AN INTRODUCTION TO NUMERICAL LINEAR ALGEBRA

Monographs on Numerical Analysis

L. FOX

AN INTRODUCTION TO NUMERICAL LINEAR ALGEBRA

BY

L. FOX, M.A., D.Sc.

DIRECTOR, UNIVERSITY COMPUTING LABORATORY, AND
PROFESSOR OF NUMERICAL ANALYSIS,
OXFORD

CLARENDON PRESS · OXFORD

Oxford University Press, Ely House, London W.1

GLASGOW NEW YORK TORONTO MELBOURNE WELLINGTON
CAPE TOWN IBADAN NAIROBI DAR ES SALAAM LUSAKA ADDIS ABABA
DELHI BOMBAY CALCUTTA MADRAS KARACHI LAHORE DACCA
KUALA LUMPUR SINGAPORE HONG KONG TOKYO

Casebound: ISBN 0 19 853402 7
Paperback: ISBN 0 19 853407 8

FIRST PUBLISHED 1964
REPRINTED LITHOGRAPHICALLY IN GREAT BRITAIN
FROM CORRECTED SHEETS OF THE FIRST EDITION
BY WILLIAM CLOWES & SONS, LIMITED
LONDON, BECCLES AND COLCHESTER
1967, 1973

Preface

This series of Monographs on Numerical Analysis owes its existence to the late Professor D. R. Hartree who, defying the walrus, thought that the time had come to talk of one thing at a time, at least in this field. Indeed the various areas of Numerical Analysis have expanded so rapidly that it is now virtually impossible to write a single book which gives more than a very elementary introduction in all fields. We even need a variety of books on each single topic, as in other branches of science and mathematics, to meet the various requirements of the undergraduate, the research student, and those who spend their working life in solving numerical problems in specific contexts.

Numerical analysis was introduced in 1959 into the Oxford undergraduate mathematical syllabus, and it seemed to me preferable to talk about numerical linear algebra in the first place and to leave for subsequent courses the theory and practice of approximation and its applications to the solution of differential and integral equations in which, of course, linear algebra plays a large part. The material of this book is therefore based on this first set of lectures, and I generally cover some two-thirds of it in about 28 lectures, treating less thoroughly Chapters 4, 5 and 7 and the later parts of Chapters 8, 10 and 11.

I had considerable difficulty with Chapter 2. Instead of introducing linear equations via vector spaces, linear transformations and matrices I started with linear equations and tried to show how the algebra of matrix manipulation 'hangs together' and simplifies not only our notation but the proofs of our numerical operations. Though this does not give a beautiful mathematical theory I made the deliberate choice for three reasons. First, the Oxford undergraduates learn the mathematical theory from other lecturers. Second, I think that with that theory they do not easily acquire the facility with matrix manipulation which they will need in numerical work and even for the study of more advanced theoretical texts. Third, the Director of a Computing Laboratory must also consider the engineers and scientists who use the computer to solve numerical problems, and in my experience these workers have some antipathy to and even fear of words like 'space', 'rank' and even 'matrix' which, they feel, represent

strange and impractical mathematical abstractions. And yet the use and manipulation of matrices, in the elementary form given here and which is sufficient for many practical purposes, is really very easy! This is true also of 'norms', introduced in an elementary way in this Chapter and which are so valuable for measuring the convergence of series and iterative processes.

And of course mathematical rank and matrix singularity have less importance in practical work. Here the data is rarely exact, and instead of a matrix A we have to consider the matrix $A+\delta A$, where we may know only upper and lower bounds to the elements of δA. Even if A is exact (in a 'mathematical' problem) our numerical methods involve arithmetic which is rarely exact. We must face the fact, and numerical analysts do not apologise for it, that the question of error analysis is profoundly important and that for this purpose we must investigate very closely the details of the arithmetic. The present tendency of error analysis, in all branches of numerical analysis, largely refrains from following through the effects on the solution of each individual error, but accepts these errors and tries to determine what problem we have actually solved. Our methods are then evaluated according to our ability to perform the appropriate analysis and to the size of the upper bounds of the perturbations.

This is considered in Chapter 6. Chapters 3 and 4 study various direct processes for solving linear equations based on elimination and triangular decomposition, and the close relations which exist between the various methods. Many of these, together with the orthogonalisation methods of Chapter 5, might well be discarded for practical purposes, but they have some mathematical interest and considerable literature, and I thought it desirable to collect in one place a summary of the relevant facts. In Chapter 7 I consider very briefly the work and storage requirements for some of the methods, with particular reference to automatic digital computers. Chapter 8 gives an introduction to a class of iterative methods for solving linear equations, whose recent developments, particularly for the large sparse matrices relevant to elliptic differential equations, have been brilliantly expounded by R. S. Varga in his 'Matrix Iterative Analysis' (Prentice-Hall, 1962).

