.126
 6.00   4.791        4.790    0.559    0.229
 6.20   4.908        4.908    0.626    0.427
 6.40   5.045        5.043    0.738    0.978
 6.60   5.211        5.204    0.934   -3.579
 6.80   5.444        5.269    0.672   21.565
 7.00   5.859        6.385    8.687   33.509
 7.20   8.617        8.201   11.006  -18.564
 7.40   9.747        9.774    3.186  -29.339
 7.60  10.134       10.174    1.222   -3.517
 7.80  10.348       10.353    0.833   -0.885
 8.00  10.491       10.494    0.621   -0.781
 8.20  10.604       10.603    0.496   -0.534
 8.40  10.692       10.692    0.399   -0.350
 8.60  10.766       10.766    0.343   -0.284
---------------------------------------------
```

As it will be discussed, while three maxima of the first derivative are observed, the second one is a consequence of the applied numerical method. Using the second derivative values in the last column, local inverse linear interpolation gives $V = 3.74$ ml and $V = 7.13$ ml for the two equivalence points. We will see later on how the false end point can be eliminated.

Exercise

o Compute the second derivative by divided finite difference approximation and compare the result with that of the Savitzky - Golay method.

4.1.4 Integration

For small samples we can integrate the global interpolating polynomial. For larger samples the trapezium rule

$$\int_{x_1}^{x_k} f(x)dx \approx \frac{1}{2h}\left[y_1 + 2\sum_{i=2}^{k-1} y_i + y_k \right] \qquad (4.14)$$

based on local linear interpolation, is usually sufficient. After the trapezium integration, textbooks on numerical analysis invariably proceed to the familiar Simpson rule, resulting in doubled weighting for each second point. Although the method has theoretical superiority over (4.14) if the distance h can be arbitrarily reduced, it is difficult to justify such weighting scheme with a priori given data.

Exercise

- Discuss the behavior of the Simpson formula on a "measured" data sequence similar to 1,-1,1,-1,...
- Show that integration amplifies the signal-to-noise ratio if the errors are independent.

4.2 SPLINE FUNCTIONS IN SIGNAL PROCESSING

Local cubic interpolation results in a function whose derivative is not necessarily continuous at the grid points. With a non-local adjustment of the coefficients we can, however, achieve global differentiability up to the second derivatives. Such functions, still being cubic polynomials between each pair of grid points, are called cubic splines and offer a "stiffer" interpolation than the strictly local approach.

4.2.1 Interpolating splines

We find the cubic spline interpolating the points $\{ (x_i,y_i),\ i = 1, 2, \ldots, n \}$. Let $p_i(d)$ denote the cubic polynomial over the interval $[x_i,x_{i+1}]$ of length $h_i = x_{i+1} - x_i$, where $d = x - x_i$. To define the n-1 cubics we need 4(n-1) coefficients. The available constraints are as follows:

(a) The cubics are interpolating ones, and hence

$$p_i(0) = y_i \ , \quad i=1,2,\ldots n-1; \qquad (4.15)$$
$$p_i(h_i) = y_{i+1} \ , \quad i=1,2,\ldots n-1. \qquad (4.16)$$

(b) The continuity of the first derivative implies

$$p'_{i-1}(h_{i-1}) = p'_i(0), \quad i=2,3,\ldots n-1, \qquad (4.17)$$

(c) whereas from the continuity of the second derivative we have

$$p''_{i-1}(h_{i-1}) = p''_i(0), \quad i=2,3,\ldots n-1. \qquad (4.18)$$

Thus we have 4n-6 equations, and need two further constraints to define the coefficients uniquely. In most cases these are chosen according to one of the following alternatives.

i) Assume that the second derivative vanishes at the end points x_1 and x_n, resulting in the equations $p''_1(0) = 0$ and $p''_{n-1}(h_{n-1}) = 0$. The derived function is called natural spline.

ii) The first derivative has arbitrarily fixed values at the end points, $p'_1(0) = y'_0$ and $p'_{n-1}(h_{n-1}) = y'_n$.

It can be verified that the set of linear equations given by the constraints has a unique solution both for cases i) and ii), if the grid points $x_1, x_2, \ldots, x_n$ are distinct (ref. 11).

To illustrate a special smoothness property of natural splines, define the quantity

$$S = \frac{1}{x_n - x_1} \int_{x_1}^{x_n} [f''(x)]^2 dx \ . \tag{4.19}$$

Obviously, $S = 0$ for a straight line. If S is small for a given function, it indicates that f does not wildly oscillate over the interval $[x_1, x_n]$ of interest. It can be shown that among all functions that are twice continuously differentiable and interpolate the given points, S takes its minimum value on the natural cubic interpolating spline (ref. 12).

It remains to calculate the coefficients that define the interpolating spline. One can obviously solve the $4(n-1)$ constraint equations directly, but there exists a much more efficient algorithm. Let m_i and m_{i+1} denote the second derivatives of the cubic p_i at $d = 0$ and $d = h_i$, respectively. The derivative is a linear function, given by

$$p''_i(d) = \frac{h_i - d}{h_i} m_i + \frac{d}{h_i} m_{i+1} \ . \tag{4.20}$$

Integrating the function (4.20) twice and determining the two integration constants from the constraints (4.15) and (4.16), the cubic polynomial p_i is obtained in the form

$$p_i(d) = \frac{m_i}{6h_i}(h_i - d)^3 + \frac{m_{i+1}}{6h_i}d^3 + \left[\frac{y_{i+1}}{h_i} - \frac{h_i m_{i+1}}{6}\right]d + \left[\frac{y_i}{h_i} - \frac{h_i m_i}{6}\right](h_i - d) \ ,$$

$$i = 1,2,\ldots n-1 \ . \tag{4.21}$$

Now we differentiate (4.21) once and exploit the constraints (4.17). The resulting equations are

$$h_{i-1}m_{i-1} + 2(h_i + h_{i+1})m_i + h_i m_{i+1} = 6\left\{\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right\} ,$$

$$i = 2, 3, \ldots, n-1 . \qquad (4.22)$$

If we select the end conditions i), the further equations are

$$m_1 = 0 \quad \text{and} \quad m_n = 0 . \qquad (4.23)$$

Adopting assumption ii) instead, equations (4.23) are replaced by

$$2h_1m_1 \quad + \quad h_1m_2 = 6\left[\frac{y_2 - y_1}{h_1} - y'_0\right] \quad \text{and}$$

$$h_{n-1}m_{n-1} + 2h_{n-1}m_n = 6\left\{y'_n - \frac{y_n - y_{n-1}}{h_{n-1}}\right\} . \qquad (4.24)$$

In both cases the resulting system of equations is tridiagonal and can be easily solved by the special method presented in Section 1.5. Once the m_i values are known, equations (4.21) can be easily rearranged to obtain the polynomial coefficients. Computing the function value and the derivatives at any point x is then straightforward, whereas integration is facilitated by the relationship

$$\int_a^b p_3(x)dx = \frac{1}{2}(b - a)[p_3(a) - p_3(b)] - \frac{1}{24}(b - a)^3[p''_3(a) + p''_3(b)] , \qquad (4.25)$$

valid for any cubic.

Program module M63

```
6300 REM ####################################################
6302 REM #   DETERMINATION OF INTERPOLATING CUBIC SPLINE    #
6304 REM ####################################################
6306 REM INPUT:
6308 REM     N      NUMBER OF GRID POINTS
6310 REM     Z(N)   GRID POINTS (KNOTS)
6312 REM     F(N)   FUNCTION VALUES
6314 REM     EC     IDENTIFIER FOR SELECTING END CONDITIONS
6316 REM              0   - NATURAL SPLINE
6318 REM            NOT 0 - FIRST DERIVATIVES GIVEN AT END POINTS
6320 REM                    THIS CASE REQUIRES FURTHER INPUTS:
6322 REM     D1     FIRST DERIVATIVE AT X=Z(1)
6324 REM     DN     FIRST DERIVATIVE AT X=Z(N)
6326 REM OUTPUT:
6328 REM   S(4,N)   S(J,I) COEFFICIENTS OF THE J-TH DEGREE TERMS (J=0...3)
6330 REM            S(4,I) INTEGRAL VALUE FROM Z(1) TO Z(I)
```

```
6332 FOR I=1 TO N-1
6334  S(0,I)=Z(I+1)-Z(I) :S(1,I)=(F(I+1)-F(I))/S(0,I)
6336 NEXT I
6338 S(0,N)=0
6340 S(3,1)=2*S(0,1)
6342 IF EC<>0 THEN S(2,1)=3*(S(1,1)-D1)
6344 FOR I=2 TO N
6346  S=S(0,I-1)/S(3,I-1)
6348  IF EC=0 AND I=2 THEN S=0
6350  S(3,I)=2*(S(0,I)+S(0,I-1))-S*S(0,I-1)
6352  IF I<N THEN S(2,I)=3*(S(1,I)-S(1,I-1))-S*S(2,I-1)
6354 NEXT I
6356 IF EC<>0 THEN S(2,N)=3*(DN-S(1,N-1))-S*S(2,N-1)
6358 IF EC=0 THEN S(2,N)=0
6360 S(2,N)=S(2,N)/S(3,N) :S(3,N)=0
6362 FOR I=N-1 TO 1 STEP -1
6364  S(2,I)=(S(2,I)-S(0,I)*S(2,I+1))/S(3,I)
6366  IF EC=0 AND I=1 THEN S(2,1)=0
6368  S(1,I)=S(1,I)-S(0,I)*(2*S(2,I)+S(2,I+1))/3
6370  S(3,I)=(S(2,I+1)-S(2,I))/S(0,I)/3
6372 NEXT I
6374 S(1,N)=S(1,N-1)+S(0,N-1)*(S(2,N-1)+S(2,N))
6376 S(4,1)=0
6378 FOR I=2 TO N
6380  S=S(4,I-1)+S(0,I-1)*(F(I)+F(I-1))/2
6382  S(4,I)=S-S(0,I-1)^3*(S(2,I)+S(2,I-1))/12
6384  S(0,I-1)=F(I-1)
6386 NEXT I
6388 S(0,N)=F(N)
6390 RETURN
6392 REM ####################################################
```

With the end condition flag EC = 0 on the input, the module determines the natural cubic spline function interpolating the function values stored in vector F. Otherwise, D1 and DN are additional input parameters specifying the first derivatives at the first and last points, respectively. Results are returned in the array S such that S(J,I), J = 0, 1, 2, 3 contain the 4 coefficients of the cubic defined on the I-th segment between Z(I) and Z(I+1). Note that the i-th cubic is given in a coordinate system centered at Z(I). The module also calculates the area under the curve from the first point Z(1) to each grid point Z(I), and returns it in S(4,I). The entries in the array S can be directly used in applications, but we provide a further module to facilitate this step.

Program module M64

```
6400 REM ****************************************************
6402 REM *    FUNCTION VALUE, DERIVATIVES AND DEFINITE      *
6404 REM *   INTEGRAL OF A CUBIC SPLINE AT A GIVEN POINT    *
6406 REM ****************************************************
6408 REM INPUT:
6410 REM     N      NUMBER OF KNOTS
6412 REM    Z(N)    GRID POINTS (KNOTS)
6414 REM   S(4,N)   SPLINE COEFFICIENTS (FROM M63 OR M65)
6416 REM     X      GIVEN POINT
6418 REM OUTPUT:
6420 REM    S0      SPLINE FUNCTION VALUE AT X
6422 REM    S1             FIRST DERIVATIVE
6424 REM    S2             SECOND DERIVATIVE
6426 REM    S3             THIRD DERIVATIVE
6428 REM    S4             DEFINITE INTEGRAL FROM Z(1) TO X
6430 FOR I=N TO 1 STEP -1
6432  IF X<Z(I) THEN 6442
6434  S=X-Z(I) :S0=((S(3,I)*S+S(2,I))*S+S(1,I))*S+S(0,I)
6436  S1=(3*S(3,I)*S+2*S(2,I))*S+S(1,I) :S2=6*S(3,I)*S+2*S(2,I)
6438  S3=6*S(3,I) :S4=S(4,I)+S*(S0+S(0,I))/2-S*S*S*(S2+S(2,I))/12
6440  GOTO 6448
6442 NEXT I
6444 S=X-Z(1) :S0=(S(2,1)*S+S(1,1))*S+S(0,1) :S1=2*S(2,1)*S+S(1,1)
6446 S2=S(2,1) :S3=0 :S4=S*(S0+S(0,1))/2-S*S*S*S2/6
6448 RETURN
6450 REM ****************************************************
```

In addition to the grid points stored in the vector Z and the array S of coefficients created by the module M63 (or by M65), the input to this module is a specified point X . This module returns the function value in S0 , and the values of the first, second and third derivatives in S1, S2 and S3 , respectively. The area under the curve from Z(1) to the specified X is returned in S4. If X is outside the range of the grid points, the extrapolation involves a straight line tangential to the function at the corresponding end point.

Example 4.2.1 Enthalpy and heat capacity by spline interpolation

We use the modules M63 and M64 to solve the interpolation problem discussed in Example 4.1.1. In addition to the enthalpy at T = 298.15 K , the heat capacity (i.e., the first derivative) is also computed at this point. The main program and the results are as follows.

```
100 REM ------------------------------------------------------------
102 REM EX. 4.2.1 SPLINE INTERPOLATION
104 REM MERGE M63, M64
106 REM ---------- DATA
108 REM (NUMBER OF POINTS)
110 DATA 9
112 REM (T,K  H-H0,kJ/mol)
114 DATA  200, 8.722
116 DATA  300, 15.492
118 DATA  400, 23.367
120 DATA  500, 32.026
122 DATA  600, 41.225
124 DATA  700, 50.799
126 DATA  800, 60.637
128 DATA  900, 70.666
130 DATA 1000, 80.836
200 REM ---------- READ DATA
202 READ N
204 DIM Z(N),F(N),S(4,N)
206 FOR I=1 TO N
208  READ Z(I),F(I)
210 NEXT I
212 LPRINT "NATURAL CUBIC SPLINE INTERPOLATION"
214 REM ---------- CALL SPLINE DETERMINATION AND EVALUATION MODULES
216 EC=0     :GOSUB 6300
218 X=298.15 :GOSUB 6400
220 V$=STRING$(45,"-")
222 LPRINT V$
224 LPRINT USING "T=###.## K        H(T)-H(0)=##.### kJ/mol";X,S0
226 LPRINT USING "                          Cp=##.###  J/(mol K)";S1*1000
228 LPRINT V$
230 STOP
```

```
NATURAL CUBIC SPLINE INTERPOLATION
---------------------------------------------
T=298.15 K        H(T)-H(0)=15.358 kJ/mol
                         Cp=72.398  J/(mol K)
---------------------------------------------
```

4.2.2 Smoothing splines

If the data $\{ (x_i,\tilde{y}_i),\ i = 1, 2, \ldots, n \}$ are noisy, it is not reasonable to force a function f to pass through the measured values. Suppose we have an estimate d_i of the standard error of the i-th function value. Then a suitable measure of the distance of any smoothing function f from the measurement points is the sum of squares

$$F^2 = \sum_{i=1}^{n} \left[\frac{f(x_i) - \tilde{y}_i}{d_i} \right]^2 . \qquad (4.26)$$

If the squared distance F^2 is greater than the number of points n, then the function f is too far from the measurement points. Therefore, we restrict consideration to functions satisfying the constraint

$$F^2 \le n \ , \qquad (4.27)$$

i.e., we attempt to fit the data within the range of measurement errors. In addition, we are interested in functions that are at least twice continuously differentiable. One can draw several such curves satisfying (4.27), and the "smoothest" of them is the one minimizing the integral (4.19). It can be shown that the solution of this constrained minimization problem is a natural cubic spline (ref. 12). We call it smoothing spline.

The smoothing spline converges to the interpolating spline if $d_i \to 0$. While this function is unique for reasonable values of d_i, with too large standard errors an entire family of straight lines satisfy (4.27) thus yielding zero value for S in (4.19). This family includes the straight line fitted by weighted linear regression, and hence in this case it is not justified to seek the solution in spline form.

If the solution is unique, it can be obtained by the method of Lagrange multipliers (ref. 13). We look for the minimum of the Lagrange function

$$L = \int_{x_1}^{x_n} [f''(x)]^2 dx + \frac{1}{p}(F^2 - n) \qquad (4.28)$$

where p is the reciprocal Lagrange multiplier. For any fixed value of p the 4(n-1) equations for the 4(n-1) coefficients of the natural cubic spline function minimizing (4.28) can be obtained from the Euler - Lagrange relations (ref. 13). Introducing the second derivatives m_i as in the previous section, the system can be reduced to simultaneous linear equations with a coefficient matrix of band structure. The matrix has 5 nonvanishing diagonals. Therefore, the spline is relatively easy to determine for a given value of p , and it yields the actual squared distance (4.26) denoted by $F^2(p)$.

The additional problem we face is determining the optimal value for p. It is important to note that the squared distance $F^2(p)$ increases with the value of p . Therefore, the algorithm can be viewed as starting with an interpolating spline obtained at p = 0 , and then "streching" this function by gradually increasing the value of p until (4.27) holds. To find this particular p we solve the nonlinear equation

$$F(p) - n^{1/2} = 0 \ . \qquad (4.29)$$

A Newton method can be used, since the derivative F'(p) is relatively easy to compute. The following program module is based on the procedure proposed by Reinsch (ref. 13). The only essential deviation from the original algorithm is in the formula for the correction Δp :

$$\Delta p = - \frac{F(p)-n^{1/2}}{F'(p)} \left[\frac{n}{F^2(p)} \right]^{1/2} , \tag{4.30}$$

where the additional square root convergence promotion factor can somewhat improve the convergence at the beginning of the iteration where $F^2(p)$ satisfies the inequality $0 < F^2(p) << n$.

Program module M65

```
6500 REM ####################################################
6502 REM #     DETERMINATION OF SMOOTHING CUBIC SPLINE      #
6504 REM #           METHOD OF  C. H. REINSCH               #
6506 REM ####################################################
6508 REM INPUT:
6510 REM     N      NUMBER OF GRID POINTS
6512 REM    Z(N)    GRID POINTS (KNOTS)
6514 REM    F(N)    FUNCTION VALUES
6516 REM    D(N)    STANDARD ERRORS AT GRID POINTS
6518 REM    IM      MAXIMUM NUMBER OF ITERATIONS
6520 REM OUTPUT:
6522 REM    ER      STATUS FLAG
6524 REM              0   SUCCESSFUL COMPLETITION
6526 REM              1   SOLUTION IS A STRAIGHT LINE
6528 REM              2   NUMBER OF ITERATIONS IS INSUFFICIENT
6530 REM   S(4,N)   S(J,I) COEFFICIENTS OF THE J-TH DEGREE TERMS (J=0...3)
6532 REM            S(4,I) INTEGRAL VALUES FROM Z(1) TO Z(I)
6534 REM AUXILIARY ARRAY:
6536 REM   R(6,N)
6538 R(5,0)=0 :R(5,1)=0 :P=0
6540 R(0,0)=0 :R(0,1)=0 :R(0,N)=0 :R(2,N)=0
6542 H=Z(2)-Z(1) :F=(F(2)-F(1))/H
6544 FOR I=2 TO N-1
6546  G=H :H=Z(I+1)-Z(I)
6548  E=F :F=(F(I+1)-F(I))/H
6550  S(0,I)=F-E :R(3,I)=2*(G+H)/3 :R(4,I)=H/3
6552  R(2,I)=D(I-1)/G :R(0,I)=D(I+1)/H
6554  R(1,I)=-D(I)/G-D(I)/H
6556 NEXT I
6558 FOR I=2 TO N-1
6560  S(1,I)=R(0,I)*R(0,I)+R(1,I)*R(1,I)+R(2,I)*R(2,I)
6562  S(2,I)=R(0,I)*R(1,I+1)+R(1,I)*R(2,I+1)
6564  IF I<N-1 THEN S(3,I)=R(0,I)*R(2,I+2) ELSE S(3,I)=0
6566 NEXT I
```

```
6568 REM ---------- START OF ITERATION
6570 FOR IT=1 TO IM
6572  FOR I=2 TO N-1
6574   R(1,I-1)=F*R(0,I-1) :R(2,I-2)=G*R(0,I-2)
6576   R(0,I)=1/(P*S(1,I)+R(3,I)-F*R(1,I-1)-G*R(2,I-2))
6578   R(5,I)=S(0,I)-R(1,I-1)*R(5,I-1)-R(2,I-2)*R(5,I-2)
6580   F=P*S(2,I)+R(4,I)-H*R(1,I-1) :G=H :H=S(3,I)*P
6582  NEXT I
6584  FOR I=N-1 TO 2 STEP -1
6586   R(5,I)=R(0,I)*R(5,I)-R(1,I)*R(5,I+1)
6588   IF I<N-1 THEN R(5,I)=R(5,I)-R(2,I)*R(5,I+2)
6590  NEXT I
6592  E=0 :H=0
6594  FOR I=1 TO N-1
6596   G=H :H=(R(5,I+1)-R(5,I))/(Z(I+1)-Z(I))
6598   R(6,I)=(H-G)*D(I)*D(I) :E=E+R(6,I)*(H-G)
6600  NEXT I
6602  G=-H*D(N)*D(N) :R(6,N)=G :E=E-G*H :F2=E*P*P
6604  IF ABS(P*(Z(N)-Z(1))) > 1E+08  AND  F2<N THEN ER=1 :GOTO 6630
6606  IF ABS(F2-N)<=N/10000 THEN ER=0 :GOTO 6630
6608  F=0 :H=(R(6,2)-R(6,1))/(Z(2)-Z(1))
6610  FOR I=2 TO N-1
6612   G=H :H=(R(6,I+1)-R(6,I))/(Z(I+1)-Z(I))
6614   G=H-G-R(1,I-1)*R(0,I-1)-R(2,I-2)*R(0,I-2)
6616   F=F+G*R(0,I)*G :R(0,I)=G
6618  NEXT I
6620  H=E-P*F : IF H=0 THEN ER=0 :GOTO 6630
6622  E=(N-F2)/((SQR(N/E)+P)*H)
6624  IF IT=1 THEN P=P+E ELSE P=P+E*SQR(N/F2)
6626 NEXT IT
6628 ER=2
6630 REM ---------- SPLINE COEFFICIENTS INTO S
6632 S(0,N)=F(N)-P*R(6,N) :S(2,N)=0
6634 FOR I=N-1 TO 1 STEP -1
6636  H=Z(I+1)-Z(I)
6638  S(2,I)=R(5,I)
6640  S(0,I)=F(I)-P*R(6,I)
6642  S(1,I)=(S(0,I+1)-S(0,I))/H-H*(2*S(2,I)+S(2,I+1))/3
6644  S(3,I)=(S(2,I+1)-S(2,I))/(3*H)
6646 NEXT I
6648 S(1,N)=S(1,N-1)+(Z(N)-Z(N-1))*(S(2,N-1)+S(2,N))
6650 S(3,N)=0 :S(4,1)=0
6652 FOR I=2 TO N
6654  H=Z(I)-Z(I-1)
6656  S(4,I)=S(4,I-1)+H*(S(0,I)+S(0,I-1))/2-H*H*H*(S(2,I)+S(2,I-1))/12
6658 NEXT I
6660 RETURN
6662 REM ********************************************************
```

The input is similar to that of the module M63. No end condition flag is used since only natural splines can be fitted. On the other hand, you should specify the maximum number IM of iterations. The module returns the array S defined in the description of the module M63, and hence the function value, the derivatives and the integral at a specified X can be computed by calling the module M64. The important additional inputs needed by the module M65 are the standard errors given in the vector D . With all D(I) = 0 , the module returns the interpolating natural cubic spline, whereas too large D(I) values may result in a straight line idicated by the error flag ER = 1 .

Example 4.2.2 Detection of end points in potentiometric titration by spline smoothing

The problem of Example 4.1.3 is revisited here. We determine the smoothing spline function and its derivatives assuming identical standard errors d_i = 0.25 in the measured pH.

```
100 REM ------------------------------------------------------------
102 REM EX. 4.2.2  SMOOTHING BY SPLINE
104 REM MERGE M65
106 REM ---------- DATA
108 REM  (NUMBER OF POINTS)
110 DATA 32
112 REM  (V,ml;  pH)
114 DATA  2.4, 2.642,  2.6, 2.706,  2.8, 2.786,  3.0, 2.877
116 DATA  3.2, 2.986,  3.4, 3.126,  3.6, 3.295,  3.8, 3.480
118 DATA  4.0, 3.659,  4.2, 3.816,  4.4, 3.952,  4.6, 4.074
120 DATA  4.8, 4.183,  5.0, 4.285,  5.2, 4.384,  5.4, 4.480
122 DATA  5.6, 4.579,  5.8, 4.682,  6.0, 4.791,  6.2, 4.908
124 DATA  6.4, 5.045,  6.6, 5.211,  6.8, 5.444,  7.0, 5.859
126 DATA  7.2, 8.617,  7.4, 9.747,  7.6,10.134,  7.8,10.348
128 DATA  8.0,10.491,  8.2,10.604,  8.4,10.692,  8.6,10.766
200 REM ---------- READ DATA
202 READ N
204 DIM Z(N),F(N),D(N),S(4,N),R(6,N)
206 FOR I=1 TO N
208  READ Z(I),F(I)
210 NEXT I
212 REM ---------- CONSTANT STANDARD ERROR
214 SD=.25
216 FOR I=1 TO N :D(I)=SD :NEXT I
218 REM ---------- CALL SMOOTHING SPLINE MODULE
220 IM=20 :GOSUB 6500
222 IF ER=1 THEN LPRINT "STRAIGHT LINE"
224 IF ER=2 THEN LPRINT "MAX NUMBER OF ITERATIONS IM IS EXCEEDED" :STOP
226 REM ---------- PRINT RESULTS
228 V$=STRING$(65,"-")
230 A$=" ##.##      ##.###      ##.###      ####.##      #####.##"
232 LPRINT USING "SMOOTHING SPLINE,  ST. ERR: ##.##";SD :LPRINT
234 LPRINT V$
236 LPRINT "V, ml      MEAS. pH    SMOOTHED pH    FIRST DER.   SECOND DER."
238 LPRINT V$
240 FOR I=1 TO N
242  LPRINT USING A$;Z(I),F(I),S(0,I),S(1,I),2*S(2,I)
244 NEXT I
246 LPRINT V$
248 STOP
```

Note that the coefficients S(2,I) are multiplied by 2 to obtain the second derivatives shown in the last column of the following output.

