WILKINSON

Rounding Errors in Algebraic Processes

PRENTICE-HALL SERIES IN AUTOMATIC COMPUTATION

Rounding Errors in Algebraic Processes

J. H. WILKINSON M.A., Sc.D.
National Physical Laboratory
Teddington, Middlesex, England.

PRENTICE-HALL, INC.
Englewood Cliffs, N. J.

Published in the U.S.A. by
PRENTICE-HALL, INC.
Englewood Cliffs, N.J.

Printed in England by Her Majesty's Stationery Office.

PREFACE

The development of automatic digital computers has made it possible to carry out computations involving a very large number of arithmetic operations and this has stimulated a study of the cumulative effect of rounding errors. In this book I have given an elementary introduction to this subject which is based on the work which has been done in the Mathematics Division of N.P.L. in the past few years. Some of the material presented here has already appeared in published papers, but much of it has hitherto been available only in the form of rough notes for lectures given in this country and the United States.

Error analysis is regarded by many users of digital computers as a subject for experts. This view is probably the result of attempting to read, at widely separated intervals, isolated analyses presented from entirely different standpoints. In this book I have attempted to present a number of simple analyses in a uniform manner. This has made it necessary to be somewhat selective in the choice of material but it is hoped that, in spite of this, the analyses are reasonably representative. It is my hope that in this way the ideas will be readily assimilated by anyone who can understand the techniques which are analysed.

The first draft was read and criticized helpfully by a number of colleagues in Mathematics Division and I am particularly indebted to E. T. Goodwin and D. W. Martin. I would also like to thank G. E. Forsythe, Director of the Computation Center, Stanford, for numerous helpful suggestions. I am very grateful to Mrs. I. Goode for her invaluable assistance in seeing the book through the press.

J. H. WILKINSON
DEPUTY CHIEF SCIENTIFIC OFFICER
NATIONAL PHYSICAL LABORATORY

NATIONAL PHYSICAL LABORATORY
TEDDINGTON, MIDDLESEX
1963
January 1963

(86262) A 2

CONTENTS

I

THE FUNDAMENTAL ARITHMETIC OPERATIONS

DIGITAL COMPUTATION

1. In this book we shall be concerned exclusively with computation using digital computers. Usually the results we give are important only in the cases when the computation is so extensive that the use of an automatic digital computer is essential, but the analysis will apply equally well to calculations on a desk machine.

The majority of automatic digital computers which are used for scientific computation, work with numbers which are expressed in the binary scale, but since it is impractical to display binary computations, all the examples have been computed on a desk machine working in the decimal scale. Usually we shall express the error analysis directly in terms of binary computation; the corresponding results for decimal computation follow almost immediately, and apart from describing the essential differences in the next few sections we shall not refer to this matter again.

FIXED-POINT AND FLOATING-POINT COMPUTATION

2. There are two main modes of operation which are commonly used on automatic computers. The first mode is called *fixed-point* computation. In this mode the computation must be framed so that every computed number x, satisfies certain inequalities such as

$$-1 \leqslant x \leqslant 1. \tag{2.1}$$

Often the number $+1$ is excluded and, less frequently, the number -1, but as this complicates the details of the analysis in an inessential way, we shall assume that the permissible range is that given by (2.1). In general, each number will be allowed a fixed number t, of binary (decimal) digits for its representation, and we shall say that the computer works with *words* of t binary (decimal) digits. If it is necessary to work to a higher precision than 1 part in 2^t then we may employ numbers which are represented by a multiple of t binary digits, and we shall refer to this as *multiple-precision computation* as distinct from the standard *single-precision computation.* Normally, multiple-precision computation will require the use of special subroutines and each arithmetic operation will take several times as long as the corresponding single-precision operation.

3. The other mode of computation is called *floating-point* computation. In this mode each number x is represented by an ordered pair a and b such that $x = 2^b(a)$. Here b is an integer, positive or negative, and a is a number satisfying

$$-\tfrac{1}{2} \geqslant a \geqslant -1 \quad \text{or} \quad \tfrac{1}{2} \leqslant a \leqslant 1. \tag{3.1}$$

The number b is commonly called the *exponent* or the *index*, and the number a is called the *mantissa* or the *fractional* part. Again one of the end points of each of the intervals (3.1) may be excluded, but as with fixed-point computation we ignore this possibility.