Chapters 9 and 10 discuss the determination of the latent roots and vectors of general matrices, both by iterative methods and by the search for similarity transformations of various kinds, associated with the names of Jacobi, Givens, Householder, Lanczos, Rutishauser,

Francis and others. These have been further developed by J. H. Wilkinson, together with a systematic error analysis whose general features are indicated in Chapter 11, and which are described in comprehensive detail in his forthcoming 'The algebraic eigenvalue problem' (Oxford, this series, in press).

There are, of course, some omissions which will displease many students and teachers. For example I have said very little about computing machines, and have not made detailed distinction between the error analysis of 'fixed-point' and 'floating-point' machine arithmetic. Coding and programming are mentioned only briefly in the introductory chapter, with no details of languages like FORTRAN or ALGOL. My personal opinion is that these things, while relatively easy to learn and master, take much space to describe and the mathematical undergraduate needs essentially the principles expressed in a book which is reasonably short and correspondingly inexpensive. Those who teach ALGOL, moreover can easily use as exercises the algorithms of this book, all of which, I hope, are expressed unambiguously in the language of English and of standard mathematical notation. In a few cases the algorithmic language would simplify the description, and in these cases it is interesting to note that hand computation is relatively tedious; the method of § 30 in Chapter 4 is one example of this. There is, of course, some advantage in using digital computers at the undergraduate stage, and I hope to introduce this at Oxford when we acquire facilities which are not completely saturated by the demands of research.

With regard to notation I have used the prime rather than the superscript T to denote matrix transposition, and usually capital letters denote matrices and lower-case letters denote vectors, in ordinary italic type. Exceptions are the row or column vectors of a matrix, usually denoted respectively by $R_s(A)$ and $C_s(A)$, and I fear that consistency lapses for the residual vector, sometimes called r and sometimes R, I suspect for personal historical reasons. All my matrices, incidentally, have distinct latent roots (which word I use consistently instead of eigenvalues) and consequently a full set of independent latent vectors, with obvious simplifications in the theory and no considerable restriction in practice.

Most of the material is already published in learned journals, and most books on numerical analysis have some account of parts of it. Few similar books, however, are available in English. Predecessors not mentioned in the text include P. S. Dwyer's 'Linear Computations'

(Wiley, 1951), written before the advent of the digital computer and the advances in error analysis, and E. Bodewig's 'Matrix Calculus' (North Holland Publishing Company, Amsterdam, 1959) which has more and deeper theoretical treatment but perhaps fewer practical details. More advanced books include those of Varga, the imminent treatise of Wilkinson, and the latter's just published 'Rounding errors in algebraic processes' (HMSO, 1963), and I hope that my readers will be able subsequently to benefit more easily from these learned works.

It is a pleasure to record my debt to Dr. E. T. Goodwin, who read the proofs and made several valuable suggestions; to Professor A. H. Taub, who invited me to Illinois for a sabbatical semester in which I found time to write several chapters; to the Clarendon Press, who made a special and successful effort to produce this book in time for the 1964 examinations; and above all to Dr. J. H. Wilkinson, who read all the first draft, made important criticisms and suggestions, and from whom I have learnt much.

L. Fox
Oxford, January 1964

Contents

1. INTRODUCTION

Numerical analysis	1
Computer arithmetic	5
Simple error analysis	7
Computing machines, programming and coding	9
Checking	15
Additional notes	16

2. MATRIX ALGEBRA

Introduction	18
Linear equations. General considerations	18
Homogeneous equations	20
Linear equations and matrices	21
Matrix addition and multiplication	22
Inversion and solution. The unit matrix	26
Transposition and symmetry. Inversion of products	29
Some special matrices	31
Triangular matrices. The decomposition theorem	32
The determinant	34
Cofactors and the inverse matrix	36
Determinants of special matrices	37
Partitioned matrices	38
Latent roots and vectors	39
Similarity transformations	42
Orthogonality	44
Symmetry, Rayleigh's Principle. Hermitian matrices	45
Limits, series and norms	48
Numerical methods	51
Additional notes and bibliography	55