```
SMOOTHING SPLINE,  ST. ERR:  0.25

------------------------------------------------------------------
V, ml      MEAS. pH    SMOOTHED pH    FIRST DER.   SECOND DER.
------------------------------------------------------------------
  2.40       2.642         2.625         0.38          0.00
  2.60       2.706         2.703         0.40          0.16
  2.80       2.786         2.788         0.45          0.34
  3.00       2.877         2.885         0.53          0.51
  3.20       2.986         3.003         0.64          0.59
  3.40       3.126         3.143         0.76          0.52
  3.60       3.295         3.303         0.84          0.28
  3.80       3.480         3.474         0.86         -0.04
  4.00       3.659         3.644         0.83         -0.30
  4.20       3.816         3.802         0.76         -0.42
  4.40       3.952         3.945         0.67         -0.40
  4.60       4.074         4.072         0.60         -0.32
  4.80       4.183         4.186         0.54         -0.23
  5.00       4.285         4.291         0.51         -0.16
  5.20       4.384         4.389         0.47         -0.16
  5.40       4.480         4.481         0.44         -0.20
  5.60       4.579         4.564         0.39         -0.25
  5.80       4.682         4.639         0.35         -0.16
  6.00       4.791         4.710         0.37          0.34
  6.20       4.908         4.800         0.57          1.62
  6.40       5.045         4.961         1.12          3.92
  6.60       5.211         5.285         2.22          7.03
  6.80       5.444         5.885         3.86          9.43
  7.00       5.859         6.834         5.57          7.63
  7.20       8.617         8.027         5.99         -3.43
  7.40       9.747         9.120         4.76         -8.88
  7.60      10.134         9.898         3.03         -8.37
  7.80      10.348        10.355         1.64         -5.61
  8.00      10.491        10.588         0.78         -2.93
  8.20      10.604        10.697         0.37         -1.16
  8.40      10.692        10.754         0.23         -0.28
  8.60      10.766        10.796         0.20          0.00
------------------------------------------------------------------
```

Using inverse linear interpolation the two titration equivalence points are obtained as the zero-crossing points of the second derivative at $V = 3.78$ ml and $V = 7.14$ ml . On Fig. 4.4 the second derivative curve of the interpolating spline (SD = 0) and that of the smoothing spline (SD = 0.25) are shown. The false zero-crossing of the second derivative present at interpolation is eliminated by smoothing.

We note that another type of smoothing spline can be fitted by the traditional least squares method. In that case, however, the q subintervals on which the individual cubics are defined should be selected prior to the fit,

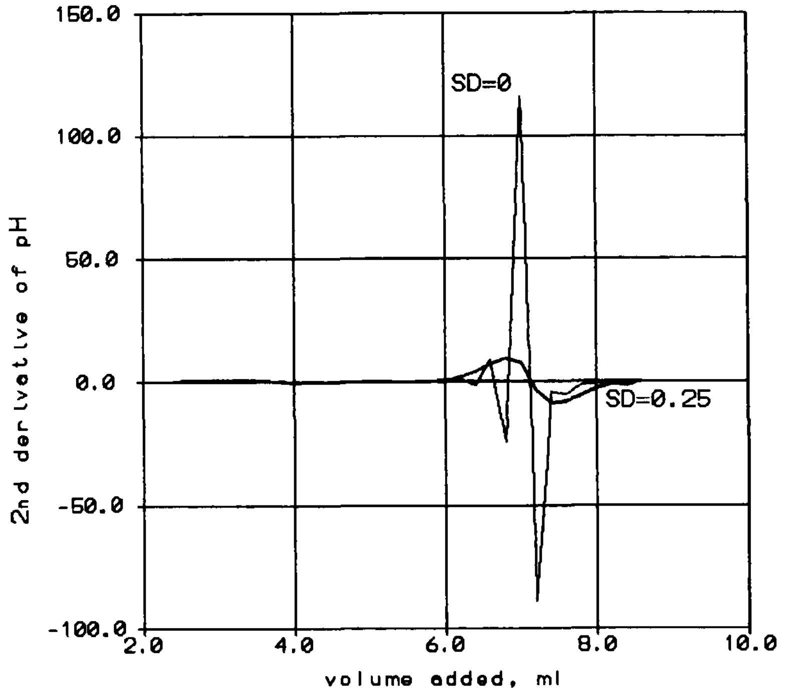


Fig. 4.4. Second derivative of smoothing (SD = 0.25) and interpolating (SD = 0) splines

where q << n . Then the squared distance F^2 between the smoothing function and the measurement points is minimized by multivariable linear regression. The extent of smoothing can be influenced only indirectly, changing the number and locations of the grid points.

4.3 FOURIER TRANSFORM SPECTRAL METHODS

Apart from some special drift processes that we will treat separately, the noise in the measurements is expected to be the result of random processes much faster than the changes in the useful signal itself. Fourier transform spectral methods exploit this difference in frequency for separating the two components by considering a frequency-domain representation of the signal instead of its original time domain representation.

4.3.1 Continuous Fourier transformation

The frequency domain representation F of a function f depending on time t is defined by the Fourier transform

$$\mathcal{F}[f] = F(\nu) = \int_{-\infty}^{\infty} f(t)\exp(-i2\Pi\nu t)dt \ , \tag{4.31}$$

where $i = (-1)^{1/2}$ and F depends on the frequency ν. If the integral in (4.31) converges, then the Fourier transform is one-to-one, and its inverse is given by

$$\mathcal{F}^{-1}[F] = f(t) = \frac{1}{2\Pi} \int_{-\infty}^{\infty} F(\nu)\exp(i2\Pi\nu t)d\nu \ . \tag{4.32}$$

The generally complex function F can be decomposed into real and imaginary parts according to

$$F(\nu) = \int_{-\infty}^{\infty} f(t)\cos(2\Pi\nu t)dt \ - i\int_{-\infty}^{\infty} f(t)\sin(2\Pi\nu t)dt \tag{4.33}$$

due to the Euler equality $\exp(ix) = \cos(x) + i\sin(x)$. If f is even, then its transform F is real, $F(\nu) = \mathrm{Re}\,F(\nu)$. If f is odd, then F is purely imaginary, $F(\nu) = i\,\mathrm{Im}\,F(\nu)$. Otherwise F is a complex valued function.

Some elementary properties of the Fourier transform are listed in Table 4.3.

Table 4.3
Properties of the Fourier transform

Property	Relationship
linearity	$\mathcal{F}[a_1f_1 + a_2f_2] = a_1\mathcal{F}[f_1] + a_2\mathcal{F}[f_2]$
time shifting	$\mathcal{F}[f(t-t_o)] = \mathcal{F}[f(t)]\exp(-i2\Pi\nu t_o)$
differentiation	$\mathcal{F}[\frac{d}{dt}f] = i2\Pi\nu\mathcal{F}[f]$
integration	$\mathcal{F}[\int_{-\infty}^{\infty} f(\tau)d\tau] = (i2\Pi\nu)^{-1}\mathcal{F}[f]$
convolution	$\mathcal{F}[\int_{-\infty}^{t} f(t-\tau)g(\tau)d\tau] = \mathcal{F}[f]\mathcal{F}[g]$

It is important that differentiation and integration in the time domain give multiplication and division, respectively, by the variable ν in the frequency domain. The role of convolution integrals will be further discussed in Chapter 5.

We can regard Fourier transform as decomposing f into trigonometric functions of different frequencies. This spectral decomposition is based on the property

$$\mathcal{F}[A_1\cos(2\Pi\nu_1 t)] = A_1[\delta(\nu-\nu_1) + \delta(\nu+\nu_1)] \tag{4.34}$$

where $\delta(\nu-\nu_1)$ denotes the Dirac impulse such that $\delta(\nu-\nu_1) = 0$ for $\nu \neq \nu_1$ and

$$\int_{-\infty}^{\infty}\delta(\nu-\nu_1)d\nu = 1 \ . \tag{4.35}$$

By the time shifting property shown in Table 4.3, the transform of a shifted cosine function

$$A_1\cos[2\Pi\nu_1(t-t_0)] = A_1\cos(2\Pi\nu_1 t + \varphi)$$

is given by

$$F[\nu] = \mathcal{F}[A_1\cos(2\Pi\nu_1 t + \varphi)] = A_1[\delta(\nu-\nu_1) + \delta(\nu+\nu_1)]\exp(i\varphi). \tag{4.36}$$

The transform (4.36) is complex valued and vanishes at all frequencies except $\nu = \nu_1$ and $\nu = -\nu_1$. The complex number $F[\nu]$ can be represented by its amplitude $A(\nu) = [\mathrm{Re}^2F(\nu) + \mathrm{Im}^2F(\nu)]^{1/2}$ and phase $\varphi(\nu) = \mathrm{arc\ tg}\ [\mathrm{Im}\ F(\nu)/\mathrm{Re}\ F(\nu)]$. As functions of ν, $A(\nu)$ and $\varphi(\nu)$ are called amplitude and phase spectra, respectively. In the amplitude spectrum of (4.36) we have $A(\nu_1) = A(-\nu_1) = A_1$, whereas $A(\nu) = 0$ if $|\nu| \neq \nu_1$. Since any piecewise continuous function can be expanded into a sum of trigonometric functions with different amplitudes and phases, by (4.36) and by the linearity of Fourier transform the amplitude and phase spectra $A(\nu)$ and $\varphi(\nu)$ uniquely specify the function f. The frequency domain description is frequently given only as the power spectrum $H^2(\nu) = \mathrm{Re}^2F(\nu) + \mathrm{Im}^2F(\nu)$, which does not specify f uniquely, but contains sufficient information in many applications.

This is the analytical formalism we will need in the present section. The experimental data are, however, almost invariably given by a limited set of discrete observations instead of a continuous function defined for $-\infty < t < \infty$. The next subsection extends the Fourier transformation to a finite set of sampled data.

4.3.2 Discrete Fourier transformation

Consider a sample of n observations $\{y_0, y_1, \ldots, y_{n-1}\}$ and define its discrete Fourier transform by

$$a_k = \sum_{j=0}^{n-1} y_j \exp(-i2\Pi kj/n) \ , \quad k = 0, 1, \ldots, n-1 \ , \tag{4.37}$$

where the n a_k values are generally complex numbers. The transformation is one-to-one, and its inverse is given by

$$y_j = \frac{1}{n} \sum_{k=0}^{n-1} a_k \exp(i2\Pi kj/n) \ , \quad j = 0, 1, \ldots, n-1 \ . \tag{4.38}$$

The expression (4.37) can be extended for $k < 0$ or $k > n-1$. At fixed j, however, the points $\exp(-i2\Pi kj/n)$ are on the unit circle and constitute the edges of a regular polygon, and hence the sequence $\ldots a_{-1}, a_0, a_1 \ldots$ is periodic with the period n. Thus $a_{n\times m+k} = a_k$ for all m. In addition, for a real sequence $\{y_0, y_1, \ldots, y_{n-1}\}$ we have the property $a_k = \bar{a}_{n-k}$, i.e., $\mathrm{Re}\ a_k = \mathrm{Re}\ a_{n-k}$ and $\mathrm{Im}\ a_k = -\mathrm{Im}\ a_{n-k}$.

Let $\{y_0, y_1, \ldots, y_{n-1}\}$ represent the sampled values of a continuous function f, i.e., $y_k = f(k\Delta t)$, where Δt is the length of the sampling interval. It is interesting to see how the discrete transforms a_k are related to the sampled values $F(k\Delta t)$ of the Fourier transform of the continuous function f. Assume first that f vanishes outside the interval $[0,T]$, where $T = n\Delta t$ is the sampling time, and $f(0) = f(T)$. Estimating the integral in (4.31) by the trapezium rule we have

$$F(\nu) = \int_0^T f(t)\exp(-2\Pi\nu t)dt \approx \Delta t \sum_{j=1}^{n-1} y_j \exp(-i2\Pi\nu j\Delta t) \ . \tag{4.39}$$

Let $\tilde{F}(\nu)$ denote the sum on the right hand side of (4.39), and define the sampling interval $\Delta\nu$ in the frequency domain by

$$\Delta\nu = 1/(n\Delta t) \ . \tag{4.40}$$

Then $\nu_k = k\Delta\nu$ and

$$\tilde{F}(\nu_k) = \sum_{j=1}^{n-1} y_j \exp(-i2\Pi kj/n) = a_k \ . \tag{4.41}$$

Thus, for our special function f, $F(k\Delta\nu) \approx \Delta t a_k$. If the y_k values are real, then by (4.40) the points of the discrete spectrum are obtained at the frequencies $\nu_0 = 0$, $\nu_1 = 1/T$, $\nu_2 = 2/T, \ldots, \nu_{n/2} = n/(2T)$, where $\nu_{n/2}$ is called the Nyquist critical frequency. The further points of the spectrum are determined by the relation $a_{n-k} = \bar{a}_k$, and hence do not offer any additional information.

However, unless special care is exercised, generally the discrete spectrum does not estimate very well the sampled continuous spectrum. The problems we face are as follows.

(a) Aliasing is present if the function f contains a periodic component with a frequency ν higher than $\nu_{n/2}$, say $\nu = \nu_{n/2} + \Delta\nu$. This component shows up in the spectrum at the frequency $\nu_{n/2} - \Delta\nu$. Thus the spectrum is distorted unless f is bandwidth limited to less than the Nyquist critical frequency. This relationship is the sampling theorem implying that the sampling interval Δt should be chosen sufficiently small, depending on the estimated bandwith of f.

(b) Broadening and "leakage" of the spectrum is the consequence of the finite interval $[0,T]$ of integration in (4.39), if f does not vanish outside this interval. In fact, (4.39) then means estimating the Fourier transform of the product $f(t)W_{[0,T]}$, where $W_{[0,T]}$ is the square window function defined by

$$W_{[0,T]} = \begin{cases} 1, & \text{if } 0 \le t \le T \\ 0, & \text{otherwise} . \end{cases} \qquad (4.42)$$

Thus $\Delta t a_k \approx \mathcal{F}[fW_{[0,T]}]$, which is the convolution integral of the transforms $\mathcal{F}[f]$ and $\mathcal{F}[W_{[0,T]}]$. The latter has rather unpleasant properties. For example, Fig. 4.5 shows the even square window $W_{[-T,T]}$ and its (purely real) transform. $\mathcal{F}[W_{[0,T]}]$ is complex valued, but has similar sidelobs. The undesired convolution of $\mathcal{F}[f]$ with such a boxcar function implies that the spectrum is broadened and has several sidelobs near the critical frequency $\nu_{n/2}$. It can be improved by increasing the sample size.

Although one has to deal with the above problems, the discrete Fourier transformation is still a powerful tool, mainly because of its numerical efficiency. The efficiency does not follow from (4.37) that requires n^2 complex multiplications. As shown, however, by Cooley and Tukey (ref. 14), the transform can be computed in $n \times \log_2 n$ operations with an ingenious algorithm called Fast Fourier Transformation (FFT). The original Radix-2 version of FFT applies to sample sizes $n = 2^m$, where m is a positive integer. This assumption is not very restrictive, since we can always add a sufficient number of zeros to the sample in order to reach the nearest power of 2. As we will discuss, such zero addition might even improve the spectrum.

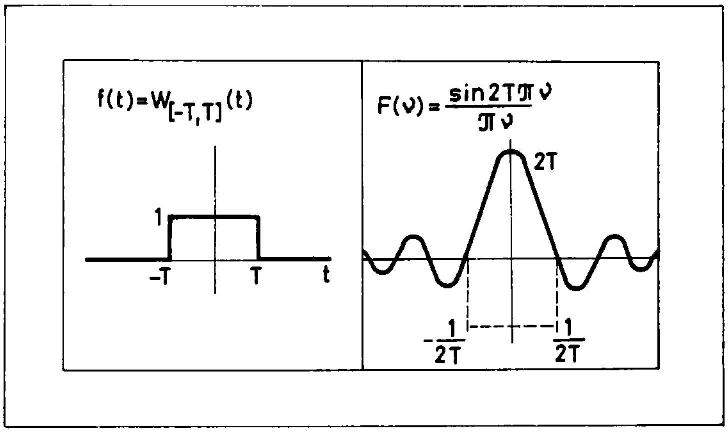


Fig. 4.5. The boxcar function $f(t) = W_{[-T,T]}$ and its Fourier transform $F(\nu)$

The following module is based on the FORTRAN program of Cooley, Lewis and Welch (ref. 15).

Program module M67

```
6700 REM ####################################################
6702 REM #              FAST FOURIER TRANSFORM              #
6704 REM #      RADIX-2 ALGORITHM OF COOLEY AND TUKEY       #
6706 REM ####################################################
6708 REM INPUT:
6710 REM     M       LOG2(NUMBER OF POINTS)
6712 REM A(1...2^M)  REAL PART OF FUNCTION VALUES
6714 REM B(1...2^M)  IMAGINARY PART OF FUNCTION VALUES
6716 REM    IN       IDENTIFIER OF INVERSE TRANSFORMATION
6718 REM                0  - DIRECT TRANSFORMATION
6720 REM              NOT 0 - INVERSE TRANSFORMATION
6722 REM OUTPUT:
6724 REM A(1...2^M)  REAL PART OF TRANSFORMED SEQUENCE
6726 REM B(1...2^M)  IMAGINARY PART OF TRANSFORMED SEQUENCE
```

```
6728 NP=2^M
6730 IF IN THEN FOR I=1 TO NP :A(I)=A(I)/NP :B(I)=-B(I)/NP :NEXT I
6732 REM ---------- REVERSED BIT ORDER
6734 J=1 :ND=NP/2
6736 FOR I=1 TO NP-1
6738  IF I<J THEN TR=A(I) :TI=B(I) :A(I)=A(J) :B(I)=B(J) :A(J)=TR :B(J)=TI
6740  K=ND
6742  IF K<J THEN J=J-K :K=INT(K/2) :GOTO 6742
6744  J=J+K
6746 NEXT I
6748 REM ---------- RADIX-2
6750 LE=1
6752 FOR L=1 TO M
6754  LD=LE :LE=LE+LE
6756  UR=1 :UI=0 :AN=3.14159/LD
6758  WR=COS(AN) :WI=-SIN(AN)
6760  FOR J=1 TO LD
6762   FOR I=J TO NP STEP LE
6764    IP=I+LD
6766    TR=A(IP)*UR-B(IP)*UI :TI=A(IP)*UI+B(IP)*UR
6768    A(IP)=A(I)-TR :B(IP)=B(I)-TI :A(I)=A(I)+TR :B(I)=B(I)+TI
6770   NEXT I
6772   TR=UR*WR-UI*WI :UI=UR*WI+UI*WR :UR=TR
6774  NEXT J
6776 NEXT L
6778 IF IN THEN FOR I=1 TO NP :B(I)=-B(I) :NEXT I
6780 RETURN
6782 REM ****************************************************
```

The module assumes that the sample points are complex. The real components are placed in vector A, i.e., Re y_0 is stored in A(1) on input. For a real valued sample (like a titration curve) vector B should contain zeros. On output the transform is stored in the same vectors, i.e., Re a_0 can be found in A(1) and Im a_0 in B(1). The module computes the inverse transform (4.38) if the inverse transformation flag IN has a nonzero value.

Before considering a numerical example we discuss some of the most fundamental potential applications.

4.3.3 Application of Fourier transform techniques

Smoothing. The basic idea is to eliminate the high-frequency part of the spectrum and obtain a smoothed function by inverse transformation. Applying such a square window to the spectrum gives, however, poor results due to the phenomena of broadening and leakage. Windowing the spectrum by a smoother function is much better (ref. 16). Fig. 4.6 shows the simple triangle window we will use in Example 4.3.3.

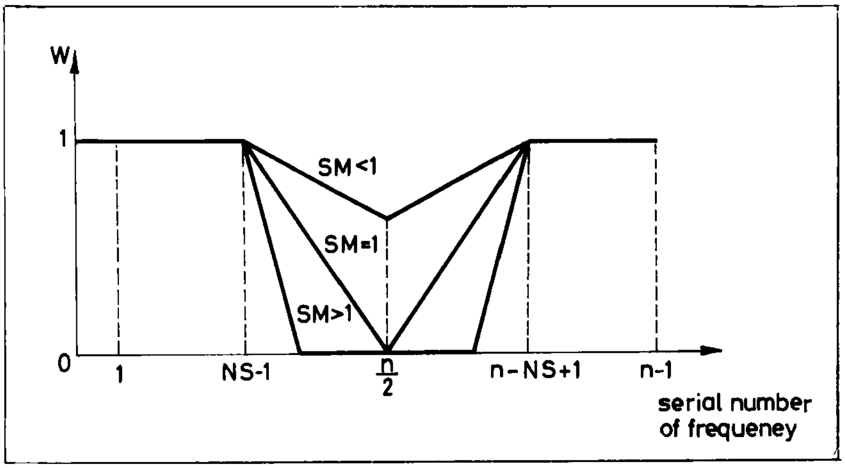


Fig. 4.6. A simple window $W_{[0,n-1]}$ for smoothing real data

The multiplication of the spectrum by a window is equivalent to a convolution in the time domain, and hence the approach is related to the Savitzky-Golay procedure. Indeed, by (4.10) this latter is also a convolution of the function values and the coefficients c_i/F .

Another approach to smoothing involves several segments of the sample, averaging their spectra, and applying the inverse transformation to their mean (ref. 17). Eliminating the high-frequency part of the spectrum, both approaches are also called low-pass filtering.

Base line correction. In a number of applications the signal is distorted by slow effects, resulting in the drift of the base line of the output signal of the instrument. Such slow processes are, for example, the electrochemical changes on the electrode surface in EEG measurements (ref. 18), and the fluorescence signal in Raman spectroscopy (ref. 16). The data are then first smoothed by low-pass filtering, and substracted from the original signal, thereby eliminating the low frequency components.

Interpolation and smoothing by addition of zeros. We may need to add zeros to the sample simply in order to obtain 2^m points. The addition of zeros, however, also increases the length of the observation interval [0,T], and hence the number of frequences in the discrete spectrum. Smoothing the spectrum by an appropriate window and applying the inverse transformation then results in an enhanced sample with new data points between the original ones.

Differentiation and integration. As seen from Table 4.3, we can estimate the derivative of the sampled function if we multiply a_k by the factor $(i2\Pi k\Delta\nu)$ before the inverse transformation. This operation amplifies the high frequency components, and hence it can be used only with a smoothing window as a further multiplier. On the other hand the spectrum is divided by the same factor in order to estimate the integral of the sampled function. Therefore, at sufficiently large values of k the high-frequency components does not disturb the integration. This shows why integration always leads to some smoothing.

Numerical deconvolution. A number of techniques theoretically result in line spectra, with nonzero values only at well defined values of the independent variable. Due to scattering phenomena, however, the separate lines are broadened into peaks of a continuous curve that may be viewed as the convolution of the original line spectrum with the Gaussian function $g(t) = \exp(-at^2)$ (ref. 16). By the last relation in Table 4.3 we can restore the theoretical line structure, or at least significantly narrow the peaks by dividing the transform of the output signal by the transform of the Gaussian and then performing inverse transformation. This procedure is of considerable importance if the peaks overlap and their number is not a priori known.

Feature extraction and data reduction. A sampled continuous signal can frequently be well described in terms of a few low-frequency components of its discrete Fourier transform. This enables us to study, store and compare relatively short vectors in large data bases.

Example 4.3.3 Detection of end points in potentiometric titration by Fourier transform techniques

Our goal is again to find the maxima of the smoothed first derivative of the titration curve first studied in Example 4.1.3. Recall that the discrete transform of a real sample satisfies the relationship $a_{n/2+j} = \bar{a}_{n/2-j}$ for all $j = 1, 2, \ldots, n/2-1$.

Multiplying the transform by the window $W_{[0,n-1]}$ shown in Fig. 4.6 this property is preserved, and hence the inverse transform of the product is purely real. The window (or low-pass filter) is described in terms of two parameters, the index NS of the frequency where smoothing is started, and the smothing factor SM that determines the slope of the decreasing part of the window as shown on Fig. 4.6. The transform of the smoothed function is then the product $\mathcal{F}[f]\mathcal{F}[W]$. To obtain the smoothed derivative of f , we multiply this product by the coefficient (i2ΠkΔν) and perform inverse transformation, whereas the smoothed curve is the inverse transform of the product itself.

The following main program includes the above steps.