Quite commonly the number of digits allocated to a and b together, is the same as that available for a fixed-point number. A typical division of the word is 8 binary digits in b and 32 binary digits in a for a computer with a word of 40 binary digits. This means that fixed-point computation is usually *capable* of giving higher precision, though often, in order to ensure that numbers do not grow out of the permissible range, it is convenient to work for most of the time with numbers which are appreciably smaller than unity. Some or all of the potential advantage is then thrown away in order to avoid the very detailed analysis which would otherwise be necessary.

4. On computers in which floating-point operations are not provided directly by the machine code, it is not uncommon to allocate two words for floating-point numbers, one for a and one for b, since this makes the subroutines faster. We shall use the same sign t, to denote the number of digits in a fixed-point number and in the mantissa of a floating-point number, and shall refer in both cases to computation with a precision of t binary digits. It will be assumed that the number of digits in b is adequate to ensure that no computed floating-point number will lie outside the permissible range. As in fixed-point computation, it is possible to use multiple-precision arithmetic in which the mantissa is represented by a multiple of t binary digits.

5. The floating-point representation of the number zero differs from one computer to another. On some computers it has a special representation, the mantissa having the value zero. On others, it is represented by taking the index b to be some very large negative number. The decision is not very important, and we shall assume here that the zero mantissa is used.

NOTATIONAL CONVENTIONS

6. Two main forms of error analysis are used in this book and are referred to as *forward* and *backward* analysis respectively. The precise significance of the two techniques will become apparent only when we consider specific analyses in detail, but the general principles may be described as follows.

We consider forward analysis first. We may regard the computation in question as being described by a number of mathematical equations. In each equation some new quantity x is defined in terms of previously computed quantities $a_1, a_2, \ldots, a_n$ (say), where some of the quantities may be initial

data. We may write the mathematical equation in the form

$$x = g(a_1, a_2, \ldots, a_n) = g(a_i). \tag{6.1}$$

The derivation of x from the a_i must involve only the fundamental arithmetic operations. (We are not concerned here with the errors implicit in replacing continuous operations such as integration by finite-difference or other approximations.) Now because of the rounding errors made in the calculations, the computed value of x will be different from that obtained if $g(a_i)$ were evaluated exactly. In forward analysis we denote the computed value by $\bar{x}$, and attempt to obtain a bound for $|\bar{x} - g(a_i)|$. An essential feature of the method then is a comparison of $\bar{x}$ with x.

In backward error analysis we do not concern ourselves with the differences between the computed values and the true values at each step. Instead we endeavour at a typical stage to show that the computed value obtained by interpreting (6.1) is exactly equal to $g(a_1+\epsilon_1, a_2+\epsilon_2, \ldots, a_n+\epsilon_n)$ for some values of the ϵ_i, and to give bounds for these ϵ_i. *It is not essential that every individual computed value should be representable in this form.* Obviously the ϵ_i will not in general be unique.

Since, in the backward analysis, we are not concerned with a comparison between x and $\bar{x}$, and indeed will make no reference to the true x, there seems little point in using a separate symbol to denote the computed value. We therefore dispense with the bar and use x to denote the computed value itself. Hence corresponding to the mathematical equation (6.1) defining x, we shall have a computational equation defining the computed x, of the form

$$x \equiv g(a_1+\epsilon_1, a_2+\epsilon_2, \ldots, a_n+\epsilon_n) \tag{6.2}$$

followed by inequalities satisfied by the ϵ_i. To emphasize that it is a computational equation we shall use an equivalence sign. It might be felt that this notation would cause confusion because x is used in a different sense in (6.1) and (6.2) for example. In fact confusion is avoided because at no stage do we concern ourselves with the value of x defined by (6.1).