3. ELIMINATION METHODS OF GAUSS, JORDAN AND AITKEN

Introduction	60
The Gauss methods	61
Jordan elimination	65
Calculation of the inverse	66
Matrix equivalent of elimination	68
The method of Aitken	75
The symmetric case	79
The symmetric, positive-definite case	80
Exact and approximate solutions. Integer coefficients	82
Determination of rank	87
Complete pivoting	91
Compatibility of linear equations	93
Note on comparison of methods	96
Additional notes and bibliography	97

4. COMPACT ELIMINATION METHODS OF DOOLITTLE, CROUT, BANACHIEWICZ AND CHOLESKY

Introduction 99
The method of Doolittle 99
Connexion with decomposition 102
The method of Crout 102
Symmetric case 104
The methods of Banachiewicz and Cholesky 106
Inversion. Connexion with Doolittle and Crout 110
Inversion. Symmetric case 113
Connexion with Jordan and Aitken 115
Row interchanges 117
Operations with complex matrices 121
Additional notes and bibliography 124

5. ORTHOGONALISATION METHODS

Introduction 125
Symmetric case 126
Unsymmetric case 128
Matrix orthogonalisation 130
Additional notes and bibliography 135

6. CONDITION, ACCURACY AND PRECISION

Introduction 136
Symptoms, causes and effects of ill-conditioning 137
Measure of condition 141
Exact and approximate data 143
Mathematical problems. Correction to approximate solution 143
Mathematical problems. Correction to the inverse 155
Physical problems. Error analysis 158
Relative precision of components of solution 167
Additional notes and bibliography 169

7. COMPARISON OF METHODS. MEASURE OF WORK

Introduction 175
Gauss elimination 175
Jordan elimination 179
Matrix decomposition 180
Aitken elimination 183
Other elimination methods. Symmetry 183
Evaluation and comparison 185
Additional notes 186

8. ITERATIVE AND GRADIENT METHODS

Introduction 189
General nature of iteration 190
Jacobi and Gauss-Seidel iteration 191
Acceleration of convergence 194
Labour and accuracy 202
Consistent ordering 203
Gradient methods 205

Symmetric positive-definite case 207
A finite iterative process 208
Additional notes and bibliography 213

9. ITERATIVE METHODS FOR LATENT ROOTS AND VECTORS

Introduction 215
Direct iteration 216
Acceleration of convergence 220
Other roots and vectors. Inverse iteration 223
Matrix deflation 228
Connexion with similarity transformation 232
Additional notes and bibliography 233

10. TRANSFORMATION METHODS FOR LATENT ROOTS AND VECTORS

Introduction 238
Method of Jacobi, symmetric matrices 238
Method of Givens, symmetric matrices 241
Method of Householder, symmetric matrices 247
Example of Givens and Householder 249
Uniqueness of triple-diagonal form 251
Method of Lanczos, symmetric matrices 252
Method of Lanczos, unsymmetric matrices 255
Vectors of triple-diagonal matrices 258
Other similarity transformations. The L-R method 259
The Q-R method 263
Reduction to Hessenberg form 264
Roots and vectors of Hessenberg matrix 266
Additional notes and bibliography 268

11. NOTES ON ERROR ANALYSIS FOR LATENT ROOTS AND VECTORS

Introduction 275
Ill-conditioning 275
Corrections to approximate roots and vectors 278
General perturbation analysis 281
Deflation perturbation 286
Additional notes and bibliography 288

INDEX 291

NOTE. † indicates that there is a further mention of the section in *Additional notes and bibliography*, given at the end of each Chapter.

1

Introduction

Numerical analysis

1. THIS book is concerned with topics in the field of linear algebra, in particular with the solution of linear equations and the inversion of matrices, and the determination of the latent roots and vectors of matrices. Before embarking on our exposition it is desirable to make some introductory remarks on the nature and general aims of numerical analysis, and on the computing equipment which will enable us, without undue fatigue and in reasonable time, to obtain numerical answers to our problems.

The numerical answer is our aim. The roots of the quadratic equation $x^2+2bx+c = 0$ are

$$x_1, x_2 = -b \pm (b^2-c)^{\frac{1}{2}}, \tag{1}$$

but we are concerned with the evaluation of x_1 and x_2 for given numerical values of b and c. We might, as here, have a 'closed expression' for the answer, in which we merely have to substitute the given numbers, the data of the problem. More commonly there is no simple formula, but there may be an algorithm, represented by an ordered sequence of numerical operations, additions, subtractions, multiplications and divisions, which is known to give the required result. The construction of such algorithms is one of the research activities of numerical analysis.