```
100 REM ------------------------------------------------------------
102 REM EX. 4.3.3  APPLICATION OF FFT TECHNIQUES
104 REM MERGE M67
106 REM ---------- DATA
108 REM  (NUMBER OF POINTS)
110 DATA 5
112 REM  (V,ml; pH)
114 DATA  2.4, 2.642,  2.6, 2.706,  2.8, 2.786,  3.0, 2.877
116 DATA  3.2, 2.986,  3.4, 3.126,  3.6, 3.295,  3.8, 3.480
118 DATA  4.0, 3.659,  4.2, 3.816,  4.4, 3.952,  4.6, 4.074
120 DATA  4.8, 4.183,  5.0, 4.285,  5.2, 4.384,  5.4, 4.480
122 DATA  5.6, 4.579,  5.8, 4.682,  6.0, 4.791,  6.2, 4.908
124 DATA  6.4, 5.045,  6.6, 5.211,  6.8, 5.444,  7.0, 5.859
126 DATA  7.2, 8.617,  7.4, 9.747,  7.6,10.134,  7.8,10.348
128 DATA  8.0,10.491,  8.2,10.604,  8.4,10.692,  8.6,10.766
200 REM ---------- READ DATA
202 READ M :N=2^M
204 DIM Z(N),F(N),S(N),D(N),A(N),B(N),U(N),V(N)
206 FOR I=1 TO N
208  READ Z(I),F(I)
210  A(I)=F(I) :B(I)=0
212 NEXT I :DX=Z(2)-Z(1)
214 REM ---------- CALL FOURIER TRANSFORMATION MODULE
216 IN=0 :GOSUB 6700
218 REM ---------- SMOOTHING FROM THE NS-TH FREQUENCY
220 REM            SM: SMOOTHING FACTOR
222 NS=N/8 :SM=1
224 FOR I=2 TO N/2
226  S=1 :IF I>=NS THEN S=1-(I+1-NS)/(N/2+2-NS)*SM
228  IF S<0 THEN S=0
230  A(I)=S*A(I)   :B(I)=S*B(I)
232  A(N+2-I)=A(I) :B(N+2-I)=-B(I)
234 NEXT I
236 S=1-SM :IF S<0 THEN S=0
238 A(N/2+1)=A(N/2+1)*S
240 REM ---------- STORE SMOOTHED TRANSFORM
242 FOR I=1 TO N
244  U(I)=A(I) :V(I)=B(I)
246 NEXT I
248 REM ---------- INVERSE TRANSFORMATION
250 IN=1 :GOSUB 6700
252 REM ---------- STORE SMOOTHED FUNCTION VALUES
254 FOR I=1 TO N :S(I)=A(I) :NEXT I
256 REM ---------- TRANSFORM OF THE FIRST DERIVATIVE
258 D=6.28319/N/DX
260 A(1)=0 :B(1)=0
262 FOR I=2 TO N/2+1
264  A(I)=-V(I)*D*(I-1) :B(I)=U(I)*D*(I-1)
266  A(N+2-I)=A(I) :B(N+2-I)=-B(I)
268 NEXT I
270 REM ---------- INVERSE TRANSFORMATION
272 IN=1 :GOSUB 6700
274 REM ---------- STORE DERIVATIVES
276 FOR I=1 TO N :D(I)=A(I) :NEXT I
```

```
278 REM ---------- PRINT RESULTS
280 V$=STRING$(50,"-")
282 A$=" #.##      ##.##       ##.##      ###.###"
284 LPRINT "   SMOOTHING BY DISCRETE FOURIER TRANSFORMATION": LPRINT
286 LPRINT "NUMBER OF FREQUENCY WHERE SMOOTHING STARTS, NS ..";NS
288 LPRINT "SMOOTHING FACTOR, SM ...........................";SM
290 LPRINT :LPRINT V$
292 LPRINT "V, ml      MEAS pH    SMOOTHED pH    DERIVATIVE"
294 LPRINT V$
296 FOR I=1 TO N
298  LPRINT USING A$;Z(I),F(I),S(I),D(I)
300 NEXT I
302 LPRINT V$
304 STOP
```

The following output should be evaluated with care.

```
   SMOOTHING BY DISCRETE FOURIER TRANSFORMATION

NUMBER OF FREQUENCY WHERE SMOOTHING STARTS, NS .. 4
SMOOTHING FACTOR, SM ........................... 1

--------------------------------------------------
V, ml      MEAS pH    SMOOTHED pH    DERIVATIVE
--------------------------------------------------
  2.40       2.64         4.44       -19.323
  2.60       2.71         2.67        -0.319
  2.80       2.79         2.81        -0.250
  3.00       2.88         2.75         0.521
  3.20       2.99         2.92         0.632
  3.40       3.13         3.04         0.910
  3.60       3.30         3.26         0.995
  3.80       3.48         3.44         1.043
  4.00       3.66         3.66         0.942
  4.20       3.82         3.82         0.824
  4.40       3.95         3.99         0.696
  4.60       4.07         4.10         0.601
  4.80       4.18         4.22         0.511
  5.00       4.29         4.31         0.459
  5.20       4.38         4.41         0.414
  5.40       4.48         4.48         0.399
  5.60       4.58         4.57         0.404
  5.80       4.68         4.65         0.438
  6.00       4.79         4.74         0.485
  6.20       4.91         4.85         0.610
  6.40       5.05         4.98         0.782
  6.60       5.21         5.19         1.244
  6.80       5.44         5.47         1.935
  7.00       5.86         6.36         7.811
  7.20       8.62         8.29         9.506
  7.40       9.75         9.63         3.990
  7.60      10.13        10.18         1.880
  7.80      10.35        10.43         0.972
  8.00      10.49        10.64         0.679
  8.20      10.60        10.61        -0.193
  8.40      10.69        10.75        -0.287
  8.60      10.77         8.99       -19.308
--------------------------------------------------
```

Indeed, both the smoothed curve and the derivative have sidelobs at both ends of the sample, but the results are satisfying at most of the internal points. Since Fourier transform spectral methods are usually applied to samples much larger than the one considered here, the distortion at a few outermost points is not a serious drawback.

Exercise

- Repeat the computations with other NS and SM values and investigate how the number of maxima of the derivative changes.

4.4 APPLICATIONS AND FURTHER PROBLEMS

4.4.1 Heuristic methods of local interpolation

Spline interpolation is a global method, and this property is not necessarily advantageous for large samples. Several authors proposed interpolating formulas that are "stiffer" than the local polynomial interpolation, thereby reminding spline interpolation, but are local in nature. The cubic polynomial of the form

$$p_i(d) = y_i + t_i d + \left[\frac{3(y_{i+1} - y_i)}{h_i} - 2t_i - t_{i+1} \right] \frac{d^2}{h_i} + \left[t_i + t_{i+1} - \frac{2(y_{i+1} - y_i)}{h_i} \right] \frac{d^3}{h_i^2} \tag{4.43}$$

has been used in the i-th interval by Akima (ref. 19), where $d = x - x_i$, $h_i = x_{i+1} - x_i$, whereas t_i and t_{i+1} denote the derivative of the polynomial at $d = 0$ and $d = h_i$, respectively. Concatenation of the polynomials (4.43) gives a continuous and once continuously differentiable interpolating function. (Notice that cubic splines are twice continuously differentiable.) The heuristics lies in the choice of t_i . The weighted sum

$$t_i = \frac{m_{i-1}|m_{i+1} - m_i| + m_i|m_{i-1} - m_{i-2}|}{|m_{i+1} - m_i| + |m_{i-1} - m_{i-2}|} , \tag{4.44}$$

has been proved useful where $m_i = (y_{i+1} - y_i)/h_i$. Slightly different formulas have been suggested by Butland (refs. 20,21).

Exercise

□ Interpolate the titration curve implementing Akima's method. Compare the interpolating curve with the results of local cubic interpolation and spline interpolation.

4.4.2 Processing of spectroscopic data

In order to maximize information obtained from raw spectroscopic data, analytical chemists and instrumental specialists depend on signal processing and apply a large number of specialized versions of the basic methods considered in this chapter, as well as the parametric methods discussed in the previous chapter, see, e.g. (ref. 22). Here we provide only an example of parametric methods. Table 4.4 shows 20 points of the electronic absorption spectrum of o-phenilenediamidine in ethanol (ref. 23).

Table 4.4
Points of an electronic absorption spectrum

Frequency, cm^{-1}	50000	49000	48000	47000	46000	45000	44000	43000
Absorptivity	20000	29000	38000	32000	19000	9000	6000	6200

Frequency, cm^{-1}	42000	41000	40000	39000	38000	37000	36000	35000
Absorptivity	6500	6000	3800	1800	880	950	1800	2700

Frequency, cm^{-1}	34000	33000	32000	31000
Absorptivity	3200	2500	850	170

Following the treatment in (ref. 23) we separate the spectrum into 3 Gaussians, and estimate the parameters A_i, B_i and C_i, i = 1, 2, 3, of the function

$$y = \sum_{i=1}^{3} A_i \exp\left[-\left(\frac{\nu_i - C_i}{B_i} \right)^2 \right] \qquad (4.45)$$

using the nonlinear least squares module M45. The initial estimates of the parameters shown in Table 4.5 can be obtained by inspecting the curve. Indeed, C_i is the location of the i-th peak, A_i is its height and B_i is its half-width multiplied by $\sqrt{2}$. The initial estimates and the ones resulting from the fit are shown in Table 4.5. The fit is plotted on Fig. 4.7.

Table 4.5
Parameter estimates in model (4.45)

	Initial	Final	95% confidence interval
A_1	39000	37165	30108 - 44223
B_1	2500	2365	1946 - 2784
C_1	48000	48046	47774 - 48319
A_2	6500	6409	5230 - 7588
B_2	2500	2754	2210 - 3299
C_2	42000	42080	41419 - 42741
A_3	3200	3389	2271 - 4008
B_3	2500	1844	1684 - 2003
C_3	34000	34434	34247 - 34620

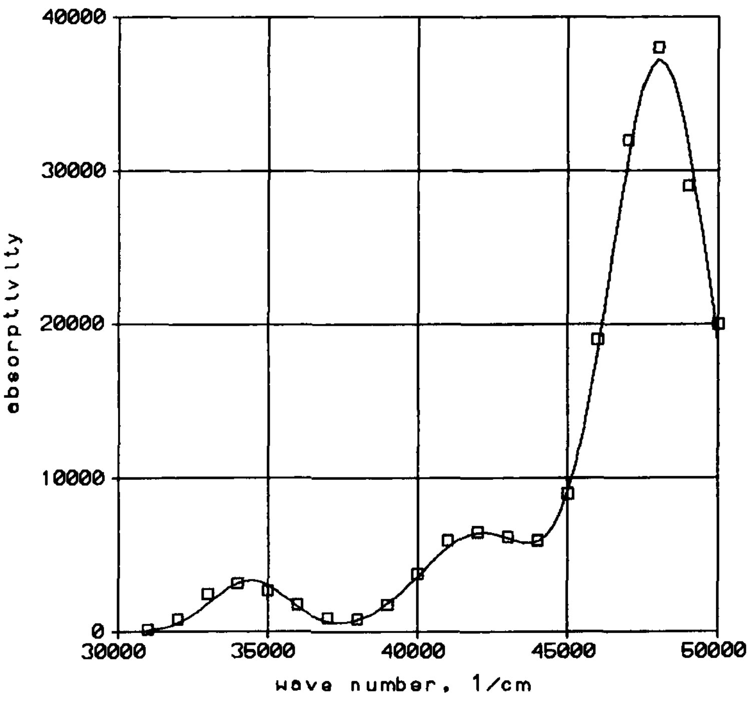


Fig. 4.7. Observed (points) and fitted (continuous) electronic absorption spectrum

Exercise

◻ Use smoothing spline to obtain the initial estimates for peak location (the location of the maximum), peak height (function value at the maximum point) and half-width (the distance between the maximum point and the inflection point).

REFERENCES

1 S.M. Bozic, Digital and Kalman Filtering. Arnold, London, 1979.
2 H.C. Smit, Specification and Estimation of Noisy Analytical Signals, Laboratory of Analytical Chemistry, University of Amsterdam, 1986, (manuscript)
3 A. Ralston, A First Course in Numerical Analysis, McGraw-Hill, New York, 1965.
4 P. Henrici, Elements of Numerical Analysis, John Wiley, New York, 1964.
5 F. B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, New York, 1956.
6 V.P. Glushko (Editor), Thermodynamic properties of Individual Substances, (in 4 volumes) Nauka, Moscow, 1978-1982 (in Russian)
7 A. Savitzky and M.I.E. Golay, Smoothing and differentiation of data by simplified least squares procedures, Analytical Chemistry, 36 (1964) 1627-1639.
8 J. Steiner, Y. Fermonia and J. Deltour, Comment on smoothing and differentiation of data, Analytical Chemistry, 44 (1972) 1906-1909.
9 A. Proctor and P.M.A. Sherwood, Analytical Chemistry, 52 (1980) 2315.
10 J. Házi, Eötvös Loránd University Budapest, Personal communication.
11 C. de Boor, A Practical Guide to Splines, Springer, New York, 1978.
12 I.J. Schoenberg, On interpolation by spline functions and its minimal properties. On Approximation Theory, p109. Proceedings of the Conference held in the Mathematical Research Institute at Oberwolfach, Black Forest, August 4-10, 1963. Birkhäuser, Basel-Stuttgart, 1964.
13 C.H. Reinsch, Smoothing by spline functions I-II. Numerische Mathematik, 10 (1967) 177-183 and 16 (1971) 451-454.
14 J.W. Cooley and J.W. Tukey, An algorithm for the machine calculation of complex Fourier series, Mathematics of Computation, 19 (1965) 297.
15 J.W. Cooley, P.A. Lewis and P.D. Welch, The fast Fourier transform and its applications. in K. Enslein, A. Ralston and H.S. Wilf (Editors) Statistical Methods for Digital Computers, Vol. 3, John Wiley, New York, 1977.
16 R.R. Griffiths (Editor), Transform Techniques in Chemistry, Plenum Press, New York, 1978.
17 P.D. Welch, The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodigrams. IEEE Trans. Audio and Electroacustics. AU-15 (1967) 70-73.
18 J. Berkhout and B.O. Walter, Temporal stability and individual differences in human EEG. An analysis of variance of spectral values. IEEE Trans. Biomed. Engng. 15 (1968) 165-173.
19 H. Akima, A new method of interpolation and smooth curve fitting based on local procedures. Journal of the ACM, 17 (1970) 589-602.
20 I. Butland, in K.W. Brodie (Editor) IMA Conf. on Mathematical Methods in Comp. Graphics and Design '79. Academic Press, Toronto, 1980.
21 R.I. Tervo, T.I. Kenett and W.V. Prestwich, An automated background estimation procedure for gamma ray spectra. Nucl. Instr. Meth. Phys. Res., 216 (1983) 205-208.
22 P. Gaus and I.B. Gill, Smoothing and differentiation of spectroscopic curves using spline functions, J. Applied Spectroscopy, 38 (1984) 370-376.
23 G. Beech, FORTRAN IV in Chemistry, John Wiley, New York, 1975.

Chapter 5

DYNAMICAL MODELS

This chapter is devoted to predicting the behavior of systems modelled by ordinary differential equations of the form

$$\frac{d}{dt} y = f(t,y) \ , \qquad (5.1)$$

that account for relationships between the dependent variables $y = (y_1, y_2, \ldots, y_n)^T$ and their time derivatives $dy/dt = (dy_1/dt, dy_2/dt, \ldots, dy_n/dt)^T$. To obtain such models, one first usually formulates balance equations for extensive quantities such as mass, energy or momentum, considering all changes that occur in the system during a small time interval Δt. If these changes are smooth, and the system is homogeneous, i.e., its variables do not significantly depend on the spatial coordinates, then the assymptotic treatment $\Delta t \rightarrow 0$ results in a model of the form (5.1). For example, the rate of the radioactive decay of y atoms is proportional to the number of atoms. Thus $\Delta y = - ky\Delta t$, where k is the positive decay constant, and $\Delta t \rightarrow 0$ gives the well known differential equation

$$\frac{dy}{dt} = - ky \ . \qquad (5.2)$$

Equations (5.1) define a direction vector at each point (t,y) of the $n+1$ dimensional space. Fig. 5.1 shows the field of such vectors for the radioactive decay model (5.2). Any function $y(t)$, tangential to these vectors, satisfies (5.2) and is a solution of the differential equation. The family of such curves is the so called general solution. For (5.2) the general solution is given by

$$y = c \times \exp(-kt) \ , \qquad (5.3)$$

where c is an arbitrary constant. There exists, however, only one particular solution through any fixed point (t,y^o), where $y^o = y(0)$ is called initial condition or initial value, and it uniquely determines the value of c in the expression (5.3).

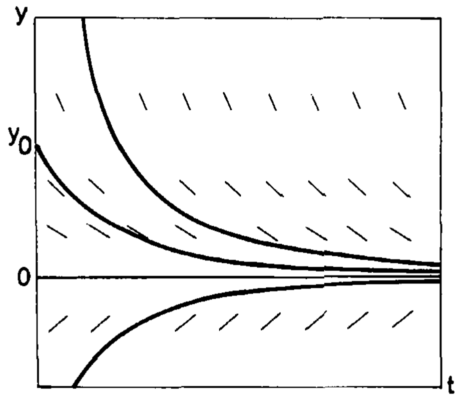


Fig. 5.1. Vector field defined by the differential equation

Existence and uniqueness of the particular solution of (5.1) for an initial value y^o can be shown under very mild assumptions. For example, it is sufficient to assume that the function f is differentiable and its derivatives are bounded. Except for a few simple equations, however, the general solution cannot be obtained by analytical methods and we must seek numerical alternatives. Starting with the known point (t_o,y^o), all numerical methods generate a sequence (t_1,y^1), (t_2,y^2), ..., (t_i,y^i), approximating the points of the particular solution through (t_o,y^o). The choice of the method is large and we shall be content to outline a few popular types. One of them will deal with stiff differential equations that are very difficult to solve by classical methods. Related topics we discuss are sensitivity analysis and quasi steady state approximation.

Both the function f and the initial condition y^o may depend on unknown parameters p :

$$\frac{d}{dt} y = f(t,y,p), \quad y(0) = y^o(p) . \tag{5.4}$$

A frequent problem is to estimate p from the sample $\{ (t_i,\tilde{y}_i)$, $i = 1, 2, \ldots, nm \}$, where $\tilde{y}_i$ denotes an error-corrupted observation of the solution. If the solution is known in analytic form, we have a parameter estimation problem treated in Chapter 3. In principle, one can use the same methods even without an analytical solution, solving the differential equations numerically in each iteration of the estimation procedure. The computational cost of such treatment is, however, too high for most personal computers, and we will propose a special technique with improved numerical efficiency.

Modeling of some systems leads to higher order differential equations of the form

$$y^{(m)} = f(t,y,y^{(1)},\ldots,y^{(m-1)}) \ . \tag{5.5}$$

The additional variables $x_1 = y$, $x_2 = y^{(1)},\ldots, x_m = y^{(m-1)}$ reduce (5.5) to a set (5.1) of m first-order differential equations, and hence you do not need special methods to solve (5.5). Nevertheless, we will treat separately the problems of identifying and inverting single-input, single-output linear systems described by the equation

$$y^{(m)} + a_1y^{(m-1)} + \ldots + a_my = b_1u^{(m-1)} + \ldots + b_mu \ , \tag{5.6}$$

where $u(t)$ is the input and $y(t)$ is the output of the system.

Ordinary differential equations are suitable only for describing homogeneous systems, and we need partial differential equations if the variables depend also on spatial coordinates. The solution of such equations is beyond the scope of this book.

5.1 NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

Although not recommended for practical use, the classical Euler extrapolation is a convenient example to illustrate the basic ideas and problems of numerical methods. Given a point (t_i,y^i) of the numerical solution and a step size h, the explicit Euler method is based on the approximation $(y^{i+1} - y^{(i)})/(t_{i+1} - t_i) \approx dy/dt$ to extrapolate the solution to $t_{i+1} = t_i + h$ by the expression

$$y^{i+1} = y^i + hf(t_i,y^i) \ . \tag{5.7}$$

As seen from Fig. 5.2, reducing the step size h improves the accuracy of this estimation.

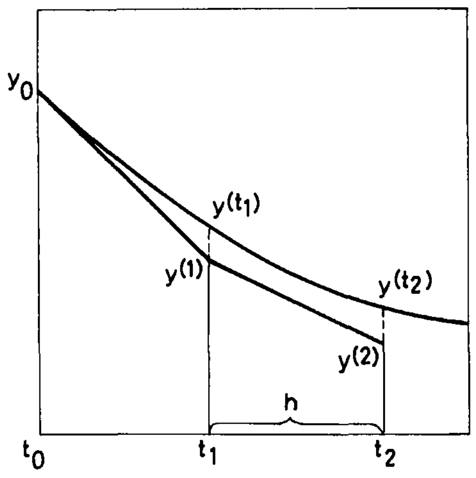


Fig 5.2. True values $y(t)$ and computed values $y^{(i)}$ in the Euler method

While in the first step the deviation from the exact solution stems only from approximating the solution curve by its tangent line, in further steps we calculate the slope at the current approximation y^i instead of the unknown true value $y(t_i)$, thereby introducing additional errors. The solution of (5.2) is given by (5.3), and the total error $E_i = y(t_i) - y^i$ for this simple equation is

$$E_i = y(t_{i-1})\exp(-kh) - (1 - kh)y^{i-1} . \tag{5.8}$$

Since $y^{i-1} = y(t_{i-1}) - E_{i-1}$, (5.8) yields the recursive relation

$$E_i = [\exp(-kh) - (1 - kh)]y(t_{i-1}) + (1 - kh)E_{i-1} . \tag{5.9}$$

The first term in (5.9) is the local truncation or step error that occurs in a single step and does not take into account the use of y^{i-1} instead of $y(t_{i-1})$. The second term shows the propagation of the error E_{i-1}. It is of primary importance to keep the effect of E_i decreasing in latter steps, resulting in the stability of the method. In this simple example the requirement of stability implies $|1-kh| \le 1$, so that

$$h \le \frac{2}{k} . \tag{5.10}$$

Thus, stability can be achieved only at sufficiently small step sizes. Such steps decrease also the truncation error, but increase the required computational effort. Therefore, a common goal of all numerical methods is to provide stability and relatively small truncation errors at a reasonably large step size (refs. 1-2).

The stability of the Euler method is improved by using interpolation instead of extrapolation, and considering the tangent evaluated at t_{i+1} :

$$y^{i+1} = y^i + hf(t_i,y^{i+1}) . \tag{5.11}$$

For the special case of (5.2) we can solve (5.11) as

$$y^{i+1} = \frac{1}{1 + kh}y^i ,$$

and then the total error is given by

$$E_i = \left[\exp(-kh) - \frac{1}{1 + kh}\right]y(t_{i-1}) + \frac{1}{1 + kh}E_{i-1} . \tag{5.12}$$

The truncation errors in (5.9) and (5.12) are of the same magnitude, but the implicit Euler method (5.11) is stable at any positive step size h. This conclusion is rather general, and the implicit methods have improved stability properties for a large class of differential equations. The price we have to pay for stability is the need for solving a set of generally nonlinear algebraic equations in each step.

To compare the explicit and implicit Euler methods we exploited that the solution (5.3) of (5.2) is known. We can, however, estimate the truncation error without such artificial information. Considering the truncated Taylor series of the solution, for the explicit Euler method (5.7) we have

$$y(t_{i+1}) - y^{i+1} = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(\theta) - y^i - hf(t_i,y^i) = \frac{h^2}{2}y''(\theta) \tag{5.13}$$

where we assumed $y^i = y(t_{i-1})$ to obtain the local truncation error. The value of θ is between t_i and t_{i+1}, but otherwise unknown. Nevertheless, (5.13) shows that the truncation error is proportional to h^2. We can derive a similar expression for each method, and express the truncation error in the form $C \times h^{p+1}$, where the integer p is said to be the order of the method. The explicit and implicit Euler methods are both first order ones. While a higher order implies smaller truncation error, this does not necessarily mean improved efficiency, since the computational costs are usually increased.

5.1.1 Runge - Kutta methods

The formulas (5.7) and (5.11) of explixit and implicit Euler methods, respectively, are unsymmetrical, using derivative information only at one end of the time interval of interest. Averaging the slopes of the two tangent lines means using more information, and gives

$$y^{i+1} = y^i + \frac{h}{2}[f(t_i,y^i) + f(t_{i+1},y^{i+1})] \ . \qquad (5.14)$$

Since the formula (5.14) is implicit, we must solve a (generally) nonlinear equation to obtain y^{i+1}. To simplify the calculation, consider the prediction

$$\bar{y}^{i+1} = y^i + k_1 \ , \qquad (5.15)$$

of y^{i+1} , where

$$k_1 = hf(t_i,y^i) \ . \qquad (5.16)$$

Thus the prediction is based only on the explicit formula (5.7). Using this prediction, let

$$k_2 = hf(t_{i+i},\ y^i + k_1) \ , \qquad (5.17)$$

then

$$y^{i+1} = y^i + \frac{1}{2}(k_1 + k_2) \qquad (5.18)$$

approximates the formula (5.14), but is explicit. The improvement (5.18) makes the Euler method second order. The generalization of the above idea leads to the family of Runge - Kutta methods in the form of

$$y^{i+1} = y^i + (b_1k_1 + b_2k_2 + \ldots + b_sk_s) \ , \qquad (5.19)$$

where

$$k_m = hf(t_i + d_mh,\ y^i + a_{m1}k_1 + \ldots + a_{m,m-1}k_{m-1}) \ , \quad 1 \le m \le s \ . \qquad (5.20)$$

The constants a_{ij}, b_i and d_i are chosen to maximize the order p of the method. For any given p we need at least s terms in (5.19), where s depends on p (ref. 2). In particular, if p equals 1, 2, 3 or 4, then $s = p$. For $p = 5$, however, we need $s = 6$ terms, i.e., 6 function evaluations in each time step. This partly explains the popularity of the fourth-order Runge - Kutta method :

$$y^{i+1} = y^{i} + k_1/6 + k_2/3 + k_3/3 + k_1/6 \ , \tag{5.21}$$

where

$$\begin{aligned} k_1 &= hf(t_i, y^i) \\ k_2 &= hf(t_i + h/2,\ y^i + k_1/2) \\ k_3 &= hf(t_i + h/2,\ y^i + k_2/2) \\ k_4 &= hf(t_i + h,\ y^i + k_3/2) \ . \end{aligned} \tag{5.22}$$

The following program module extends the formula (5.21) to vector differential equations of the form (5.1), simply by considering **y** , **f** and $\mathbf{k}_i$ as vectors.

Program module M70

```
7000 REM ****************************************************
7002 REM *  SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS     *
7004 REM *         FOURTH ORDER RUNGA-KUTTA METHOD          *
7006 REM ****************************************************
7008 REM INPUT:
7010 REM     N      NUMBER OF DEPENDENT VARIABLES
7012 REM     T      INITIAL TIME
7014 REM     Y(N)   INITIAL CONDITIONS
7016 REM     H      TIME STEP SIZE
7018 REM     NS     REQUIRED NUMBER OF STEPS
7020 REM OUTPUT:
7022 REM     T      END TIME
7024 REM     Y(N)   SOLUTION AT END TIME
7026 REM USER SUPPLIED SUBROUTINE:
7028 REM    FROM LINE 900:   T,Y(N) --> D(N)   ( RHS EVALUATION )
7030 REM AUXILIARY ARRAYS:
7032 REM   R(N),Q(N)
7034 FOR L=1 TO NS
7036  FOR I=1 TO N :R(I)=Y(I) :NEXT I
7038  GOSUB 900
7040  FOR I=1 TO N :Q(I)=D(I) :Y(I)=R(I)+.5*H*D(I) :NEXT I
7042  T=T+.5*H :GOSUB 900
7044  FOR I=1 TO N :Q(I)=Q(I)+2*D(I) :Y(I)=R(I)+.5*H*D(I) :NEXT I
7046  GOSUB 900
7048  FOR I=1 TO N :Q(I)=Q(I)+2*D(I) :Y(I)=R(I)+H*D(I) :NEXT I
7050  T=T+.5*H :GOSUB 900
7052  FOR I=1 TO N :Y(I)=R(I)+H/6*(Q(I)+D(I)) :NEXT I
7054 NEXT L
7056 RETURN
7058 REM ****************************************************
```

The module calls the user supplied subroutine starting at line 900 that evaluates the right hand sides of (5.1) at the current values of the Y vector and time T and put them into the vector D. For a single equation only Y(1) and D(1) are used. The step size H and the number NS of steps are selected by the user.

Example 5.1.1 Solving a microbial growth model by Runge - Kutta method

In a batch fermentation process studied by Holmberg (ref. 3) the substrate is converted to biomass. The specific growth rate $\mu(y_2)$ is described by the Michaelis - Menten equation

$$\mu(y_2) = \frac{V_m y_2}{K_s + y_2} \tag{5.23}$$

where V_m is the maximum specific growth rate, K_s is the so called Michaelis - Menten constant and y_2 denotes the substrate concentration. The concentration y_1 of the microorganisms and the concentration of the substrate are governed by the system of differential equations

$$\begin{aligned} \frac{dy_1}{dt} &= \mu(y_2)y_1 - K_d y_1 , \\ \frac{dy_2}{dt} &= - \frac{1}{Y}\mu(y_2)y_1 , \end{aligned} \tag{5.24}$$

where K_d is the decay rate coefficient and Y is the yield coefficient. Typical values of the coefficients and initial conditions are $V_m = 0.5\ h^{-1}$, $K_s = 3$ g/l, $Y = 0.6$, $K_d = 0.05\ h^{-1}$, $y_1{}^o = 1$ g/l and $y_2{}^o = 30$ g/l. The following main program determines the concentrations during a 10 hours period.