2. But we must be careful with the phrase 'required result'. An answer is rarely obtainable exactly as an integer or the ratio of two integers. Even for a simple problem like that represented by equation (1) we shall have to compute an irrational number, or non-terminating decimal, for most values of b and c. For example if $b = 1$ and $c = -1$ the required roots are $-1 \pm \sqrt{2}$, and if we want this as a single number we have to specify in advance the precision of our result, that is the number of figures which we should like to have correct. In the decimal scale the number $\sqrt{2}$ is 1·41421356..., and if we specify a precision of p decimals we have to *round* the number appropriately, and in such a way that the error committed is as small as possible.

To do this we *truncate* the number to the precision required, increasing by unity the last digit retained if the first neglected digit is

5, 6, 7, 8 or 9. We thereby ensure that the maximum error committed is not more than five units in the first neglected place, or half a unit in the last figure given, or $0{\cdot}5 \times 10^{-p}$. To three decimals $\sqrt{2} = 1{\cdot}414$, to seven decimals it is 1·4142136, and so on.

3. Even if the computation can in theory be performed with exact integers, moreover, we shall find that our computing machine cannot usually handle the large numbers involved in the arithmetic processes. For example, if we are solving simultaneous linear algebraic equations in n unknowns, in which the coefficients and right-hand sides are given as p-figure integers, an exact process could give the results as the ratios of two integers, both of which would contain np digits. In the more practicable methods which we discuss in this book the integers might contain $p \times 2^{n-1}$ digits. If n is 20, which is by no means large in practical problems, and p is say four, this number is of the order of 2×10^6, and no computing machine can store numbers of this size without complicating prohibitively the task of 'programming' and increasing prohibitively the time of operation.

If the coefficients are given as rational fractions, such as $\frac{1}{7}$, or as irrational numbers like e, π, $\sqrt{2}$ or sin 0·72 (radians), we shall have to round them to a given number of digits. The problem we are solving is then not quite the original problem, and one of our tasks will be to decide how many figures we need to keep in the original data, and also in the process of the computation, to obtain the required precision in the results.

4. Problems in which the data are known exactly, either as integers, rational or irrational numbers, I call *mathematical*. The author of such a problem has a perfect right to ask for any degree of precision which he needs for his purpose. On the other hand most problems with a scientific context will involve data obtained as a result of measurement, in some degree inaccurate, and our task now is to decide the *worthwhile* precision of the answers. Such problems are called *physical*, and it is self-deceptive to quote as answers more digits than those which remain unchanged however the data is varied within its limits of 'tolerance'. The 'required result' now becomes the 'meaningful result', and our methods should decide this for us.

As a trivial example, if we are asked to compute $\sin x$, and a measurement of x gives the value $x = \frac{1}{4}\pi \pm 0{\cdot}005$, we see that there is a range of values of the answer, from about 0·7036 to 0·7106, and a quoted result of 0·7071 has a possible error of $\pm 0{\cdot}0035$. It would clearly

be stupid to quote more than three decimals in the result. We shall see later that the precision of the answer compared with that of the data varies considerably with the problem, and in complicated algorithms our work of determining this might be formidable and challenging.

5. We note also that we would often prefer to use an algorithm, rather than evaluate a closed solution, even when the latter exists. In the field of differential equations, for example, the solution of the first-order equation

$$\frac{dy}{dx} - \frac{2y}{1-x^4} = 0 \tag{2}$$

is

$$y = A\left(\frac{1+x}{1-x}\right)^{\frac{1}{2}} e^{\tan^{-1}x}, \tag{3}$$

where A is an arbitrary constant to be fixed by the specification of y for a particular value of x.

Now this is a useful formula for the computation of y for one or two particular values of x. But it is quite common to want a *graph*, or preferably a *table* of values of y for a set of (usually) equidistant values of x over a lengthy range. The calculation of the expression (3) is then not trivial, involving the evaluations of a square root, an inverse tangent, and an exponential function, in addition to one division and several multiplications. In the computation of these elementary functions, moreover, we shall either have to use some form of series or to interpolate in mathematical tables, and the whole operation is somewhat lengthy. We have numerical methods for solving such problems, though they belong to a field outside our present interest, which perform much less arithmetic and which produce successive values in the table without ever knowing the closed solution (3).

6. The closed solution, of course, is extremely valuable for many purposes, but unfortunately it can rarely be obtained in terms of the so-called 'elementary' functions. For example an apparently innocent change in (2), to the form

$$\frac{dy}{dx} - \frac{2y}{1-x^4} = x, \tag{4}$$

produces the more formidable-looking solution

$$y = \left(\frac{1+x}{1-x}\right)^{\frac{1}{2}} e^{\tan^{-1}x}\left(\int x\left(\frac{1-x}{1+x}\right)^{\frac{1}{2}} e^{\tan^{-1}x}\,dx + A\right). \tag{5}$$

This can hardly be called a solution at all, since we have no analytical methods for evaluating the indefinite integral in terms of elementary

functions, and some *numerical* process has to be used for this purpose. We might just as well use our algorithmic numerical method for the equation (4) without recourse to (5), and in fact the extra numerical work in (4) compared with that of (2) is almost negligible.

7. Again, however, we should not ignore the possibility of obtaining a closed solution, and it is very important that we should understand the mathematics and mathematical methods for our problems, as well as the numerical analysis and possible algorithms. In particular we should try to decide in advance whether our given problem really has a solution, that is whether there is an *existence theorem* for it. With the development of automatic computing machines the mathematical analysis is increasingly important, and it should never be thought that the machine will do the mathematics for us.

Our algorithm may sometimes decide for us whether or not our problem has a solution, or at least a unique solution. For example it is usually the case that a set of simultaneous linear algebraic equations has a unique solution when the number of equations is equal to the number of unknowns. But it is clear that the equations

$$\left.\begin{aligned} x+y &= 3 \\ 2x+2y &= 6 \end{aligned}\right\} \tag{6}$$

do not define a unique solution, the second equation being effectively a restatement of the first. If in the second of (6) the right-hand side were a number other than six it is clear, moreover, that the equations would have no solution at all. This is less obvious with the equations

$$\left.\begin{aligned} x+y+z &= \alpha \\ x-y-z &= \beta \\ 2x+4y+4z &= \gamma \end{aligned}\right\}, \tag{7}$$

which have no unique solution for any α, β and γ, and no solution at all unless $\gamma = 3\alpha-\beta$.

With many equations, and with more digits in the coefficients, we may have some trouble in this context, and the necessity for rounding may produce a solution from our computing machine when in fact no solution exists. We shall give examples of this in a later chapter and show how our algorithm can help to decide the questions. In other fields, notably in the solution of differential equations, our algorithm may be less valuable in the determination of existence, and mathematical analysis is essential.

8. Summarizing, we can say that numerical analysis is concerned with the production of numerical solutions to scientific and mathematical problems. Our aim is to find methods which are economic in time, which produce the results to the accuracy requested in mathematical problems, and which tell us how many figures are worth quoting in physical problems. To the numerical analysis we should add any mathematical knowledge we have or can find about the existence of solutions, and in some sense our methods, like those of mathematics itself, should be elegant!

As a rather trivial example of elegance we might consider the formula (1) for the solution of quadratic equations. If b^2-c is reasonably small, and we compute its square root to a given number of decimal places, the formula gives roughly the same number of correct digits in both roots. But if c is small, so that $(b^2-c)^{\frac{1}{2}} = b+\epsilon$, where ϵ is small, then $x_2 = -2b-\epsilon$, $x_1 = \epsilon$, and x_2 is given accurately with many more digits than x_1. To avoid computing the square root to more figures we use our mathematics to note that $x_1x_2 = c$, so that $x_1 = c/x_2$ and can be computed from this formula with a relative accuracy similar to that of x_2.

The loss of significant digits in subtracting large numbers is a common phenomenon, and we use all possible methods to avoid or mitigate the consequences thereof.

Computer arithmetic

9. There are two methods in common use for operating with numbers in a computing machine. In both cases the numbers are stored in registers of fixed length, so that we can retain only p digits say, in any given number, and a number containing more than p digits must be truncated or, with extra effort, stored in two or more such registers. In what follows we assume that we are working in the common decimal system.

With 'single-length' arithmetic, with p digits, we have either the *fixed-point* or the *floating-point* method of operation. In the fixed-point method it is customary to limit the size of numbers which may occur to the range -1 to $+1$, and any number outside this range must be scaled appropriately by dividing by a power of 10. The programmer must take definite steps to keep track of these scale factors so that the correct result can finally be obtained.

Since our machine can only store digits we must turn the positive and negative signs into quasi-digital form, and this we do with the

convention that all positive numbers have their first digit zero. The decimal point will normally be thought to follow this digit, so that in a four-digit register we can effectively store three figures. The number 0·924 will actually appear in that form, and the largest positive number we can store is 0·999, the integer after the decimal point being 10^p-1 in a $(p+1)$ register machine.