```
100 REM ------------------------------------------------------------
102 REM EX. 5.1.1. FERMENTATION KINETICS BY RUNGE-KUTTA METHOD
104 REM MERGE M70
106 REM ---------- DATA
108 N=2 :VM=.5 :KS=3 :YY=.6 :KD=.05
200 REM ---------- DIMENSIONS
202 DIM Y(N),D(N),R(N),Q(N)
204 REM ---------- INITIAL CONDITIONS, STEP SIZE, NUMBER OF STEPS
206 Y(1)=1 :Y(2)=30 :T=0 :H=.05 :NS=1/H
208 V$=STRING$(40,"-")
210 LPRINT "FOURTH-ORDER RUNGE-KUTTA,  STEP SIZE H=";H :LPRINT
212 LPRINT V$
214 LPRINT "TIME, h        y1, g/l        y2, g/l"
216 LPRINT V$
218 LPRINT USING" ##.##         ##.###         ##.####";T,Y(1),Y(2)
220 FOR ID=1 TO 10
222  GOSUB 7000
224  LPRINT USING" ##.##         ##.###         ##.####";T,Y(1),Y(2)
226 NEXT ID
228 LPRINT V$ :LPRINT
230 STOP
900 REM ---------- RIGHT HAND SIDE EVALUATION
902 MS=VM*Y(2)/(KS+Y(2))
904 D(1)=MS*Y(1)-KD*Y(1)
906 D(2)=-1/YY*MS*Y(1)
908 RETURN
```

A way to check the accuracy of the solution is repeating the procedure with a smaller step size until the significant digits will be unchanged. More efficient methods of step size control will be discussed in Section 5.1.3. In this example the step size $h = 0.05$ hours has been proved appropriate and results in the following solution:

```
FOURTH-ORDER RUNGE-KUTTA,  STEP SIZE H= .05

----------------------------------------
TIME, h         y1, g/l        y2, g/l
----------------------------------------
  0.00           1.000         30.0000
  1.00           1.498         29.0678
  2.00           2.239         27.6780
  3.00           3.339         25.6158
  4.00           4.955         22.5806
  5.00           7.290         18.1852
  6.00          10.524         12.0600
  7.00          14.386          4.5845
  8.00          16.204          0.2518
  9.00          15.557          0.0033
 10.00          14.800          0.0000
----------------------------------------
```

5.1.2 Multistep methods

In the improved Euler method (5.14) we use derivative information at two points of the time interval of interest, thereby *increasing the order* of the method. A straightforward extension of this idea is to use the derivative at several grid points, leading to the k-step formulas

$$y^{i+1} = \sum_{m=0}^{k} b_m f(t_{i-m+1}, y^{i-m+1}) \qquad (5.25)$$

of Adams (ref. 2). More general multistep formulas can be derived using not only the derivatives, but also function values y^i computed at previous grid points when estimating y^{i+1}.

The multistep method (5.25) is explicit if $b_0 = 0$, otherwise it is *implicit*. *These latter are the best ones due to their improved stability* properties. To use an implicit formula, however, we need an initial estimate of y^{i+1}. The basic idea of the predictor - corrector methods is to estimate y^{i+1} by a p-th order explicit formula, called predictor, and then to refine y^{i+1} by a p-th order implicit formula, which is said to be the corrector. Repeating the correction means solving the algebraic equation (5.25) by successive substitution. The use of more than two iterations is not efficient.

The great advantage of the predictor - corrector methods is that in addition to y^{i+1}, in expression (5.25) we need only previously computed (and saved) function values. Thus, the computational cost depends on the number of corrections and does not depend on the order p of the particular formula.

Starting a multistep method is an additional problem, since no previous function values are yet available. One can start with a one step formula and a small step size, then gradually increase k to the desired value. A more common approach is to use Runge - Kutta steps of the same order at the beginning.

The module included here is based on the fourth order method of Milne (ref. 4), where the predictor

$$\bar{y}^{i+1} = y^{i-3} + \frac{4h}{3}(2f_{i-2} - f_{i-1} + 2f_i) \tag{5.26}$$

is combined with the corrector

$$y^{i+1} = y^{i-1} + \frac{h}{3}[\ 2f_{i-1} - 4f_i + f(t_{i+1},\bar{y}^{i+1})\ ]\ . \tag{5.27}$$

Only one correction is made and the procedure is started calling the fourth order Runge - Kutta module M70 .

<u>Program module M71</u>

```
7100 REM ####################################################
7102 REM #   SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS    #
7104 REM #        PREDICTOR-CORRECTOR METHOD OF MILNE       #
7106 REM ####################################################
7108 REM INPUT:
7110 REM     N      NUMBER OF DEPENDENT VARIABLES
7112 REM     T      INITIAL TIME
7114 REM     Y(N)   INITIAL CONDITIONS
7116 REM     H      TIME STEP SIZE
7118 REM     NS     REQUIRED NUMBER OF STEPS (AT FIRST CALL NS>=4)
7120 REM     FC     IDENTIFIER OF FIRST CALL
7122 REM                 0   - NOT FIRST CALL, THUS VALUES
7124 REM                       Y1(N),Y2(N),Y3(N),D1(N),D2(N) ARE KNOWN
7126 REM               NOT 0 - FIRST CALL
7128 REM                       ( REQUIRES NS>=4 )
7130 REM OUTPUT:
7132 REM     T      END TIME
7134 REM     Y(N)   SOLUTION AT END TIME
7136 REM   ( AND UPDATED VALUES OF Y1(N),Y2(N),Y3(N),D1(N),D2(N),FC )
7138 REM USER-SUPPLIED SUBROUTINE
7140 REM   FROM LINE 900:   T,Y(N) --> D(N)   ( RHS EVALUATION )
7142 REM AUXILIARY ARRAYS:
7144 REM   R(N),Q(N)
7146 REM MODULE CALLED: M70
```

```
7148 IF FC=0 THEN N1=1 :N2=NS :GOTO 7158
7150 N1=4 :N2=NS :NS=1
7152 FOR I=1 TO N :Y3(I)=Y(I) :NEXT I :GOSUB 7000
7154 GOSUB 900 :FOR I=1 TO N :Y2(I)=Y(I) :D2(I)=D(I) :NEXT I :GOSUB 7000
7156 GOSUB 900 :FOR I=1 TO N :Y1(I)=Y(I) :D1(I)=D(I) :NEXT I :GOSUB 7000
7158 FOR L=N1 TO N2
7160  REM ---------- PREDICT
7162  GOSUB 900
7164  FOR I=1 TO N
7166   Y=Y(I) :Y(I)=Y3(I)+1.333333*H*(2*D2(I)-D1(I)+2*D(I))
7168   Y3(I)=Y2(I) :Y2(I)=Y1(I) :Y1(I)=Y :D2(I)=D1(I) :D1(I)=D(I)
7170  NEXT I
7172  REM ---------- CORRECT
7174  T=T+H :GOSUB 900
7176  FOR I=1 TO N
7178   Y(I)=Y2(I)+H/3*(D2(I)+4*D1(I)+D(I))
7180  NEXT I
7182 NEXT L
7184 FC=0 :NS=N2
7186 RETURN
7188 REM ****************************************************
```

The use of this module is similar to that of the module M70 . The only new variable is the first call flag FC. You should put a nonzero value into FC before the first call. In subsequent calls FC will remain zero.

Example 5.1.2 Solving the microbial growth model by Milne method

Here we list only the lines differing from the ones of the main program in Example 5.1.1.

```
102 REM EX. 5.1.2. FERMENTATION KINETICS BY MILNE METHOD
104 REM MERGE M70,M71

202 DIM Y(N),D(N),R(N),Q(N),Y1(N),Y2(N),Y3(N),D1(N),D2(N)

206 Y(1)=1 :Y(2)=30 :T=0 :H=.05 :NS=1/H :FC=1

210 LPRINT "MILNE METHOD,  STEP SIZE H=";H :LPRINT

222  GOSUB 7100
```

The given step size results in the same solution, not repeated here.

An important question is the relative numerical efficiency of the two methods or, more generally, the two families of methods. At a fixed step size the predictor - corrector methods clearly require fewer function evaluations. This does not necessarily means, however, that the predictor - corrector methods are superior in every application. In fact, in our present example increasing the step size leaves the Runge - Kutta solution almost unchanged, whereas the Milne solution is deteriorating as shown in Table 5.1.

Table 5.1
Substrate (y_2, g/l) computed at different step sizes H (in hours)

Time, h	Runge - Kutta			Milne		
	H = 0.1	H = 0.2	H = 0.25	H = 0.1	H = 0.2	H = 0.25
6	12.060	12.060	12.060	12.060	12.060	12.060
7	4.585	4.585	4.585	4.585	4.585	4.585
8	0.252	0.253	0.257	0.252	0.253	0.242
9	0.003	0.003	0.004	0.003	0.003	0.018
10	0.000	0.000	0.000	0.000	-0.001	-0.022

Experience shows that the relatively slow Runge - Kutta procedure is quite robust and hence it is a good choice for a first try.

5.1.3 Adaptive step size control

To control the step size adaptively we need an estimate of the local truncation error. With the Runge - Kutta methods a good idea is to take each step twice, using formulas of different order, and judge the error from the deviation between the two predictions. Selecting the coefficients in (5.20) to give the same a_{ij} and d_i values in the two formulas at least for some of the internal function evaluations reduces the overhead in calculation. For example, 6 function evaluations are required with an appropriate pair of fourth-order and fifth-order formulas (ref. 5).

In the predictor - corrector methods the magnitude of the first correction is an immediate error estimate with no additional cost.

From the actual step size h_{act}, error estimate E_{est} and the desired error bound E_{des} a new step size h_{new} can be selected according to

$$\frac{|E_{des}|}{|E_{est}|} \approx \frac{|h_{new}|^{p+1}}{|h_{act}|^{p+1}} , \qquad (5.28)$$

where p is the order of the method. The exponent p instead of (p+1) in (5.28) results in a more conservative step size control, taking into account also the propagation of errors.

The most sophisticated differential equation solver considered in this book and discussed in the next section includes such step size control. In contrast to most integrators, however, it takes a full back step when facing a sudden increase of the local error. If the back step is not feasible, for example at start, then only the current step is repeated with the new step size.

5.2 STIFF DIFFERENTIAL EQUATIONS

Stiffness occures in a problem if there are two or more very different time scales on which the dependent variables are changing. Since at least one component of the solution is "fast", a small step size must be selected. There is, however, also a "slow" variable, and the time interval of interest is large, requiring to perform a large number of small steps. Such models are common in many areas, e.g., in chemical reaction kinetics, and solving stiff equations is a challenging problem of scientific computing.

The eigenvalues λ_i of the Jacobian matrix

$$[J]_{jk} = \frac{\partial f_j(t_i,y)}{\partial y_k} \tag{5.29}$$

of the function f in (5.1) provide some information on the stiffness of a particular system. Local linearization of f gives a linear combination of the exponentials $\exp(\lambda_i t)$ as a local estimate of the behavior of the solution. Let λ_{min} and λ_{max} denote the smallest and largest eigenvalues, respectively. (In case of complex eigenvalues we can use their moduli.) Then the ratio $\lambda_{max}/\lambda_{min}$ shows the ratio of the involved time scales and measures the stiffness, varying along the solution if the equations (5.1) are nonlinear.

Implicit methods, including predictor - corrector ones, are of primary importance in solving stiff equations. The traditional successive approximation correction procedures, however, do not converge, so that are usually replaced by a Newton - Raphson iteration. This idea applies to any implicit method, and the multistep procedure of Gear (ref. 6) has been particularly successful in this respect. We provide, however, a program module based on the so called ROW4A procedure, that is much simpler than the Gear program, in spite of its comparable performance (ref. 7). The ROW4A procedure realizes a semi-implicit Runge - Kutta method introduced by Rosenbrock and modified by Gottwald and Wanner (ref. 8).

The basic formula of the semi-implicit Runge-Kutta methods is similar to (5.20) , but k_m appears also on the right hand side. Since the method is restricted to autonomous differential equations (i.e., the function f does not explicitly depend on time), we drop the argument t and replace (5.20) by the expression

$$k_m = hf\left(y^i + \sum_{q=1}^{m} a_{mq}k_q \right) , \quad m = 1, \ldots, s . \tag{5.30}$$

We need to solve s sets of nonlinear equations, but Rosenbrock devised a much simpler procedure. Linearization of the m-th set of equations in (5.30) around the point

$$y = y^i + \sum_{q=1}^{m-1} a_{mq}k_q \qquad (5.31)$$

gives the equations

$$[I - a_{mm}hJ]k_m = hf\left(y^i + \sum_{q=1}^{m-1} a_{mq}k_q\right) \qquad (5.32)$$

for k_m, where I denotes the n×n identity matrix, and n is the number of dependent variables (the dimension of the y vector). Furthermore, the Jacobian matrix J is evaluated only at the beginning of the current time interval, and the a_{mm} coefficients are identical for any m. The fourth - order method then requires the solution of 4 sets of linear equations

$$Ek_m = r_m \text{ , } m = 1, 2, 3, 4 \qquad (5.33)$$

where

$E = I - a_{11}hJ$

$r_1 = hf(y^i)$

$r_2 = hf(y^i + a_{21}k_1) + c_{21}k_1$

$r_3 = hf(y^i + a_{31}k_1 + a_{32}k_2) + c_{31}k_1 + c_{32}k_2$

$r_4 = hf(y^i + a_{41}k_1 + a_{42}k_2 + a_{43}k_3) + c_{41}k_1 + c_{42}k_2 + c_{43}k_3$.

Since all 4 sets of equations in (5.33) have the same coefficient matrix E, a single LU decomposition is sufficient as described in Sections 1.3.2 and 1.3.3. The next point of the solution is predicted by

$$y^{i+1} = y^i + b_1k_1 + b_2k_2 + b_3k_3 + b_4k_4 \qquad (5.34)$$

whereas

$$E^{i+1} = e_1k_1 + e_2k_2 + e_3k_3 + e_4k_4 \qquad (5.35)$$

is an estimate of the local error vector. The values of the coefficients involved can be found in the line 7260 through 7272 of the following program module.

Program module M72

```
7200 REM ****************************************************
7202 REM *    SOLUTION OF STIFF DIFFERENTIAL EQUATIONS      *
7204 REM * SEMI IMPLICIT-RUNGE KUTTA METHOD WITH BACKSTEPS *
7206 REM *          ROSENBROCK-GOTTWALD-WANNER              *
7208 REM ****************************************************
7210 REM INPUT:
7212 REM     N      NUMBER OF DEPENDENT VARIABLES
7214 REM     T      INITIAL TIME
7216 REM    Y(N)    INITIAL CONDITIONS
7218 REM     TE     REQUIRED END TIME
7220 REM     EP     RELATIVE ERROR TOLERANCE
7222 REM     H      INITIAL TIME STEP SIZE
7224 REM     IM     MAXIMUM NUMBER OF STEPS
7226 REM OUTPUT:
7228 REM     ER     STATUS FLAG
7230 REM              0  SUCCESSFULL SOLUTION
7232 REM              1  NUMBER OF STEPS INSUFFICIENT
7234 REM     T      END TIME
7236 REM     Y(N)   SOLUTION AT END TIME
7238 REM     H      SUGGESTED SIZE OF NEXT STEP
7240 REM     IP     NUMBER OF ACCEPTED STEPS
7242 REM     IR     NUMBER OF REPEATED AND BACKWARD STEPS
7244 REM USER SUPPLIED SUBROUTINE:
7246 REM    FROM LINE 900:   T,Y(N) --> D(N)    ( RHS EVALUATION )
7248 REM AUXILIARY ARRAYS:
7250 REM        E(N,N),A(N,N),R(N),YO(N),YL(N)
7252 REM        R1(N),R2(N),R3(N),R4(N),X(N)
7254 REM MODULES CALLED: M14,M15
7256 IF T>=TE THEN ER=0 :GOTO 7414
7258 REM ---------- INITIALIZATION
7260 A1=.438           :A2=.9389487     :A3=7.387954E-02
7262 C1=-1.943474      :C2=.4169575     :C3=1.323968
7264 C4=1.519513       :C5=1.353708     :C6=-.8541515
7266 B1=.7290448       :B2=5.410698E-02
7268 B3=.2815994       :B4=.25
7270 E1=-1.908589E-02  :E2=.2556088
7272 E3=-8.638163E-02  :E4=.25
7274 IP=0 :IR=0 :LS=-1 :LE=0 :SF=1 :TR=T
7276 FOR I=1 TO N :YO(I)=Y(I) :NEXT I
7278 REM ---------- MAX NUMBER OF STEPS OR END TIME REACHED
7280 IF IP>=IM THEN ER=1 :GOTO 7414
7282 IF T+H>=TE THEN LE=-1 :HO=H :H=TE-T
7284 REM ---------- JACOBIAN MATRIX
7286 GOSUB 900
7288 FOR I=1 TO N :R(I)=D(I) :NEXT I
7290 FOR J=1 TO N
7292  Y=Y(J) :D=ABS(Y)*.001+1E-15 :Y(J)=Y+D
7294  GOSUB 900
7296  FOR I=1 TO N :E(I,J)=(D(I)-R(I))/D :NEXT I
7298  Y(J)=Y
7300 NEXT J
7302 REM ---------- LU DECOMPOSITION
7304 FOR I=1 TO N :FOR J=1 TO N
7306  A(I,J)=-.395*H*E(I,J)-(I=J)
7308 NEXT J :NEXT I
7310 GOSUB 1400
7312 IF ER THEN H=H/2 :GOTO 7304
```

```
7314 REM ---------- COMPUTE STEP
7316 FOR I=1 TO N :X(I)=H*R(I) :NEXT I
7318 GOSUB 1500
7320 FOR I=1 TO N
7322  R1(I)=X(I) :Y(I)=YO(I)+A1*X(I)
7324 NEXT I
7326 GOSUB 900
7328 FOR I=1 TO N :X(I)=H*D(I)+C1*R1(I) :NEXT I
7330 GOSUB 1500
7332 FOR I=1 TO N
7334  R2(I)=X(I) :Y(I)=YO(I)+A2*R1(I)+A3*R2(I)
7336 NEXT I
7338 GOSUB 900
7340 FOR I=1 TO N :X(I)=H*D(I)+C2*R1(I)+C3*R2(I) :NEXT I
7342 GOSUB 1500
7344 FOR I=1 TO N
7346  R3(I)=X(I) :X(I)=H*D(I)+C4*R1(I)+C5*R2(I)+C6*R3(I)
7348 NEXT I
7350 GOSUB 1500
7352 FOR I=1 TO N
7354  R4(I)=X(I) :Y(I)=YO(I)+B1*R1(I)+B2*R2(I)+B3*R3(I)+B4*R4(I)
7356 NEXT I
7358 T=T+H
7360 REM ---------- ESTIMATE ERROR
7362 ES=EP/16
7364 FOR I=1 TO N
7366  S1=ABS(E1*R1(I)+E2*R2(I)+E3*R3(I)+E4*R4(I))
7368  S2=ABS(Y(I)) :S3=ABS(YO(I))
7370  S=2*S1/(S2+S3+EP/1E10)
7372  IF S>ES THEN ES=S
7374 NEXT I
7376 REM ---------- NEW STEP SIZE
7378 S=.9*(EP/ES)^.25
7380 H=S*SF*H
7382 REM ---------- CHECK ERROR
7384 IF ES>EP THEN 7400
7386 REM ---------- ACCEPT STEP AND INCREASE STEP FACTOR SF
7388 IP=IP+1
7390 IF LE THEN H=HO :ER=0 :GOTO 7414
7392 FOR I=1 TO N :YL(I)=YO(I) :YO(I)=Y(I) :NEXT I
7394 TL=TR :TR=T
7396 LS=0 :SF=1.01*SF :IF SF>1 THEN SF=1
7398 GOTO 7280
7400 IR=IR+1 :LE=0: IF NOT LS THEN 7408
7402 REM ---------- REPEAT CURRENT STEP IF BACKSTEP IS NOT POSSIBLE
7404 FOR I=1 TO N :Y(I)=YO(I) :NEXT I
7406 T=TR :GOTO 7304
7408 REM ---------- STEP BACK AND MODERATE STEP FACTOR SF
7410 FOR I=1 TO N :Y(I)=YL(I) :YO(I)=YL(I) :NEXT I
7412 IP=IP-1 :T=TL :TR=T : LS=-1: SF=.9*SF :GOTO 7286
7414 RETURN
7416 REM ****************************************************
```

In contrast to the modules M70 and M71 , here we specify the end time TE instead of the number of steps, since the initial step size H is adaptively decreased or increased in order to keep the relative error just below the threshold EP . The suggested values of the threshold are between 0.01

and 0.0001 . The module returns the value ER = 1 if the maximum allowed number IM of steps does not suffice to obtain the desired accuracy. The number of accepted steps and the number of repeated or backward steps are stored in variables IP and IR , respectively. This information is useful in evaluating the performance of the integrator. The Jacobian matrix is approximated by divided differences, so you need to supply only one subroutine for evaluating the right hand sides of the differential equations, similarly to the previous two modules.

Example 5.2 Solving the model of an oscillating reaction

The famous Oregonator model (ref. 9) is a highly simplified (but very successful) description of the Belousov - Zhabotinsky oscillating reaction :

$$\frac{dy_1}{dt} = k_1[y_2 + y_1(1 - k_2 y_1 - y_2)]$$

$$\frac{dy_2}{dt} = [y_3 - (1 + y_1)y_2]/k_1 \qquad (5.36)$$

$$\frac{dy_3}{dt} = k_3(y_1 - y_3) \ .$$

where y_1, y_2 and y_3 denote the normalized concentrations of $HBrO_2$, Br^- and Ce^{4+}, respectively, and t is the dimensionless time. The dimensionless parameters are $k_1 = 77.27$, $k_2 = 8.375E-6$ and $k_3 = 0.161$ (ref. 8). The initial values $y_1^o = 4$, $y_2^o = 1.33139$ and $y_3^o = 2.85235$ result in a periodic solution with period length $t \approx 302.9$. Within a period there are sudden changes in the variables, more than seven orders of magnitude. In the following main program we compute the solution at selected time points.

```
100 REM ------------------------------------------------------------
102 REM EX. 5.2. SOLUTION OF OREGONATOR MODEL BY SEMI-IMPLICIT METHOD
104 REM MERGE M14,M15,M72
106 REM ---------- NUMBER OF TIME POINTS AND TIME POINTS
108 DATA 12,0,1,2,3,4,5,6,10,100,200,300,302.9
110 READ NT
112 DIM TW(NT)
114 FOR I=1 TO NT :READ TW(I) :NEXT I
200 REM ---------- PROBLEM SIZE
202 N=3
204 DIM Y(N),D(N),E(N,N),A(N,N),R(N),YO(N),YL(N),X(N)
206 DIM R1(N),R2(N),R3(N),R4(N)
208 REM ---------- INITIAL VALUES, FIRST H, ACCURACY
210 T=TW(1) :Y(1)=4 :Y(2)=1.33139 :Y(3)=2.85235 :H=.1 :EP=.001 :IM=1000
212 V$=STRING$(56,"-")
214 A$="###.#  ######.#####   ###.#####  #####.#####  ####  ####"
216 LPRINT "OREGONATOR MODEL BY SEMI-IMPLICIT RUNGE-KUTTA M., ";
218 LPRINT "TOLERANCE=";EP :LPRINT :LPRINT V$
220 LPRINT " TIME        y(1)        y(2)          y(3)      ip    ir"
```

```
222 LPRINT V$
224 LPRINT USING A$;T,Y(1),Y(2),Y(3)
226 REM ---------- CALL SOLUTION MODULE FOR EACH TIME POINT
228 FOR ID=2 TO NT
230  TE=TW(ID) :GOSUB 7200
232  LPRINT USING A$;T,Y(1),Y(2),Y(3),IP,IR
234 NEXT ID
236 LPRINT V$ :LPRINT
238 STOP
900 REM ---------- RIGHT HAND SIDE EVALUATION
902 D(1)=77.27*(Y(2)+Y(1)*(1-8.375E-06*Y(1)-Y(2)))
904 D(2)=(Y(3)-(1+Y(1))*Y(2))/77.27
906 D(3)=.161*(Y(1)-Y(3))
908 RETURN
```

In addition to the solution, the number of accepted steps (ip) and the number of back steps or repeated steps (ir) are also printed to show how the step size control works.