For a negative number x we store the complement $10^{p+1}-|x|$, so that the first digit is always 9, and the number $-0{\cdot}924$ appears as 9·076. All negative numbers have nine as the first digit, and the largest negative number we can store is 9·000, which is -1 in the 'signed' convention, the 'fractional part' representing the integer 10^p.

It is easy to see that addition and subtraction, using the complements of negative numbers, will always give the correct answers in the 'signed' convention provided that the result is in the allowed range. In fact in a sequence of such operations the intermediate results are allowed to exceed the range. For example $0{\cdot}126-0{\cdot}125 = 0{\cdot}126+9{\cdot}875 = 10{\cdot}001$. The first digit is 'lost' and we are left with 0·001, the true result. Again, $0{\cdot}125-0{\cdot}126 = 0{\cdot}125+9{\cdot}874 = 9{\cdot}999 = -0{\cdot}001$, again correct. The sum $0{\cdot}986+0{\cdot}125 = 1{\cdot}111$ cannot be allowed, however, and we would have to store this in the rounded form $0{\cdot}111\times10^1$, remembering the power of 10 involved. But

$$0{\cdot}986+0{\cdot}125-0{\cdot}389 = 0{\cdot}986+0{\cdot}125+9{\cdot}611 = 10{\cdot}722 = 0{\cdot}722,$$

and this is correct.

When we multiply together two permissible numbers the result is certain to be within range. But the exact product of two numbers of p digits has $2p$ digits, and we need two registers to store it exactly, a so called 'double-length' accumulator. If we have to round it to single length we commit an error of maximum amount $0{\cdot}5\times10^{-p}$. The division a/b is out of range if $a > b$, but otherwise we can perform the calculation. In a 'single-length' register the stored result will have a maximum error of $0{\cdot}5\times10^{-p}$, unless the resulting decimal number terminates in at most p digits.

10. In the floating-point system our numbers can be of almost any size, and we store them in the form $10^a\times b$, making space in our register for both a and b. This representation is not unique, but we standardize by choosing b in the range $0{\cdot}1 \leqslant |b| < 1$. For example the number 1562 is stored as $0{\cdot}1562\times10^4$, 0·001562 is given as $0{\cdot}1562\times10^{-2}$. Both a and b can be negative, and are stored with the signed

convention, though a is always an integer and we can forget about the decimal point in its register.

Here the user is not worried by scaling problems and the machine automatically keeps track of the relevant powers of ten. 'Overflow' of the accumulator is now almost solely restricted to the case of division by zero, and otherwise the size of allowable numbers is governed by the size of the register we allow for the representation of the exponent a. We shall mention some other relevant facts about arithmetic in the appropriate contexts.

Simple error analysis

11. The fixed-point and floating-point representations introduce the ideas of *decimal places* and *significant figures*. Both the numbers 0·9246 and 0·0002 have four decimal places and would be stored in this form in the fixed-point method. The first number, however, has four significant figures whereas the second has only one significant figure. The point about the word 'significant' is that, if these numbers were obtained as a result of rounding with a possible maximum error of half a unit in the last place retained, each has a possible *absolute* error of $\pm 0{\cdot}00005$, but the former has a much smaller *relative* error. It is correct to approximately one part in 20,000, while the number 0·0002 is correct only to one part in 4.

In the floating-point representation these numbers are stored respectively as $0{\cdot}9246 \times 10^{0}$ and $0{\cdot}2000 \times 10^{-3}$. Here the number of non-zero digits in the fractional part represents the number of significant figures present, the three zeros in the second example being inserted to fill up the register. If we had more *significant* information about this value, for example that it was 0·0002329..., or 0·0002000 where the last three zeros are known to be correct, we could store it in a floating-point form like $0{\cdot}2329 \times 10^{-3}$ with a small relative error, whereas the rounded fixed-point number 0·0002 has a small absolute error but a large relative error. This, incidentally, does not imply that the floating-point representation is superior. There are many factors involved, some of which we shall mention later. We note immediately, however, that in an addition like $0{\cdot}9246 \times 10^{0} + 0{\cdot}2329 \times 10^{-3}$ we have first to express the smaller number in the rounded form $0{\cdot}0002 \times 10^{0}$ in order to add it to the first, and we have had to discard its last three digits.

12. We shall need rules for assessing both types of error in simple operations, so that we can extend them to complicated situations.