```
OREGONATOR MODEL BY SEMI-IMPLICIT RUNGE-KUTTA M., TOLERANCE= .001

----------------------------------------------------------
 TIME        y(1)         y(2)          y(3)      ip   ir
----------------------------------------------------------
  0.0       4.00000      1.33139       2.85235
  1.0       4.52980      1.28090       3.06099     5    0
  2.0       5.35444      1.22638       3.33716     4    0
  3.0       6.93755      1.16312       3.74244     4    0
  4.0      12.43307      1.07232       4.52226     6    1
  5.0  116764.80000      0.02421    2839.26000    58    5
  6.0   97271.55000      0.18801   18304.12000    16    0
 10.0       1.00214    785.15810   21132.22000   183    3
100.0       1.00368    273.45220       1.01394   224    0
200.0       1.05098     20.54734       1.04376    31    2
300.0       3.13234      1.46751       2.44608    30    1
302.9       4.01773      1.32940       2.85972     6    0
----------------------------------------------------------
```

<u>Exercise</u>

□ Try to solve the Oregonator model using a non - stiff integrator as the module M70 . Comment on the step size needed for a reasonable accuracy.

5.3 SENSITIVITY ANALYSIS

In this section we consider the parametrized vector differential equation

$$\frac{d}{dt} y(t,p) = f\left[y(t,p),p\right] \quad , \quad y(0) = y^{o}(p) \quad , \qquad (5.37)$$

where p denotes the np-vector of parameters. The vector of sensitivity coefficients to the parameter p_j is defined by

$$s_j(t,p) = \frac{\partial y(t,p)}{\partial p_j} . \tag{5.38}$$

These partial derivatives provide a lot of information (ref. 10). They show how parameter perturbations (e.g., uncertainties in parameter values) affect the solution. Identifying the unimportant parameters the analysis may help to simplify the model. Sensitivities are also needed by efficient parameter estimation procedures of the Gauss - Newton type. Since the solution $y(t,p)$ is rarely available in analytic form, calculation of the coefficients $s_j(t,p)$ is not easy. The simplest method is to perturb the parameter p_j , solve the differential equation with the modified parameter set and estimate the partial derivatives by divided differences. This "brute force" approach is not only time consuming (i.e., one has to solve np+1 sets of ny differential equations), but may be rather unreliable due to the roundoff errors. A much better approach is solving the sensitivity equations

$$\frac{d}{dt}s_j(t,p) = J\left[y(t,p),p\right]s_j(t,p) + \frac{\partial}{\partial p_j}f\left[y(t,p),p\right] , \tag{5.39}$$

where the i,j-th element of the Jacobian is given by

$$\left[J(y,p)\right]_{ij} = \frac{\partial}{\partial y_j}f_i(y,p) . \tag{5.40}$$

The sensitivity equations (5.39) are derived by differentiating (5.37) with respect to p_j , and changing the order of differentiation on the left hand side. The initial values to (5.39) are given by

$$s_j(0,p) = \frac{\partial}{\partial p_j} y^0(p) . \tag{5.41}$$

The sensitivity equations (5.39) can be solved simultaneously with the original equations (5.37). Although the special structure of this extended system of differential equations enables one to devise more efficient special methods (see, for example, refs. 11-13), in the following example we solve the equations using the general purpose integrator module M72 . The straightforward method not making use of the special structure of the sensitivity equations is called direct method of sensitivity analysis.

Example 5.3 Parameter sensitivities in the microbial growth model

In order to discuss the practical identifiability of the model studied in Examples 5.1.1 and 5.1.2, Holmberg (ref. 3) computed the sensitivities of the microorganism concentrations y_1 and substrate concentration y_2 with respect to the parameters V_m, K_s, K_d and Y. To repeat the computations, we need the partial derivatives

$$\mathbf{J} = \begin{bmatrix} \dfrac{V_m y_2}{K_s + y_2} - K_d & \dfrac{V_m K_s y_1}{(K_s + y_2)^2} \\ -\dfrac{1}{Y}\dfrac{V_m y_2}{K_s + y_2} & -\dfrac{1}{Y}\dfrac{V_m K_s y_1}{(K_s + y_2)^2} \end{bmatrix}, \qquad (5.42)$$

and

$$\frac{\partial f}{\partial p} = \begin{bmatrix} \dfrac{y_1 y_2}{K_s + y_2} & -\dfrac{V_m y_1 y_2}{(K_s + y_2)^2} & -y_2 & 0 \\ -\dfrac{1}{Y}\dfrac{y_1 y_2}{K_s + y_2} & \dfrac{1}{Y}\dfrac{V_m y_1 y_2}{(K_s + y_2)^2} & 0 & \dfrac{V_m y_1 y_2}{Y^2(K_s + y_2)} \end{bmatrix}. \qquad (5.43)$$

The initial values are $s_j(0,p) = 0$, $j = 1, 2, 3$ and 4 . (Note that the initial values of the concentrations y_1 and y_2 do not depend upon the parameters investigated.) To solve the extended system of differential equations the following main program is used:

```
100 REM ------------------------------------------------------------
102 REM EX. 5.3. SENSITIVITY ANALYSIS OF A MICROBIAL GROWTH PROCESS
104 REM MERGE M14,M15,M72
106 REM ---------- NUMBER OF TIME POINTS AND TIME POINTS
108 DATA 11,0,1,2,3,4,5,6,7,8,9,10
110 READ NT
112 DIM TW(NT)
114 FOR I=1 TO NT :READ TW(I) :NEXT I
200 REM ---------- PARAMETERS
202 VM=.5 :KS=3 :YY=.6 :KD=.05
204 REM ---------- PROBLEM SIZE
206 N=10
208 DIM Y(N),D(N),E(N,N),A(N,N),R(N),YO(N),YL(N),X(N)
210 DIM R1(N),R2(N),R3(N),R4(N)
212 REM ---------- INITIAL VALUES, FIRST H, ACCURACY
214 T=TW(1) :Y(1)=1 :Y(2)=30 :H=.1 :EP=.001 :IM=1000
216 V$=STRING$(56,"-")
218 A$="###.#   y1   ###.###  ###.###  ###.###  ###.###  ###.###"
220 B$="        y2   ###.###  ###.###  ###.###  ###.###  ###.###"
222 LPRINT "SEMI - LOGARITHMIC (dYi/dlnPi) SENSITIVITY MATRIX"
224 LPRINT :LPRINT V$
226 LPRINT "TIME,h   CONCENTRATION        PARAMETER SENSITIVITY"
228 LPRINT "                 g/l       Vm       Ks       Kd       Y"
230 LPRINT V$
232 LPRINT USING A$;T,Y(1),0,0,0,0
234 LPRINT USING B$; Y(2),0,0,0,0
236 REM ---------- CALL SOLUTION MODULE FOR EACH TIME POINT
238 FOR ID=2 TO NT
240  TE=TW(ID) :GOSUB 7200
242  LPRINT USING A$;T,Y(1),VM*Y(3),KS*Y(5),KD*Y(7),YY*Y(9)
244  LPRINT USING B$; Y(2),VM*Y(4),KS*Y(6),KD*Y(8),YY*Y(10)
246 NEXT ID
248 LPRINT V$ :LPRINT
250 STOP
```

```
900 REM ---------- RIGHT HAND SIDE EVALUATION
902  M0=KS+Y(2)   :M1=Y(2)/M0  :M2=M1/M0     :M3=KS/M0/M0
904  M4=VM*M1-KD  :M5=VM*M3*Y(1) :M6=-VM/YY*M1 :M7=-VM/YY*M3*Y(1)
906 REM - ORIGINAL EQUATIONS
908    D(1) = VM*M1*Y(1)-KD*Y(1)
910    D(2) =-VM/YY*M1*Y(1)
912 REM - SENSITIVITY EQUATIONS WITH RESPECT TO Vm
914    D(3) = M1*Y(1)         +M4*Y(3)+M5*Y(4)
916    D(4) =-VM/YY*M1*Y(1)    +M6*Y(3)+M7*Y(4)
918 REM - SENSITIVITY EQUATIONS WITH RESPECT TO Ks
920    D(5) =-VM*M2*Y(1)       +M4*Y(5)+M5*Y(6)
922    D(6) = VM/YY*M2*Y(1)    +M6*Y(5)+M7*Y(6)
924 REM - SENSITIVITY EQUATIONS WITH RESPECT TO Kd
926    D(7) =-Y(1)          +M4*Y(7)+M5*Y(8)
28    D(8) =          +M6*Y(7)+M7*Y(8)
930 REM - SENSITIVITY EQUATIONS WITH RESPECT TO Y
932    D(9) =          +M4*Y(9)+M5*Y(10)
934    D(10)= VM/YY/YY*M1*Y(1) +M6*Y(9)+M7*Y(10)
936 RETURN
```

Instead of the sensitivities $s_j(t,p)$, in most applications we use the vectors of semi-logarithmic or normalized sensitivities, defined by

$$\partial y/\partial \log p_j = p_j \partial y/\partial p_j \ . \qquad (5.44)$$

The last four columns of the following output list the matrix **S** of semi-logarithmic sensitivities consisting of $ny \times nt = 22$ rows and $np = 4$ columns. This matrix is called normalized sensitivity matrix.

```
SEMI - LOGARITHMIC (dYi/dlogPj) SENSITIVITY MATRIX

-------------------------------------------------------
TIME,h  CONCENTRATION        PARAMETER SENSITIVITY
              g/l        Vm       Ks       Kd       Y
-------------------------------------------------------
  0.0   y1    1.000    0.000    0.000    0.000    0.000
        y2   30.000    0.000    0.000    0.000    0.000
  1.0   y1    1.498    0.679   -0.063   -0.075    0.001
        y2   29.068   -0.691    0.107    0.025    0.931
  2.0   y1    2.239    2.023   -0.190   -0.224    0.007
        y2   27.678   -2.343    0.328    0.131    2.310
  3.0   y1    3.339    4.496   -0.431   -0.499    0.029
        y2   25.616   -5.695    0.756    0.389    4.335
  4.0   y1    4.955    8.790   -0.872   -0.982    0.101
        y2   22.581  -11.871    1.543    0.913    7.245
  5.0   y1    7.290   15.770   -1.645   -1.782    0.333
        y2   18.185  -22.309    2.933    1.854   11.237
  6.0   y1   10.524   25.672   -2.900   -2.971    1.156
        y2   12.060  -37.461    5.211    3.295   15.936
  7.0   y1   14.385   31.820   -4.093   -4.010    4.864
        y2    4.586  -46.466    7.499    4.289   17.020
  8.0   y1   16.203    8.872   -0.544   -2.511   15.644
        y2    0.252   -7.873    1.812    0.769    2.525
  9.0   y1   15.557    4.036    0.507   -2.728   16.462
        y2    0.003   -0.123    0.040    0.013    0.035
 10.0   y1   14.800    3.770    0.505   -3.328   15.681
        y2    0.000   -0.002    0.001    0.000    0.001
-------------------------------------------------------
```

According to (5.44) the semi-logarithmic sensitivity coefficients show the local change in the solutions when the given parameter is perturbed by unity on the logarithmic scale and are invariant under the scaling of the parameters.

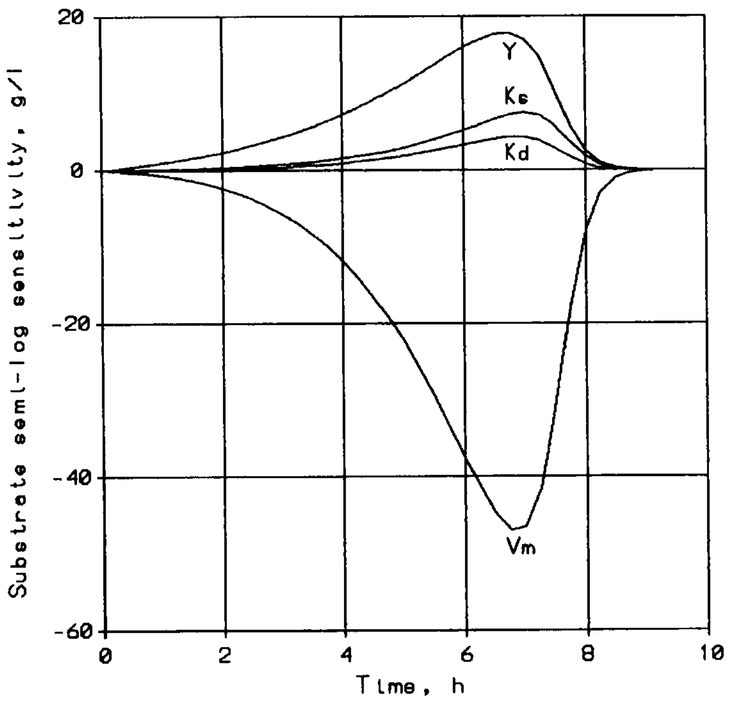


Fig. 5.3. Semi-logarithmic sensitivity of the substrate with respect to the parameters V_m, K_s, K_d and Y

As seen from Fig. 5.3, the substrate concentration is most sensitive to the parameters around $t = 7$ hours. It is therefore advantageous to select more observation points in this region when designing identification experiments (see Section 3.10.2). The sensitivity functions, especially with respect to K_s and K_d, seem to be proportional to each other, and the near-linear dependence of the columns in the Jacobian matrix may lead to ill-conditioned parameter estimation problem. Principal component analysis of the matrix S^TS is a powerful help in uncovering such parameter dependences. The approach will be discussed in Section 5.8.1.

5.4 QUASI STEADY STATE APPROXIMATION

The quasi steady state approximation is a powerful method of transforming systems of very stiff differential equations into non-stiff problems. It is the most important, although somewhat contradictive technique in chemical kinetics. Before a general discussion we present an example where the approximation certainly applies.

Example 5.4A Detailed model of the fumarase reaction

The basic mechanism of enzyme reactions is

$$E + S \underset{k_2}{\overset{k_1}{\rightleftarrows}} ES \underset{k_4}{\overset{k_3}{\rightleftarrows}} E + P \qquad (5.45)$$

where E, S, ES and P denote the enzyme, the substrate, the intermediate enzyme-substrate complex and the product, respectively. The rate expressions are mass action type with rate coefficients k_1, k_2, k_3 and k_4, resulting in the kinetic differential equations

$$\frac{d}{dt}[E] = - k_1[E][S] + k_2[ES] + k_3[ES] - k_4[E][P] \qquad (5.46)$$

$$\frac{d}{dt}[S] = - k_1[E][S] + k_2[ES] \qquad (5.47)$$

$$\frac{d}{dt}[ES] = k_1[E][S] - k_2[ES] - k_3[ES] - k_4[E][P] \qquad (5.48)$$

$$\frac{d}{dt}[P] = k_3[ES] - k_4[E][P] \qquad (5.49)$$

where the brackets denote concentrations of the species. If the substrate is fumarate and the enzyme is fumarase, at $T = 25$ °C and pH = 7 the rate constants are $k_1 = 140\times10^6$ l mol^{-1} s^{-1} , $k_2 = 200$ s^{-1} , $k_3 = 330$ s^{-1} and $k_4 = 51\times10^6$ l mol^{-1} s^{-1} (ref. 14). We solve equations (5.46) - (5.49) up to the reaction time $t = 120$ s with the initial concentrations $[E]^o = 2\times10^{-9}$ mol l^{-1} and $[S]^o = 20\times10^{-6}$ mol l^{-1} . The initial concentrations of the enzyme-substrate and the product are zero. Since the system is closed, due to the balance equations

$$[E] = [E]^o - [ES] \qquad (5.50)$$

and

$$[S] = [S]^o - [ES] - [P] \qquad (5.51)$$

it is sufficient to consider the two linearly independent differential equations

$$\frac{d}{dt}[ES] = k_1([E]^o - [ES])([S]^o - [ES] - [P]) - (k_2 + k_3)[ES] + k_4([E]^o - [ES])[P] \qquad (5.52)$$

and

$$\frac{d}{dt}[P] = k_3[ES] - k_4([E]^o - [ES])[P] \ . \qquad (5.53)$$

With the initial step size H = 0.1 s and threshold EP = 0.0001, the module M72 gives the following results:

```
ENZYME CATALYSIS - DETAILED MECHANISM

k1= 1.4E+08   k2= 200
k3= 330       k4= 5.1E+07

 TIME,s    ES,mol/l      P,mol/l    ip    ir
---------------------------------------------
   0.0    0.0000E+00   0.0000E+00
   6.0    0.1653E-08   0.3136E-05  41     9
  12.0    0.1622E-08   0.5872E-05   4     0
  18.0    0.1592E-08   0.8203E-05   3     0
  24.0    0.1563E-08   0.1014E-04   3     0
  30.0    0.1537E-08   0.1170E-04   3     0
  36.0    0.1514E-08   0.1292E-04   3     0
  42.0    0.1494E-08   0.1386E-04   3     0
  48.0    0.1479E-08   0.1457E-04   2     0
  54.0    0.1467E-08   0.1509E-04   2     0
  60.0    0.1457E-08   0.1546E-04   2     0
  66.0    0.1451E-08   0.1573E-04   2     0
  72.0    0.1446E-08   0.1593E-04   1     0
  78.0    0.1442E-08   0.1606E-04   1     0
  84.0    0.1440E-08   0.1616E-04   1     0
  90.0    0.1438E-08   0.1623E-04   1     0
  96.0    0.1437E-08   0.1627E-04   1     0
 102.0    0.1436E-08   0.1631E-04   1     0
 108.0    0.1435E-08   0.1633E-04   1     0
 114.0    0.1435E-08   0.1634E-04   1     0
 120.0    0.1434E-08   0.1636E-04   1     0
---------------------------------------------
```

The enzyme – substrate complex concentration reaches its maximum value in a very short time, and decays very slowly afterwards. To explain this special behavior of the concentration [ES], write its kinetic equation in the form $d[ES]/dt = r_p - r_c$, where r_p and r_c denote the total production and consumption rates of the enzyme – substrate, respectively. Since r_p and r_c are very large, any deviation $r_p - r_c \neq 0$ yields a quick change in [ES]. Thus [ES] quickly reaches its value where $r_p = r_c$. Therefore, it is a good approximation to assume that $r_p = r_c$ at every instant of time, i.e., to find $[ES]_{SS}$ for which the right hand side of (5.52) is zero :

$$0 = k_1([E]^o - [ES])([S]^o - [ES] - [P]) - (k_2 + k_3)[ES] + \\ + k_4([E]^o - [ES])[P] \;. \qquad (5.54)$$

Replacing (5.52) by the algebraic equation (5.54) we can solve (5.54) for $[ES]_{ss}$, the quasi steady state concentration of the enzyme - substrate. The solution depends on the actual value $[P]$, therefore $[ES]_{ss}$ is not at all constant, and hence the usual equation

$$\frac{d}{dt}[ES] \approx 0 \qquad (5.55)$$

can be used only as a short hand notation for (5.54). The quasi steady state assumption simply means that $[ES]$ can be replaced by $[ES]_{ss}$ without any reasonable loss in accuracy.

As seen from the output of Example 5.4A, the solution of the system (5.50-53) is far from easy even for the stiff integrator M72 . In the following we solve the same problem applying the quasi steady state approximation.

<u>Example 5.4B</u> Quasi steady state model of the fumarase reaction

From equation (5.54)

$$[ES]_{ss} = [E]^o \frac{k_1[S]^o + (k_4 - k_1)[P]}{k_2 + k_3 + k_1[S]^o + (k_4 - k_1)[P]} \;. \qquad (5.56)$$

Substituting this expression into (5.53), the kinetics is described by the single differential equation

$$\frac{d}{dt}[P] = \frac{k_1k_3[E]^o[S]^o}{k_2 + k_3 + k_1[S]^o} \times \frac{1 - \dfrac{k_1k_3 + k_2k_4}{k_1k_3[S]^o}[P]}{1 + \dfrac{k_4 - k_1}{k_2 + k_3 + k_1[S]^o}[P]} \qquad (5.57)$$

usually written in the form

$$\frac{d}{dt}[P] = \frac{(V_S/K_S)([S]^o - [P]) - (V_P/K_P)[P]}{1 + ([S]^o - [P])/K_S + [P]/K_P} \qquad (5.58)$$

where $V_S = k_3[E]^o$, $V_P = k_2[E]^o$, $K_S = (k_2 + k_3)/k_1$ and $K_P = (k_2 + k_3)/k_4$ are the Michaelis - Menten parameters first introduced in Example 2.5.1.

Selecting the same initial step size and threshold as in Example 5.4A, we solve the differential equation (5.58). In order to compare the results, $[ES]_{ss}$ computed form (5.56) is also listed on the following output.

```
ENZYME CATALYSIS   -   MICHAELIS - MENTEN RATE EXPRESSION

Vs= 6.6E-07   Ks= 3.785715E-06
Vp= .0000004  Kp= 1.039216E-05

 TIME,s    ES,mol/l      P,mol/l     ip    ir
---------------------------------------------
  0.0     0.1682E-08   0.0000E+00
  6.0     0.1653E-08   0.3136E-05   9     3
 12.0     0.1622E-08   0.5873E-05   2     0
 18.0     0.1592E-08   0.8204E-05   2     0
 24.0     0.1563E-08   0.1014E-04   2     0
 30.0     0.1537E-08   0.1170E-04   1     0
 36.0     0.1514E-08   0.1292E-04   1     0
 42.0     0.1494E-08   0.1386E-04   1     0
 48.0     0.1479E-08   0.1457E-04   1     0
 54.0     0.1467E-08   0.1509E-04   1     0
 60.0     0.1457E-08   0.1546E-04   1     0
 66.0     0.1451E-08   0.1573E-04   1     0
 72.0     0.1446E-08   0.1593E-04   1     0
 78.0     0.1442E-08   0.1606E-04   1     0
 84.0     0.1440E-08   0.1616E-04   1     0
 90.0     0.1438E-08   0.1623E-04   1     0
 96.0     0.1437E-08   0.1627E-04   1     0
102.0     0.1436E-08   0.1631E-04   1     0
108.0     0.1435E-08   0.1633E-04   1     0
114.0     0.1435E-08   0.1635E-04   1     0
120.0     0.1434E-08   0.1636E-04   1     0
---------------------------------------------
```

As seen from the output, the number of steps required is significantly reduced. Nevertheless, apart from a very short induction period, the solution essentially agrees with that of the detailed model.

Exercise

- Solve Example 5.4B using the program module M70 . Try to solve Example 5.4A with the same method. Comment on the differences in accuracy, required step size, etc.

5.5 ESTIMATION OF PARAMETERS IN DIFFERENTIAL EQUATIONS

In this section we deal with estimating the parameters p in the dynamical model of the form (5.37). As we noticed, methods of Chapter 3 directly apply to this problem only if the solution of the differential equation is available in analytical form. Otherwise one can follow the same algorithms, but solving differential equations numerically whenever the computed responses are needed. The partial derivations required by the Gauss - Newton type algorithms can be obtained by solving the sensitivity equations. While this indirect method is very general (ref. 15), it is so time consuming that may be not feasible on a personal computer.

The direct integral approach to parameter estimation we will discuss here applies only with all variables $y_1, y_2, \ldots, y_{ny}$ observed, but then it offers a more efficient alternative. Let $t_1, t_2, \ldots, t_{nm}$ denote the sample time points with $t_1 = 0$. The unknown parameters $\mathbf{p}$ are to be estimated from the set of observations $\left\{ (t_i, \tilde{y}_i) , i = 1, 2, \ldots, nm \right\}$. The basic idea of the direct integral method (refs. 16-17) is transforming the vector differential equation (5.37) into the equivalent integral equation

$$y(t_i,\mathbf{p}) = y^0(\mathbf{p}) + \int_0^{t_i} f\left[y(t,\mathbf{p}),\mathbf{p}\right] dt , \qquad (5.59)$$

and approximating the integrand by cubic spline functions that interpolate the points $\left\{ \left[t_i, f(\tilde{y}_i,\mathbf{p})\right] , i = 1, 2, \ldots, nm \right\}$. Evaluating the integrals at the current estimate of the parameters $\mathbf{p}$ converts the problem into an algebraic one which can be solved by the nonlinear least squares algorithm of Section 3.3.

Let $S_p^f(t)$ denote the ny-vector of natural cubic splines interpolating the values $\left\{ \left[t_i, f(\tilde{y}_i,\mathbf{p})\right] , i = 1, 2, \ldots, nm \right\}$. Introducing the vector

$$F(\mathbf{p}) = \begin{bmatrix} y^0(\mathbf{p}) \\ y^0(\mathbf{p}) + \int_0^{t_2} S_p^f(t)\, dt \\ \vdots \\ y^0(\mathbf{p}) + \int_0^{t_{nm}} S_p^f(t)\, dt \end{bmatrix} \qquad (5.60)$$

of nm×ny elements we can write the objective function of the direct integral method in the usual form (3.39).

The Jacobian matrix defined in (3.41) can be easily computed by the same interpolation technique. The idea is to differentiate (3.60) with respect to the parameters changing the order of differentiation and spline integration.

Since all the involved operations are linear we obtain

$$\frac{\partial}{\partial p_j} \int_0^{t_i} S_p^f(t)\,dt = \int_0^{t_i} S_p^{f_j}(t)\,dt \tag{5.61}$$

where $S_p^{f_j}(t)$ is the ny-vector of natural cubic splines interpolating the values $\left\{ \left[t_i, \partial f(\tilde{y}_i,p)/\partial p_j\right] , i = 1, 2, \ldots, nm \right\}$. Thus the Jacobian matrix $J(\mathbf{p})$ of (5.60) is given by

$$\begin{bmatrix} \partial y^o(\mathbf{p})/\partial p_1 & \ldots & \partial y^o(\mathbf{p})/\partial p_{np} \\ \partial y^o(\mathbf{p})/\partial p_1 + \int_0^{t_2} S_p^{f_1}(t)\,dt & \ldots & \partial y^o(\mathbf{p})/\partial p_{np} + \int_0^{t_2} S_p^{f_{np}}(t)\,dt \\ \vdots & & \\ \partial y^o(\mathbf{p})/\partial p_1 + \int_0^{t_{nm}} S_p^{f_1}(t)\,dt & \ldots & \partial y^o(\mathbf{p})/\partial p_{np} + \int_0^{t_{nm}} S_p^{f_{np}}(t)\,dt \end{bmatrix} \tag{5.62}$$

The algorithm of the direct integral method is as follows.

(i) Select a first guess of the parameters and compute the values $f(\tilde{y}_i,\mathbf{p})$ and $\partial f(\tilde{y}_i,\mathbf{p})/\partial p_j$,

(ii) Determine the interpolating splines and compute the integrals involved in (5.60) and (5.62)

(iii) Knowing the vector $F(\mathbf{p})$ and matrix $J(\mathbf{p})$ compute the Gauss - Newton - Marquardt step as discussed in Section 3.3

(iv) Return to (ii) until convergence.

Completing the procedure we obtain only the approximation (5.60) of the solution of the differential equations (5.37). To see the real goodness-of-fit we must solve the differential equations (5.37) numerically with the parameter estimates $\hat{p}$ and initial conditions $y^{o}(\hat{p})$, but only once.

Since spline interpolation and integration is mucht faster than solving the sensitivity equations and the original differential equations, the direct method is superior to the indirect one in terms of numerical efficiency, whenever it is feasible.

In spite of its simplicity the direct integral method has relatively good statistical properties and it may be even superior to the traditional indirect approach in ill-conditioned estimation problems (ref. 18). Good performance, however, can be expected only if the sampling is sufficiently dense and the measurement errors are moderate, since otherwise spline interpolation may lead to severely biased estimates.

The following program module is a modification of the nonlinear least squares module M45. Because of spline interpolation and differential equation solution involved it is rather lengthy.

Program module M75

```
7500 REM ###################################################
7502 REM #    ESTIMATION OF PARAMETERS IN DIFFERENTIAL      #
7504 REM #      EQUATIONS BY DIRECT INTEGRAL METHOD         #
7506 REM #EXTENSION OF THE HIMMELBLAU-JONES-BISCHOFF METHOD#
7508 REM ###################################################
7510 REM INPUT:
7512 REM     NM      NUMBER OF SAMPLE POINTS
7514 REM     NY      NUMBER OF DEPENDENT VARIABLES
7516 REM     NP      NUMBER OF PARAMETERS
7518 REM    T(NM)    SAMPLE TIME POINTS
7520 REM   V(NM,NY)  TABLE OF OBSERVATIONS
7522 REM     WI      IDENTIFIER OF WEIGHTING OPTIONS
7524 REM              0  IDENTICAL WEIGHTS ( W(I,I)=1, W(I,J)=0 )
7526 REM              1  RELATIVE WEIGHTS ( W(I,I)=CONST/V(M,I)^2,W(I,J)=0)
7528 REM              2  USER-SPECIFIED WEIGHTS
7530 REM                 GIVEN BY FURTHER INPUT AS
7532 REM   W(NY,NY)  MATRIX OF WEIGHTING COEFFICIENTS ( ONLY FOR WI=2 )
7534 REM    P(NP)    INITIAL PARAMETER ESTIMATES
7536 REM     EP      THRESHOLD ON RELATIVE STEP LENGTH
7538 REM     IM      MAXIMUM NUMBER OF ITERATIONS
7540 REM OUTPUT:
7542 REM     ER      STATUS FLAG
7544 REM                0  SUCCESSFUL ESTIMATION
7546 REM                1  REQUIRED THRESHOLD NOT ATTAINED
7548 REM    P(NP)    PARAMETER ESTIMATES
7550 REM    .....    FURTHER RESULTS ARE PRINTED IN THE MODULE
7552 REM USER-SUPPLIED SUBROUTINES:
7554 REM   FROM LINE 900:
7556 REM             Y(1,...,ny) AND P(1,...,np)  -->  D(1,...,ny)
7558 REM              ( EVALUATE RHS OF DIFF. EQUATIONS )
7560 REM
7562 REM   FROM LINE 800:
7564 REM             P(1,...,np)  --> YI(1,...,ny)
7566 REM              ( EVALUATES INITIAL CONDITIONS FOR VARIABLES )
```

```
7568 REM AUXILIARY ARRAYS:
7570 REM  A([NP MAX NY],[NP MAX NY]),C(NP,NP),U(NP,NP),B(NP),DE(NP),G(NY,NP)
7572 REM  F(NM),Z(NM),S(4,NM),SF(NM,NY),SG(NM,NY,NP),YG(NY,NP),W(NY,NY)
7574 REM  E(NY,NY),R(NY),YO(NY),YL(NY),R1(NY),R2(NY),R3(NY),R4(NY),X(NY)
7576 REM MODULES CALLED: M14,M15,M16,M10,M41,M63,M72
7578 REM ---------- SPLINE KNOTS
7580 FOR M=1 TO NM :Z(M)=T(M) :NEXT M
7582 REM ---------- GENERATE WEIGHTING COEFFICIENTS
7584 IF WI<>0 THEN 7588
7586 FOR I=1 TO NY :FOR J=1 TO NY :W(I,J)=-(I=J) :NEXT J :NEXT I
7588 EI=0 :ES=0 :PM=.01
7590 REM ---------- SUM OF SQUARES AT STARTING POINT
7592 GOSUB 7816 :GOSUB 7772
7594 REM ---------- START OF ITERATION
7596 LPRINT :LPRINT "STARTING POINT";TAB(25);"SUM SQ=";F :LPRINT
7598 FOR K=1 TO NP :LPRINT TAB(25);"P(";K;")=";P(K) :NEXT K
7600 FOR IT=1 TO IM
7602  FOR K=1 TO NP :U(K,0)=P(K) :NEXT K :FR=F
7604  REM ---------- COMPUTE T'WT AND WT'Y
7606  FOR K=1 TO NP :B(K)=0 :FOR L=1 TO K :C(K,L)=0 :NEXT L :NEXT K
7608  GOSUB 7842
7610  FOR M=1 TO NM
7612   IF WI=1 THEN GOSUB 7792
7614   GOSUB 7804
7616   FOR K=1 TO NP
7618 FOR L=1 TO K
7620  A=0
7622  FOR I=1 TO NY:FOR J=1 TO NY
7624   A=A+W(I,J)*G(I,L)*G(J,K)*P(L)*P(K)
7626  NEXT J :NEXT I :C(K,L)=C(K,L)+A
7628 NEXT L
7630 A=0
7632 FOR I=1 TO NY:FOR J=1 TO NY
7634  A=A+W(I,J)*G(J,K)*(V(M,I)-Y(I))*P(K)
7636 NEXT J :NEXT I :B(K)=B(K)+A
7638   NEXT K
7640  NEXT M
7642  REM ---------- NORMALIZE CROSS PRODUCT MATRIX
7644  TR=0 :FOR I=1 TO NP :C(I,0)=C(I,I) :TR=TR+C(I,I) :NEXT I
7646  TR=TR/NP/1000
7648  FOR I=1 TO NP
7650   IF C(I,0)<=TR THEN C(I,0)=1 ELSE C(I,0)=SQR(C(I,0))
7652  NEXT I
7654  FOR I=1 TO NP :FOR J=1 TO I
7656   U(I,J)=C(I,J) :C(I,J)=C(I,J)/C(I,0)/C(J,0)
7658  NEXT J :NEXT I
7660  REM ---------- MARQUARDT'S COMPROMISE
7662  FOR I=1 TO NP
7664   FOR J=1 TO I-1 :A(I,J)=C(I,J) :NEXT J
7666   A(I,I)=C(I,I)+PM
7668  NEXT I
7670  REM ---------- MATRIX INVERSION
7672  ER=0 :N=NP :GOSUB 1600 :IF ER=1 THEN 7720
7674  REM ---------- COMPUTE STEP
7676  FOR I=1 TO NP
7678   D=0 :FOR J=1 TO NP :D=D+A(I,J)/C(J,0)*B(J) :NEXT J :D(I)=D/C(I,0)
7680  NEXT I
```

```
7682  REM ---------- CHECK SIGN AND REDUCE STEP IF NEEDED
7684  SL=0 :XI=1
7686  FOR I=1 TO NP
7688   IF XI*D(I)<=-.95 THEN XI=-.95/D(I)
7690   SL=SL+D(I)*D(I)
7692  NEXT I :SL=SQR(SL)*XI
7694  REM ---------- NEW ESTIMATES
7696  FOR I=1 TO NP :P(I)=U(I,0)*(1+XI*D(I)) :NEXT I
7698  GOSUB 7816 :GOSUB 7772
7700  REM ---------- PRINT ITERATION STEP
7702  F$="#.#^^^^" :LPRINT
7704  LPRINT "IT=";IT;TAB(10);"PM="; :LPRINT USING F$;PM;
7706  LPRINT TAB(25);"SUM SQ=";F;TAB(50);"SL=";SL :LPRINT
7708  IF F>=FR THEN 7712
7710  FOR K=1 TO NP :LPRINT TAB(25);"P(";K;")=";P(K) :NEXT K
7712  REM ---------- END OF PRINT
7714  IF SL<=EP THEN EI=0 :GOTO 7728
7716  REM ---------- MARQUARDT' PARAMETER
7718  IF F<=FR THEN 7722
7720  PM=10*PM :GOTO 7660
7722  PM=PM/10 :IF PM<.000001 THEN PM=.000001
7724 NEXT IT
7726 EI=1
7728 IF FR<F THEN  FOR I=1 TO NP :P(I)=U(I,0) :NEXT I
7730 REM ----------- SOLVE DIFFERENTIAL EQUATIONS
7732 GOSUB 7900
7734 REM ----------- COMPUTE EXAXT SUM OF SQUARES
7736 GOSUB 7772
7738 NF=NM*NY-NP :SE=SQR(F/NF)
7740 REM ---------- STANDARD ERROR AND CORRELATION MATRIX OF PARAMETERS
7742 FOR I=1 TO NP :FOR J=1 TO I
7744  A(I,J)=C(I,J)
7746 NEXT J:NEXT I
7748 N=NP :GOSUB 1600 :IF ER=1 THEN ES=1 :GOTO 7764 ELSE ES=0
7750 FOR I=1 TO NP
7752  B(I)=SQR(F/NF*A(I,I)/C(I,0)/C(I,0))
7754  C(0,I)=SQR(A(I,I))
7756 NEXT I
7758 FOR I=1 TO NP :FOR J=1 TO NP
7760  C(I,J)=INT(1000*A(I,J)/C(0,I)/C(0,J)+.5)/1000
7762 NEXT J:NEXT I
7764 REM ---------- PRINCIPAL COMPONENT ANALYSIS
7766 FOR I=1 TO NP :FOR J=1 TO I :A(I,J)=U(I,J) :NEXT J :NEXT I
7768 N=NP :GOSUB 1800
7770 GOTO 7920
7772 REM ---------- SUM OF SQUARES
7774 F=0
7776 FOR M=1 TO NM
7778  IF WI=1 THEN GOSUB 7792
7780  FOR I=1 TO NY :Y(I)=SF(M,I) :NEXT I
7782  FOR I=1 TO NY :FOR J=1 TO NY
7784   F=F+W(I,J)*(V(M,I)-Y(I))^2
7786  NEXT J :NEXT I
7788 NEXT M
7790 RETURN
```

```
7792 REM ---------- RELATIVE WEIGHTING
7794 FOR I=1 TO NY
7796  Y=ABS(V(M,I)) :IF Y<1E-15 THEN Y=1E-15
7798  W(I,I)=1/Y/Y
7800 NEXT I
7802 RETURN
7804 REM ---------- JACOBI MATRIX AND RESPONSE
7806 FOR I=1 TO NY :FOR J=1 TO NP
7808  G(I,J)=SG(M,I,J)
7810 NEXT J: NEXT I
7812 FOR I=1 TO NY :Y(I)=SF(M,I) :NEXT I
7814 RETURN
7816 REM ---------- DIRECT INTEGRAL RESPONSES
7818 GOSUB 800
7820 FOR M=1 TO NM
7822  FOR J=1 TO NY :Y(J)=V(M,J) :NEXT J
7824  GOSUB 900
7826  FOR J=1 TO NY :SF(M,J)=D(J) :NEXT J
7828 NEXT M
7830 FOR J0=1 TO NY
7832  FOR M=1 TO NM :F(M)=SF(M,J0) :NEXT M
7834  N=NM :EC=0 :GOSUB 6300
7836  FOR M=1 TO NM :SF(M,J0)=S(4,M)+YI(J0) :NEXT M
7838 NEXT J0
7840 RETURN
7842 REM ---------- DIRECT INTEGRAL JACOBI MATRIX - FIRST TIME POINT
7844  FOR J=1 TO NP
7846   DE=.001*ABS(P(J))+1E-10 :P(J)=P(J)+DE :GOSUB 800
7848   FOR I=1 TO NY :YG(I,J)=YI(I)/DE :NEXT I
7850   P(J)=P(J)-DE :DE(J)=DE
7852  NEXT J
7854  GOSUB 800
7856  FOR I=1 TO NY :FOR J=1 TO NP
7858   YG(I,J)=YG(I,J)-YI(I)/DE(J)
7860  NEXT J: NEXT I
7862 REM ---------- - INNER TIME POINT
7864 FOR M=1 TO NM
7866  FOR I=1 TO NY :Y(I)=V(M,I) :NEXT I
7868  FOR J=1 TO NP
7870   DE=.001*ABS(P(J))+.000001 :P(J)=P(J)+DE :GOSUB 900
7872   FOR I=1 TO NY :G(I,J)=D(I)/DE :NEXT I
7874   P(J)=P(J)-DE :DE(J)=DE
7876  NEXT J
7878  GOSUB 900
7880  FOR I=1 TO NY :FOR J=1 TO NP
7882   SG(M,I,J)=G(I,J)-D(I)/DE(J)
7884  NEXT J: NEXT I
7886 NEXT M
7888 FOR I0=1 TO NY :FOR J0=1 TO NP
7890  FOR M=1 TO NM :F(M)=SG(M,I0,J0) :NEXT M
7892  N=NM :EC=0 :GOSUB 6300
7894  FOR M=1 TO NM :SG(M,I0,J0)=S(4,M)+YG(I0,J0) :NEXT M
7896 NEXT J0 :NEXT I0
7898 RETURN
7900 REM ---------- SOLUTION OF DIFFERENTIAL EQUATIONS
7902 N=NY :IM=100 :H=(T(2)-T(1))/10
7904 GOSUB 800
7906 FOR J=1 TO NY :Y(J)=YI(J) :NEXT J
```

```
7908 FOR IG=2 TO NM
7910  T=T(IG-1) :TE=T(IG) :GOSUB 7200
7912  IF ER THEN LPRINT "ER=";ER,"ERROR IN DIFF. EQU. SOLUTION" :STOP
7914  FOR J=1 TO NY :SF(IG,J)=Y(J) :NEXT J
7916 NEXT IG
7918 RETURN
7920 REM ---------- PRINT RESULTS
7922 LPRINT :LPRINT
7924 LPRINT TAB(15);"ESTIMATION OF PARAMETERS IN DIFFERENTIAL"
7926 LPRINT TAB(17);"EQUATIONS BY DIRECT INTEGRAL METHOD"
7928 LPRINT :LPRINT :LPRINT
7930 LPRINT " NUMBER OF DEPENDENT VARIABLES ....... ";NY
7932 LPRINT " NUMBER OF PARAMETERS................. ";NP
7934 LPRINT " NUMBER OF TIME POINTS ............... ";NM
7936 LPRINT " OPTION OF WEIGHTING ................. ";WI;
7938 IF WI=0 THEN LPRINT "(IDENTICAL WEIGHTS)"
7940 IF WI=1 THEN LPRINT "(RELATIVE WEIGHTS)"
7942 IF WI=2 THEN LPRINT "(USER DEFINED WEIGHTS)"
7944 F$="#.######^^^^  " :LPRINT :LPRINT
7946 LPRINT " PRINCIPAL COMPONENT ANALYSIS OF NORMED CROSS PRODUCT MATRIX"
7948 LPRINT :LPRINT "EIGENVALUE";
7950 FOR I=1 TO NP :LPRINT TAB(10*I+5);" P(";I;") "; : NEXT I :LPRINT :LPRINT
7952 FOR I=1 TO NP
7954  LPRINT U(0,I),
7956  FOR J=1 TO NP :LPRINT USING "##.####   ";U(J,I); :NEXT J :LPRINT
7958 NEXT I
7960 LPRINT :LPRINT
7962 V$=STRING$(70,"-") :V1$=STRING$(55,"-")
7964 IF EI=1 THEN LPRINT " REQUIRED THRESHOLD NOT ATTAINED" :LPRINT :LPRINT
7966 IF ES=1 THEN LPRINT " SINGULAR CROSS PRODUCT MATRIX" :LPRINT :LPRINT
7968 FOR I=1 TO NY
7970  LPRINT :IF NY>1 THEN LPRINT "RESPONSE FUNCTION";I
7972  LPRINT V1$ :LPRINT "No"," Y MEAS"," Y COMP"," RESIDUAL" :LPRINT V1$
7974  FOR M=1 TO NM
7976   LPRINT M, :LPRINT USING F$;V(M,I),SF(M,I),V(M,I)-SF(M,I)
7978  NEXT M :LPRINT V1$
7980 NEXT I :LPRINT :LPRINT
7982 LPRINT " SUM OF SQUARES (VIA SOLUTION OF ODE). ";F
7984 LPRINT " DEGREES OF FREEDOM.................. ";NF
7986 IF WI=0 THEN LPRINT " STANDARD ERROR ...................... ";SE
7988 IF WI>0 THEN LPRINT " SIGMA FACTOR IN THE WEIGHTS ......... ";SE
7990 GOSUB 4100
7992 LPRINT " CRITICAL T-VALUE AT 95 % CONF. LEVEL  ";T
7994 LPRINT :LPRINT V$ :LPRINT "PARAMETER",
7996 IF ES=0 THEN LPRINT " ESTIMATE"," ST. ERR","LOWER BOUND","UPPER BOUND",
7998 LPRINT :LPRINT V$
8000 FOR I=1 TO NP
8002  LPRINT " P(";I;") ", :LPRINT USING F$;P(I),
8004  PB=ABS(B(I)*P(I))
8006  IF ES=0 THEN LPRINT USING F$;PB,P(I)-T*PB,P(I)+T*PB,
8008  LPRINT
8010 NEXT I
8012 LPRINT V$ :LPRINT
8014 IF ES=1 THEN 8038
8016 LPRINT " CORRELATION MATRIX OF PARAMETERS:"
8018 LPRINT
8020 FOR I=1 TO NP :LPRINT TAB(10*I);" P(";I;") "; :  NEXT I :LPRINT :LPRINT
```

```
8022 FOR I=1 TO NP
8024  LPRINT " P(";I;") ";
8026  FOR J=1 TO I
8028 LPRINT TAB(10*J);C(I,J);
8030  NEXT J :LPRINT
8032 NEXT I
8034 LPRINT :LPRINT
8036 ER=0 :IF EI=1 THEN ER=1
8038 RETURN
8040 REM ##################################################
```

The input data structure is very similar to the one in the module M45 . Two user routines are to be supplied. The first one starts at line 900 and evaluates the right hand sides of the differential equations. The second routine, starting at line 800, serves for computing the initial conditions at the current estimates of the parameters. If the initial estimates are parameter independent (we know them exactly), then this routine simply puts the known values into the variables YI(1), ..., YI(NY) . The required partial derivatives are generated using divided differences approximation. In order to ease the use of the module a very simple example is considered here.

Example 5.5 Fitting a Michaelis - Menten type kinetic model

Consider the simple model

$$\frac{dy}{dt} = - \frac{p_1 y}{p_2 + y} \tag{5.63}$$

with unknown initial condition

$$y(0) = p_3 \ . \tag{5.64}$$

The data listed in Table 5.2 are the concentrations of a drug in plasma and come from a test problem of the BMDP statistical program package (ref. 19).

Table 5.2
Observed drug concentration

No	Time, min t_i	Concentration, g/l $\tilde{y}_i$
1	0	24.44
2	23.6	19.44
3	49.1	15.56
4	74.5	10.56
5	80.0	9.07
6	100.0	6.85
7	125.5	4.07
8	144.3	1.67

To illustrate the robustness of the direct integral program module, we chose the starting estimates $p_1 = 1$, $p_2 = 1$ and $p_3 = 1$, although $p_3 = 24.44$ obviously is a better starting guess.

```
100 REM ------------------------------------------------------------
102 REM EX. 5.5 DIRECT INTEGRAL PARAMETER ESTIMATION
104 REM MERGE M14,M15,M16,M18,M41,M63,M72,M75
106 REM ---------- DATA
108 REM    NY NM NP
110 DATA    1, 8, 3
112 REM  (TIME AND CONCENTRATION)
114 DATA     0, 24.44
116 DATA  23.6, 19.44
118 DATA  49.1, 15.56
120 DATA  74.5, 10.56
122 DATA  80.0,  9.07
124 DATA 100.0,  6.85
126 DATA 125.5,  4.07
128 DATA 147.3,  1.67
200 REM ---------- READ DATA AND SPECIFY DIMENSIONS
202 READ NY,NM,NP
204 MX=NY :IF MX<NP THEN MX=NP
206 DIM  A(MX,MX),C(NP,NP),U(NP,NP),B(NP),DE(NP),G(NY,NP),W(NY,NY)
208 DIM  F(NM),Z(NM),S(4,NM),SF(NM,NY),SG(NM,NY,NP),YG(NY,NP)
210 DIM  E(NY,NY),R(NY),YO(NY),YL(NY),R1(NY),R2(NY),R3(NY),R4(NY),X(NY)
212 DIM  T(NM),V(NM,NY)
214 FOR M=1 TO NM
216  READ T(M) :FOR J=1 TO NY :READ V(M,J) :NEXT J
218 NEXT M
220 REM ---------- SET ITERATION CONTROL PARAMETERS AND CALL MODULE
222 P(1)=1 :P(2)=1 :P(3)=1
224 EP=.001 :IM=30 :WI=0
226 GOSUB 7500
228 STOP
800 REM ---------- INITIAL VALUE EVALUATION SUBROUTINE
802 YI(1)=P(3)
804 RETURN
900 REM ---------- RIGHT HAND SIDE EVALUATION SUBROUTINE
902 D(1)=-P(1)*Y(1)/(P(2)+Y(1))
904 RETURN
```

Before listing the output, recall that the objective function to be minimized is based on the approximate response (5.60). The minimum of this function is 1.0186002, whereas solving the differential equation (5.63) at the final estimate of the parameters gives the value 1.060729. The direct integral estimares are acceptable only if these two values do not significantly differ, see (ref. 18).

```
STARTING POINT          SUM SQ= 60070.19

                        P( 1 )= 1
                        P( 2 )= 1
                        P( 3 )= 1
```

```
IT= 1    PM=0.1E-01     SUM SQ= 719.6788          SL= 27.34466

                        P( 1 )= .3919189
                        P( 2 )= 2.950189
                        P( 3 )= 28.26825

IT= 2    PM=0.1E-02     SUM SQ= 9.465345          SL= .5924904

                        P( 1 )= .2471071
                        P( 2 )= 4.258754
                        P( 3 )= 24.49897

IT= 3    PM=0.1E-03     SUM SQ= 1.066295          SL= .2507575

                        P( 1 )= .2452795
                        P( 2 )= 5.326014
                        P( 3 )= 24.3834

IT= 4    PM=0.1E-04     SUM SQ= 1.018621          SL= .0472762

                        P( 1 )= .2474369
                        P( 2 )= 5.573408
                        P( 3 )= 24.38938

IT= 5    PM=0.1E-05     SUM SQ= 1.018602          SL= 2.199665E-03

                        P( 1 )= .2475825
                        P( 2 )= 5.58522
                        P( 3 )= 24.39008

IT= 6    PM=0.1E-05     SUM SQ= 1.018602          SL= 2.04832E-04



              ESTIMATION OF PARAMETERS IN DIFFERENTIAL
                EQUATIONS BY DIRECT INTEGRAL METHOD



 NUMBER OF DEPENDENT VARIABLES .......  1
 NUMBER OF PARAMETERS.................  3
 NUMBER OF TIME POINTS ...............  8
 OPTION OF WEIGHTING .................  0 (IDENTICAL WEIGHTS)


 PRINCIPAL COMPONENT ANALYSIS OF NORMED CROSS PRODUCT MATRIX

EIGENVALUE     P( 1 )    P( 2 )    P( 3 )

 6321.124    -.491798   .141432   .859146
 356.7185     .811055  -.284531   .511109
 1.76409      .316741   .948176   .025222
```

```
-----------------------------------------------------
No            Y MEAS        Y COMP        RESIDUAL
-----------------------------------------------------
 1           0.244400E+02  0.243900E+02  0.500450E-01
 2           0.194400E+02  0.197311E+02  -.291092E+00
 3           0.155600E+02  0.149628E+02  0.597219E+00
 4           0.105600E+02  0.105999E+02  -.399475E-01
 5           0.907000E+01  0.972145E+01  -.651446E+00
 6           0.685000E+01  0.678172E+01  0.682759E-01
 7           0.407000E+01  0.376393E+01  0.306074E+00
 8           0.167000E+01  0.197414E+01  -.304137E+00
-----------------------------------------------------

 SUM OF SQUARES (VIA SOLUTION OF ODE).  1.060729
 DEGREES OF FREEDOM..................  5
 STANDARD ERROR ......................  .460593
 CRITICAL T-VALUE AT 95 % CONF. LEVEL   2.57

--------------------------------------------------------------------
PARAMETER      ESTIMATE      ST. ERR      LOWER BOUND   UPPER BOUND
--------------------------------------------------------------------
 P( 1 )      0.247583E+00  0.276408E-01  0.176546E+00  0.318620E+00
 P( 2 )      0.558522E+01  0.183689E+01  0.864411E+00  0.103060E+02
 P( 3 )      0.243901E+02  0.390726E+00  0.233859E+02  0.253942E+02
--------------------------------------------------------------------

 CORRELATION MATRIX OF PARAMETERS:

         P( 1 )   P( 2 )   P( 3 )

 P( 1 )  1
 P( 2 )  .98      1
 P( 3 )  .667     .53      1
```

For comparison, the indirect least squares estimates and their standard errors are: $p_1 = 0.246 \pm 0.029$, $p_2 = 5.43 \pm 2.01$ and $p_3 = 24.401 \pm 0.39$ (ref. 19) .

<u>Exarcise</u>

- In the previous output the computed y_i values correspond to the final parameter estimates. Replace the observed y_i values by the computed y_i values in the DATA statements 114 - 128 of the main program. Rerun the modified program and compare the parameter estimates obtained by the original and the modified program. What is the reason of the difference between the two sets of parameters?

5.6 IDENTIFICATION OF LINEAR SYSTEMS

Higher order linear differential equations of the form

$$y^{(m)} + a_1y^{(m-1)} + \dots + a_my = b_1u^{(m-1)} + \dots + b_mu \qquad (5.65)$$

are important in many application areas, particularly in automatic control and in pharmacokinetics. In equation (5.65) m is the model order, $u(t)$ and $y(t)$ denote the input and output of the system, respectively. The constant coefficients $a_1, a_2, \ldots, a_m$ and $b_1, b_2, \ldots, b_m$ usually have no physical meaning. For example, in pharmacokinetics (5.65) may describe the distribution kinetics of a drug, where $y(t)$ is the plasma concentration and the input $u(t)$ represents the absorption curve following a dose administered via an extravascular route (refs. 20, 22).

We assume that the system is initially at rest, i.e., $u^{(i)}(t) = y^{(i)}(t) = 0$ for $t < 0$ and for all $i = 0, 1, \ldots, m-1$. Neither the response nor the input functions are, however, necessarily continuous at $t = 0$, and hence the initial conditions (i.e., the right-sided limits of the variables) may be nonzero.

The computational tasks in linear system modeling are

(i) prediction of the output $y(t)$ for a given model (5.65) and known input $u(t)$,

(ii) system identification, i.e., estimation of the order m and the parameters a_i, b_i from a given input - output pair $[u(t), y(t)]$,

(iii) identification of the input function $u(t)$ for the known model (5.65) and output $y(t)$.

Transforming (5.65) to a system of m first - order differential equations it can be solved numerically, and fitting models of different order we can also estimate its parameters. There exists, however, a special family of methods based on the use of the convolution integral

$$y(t) = \int_0^t u(\tau)h(\tau-t)\,d\tau \tag{5.66}$$

where $h(t)$ is the weighting function of system (5.65), i.e., the response to a unit Dirac impulse input. The correspondence between (5.65) and its weighting function is one - to - one. For models of moderate complexity the latter can be obtained by analytical methods, mainly by Laplace transformation (see e.g., ref. 23), and used to solve problem (i) by evaluating the integral (5.66).

Consider now the problem of identifying a linear system in the form of its weighting function $h(t)$, using the relationship (5.66). This problem is called deconvolution. Discrete Fourier transformation offers a standard technique performing numerical deconvolution as mentioned in Section 4.3.3. It requires, however, a large sample of equidistant data points, usually not available in pharmacokinetics. Therefore, a variety of deconvolution methods have been proposed in the pharmacokinetic literature (refs. 20, 21, 22, 24, 26, 28). The simplest and still most popular is the point - area method. Its basic idea is approximating the known input by a piecewise - constant function $\bar{u}$ such that $\bar{u}(t) = \bar{u}_i$ on the interval $[t_{i-1}, t_i]$, and $\bar{u}_i$ is defined by the integral mean

$$\bar{u}_i = \frac{1}{t_i - t_{i-1}} \int_{t_{i-1}}^{t_i} u(t)dt \ . \tag{5.67}$$

As shown in Fig. 5.4, the area under the curve of the input remains unchanged in this approximation.

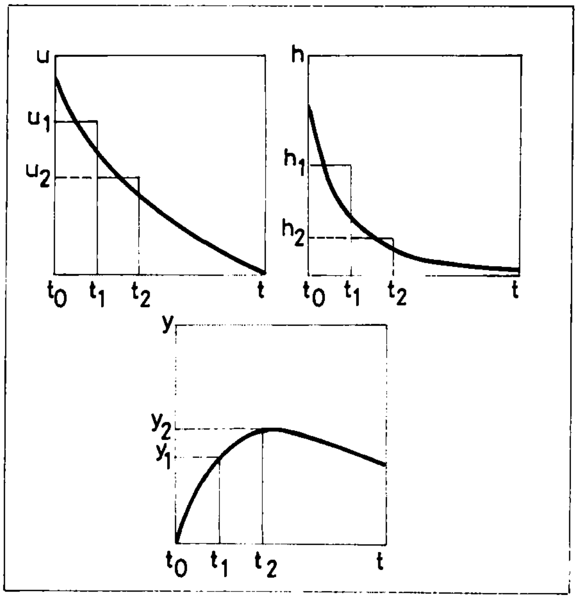


Fig. 5.4 Notations in the point - area method

Similar stepwise approximation of the weighting function $h(t)$ with the discrete values $h_1, \ldots, h_n$, and replacement of $y(t_i)$ by the observed values $\tilde{y}_i$ transform (5.66) to the system

$$\tilde{y}_j = \sum_{i=1}^{j} \bar{u}_i h_{j-i+1}(t_i - t_{i-1}),\ j= 1,\ 2,\ \ldots,\ n \qquad (5.68)$$

of linear algebraic equations. The coefficient matrix of (5.68) is triangular, and hence the equations can be easily solved for h_1, h_2, ...,h_n (ref. 22).

While the point - area method is very convenient in terms of computational efforts, it has a serious drawback. The matrix of the linear system (5.68) is inherently ill - conditioned (ref. 25), and the result is very sensitive to the errors in the observations.

More robust deconvolution methods can be derived by a parametric approach. For example, let us seek $h(t)$ in the form of a polyexponential

$$h(t) = \sum_{i=1}^{m} A_i \exp(-\lambda_i t) \qquad (5.68)$$

with unknown m and parameters A_i, λ_i . Substituting this function into (5.66) gives a (nonlinear) parameter estimation problem (ref. 26), although one must approximate the observed input values u_1, ..., u_n by some function in order to evaluate the integral in (5.66). We propose here a different parametric method that leads to a linear estimation problem.

The idea is estimating first the parameters in (5.65) by the direct integral approach discussed in the previous section, and then evaluate the weighting function analitically (ref. 27). For notational simplicity set $m = 2$ in (5.65). The equation is integrated twice to give

$$y(t_i) = - a_1\int_{0-}^{t_i} y(\tau)\, d\tau - a_2\int_{0-}^{t_i}\int_{0-}^{t} y(\tau)\, d\tau\, dt +$$

$$+ b_1\int_{0-}^{t_i} u(\tau)\, d\tau + b_2\int_{0-}^{t_i}\int_{0-}^{t} u(\tau)\, d\tau\, dt \qquad (5.69)$$

where $t = 0-$ denotes time "just before" $t = 0$. As in the previous sections, we replace the integrands by spline functions interpolating the observed values $\tilde{y}_0$, $\tilde{y}_1$, ..., $\tilde{y}_n$ and u_o, u_1, ..., u_n . It is advantageous to write the input in the form

$$u(t) = DC \times \delta(t) + US \times H(t) + u_c(t) \qquad (5.70)$$

where $\delta(t)$ and $H(T)$ are, respectively, unit Dirac impulse and unit step functions and $u_c(t)$ is a continuous function such that $u_c(0) = 0$. (In pharmacokinetic applications DC denotes the dose given as an intravenous bolus at $t = 0$.) Since

$$\int_{0-}^{t_i} DC \times \delta(\tau)\, d\tau = DC \qquad \text{and} \qquad \int_{0-}^{t_i} US \times H(\tau)\, d\tau = US \times t_i \ ,$$

we need to fit a spline function only to the points of the continuous component $u_c(t)$ of the input. Evaluating the integrals in (5.69), the parameters a_1, a_2, b_1 and b_2 can be estimated by multivariable linear regression. From these estimates the weighting function can be obtained by simple algebraic expressions (ref. 27).

In the special case the input consists of a single Dirac impulse, the first sampling time can be different from zero. Then the resulting weighting function must be appropriately adjusted (ref. 27). In any other case, however, the method applies only, if the first time point is $t = 0$.

Here we present a program that performs all the above operations for first and second order models. The input data are the model order, DC and US (use zero values if the input has only continuous component) and the number of sample points. In addition, for each sample point the sample time, the (continuous part of the) input and the observed output must be given. The program recognizes if the first time point is not at $t = 0$. Interpolating spline is used to compute the integrals and the linear regression procedure is used to estimate the parameters. The remainder of the program finds the analytical expression for the weighting function and evaluates its values at the sample time points. Before presenting the program itself we discuss a test example of system identification outlined in Fig. 5.5.

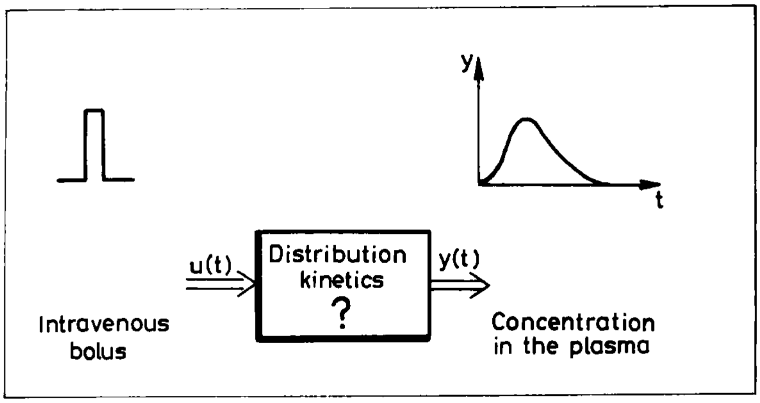


Fig. 5.5. Distribution kinetics identification

Example 5.6 Identification of a single distribution kinetics

Suppose an intravenous bolus is given at $t = 0$ and the drug concentration in the plasma is observed beginning at a time point $t > 0$. In Table 5.3 we list a data set of Cutler (ref. 20) generated by adding 1% relative errors of random character to the values of the weighting function $h(t) = \exp(-5t) + \exp(-t)$. Here we attempt to identify $h(t)$ from the error - corrupted data, naturally not making use of the "true" values given only for comparison.

Table 5.3
Data to system identification

Time, t	"True" weighting function	"Observed" response (1% error)
0.1	1.511	1.515
0.2	1.187	1.177
0.3	0.964	0.972
0.4	0.806	0.789
0.6	0.599	0.589
0.8	0.468	0.473
1.0	0.375	0.372
1.2	0.304	0.307
1.4	0.248	0.249
1.6	0.202	0.208
2.0	0.135	0.135

First we assume that the model order MD = 2 (in fact it is indeed two, but we do not need to know the exact model order). The input has an impulse component, and hence we set DC = 1. Since the input has no continuous component we give zero values in place of the (continuous) input in the DATA lines 128 - 148 . Note that no observation is available at $t = 0$.

```
100 REM ------------------------------------------------------------
102 REM EX. 5.6 DIRECT INTEGRAL IDENTIFICATION OF A LINEAR SYSTEM
104 REM MERGE M16,M18,M41,M42,M63
106 REM ---------- DATA
108 REM     MD    MODEL ORDER ( 1 OR 2 )
110 DATA     2
112 REM     DC    FLAG FOR ADDITIONAL IMPULSE COMPONENT IN THE INPUT
114 DATA     1
116 REM     US    FLAG FOR ADDITIONAL STEP COMPONENT IN THE INPUT
118 DATA     0
120 REM     NM    NUMBER OF SAMPLE POINTS
122 DATA    11
```

```
124 REM    (TIME,    INPUT,    RESPONSE)
126 REM                          ( NO OBSERVATION AT TIME=0 )
128 DATA     0.1,      0,        1.515
130 DATA     0.2,      0,        1.177
132 DATA     0.3,      0,        0.972
134 DATA     0.4,      0,        0.789
136 DATA     0.6,      0,        0.589
138 DATA     0.8,      0,        0.473
140 DATA     1.0,      0,        0.372
142 DATA     1.2,      0,        0.307
144 DATA     1.4,      0,        0.249
146 DATA     1.6,      0,        0.208
148 DATA     2.0,      0,        0.135
200 REM ---------- READ DATA AND SPECIFY DIMENSIONS
202 READ MD,DC,US,NM
204 NX=MD+MD
206 DIM Z(NM),V(NM),Y(NM),F(NM),S(4,NM),X(NM,NX),W(NM),P(NX)
208 DIM A(NX,NX),C(NX,NX),U(NX,NX),D(NX)
210 FOR I=1 TO NM :READ Z(I),V(I),Y(I) :NEXT I
212 N=NM :EC=0
214 REM ---------- COMPUTE INTEGRALS
216 FOR I=1 TO N :F(I)     =-Y(I)      :NEXT I
218 GOSUB 6300
220 FOR I=1 TO N :X(I,1)   = S(4,I)    :NEXT I
222 FOR I=1 TO N :F(I)     = V(I)      :NEXT I
224 GOSUB 6300
226 FOR I=1 TO N :X(I,MD+1)= S(4,I)    :NEXT I
228 IF MD=1 THEN 242
230 FOR I=1 TO N :F(I)     = X(I,1)    :NEXT I
232 GOSUB 6300
234 FOR I=1 TO N :X(I,2)   = S(4,I)    :NEXT I
236 FOR I=1 TO N :F(I)     = X(I,MD+1) :NEXT I
238 GOSUB 6300
240 FOR I=1 TO N :X(I,MD+2)= S(4,I)    :NEXT I
242 REM ---------- ADD EFFECT OF IMPULSE AND STEP COMPONENT
244 IF DC=0 AND US=0 THEN 252
246 FOR I=1 TO N :X(I,MD+1)=X(I,MD+1)+DC+US*(Z(I)-Z(1)) :NEXT I
248 IF MD=1 THEN 252
250 FOR I=1 TO N :X(I,MD+2)=X(I,MD+2)+DC*(Z(I)-Z(1))+US/2*(Z(I)-Z(1))^2 :NEXT I
252 REM ---------- CALL LINEAR REGRESSION MODULE
254 RP=0 :WI=0 :GOSUB 4200
256 IF MD=2 THEN 270
258 REM ---------- FIRST ORDER MODEL
260 H$="A*exp(alfa*t)"
262 A=P(2) :AL=-P(1)
264 IF Z(1)=0 THEN 322
266 REM ---------- CORRECTION FOR NON-ZERO INITIAL TIME
268 A=A*EXP(-AL*Z(1)) :GOTO 322
270 DI=P(1)*P(1)-4*P(2) :IF ABS(DI)<1E-12*P(1) THEN DI=0
272 IF DI=0 THEN 292 ELSE IF DI<0 THEN 306
274 REM ---------- SECOND ORDER MODEL - TWO DISTINCT REAL ROOTS
276 H$="A*exp(alfa*t)+B*exp(beta*t)" :ES=1
278 SD=SQR(DI)
280 AL=-P(1)/2+SD/2   :BE=-P(1)/2-SD/2
282 A = (P(3)*AL+P(4))/SD :B =-(P(3)*BE+P(4))/SD
284 IF Z(1)=0 THEN 322
286 REM ---------- CORRECTION FOR NON-ZERO INITIAL TIME
288 A=A*EXP(-AL*Z(1)) :B=B*EXP(-BE*Z(1))
290 GOTO 322
```

```
292 REM ---------- SECOND ORDER MODEL - TWO IDENTICAL REAL ROOTS
294 H$="(A+B*t)*exp(alfa*t)" :ES=2
296 AL=-P(1)/2 :A =P(3) :B = P(3)*AL+P(4)
298 IF Z(1)=0 THEN 322
300 REM ---------- CORRECTION FOR NON-ZERO INITIAL TIME
302 D=EXP(-AL*Z(1)) :A=(A-B*Z(1))*D :B=B*D
304 GOTO 322
306 REM ---------- SECOND ORDER MODEL - COMPLEX ROOTS
308 H$="( A*cos(beta*t)+B*sin(beta*t) )*exp(alfa*t)" :ES=3
310 AL=-P(1)/2 :BE=SQR(-DI)/2 :A =P(3) :B =(P(3)*AL+P(4))/BE
312 IF Z(1)=0 THEN 322
314 REM ---------- CORRECTION FOR NON-ZERO INITIAL TIME
316 S=SIN(-BE*Z(1)) :C=COS(-BE*Z(1)) :D=EXP(-AL*Z(1))
318 AA=(A*C+B*S)*D :B=(B*C-A*S)*D :A=AA
320 GOTO 322
322 REM ---------- PRINT RESULTS
324 V$=STRING$(40,"-")
326 LPRINT :LPRINT :LPRINT V$ :LPRINT V$ :LPRINT
328 LPRINT "MODEL ORDER:";MD
330 LPRINT :LPRINT :LPRINT
332 LPRINT "WEIGHTING FUNCTION:" :LPRINT
334 LPRINT "h(t) = ";H$
336 LPRINT "   A =";A
338 IF MD>1 THEN LPRINT "   B =";B
340 LPRINT "alfa =";AL
342 IF MD=2 AND ES<>2 THEN LPRINT "beta =";BE
344 LPRINT :LPRINT "WEIGHTING FUNCTION VALUES:" :LPRINT
346 LPRINT V$ :LPRINT " No"TAB(11)" t"TAB(28)"h(t)" :LPRINT V$
348 IF Z(1)<>0 THEN T=0 :GOSUB 356 :LPRINT TAB(10)T;TAB(25)H
350 FOR I=1 TO NM :T=Z(I) :GOSUB 356 :LPRINT I;TAB(10)T;TAB(25)H :NEXT I
352 LPRINT V$ :LPRINT
354 GOTO 368
356 REM ---------- COMPUTE WEIGHTING FUNCTION
358 IF MD=1 THEN H=A*EXP(AL*T) :RETURN
360 IF ES=1 THEN H=A*EXP(AL*T)+B*EXP(BE*T) :RETURN
362 IF ES=2 THEN H=(A+B*T)*EXP(AL*T) :RETURN
364 H=(A*COS(BE*T)+B*SIN(BE*T))*EXP(AL*T) :RETURN
366 REM ---------- END OF PROGRAM
368 STOP
```

The first part of the program output comes from the module M42 of multivariable linear regression. The parameters P(1), P(2), P(3) and P(4) correspond to a_1, a_2, b_1 and b_2, respectively, and have no physical meaning. The weighting function obtained from the estimates is printed as an analytical expression and its values are also listed.

```
                    MULTIVARIABLE LINEAR REGRESSION
                       METHOD OF LEAST SQUARES



NUMBER OF INDEPENDENT VARIABLES .....  4
NUMBER OF SAMPLE POINTS .............  11

PRINCIPAL COMPONENT ANALYSIS OF THE CORRELATION MATRIX
```

```
EIGENVALUE    X( 1 )      X( 2 )      X( 3 )      X( 4 )

0.36311E+01   0.520       0.498       -.461       -.518
0.33700E+00   -.039       0.512       0.814       -.271
0.31837E-01   -.756       0.545       -.352       0.080
0.41210E-04   0.395       0.438       0.012       0.807

----------------------------------------------------------------------
 I             Y MEAS        WEIGHT        Y COMP        RESIDUAL
----------------------------------------------------------------------

 1            0.15150E+01   0.10000E+01   0.15150E+01   -.28253E-04
 2            0.11770E+01   0.10000E+01   0.11841E+01   -.71427E-02
 3            0.97200E+00   0.10000E+01   0.95536E+00   0.16641E-01
 4            0.78900E+00   0.10000E+01   0.79240E+00   -.34047E-02
 5            0.58900E+00   0.10000E+01   0.60182E+00   -.12819E-01
 6            0.47300E+00   0.10000E+01   0.46816E+00   0.48361E-02
 7            0.37200E+00   0.10000E+01   0.37347E+00   -.14727E-02
 8            0.30700E+00   0.10000E+01   0.30483E+00   0.21738E-02
 9            0.24900E+00   0.10000E+01   0.24919E+00   -.19461E-03
 10           0.20800E+00   0.10000E+01   0.20467E+00   0.33343E-02
 11           0.13500E+00   0.10000E+01   0.13745E+00   -.24493E-02
----------------------------------------------------------------------

SUM OF SQUARES ......................  5.513071E-04
DEGREES OF FREEDOM .................  7
STANDARD ERROR ......................  8.874578E-03
DURBIN-WATSON D-STATISTICS ..........  2.762654
CRITICAL T-VALUE AT 95 % CONF. LEVEL   2.37

----------------------------------------------------------------------
PARAMETER     ESTIMATE      ST.ERROR      LOWER BOUND   UPPER BOUND
----------------------------------------------------------------------
 P( 1 )       0.59207E+01   0.28378E+00   0.52481E+01   0.65932E+01
 P( 2 )       0.47682E+01   0.35937E+00   0.39165E+01   0.56199E+01
 P( 3 )       0.15150E+01   0.81448E-02   0.14957E+01   0.15343E+01
 P( 4 )       0.49347E+01   0.34565E+00   0.41155E+01   0.57539E+01
----------------------------------------------------------------------

CORRELATION MATRIX OF PARAMETERS

              P( 1 )      P( 2 )      P( 3 )      P( 4 )

P( 1 )        1.000
P( 2 )        0.993       1.000
P( 3 )        0.643       0.576       1.000
P( 4 )        0.997       0.999       0.596       1.000

----------------------------------------
----------------------------------------

MODEL ORDER: 2

WEIGHTING FUNCTION:

h(t) = A*exp(alfa*t)+B*exp(beta*t)
   A = .9578139
   B = 1.059122
alfa =-.9614869
beta =-4.959169
```

```
WEIGHTING FUNCTION VALUES:

----------------------------------------
 No        t              h(t)
----------------------------------------
           0              2.016936
 1        .1              1.515029
 2        .2              1.183079
 3        .3              .9570464
 4        .4              .797706
 5        .6              .591986
 6        .8              .4638826
 7        1               .3736289
 8        1.2             .3048906
 9        1.4             .2503014
 10       1.6             .2060496
 11       2               .1400574
----------------------------------------
```

It is interesting to compare the results with the "true" weighting function values listed in Table 5.3. The agreement is fairly good. (Notice that the data came from "observing" the response to a unit impulse, and hence what we did was really a smoothing of the observed weighting function.)

At this point two remarks are appropriate. First, linear system identification is a somewhat more general problem than parameter estimation, since the order of the model (5.65) is also unknown. In (ref. 27) models of different order were fitted to the data and the Akaike Information Criterion (see Section 3.10.3) was used to select among rival model orders. In particular, considering another data set of Cutler with larger errors, it was shown that the "best" model, resulting in a statistically preferable estimate of the weighting function, might be of lower order than the "true" model used to generate the data. Second, we should admit that for higher order models the direct integral approach is not the best general parameter estimation method. In fact, with simple input functions common in pharmacokinetic applications (e.g., impulse or step function), the columns of the observation matrix **X** created from the integrals in (5.69) tend to be linearly dependent, resulting in ill - conditioned estimation problems. As discussed in the next section, this method is, however, excellent for input identification.

5.7 DETERMINING THE INPUT OF A LINEAR SYSTEM BY NUMERICAL DECONVOLUTION

The problem considered here is outlined in Fig. 5.6. The weighting function h(t) of the system and its response to an unknown input are known. We want to find the input u(t) satisfying equation (5.66).

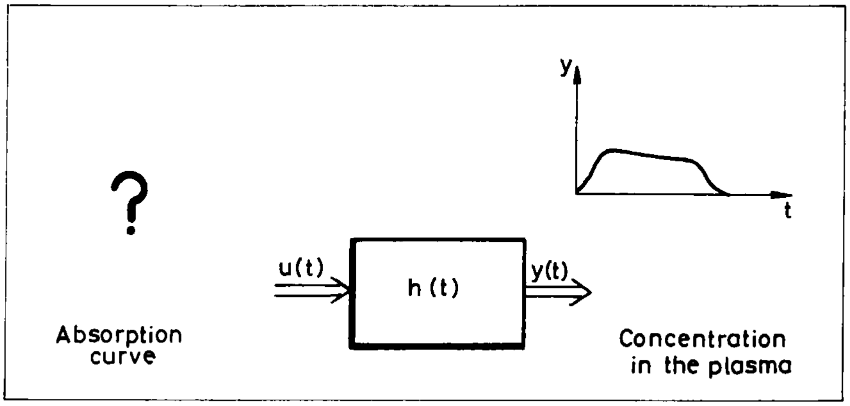


Fig. 5.6. Determining the input corresponding to a given output

Since the convolution integral is symmetrical in u(t) and h(t), this problem is similar to the one of system identification considered in the previous section. Nevertheless, it is usually easier to find the weighting function h(t) since its form is more - or - less known (e.g., as a sum of polyexponentials), and hence parametric methods apply, whereas the input function u(t) is a priori arbitrary. Therefore, the non - parametric point - area method is a popular way of performing numerical deconvolution. It is really simple: evaluating the integral means h_1, h_2, ..., h_n of the weighting function over the subinterval $[t_{i-1}, t_i]$ we can easily solve the set (6.68) of linear equations for the values $\bar{u}_1$, $\bar{u}_2$, ..., $\bar{u}_n$, of the stepwise input function. As emphasised in the previous section, this method is, however, very sensitive to the errors in the observations. Although we can overcome this difficulty by carefully smoothing the data (ref. 22), the result will much depend on the particular method of smoothing.

Another non - parametric approach is deconvolution by discrete Fourier transformation with built - in windowing. The samples obtained in pharmacokinetic applications are, however, usually short with non - equidistant sample time points. Therefore, a variety of parametric deconvolution methods have been proposed (refs. 20, 21, 26, 28). In these methods an input of known form depending on unknown parameters is assumed, and the model response predicted by the convolution integral (5.66) is fitted to the data.

The deconvolution method we propose here is also parametric and is based on direct integral parameter estimation (ref. 27). We consider a "hypothetical" linear system S^* with input $u^* = h$, where h is the known weighting function of the real system S, and the output of S^* is assumed to be $y^* = y$, the known response function. Then by (5.66) we have

$$y^*(t) = \int_0^t h^*(t-\tau)u^*(\tau)\, d\tau = \int_0^t u^*(t-\tau)h^*(\tau)\, d\tau = \int_0^t h(t-\tau)h^*(\tau)\, d\tau \,. \qquad (5.71)$$

Since $y^* = y$, comparison of equations (5.66) and (5.71) shows that the weighting function h^* of S^* equals the input function u which is being sought. Now, h^* can be estimated by identifying the weighting function of a linear model of the form (5.65) as described in the previous section. The same program can be used for input determination if the role of the variables is properly understood.

<u>Example 5.7</u> Determining the absorption curve for a given response function

We continue solving the test example of Cutler (ref. 20). In Example 5.6 we identified the weighting function of the system. Now we consider the second half of the data set generated by Cutler and shown in Table 5.4. The "true" input $u(t) = 1.2\exp(-2t)$ and the "true" weighting function were used by Cutler to generate the "true" response, then 1% random error was added to obtain the "observed" response (i.e., the observed drug concentration in the plasma). Our goal is to find the input (i.e., the absorption curve) making use of the weighting function identified in the previous example and the "observed" response.

Table 5.4
Data to determine the absorption curve

Time, t	"True" input	"True" response	"Observed" response (1% error)
0	1.2	0	0
0.1	0.9825	0.180	0.181
0.2	0.8044	0.293	0.291
0.3	0.6586	0.360	0.361
0.4	0.5392	0.394	0.388
0.6	0.3614	0.400	0.399
0.8	0.2423	0.368	0.372
1.0	0.1624	0.327	0.328
1.2	0.1089	0.288	0.286
1.4	0.0730	0.250	0.249
1.6	0.0489	0.211	0.210
2.0	0.0220	0.155	0.153

When identifying the hypothetical system S^* we need u^* . The weighting function found in Example 5.6 is substituted for the input of the hypothetical system. This input does not contain an impulse or a unit step component, and hence we set DC = 0 and US = 0 . The response of the hypothetical system equals the "Observed" response. The program is the one used in Example 5.6, only tha data lines are changed as follows:

```
100 REM ------------------------------------------------------------
102 REM EX. 5.7 INPUT FUNCTION DETERMINATION TO A GIVEN RESPONSE
104 REM MERGE M16,M18,M41,M42,M63
106 REM ---------- DATA
108 REM     MD   HYPOTHETICAL MODEL ORDER
110 DATA     1
112 REM     DC   FLAG FOR IMPULSE COMPONENT IN HYPOTHETICAL INPUT
114 DATA     0
116 REM     US   FLAG FOR STEP COMPONENT IN HYPOTHETICAL INPUTN
118 DATA     0
120 REM     NM   NUMBER OF SAMPLE POINTS
122 DATA    12
124 REM  TIME POINTS  HYPOTHETICAL INPUT   RESPONSE
126 DATA      0.0,       2.016936,          .000
128 DATA      0.1,       1.515029,          .181
130 DATA      0.2,       1.193079,          .291
132 DATA      0.3,        .9570464,         .361
134 DATA      0.4,        .797706,          .388
136 DATA      0.6,        .591986,          .399
138 DATA      0.8,        .4638826,         .372
140 DATA      1.0,        .3736289,         .328
142 DATA      1.2,        .3048906,         .286
144 DATA      1.4,        .2503014,         .249
146 DATA      1.6,        .2060496,         .210
148 DATA      2.0,        .1400574,         .153
150 REM ---------- FROM HERE THE SAME AS THE PROGRAM OF EX. 5.6
```

The assumed model order is MD = 1 . We list here only the essential parts of the output.

```
----------------------------------------
----------------------------------------

MODEL ORDER: 1


WEIGHTING FUNCTION:

h(t) = A*exp(alfa*t)
   A = 1.175618
alfa =-1.948693
```

```
WEIGHTING FUNCTION VALUES:

---------------------------------------
 No        t                h(t)
---------------------------------------
 1        0               1.175618
 2        .1              .9674652
 3        .2              .7961681
 4        .3              .6552003
 5        .4              .5391921
 6        .6              .3651591
 7        .8              .2472981
 8        1               .1674787
 9        1.2             .1134222
 10       1.4             .0768134
 11       1.6             5.202064E-02
 12       2               2.385904E-02
---------------------------------------
```

The "weighting function" we found is that of the hypothetical system, therefore it is the absorption curve we were looking for. It is useful to compare it with the "true" input given in Table 5.4. In this special case the input function found and the "true" input are of the same analytical form, so we can compare the parameters of the two functions, as well. In realistic applications, however, we are not interested in the "analytical form" of the input function and rather the table of computed values is of primary interest.

The direct integral approach to numerical deconvolution preserves the symmetry of system identification and input determination, similarly to the point - area method. By (5.71) the input function $u = h^*$ is restricted to the class of weighting functions generated by a single - input, single - output, time invariant system (5.65). This class includes polyexponentials, polynomials and trigonometric functions, so that the constraint on the form of the input is relatively mild. This constraint may in fact have a physical meaning in pharmacokinetics. For example, in the problem studied in Example 5.7 the hypotetical system S^* may be a real linear system whose response is the bioavailability of the drug following an impulse administration via an extravascular route.

Exercise

- Repeat the input identification experiment with the model order MD = 2 . Compare the linear regression residual errors for the two cases. Select the "best" model order on the basis of the Akaike Information Criterion (see Section 3.10.3 and ref. 27) .

5.8 APPLICATIONS AND FURHTER PROBLEMS

5.8.1 Principal component analysis of kinetic models

The researcher usually looks for a model that not only fits the data well, but describes the mechanism of action of the chemical or biological process. Such detailed models are, however, frequently overparameterized with respect to the available data, leading to ill-conditioned problems of parameter estimation. In Section 3.5.2 you have learned that principal component analysis of the normalized cross-product matrix $\mathbf{J}^T(\boldsymbol{\beta})\mathbf{W}\mathbf{J}(\boldsymbol{\beta})$ is a standard method of detecting ill-conditioned parameter estimation problems. In Section 5.3 we introduced the matrix $\mathbf{S}$ of normalized sensitivity coefficients. It plays the same role for dynamical models as $\mathbf{J}(\boldsymbol{\beta})$ in algebraic parameter estimation problems. Therefore, the principal component analysis of $\mathbf{S}^T\mathbf{S}$ (or of $\mathbf{S}^T\mathbf{W}\mathbf{S}$, if weighting is necessary) offers a convenient tool for extracting information from sensitivity coefficients, and it reveals whether or not there is any hope to identify the parameters of the model. Although we need initial parameter estimates to perform the calculation, such are usually available in the literature, at least in the form of some order of magnitude guesses. In this section we reconsider the sensitivity coefficients obtained in Example 5.3.

Example 5.8.1 Practical identifiability of the parameters of the microbial growth process

As shown by Holmberg (ref. 3) the four parameters V_m, K_s, K_d and Y are theoretically identifiable if both the concentration of the microorganism (y_1) and that of the substrate (y_2) are observed. Practical identifiability of the parameters is, however, a much more difficult issue. In the following four cases are investigated:

(i) Both concentrations, y_1 and y_2 are observed. The error variance is small: $\sigma^2 = 0.01$.

(ii) Both y_1 and y_2 are observed. The error variance is large: $\sigma^2 = 1$.

(iii) Only the substrate, y_2 is observed. The error variance is $\sigma^2 = 0.01$.

(iv) Only y_2 is observed. The error variance is $\sigma^2 = 1$.

To investigate cases (i) and (ii), the $\mathbf{S}$ matrix obtained in Example 5.3 is used directly. Forming $\mathbf{S}^T\mathbf{S}$ and applying eigenvalue-eigenvector decomposition (by the module M18), we obtain the results shown in Table 5.5.

Table 5.5
Principal component analysis of the normalized sensitivity matrix; both concentrations observed

Eigenvalue	Eigenvector components corresponding to V_m	K_s	K_d	Y
69429	0.957	-0.134	-0.095	-0.239
12304	0.230	0.020	-0.137	0.963
2.583	0.042	-0.518	0.846	0.121
1.724	0.172	0.845	0.507	0.013

In case (i) $100\sigma^2 = 1$, and hence the problem is not ill-conditioned, all the parameters can be identified. Unfortunately we can hardly hope such a small error variance in biotechnical applications. In the more realistic case (ii) $100\sigma^2 = 100$, thus two eigenvalues are below the threshold. As it was discussed in Section 3.5, the eigenvectors corresponding to the small eigenvalues show that there is no hope to identify parameters K_s and K_d with reasonable accuracy.

To investigate cases (iii) and (iv), we include only every second row of matrix S obtained in Example 5.3 when forming S^TS . Applying eigenvalue-eigenvector decomposition again, the results shown in Table 5.6 are obtained.

Table 5.6
Principal component analysis of the normalized sensitivity matrix; only substrate y_2 is observed

Eigenvalue	Eigenvector components corresponding to V_m	K_s	K_d	Y
51599	0.912	-0.137	-0.081	-0.378
19.225	0.334	-0.225	-0.097	0.909
0.489	0.212	0.964	0.008	0.162
0.000007	0.106	-0.041	0.991	0.057

As seen from the table, in case (iii) we can identify V_m and Y , but neither K_s nor K_d can be estimated. In the (unfortunately) more realistic case (iv) one can hope a reasonable parameter estimate only for V_m. It is advantageous to fix all the other parameters at some nominal value, so avoiding the inherent difficulties of the parameter estimation process.

Practical identifiability is not the only problem that can be adressed by principal component analysis of the sensitivity matrix. In (refs. 29-30) several examples of model reduction based on this technique are discussed.

Computing the sensitivities is time consuming. Fortunately the direct integral approximation of the sensitivity matrix and its principal component analysis can offer almost the same information whenever the direct integral method of parameter estimation applies.

5.8.2 Identification of a linear compartmental model

Assuming that a small dose of drug does not move the organism far from equilibrium state, linear differential equations are frequently used to describe the kinetics of drug distribution among different organs, and its elimination from the body. Giving some insight into the mechanism of action, linear compartmental models are particularly important and more popular than models of the form (5.65). In Example 2.2.1 a very simple compartmental model was used to describe the concentration of a certain drug in blood. Jennrich and Bright (ref. 31) estimated the parameters of the linear compartmental model shown in Fig. 5.7 from the data of Table 5.7.

Table 5.7
Sulphate kinetics data

Time, t_i	Activity, $\tilde{y}_i$	Time, t_i	Activity, $\tilde{y}_i$
0	200000	50	61554
2	151117	60	59940
4	113601	70	57689
6	97652	80	56440
8	90935	90	53915
10	84820	110	50938
15	76891	130	48717
20	73342	150	45996
25	70593	160	44968
30	67049	170	43607
40	64313	180	42668

The experiment consists of applying an intravenous bolus of sulphate traced by a radioactive isotope and measuring the activity of blood samples. The compartmental model in Fig. 5.7. leads to the differential equations

$$
\begin{aligned}
dx_1/dt &= (-k_1 + k_2)x_1 + k_3x_2 \\
dx_2/dt &= k_2x_1 - (k_3 + k_4)x_2 + k_5x_3 \qquad (5.72) \\
dx_3/dt &= k_4x_2 - k_5x_3 \ .
\end{aligned}
$$

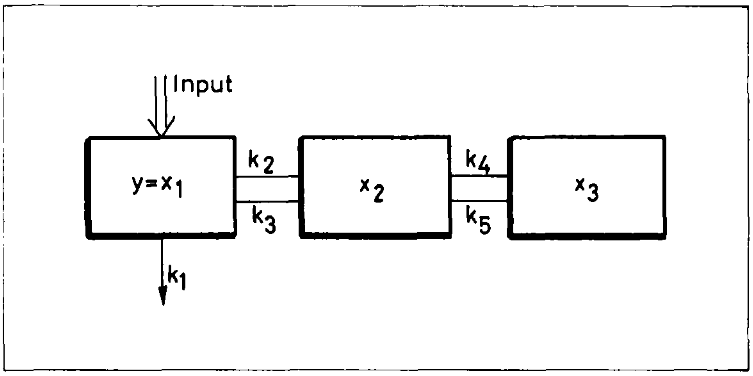


Fig. 5.7. Compartmental model of sulphate distribution kinetics

In this model x_1 is the activity in Compartment 1 representing the blood plasma volume, x_2 and x_3 are unobserved activities, and k_1, k_2, ..., k_5 are the rate constants to be determined. The initial values $x_1^o = 2\times10^5$, $x_2^o = x_3^o = 0$ assumed to be known exactly. The only observed variable is $y = x_1$. Jennrich and Bright (ref. 31) used the indirect approach to parameter estimation and solved the equations (5.72) numerically in each iteration of a Gauss-Newton type procedure exploiting the linearity of (5.72) only in the sensitivity calculation. They used relative weighting. Although a similar procedure is too time consuming on most personal computers, this does not mean that we are not able to solve the problem. In fact, linear differential equations can be solved by analytical methods, and solutions of most important linear compartmental models are listed in pharmacokinetics textbooks (see e.g., ref. 33). For the three compartment model of Fig. 5.7 the solution is of the form

$$y(t) = A_1\exp(\lambda_1 t) + A_2\exp(\lambda_2 t) + A_3\exp(\lambda_3 t) \qquad (5.73)$$

where the parameters A_1, A_2, A_3, λ_1, λ_2 and λ_3 are given as functions of the rate constants k_1, k_2, ..., k_5 and initial conditions. In addition, evaluating (5.73) at $t = 0$ shows that

$$A_1 + A_2 + A_3 = x_1^o , \qquad (5.74)$$

thereby eliminating one of the parameters of (5.73).

Now we can proceed in two different ways, either by estimating the parameters $k_1, k_2, \ldots, k_5$ directly, using the analytical solution and the module M45, or estimating first the parameters in (5.73). In this latter case we can use the very simple peeling method, also known as the method of residuals. Although the peeling procedure is of approximate character and does not take into account the available constraints such as (5.74), it still gives useful initial estimates for the least squares method.

The peeling method is based on the observation that for compartmental models $\lambda_i < 0$ in the solutions of the form (5.73). In addition, the exponents are not close to each other, since otherwise we are unable to separate the terms of (5.73) and must lump several compartments. Assume that the inequalities $\lambda_1 < \lambda_2 < \lambda_3 < 0$ hold, then the peeling consists of the following steps:

(i) Divide the time interval into 3 subintervals, containing n_1, n_2 and n_3 points, respectively, where $n_1 + n_2 + n_3 = n$, the total number of sample points.

(ii) Since λ_1 and λ_2 are smaller than λ_3, we may assume that in the last subinterval the contribution from the first two exponents is small. Therefore,

$$\log \tilde{y}_i \approx \log A_3 + \lambda_3 t_i \ , \quad i = n_1 + n_2 + 1, \ldots, n \ , \tag{5.75}$$

and A_3 and λ_3 can be found by fitting a straight line to the last n_3 point of the data.

(iii) In the second subinterval only the first term of (5.73) is assumed to be small, but $A_3\exp(\lambda_3 t_i)$ is already known from (ii). Thus again a straight line is fitted to the data

$$\log[\tilde{y}_i - A_3\exp(\lambda_3 t_i)] \approx \log A_2 + \lambda_2 t_i \ , \quad i = n_1 + 1, \ldots, n_1 + n_2 \ , \tag{5.76}$$

thereby estimating A_2 and λ_2.

(iv) Finally, a straight line is fitted to the data

$$\log[\tilde{y}_i - A_2\exp(\lambda_2 t_i) - A_3\exp(\lambda_3 t_i)] \approx \log A_3 + \lambda_3 t_i \ , \quad i = 1, \ldots, n_1 + n_2 \ , \tag{5.77}$$

in order to estimate A_1 and λ_1.

The critical point in the peeling technique is the right choice of n_3 and n_2 . By (5.75) the logarithmized observations are close to a straight line in the last subinterval, and hence a semi - logarithmic plot of the data helps to find the value of n_3 . A similar plot of the corrected and logarithmized values $\log[\tilde{y}_i - A_3\exp(\lambda_3 t_i)]$ may help to choose n_2 . For the data of Table 5.7 we select $n_1 = 6$, $n_2 = 8$ and $n_3 = 8$. Since relative error is assumed in the original data, unit weights are used when fitting the logarithmic data (see Section 3.4), and hence the modul M40 applies. The resulting estimates are

$$A_1 = 1.06\times10^5 \ , \quad A_2 = 2.19\times10^4 \ , \quad A_3 = 6.93\times10^4 \ ,$$
$$\lambda_1 = -.313 \ , \qquad \lambda_2 = -0.0562 \ , \quad \lambda_3 = -0.0027 \ .$$

These values are further refined by the module M45 applying relative weighting $w_i = 1/\tilde{y}_i^2$ and eliminating A_3 by (5.74). The following estimates and standard errors are obtained

$$A_1 = 1.092\times10^5 \ (\pm5\times10^3), \ A_2 = 2.206\times10^4 \ (\pm4\times10^3),$$
$$\lambda_1 = -.3226 \ (\pm0.019), \quad \lambda_2 = -0.05323 \ (\pm0.014), \quad \lambda_3 = -0.00267 \ (\pm0.00015) \ .$$

The weighted residual sum of squares is $Q = 0.00284$, close to the value $Q = 0.00287$ of Jennrich and Bright. Thus the fit is satisfying and the peeling method is shown to give surprisingly good initial estimates. The only remaining problem is to find the values of the original parameters k_1, k_2, ..., k_5. This can be done via the formulas listed in (ref. 32)

$$k_1 = x_1^o a_3/b_3$$
$$k_2 = a_1 - b_2/x_1^o - k_1$$
$$k_3 = b_2/x_1^o - [a_2 - k_1 b_2/x_1^o - b_3/x_1^o]/k2$$
$$k_5 = b_3/(x_1^o k_3)$$
$$k_4 = [a_2 - k_1 b_2/x_1^o - b_3/x_1^o]k_2 - k_5$$

where

$$a_1 = -(\lambda_1 + \lambda_2 + \lambda_3) \ , \qquad a_2 = \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 \ ,$$
$$a_3 = -\lambda_1\lambda_2\lambda_3 \ ,$$
$$F_1 = A_1(3\lambda_1^2 + 2a_1\lambda_1 + a_2) - x_1^o\lambda_1^2 \ , \ F_2 = A_2(3\lambda_2^2 + 2a_1\lambda_2 + a_2) - x_1^o\lambda_2^2 \ ,$$
$$b_2 = (F_1 - F_2)/(\lambda_1 - \lambda_2) \ , \qquad b_3 = (F_2\lambda_1 - F_1\lambda_2)/(\lambda_1 - \lambda_2) \ .$$

The final estimates

$k_1 = 0.0754$, $k_2 = 0.1754$, $k_3 = 0.1351$, $k_4 = 0.0156$ and $k_5 = 0.0450$

agree well with the ones of Jennrich and Bright (ref. 31).

Exercises

- Carry out numerical experiments with other choices of n_1 , n_2 and n_3 in the peeling method. Try to construct a heuristic rule for subinterval selection which can be used in a computer without human interaction.

- Compute approximate standard errors of the parameters $k_1, k_2, \ldots, k_5$, using the error propagation law

$$\sigma^2_{k_i} = \left[\frac{\partial k_i}{\partial A_1}\right]^2 \sigma^2_{A_1} + \left[\frac{\partial k_i}{\partial A_2}\right]^2 \sigma^2_{A_2} + \left[\frac{\partial k_i}{\partial \lambda_1}\right]^2 \sigma^2_{\lambda_1} + \left[\frac{\partial k_i}{\partial \lambda_2}\right]^2 \sigma^2_{\lambda_2} + \left[\frac{\partial k_i}{\partial \lambda_3}\right]^2 \sigma^2_{\lambda_3} .$$

REFERENCES

1 P. Henrici, Discrete Variable Methods in Ordinary Differential Equations, John Wiley, New York, 1962.
2 R.L. Johnston, Numerical Methods, A Software Approach, John Wiley, New York, 1982.
3 A. Holmberg, On the practical identifiability of microbial growth models incorporating Michaelis-Menten type Nonlinearities, Mathematical Biosciences, 62 (1982) 23-43.
4 B. Carnahan and J.O. Wilkes, Digital Computing and Numerical Methods, John Wiley, New York, 1973.
5 E. Fehlberg, Klassische Runge-Kutta-Formula fünfter und siebenter Ordung mit Schrittweiten-Kontrolle, Computing, 4 (1969) 93-106.
6 C.W. Gear, The automatic integration of ordinary differential equations. Communications of the ACM, 14 (1971) 176-180.
7 P. Seifert, Computational experiments with algorithms for stiff ODEs, Computing, 38 (1987) 163-176.
8 B.A. Gottwald and G. Wanner, A reliable Rosenbrock-integrator for stiff differential equations, Computing, 26 (1981) 335-357.
9 R.J. Field and R.M. Noyes, Oscillations in chemical systems. J. Chemical Physics, 60 (1974) 1877-1884.
10 H. Rabitz, Sensitivity analysis: Theory with applications to molecular dynamics and kinetics. Computers and Chemistry, 5 (1980) 167-180.
11 R.P. Dickinson and R.J. Gelinas, Sensitivity analysis of ordinary differential equation, J. Comp. Physics, 21 (1978) 123-143.
12 A.M Dunker, The decoupled direct method for calculating sensitivity coefficients in chemical kinetics, J. Chem. Phys. 81 (1984) 2385-2393.
13 P. Valkó and S. Vajda, An extended ODE solver for sensitivity calculations, Computers and Chemistry, 8 (1984) 255-271.

14 R.A. Alberty and F. Daniels, Physical Chemistry 5th ed. John Wiley, New York, 1980.

15 Y. Bard, Nonlinear Parameter Estimation. Academic Press, New York, 1974.

16 D.M. Himmelblau, C.R. Jones and K.B. Bischoff, Determination of rate constants for complex kinetic models, Ind. Eng. Chem. Fundamentals, 6 (1967) 539-546.

17 A. Yermakova, S. Vajda and P. Valkó, Direct integral method via spline approximation for estimating rate constants. Applied Catalysis, 2 (1982) 139-150.

18 S. Vajda, P. Valkó and K.R. Godfrey, Direct and indirect least squares methods in continuous-time parameter estimation, Automatica, 23 (1987) 707-718.

19 F.K. Uno, H.L. Ralston, R.I. Jennrich and P.F. Sampson, Test problems from the pharmacokinetic literature requiring fitting models defined by differential equations, Technical Report No. 61. BMDP Statistical Software, Los Angeles, 1979.

20 D.J. Cutler, Numerical deconvolution by least squares: Use of prescribed input functions, J. Pharmacokinetics and Biopharm., 6 (1978) 227-242.

21 D.J. Cutler, Numerical deconvolution by least squares: Use of polynomials to represent input function. J. Pharmacokinetics and Biopharm., 6 (1978) 243-263.

22 F. Langenbucher, Numerical convolution/deconvolution as a tool for correlating in vitro with in vivo drug availability, Pharm. Ind., 44 (1982) 1166-1172.

23 C.T. Chen, Introduction to Linear System Theory, Holt, Rinehart and Winston, New York, 1970.

24 D.P. Vaughan and M. Dennis, Mathematical basis for the point-are deconvolution method for determining in vivo input functions. J. Phar. Sci., Part I, 69 (1980) 298-305, Part II, 69 (1980) 663-665

25 B.R. Hunt, Biased estimation for nonparametric identification of linear systems, Math. Biosciences, 10 (1971) 215-237.

26 P. Veng-Pedersen, Novel deconvolution method for linear pharmacokinetic systems with polyexponential impulse response, J. Pharm. Sci., 69 (1980) 312-318.

27 S. Vajda, K.R. Godfrey and P. Valkó, Numerical deconvolution using system identification methods, J. Pharmacokinetics and Biopharm., 16 (1988) 85-107.

28 P. Veng-Pedersen, An algorithm and computer program for deconvolution in linear pharmacokinetics. J. Phar. Biopharm, 8 (1980) 463-481.

29 S.Vajda, P. Valkó and T. Turányi, Principal component analysis of kinetic models, Int. J. Chem. Kinet. 17 (1985) 55-81.

30 S. Vajda and T. Turányi, Principal component analysis for reducing the Edelson-Field-Noyes model of Belousov-Zhabotinsky reaction, J. Phys. Chem. 90 (1986) 1664.

31 R.I. Jennrich and P.B. Right, Fitting systems of linear differential equations using computer generated exact derivatives, Technometrics, 18 (1976) 385-399.

32 M.S. Gibaldi and D.Perrier, Pharmacokinetics, Marcel Dekker, New York, 1975.

SUBJECT INDEX

Oregonator model 277
orthogonal polynomials 205, 228
orthonormal eigenvectors 41
outlier 55, 210
overrelaxation 99

This Page Intentionally Left Blank