In practice we have found that the backward analysis is often much the simpler, particularly in connexion with floating-point computation. It may well be objected that backward analysis is incomplete since ultimately we must be concerned with the difference between the *computed* solution to a problem and its *exact* solution. This is indeed true and there is always a final stage at which this difference must be assessed. We may illustrate this in connexion with the eigenvalue problem. Suppose we have shown that a practical eigenvalue technique gives, for the eigenvalues of A, the exact eigenvalues of some $A + E$ and that we can give bounds for the elements of E. *Then to complete the analysis we must give bounds for the effect of the perturbation matrix E on the eigenvalues of A.*

7. Since the backward error analysis will be used more frequently, we describe the errors in the fundamental arithmetic operations in terms of the notation appropriate to this form of analysis, that is, we shall not use bars

for computed values. It will often be convenient to emphasize whether an analysis is appropriate to floating-point or fixed-point*. We shall do this on occasion by writing for example

$$d = fi(ab+c) \quad \text{or} \quad d = fl(ab+c) \tag{7.1}$$

to denote that d is the value obtained by computing $ab+c$ using fixed-point or floating-point respectively. In expressions such as (7.1) it is to be assumed that computation proceeds from left to right.

ROUNDING ERRORS IN FIXED-POINT COMPUTATION

8. We now consider the rounding errors made in the fundamental arithmetic operations. Addition and subtraction involve no round-off in fixed-point arithmetic though the possibility of the sum or difference lying outside the permitted range has to be borne in mind. If the quantity c is defined by

$$c = a \pm b \tag{8.1}$$

in the mathematical equations, then in the computational equations it will also be defined by

$$c \equiv a \pm b. \tag{8.2}$$

The exact product of two t-digit numbers lying in the interval $[-1, 1]$ is, in general, a number requiring $2t$ digits for its representation. This exact product is replaced by the t-digit approximation obtained by adding $\frac{1}{2}2^{-t}$ (or $\frac{1}{2}10^{-t}$) and discarding digits $t+1$ to $2t$. Hence corresponding to the mathematical equation

$$c = ab \tag{8.3}$$

we have the computational equation

$$\left.\begin{aligned} c \equiv ab + \epsilon \qquad & |\epsilon| \leqslant \tfrac{1}{2}2^{-t}(\text{binary}) \\ & |\epsilon| \leqslant \tfrac{1}{2}10^{-t}(\text{decimal}). \end{aligned}\right\} \tag{8.4}$$

The exact product and the rounded product always lie in the permitted range. As an example we take $a = 0{\cdot}6131$ and $b = 0{\cdot}8432$ and consider 4-digit decimal arithmetic. We have $ab = 0{\cdot}5169\ 6592$ and hence $c = 0{\cdot}5170$, giving

$$c \equiv ab + \epsilon \tag{8.5}$$

$$\epsilon \equiv 0{\cdot}0000\ 3408. \tag{8.6}$$

Note that in the computational equation (8.5), ab denotes the *exact product of a and b*, and c denotes the *computed value.*

On some binary machines a slightly different rounding procedure is used in which the last t digits of the exact product are discarded and the tth digit is replaced by a 1. This gives an error lying in the range $\pm 2^{-t}$ and

* For brevity, we shall frequently write 'floating-point' as a shorthand form of 'floating-point computation', and similarly for 'fixed-point computation'.

therefore a maximum error which is twice that of the previous method of rounding. The difference is of no great importance, and we shall assume the earlier round-off procedure throughout. It has the advantage that it gives the exact result when this is a number which does not require digits $t+1$ to $2t$ for its representation.

9. The quotient $\frac{a}{b}$ of two t-digit fixed-point binary (decimal) numbers will not lie in the permitted range unless $|b| \geqslant |a|$. This means that the use of division in fixed-point arithmetic always demands special consideration. Further, the quotient will, in general, be a non-terminating number. The exact quotient may be rounded to give a t-digit fixed-point number by adding $\frac{1}{2}2^{-t}$ $(\frac{1}{2}10^{-t})$ and retaining only digits 1 to t. Note that we need compute only the first $t+1$ digits of the exact quotient in order to derive the rounded result. In practice the rounded quotient is often obtained by adding half the divisor to the dividend before commencing the division. Only the first t digits are then required. As in the case of multiplication, we may use alternative rounding procedures but the one we have mentioned has the advantage of giving the exact quotient when it is representable by a number with t binary places.

As an example we take $a = 0{\cdot}0635$, $b = 0{\cdot}6673$ and work in 4-decimal arithmetic. The mathematical equation defining c is

$$c = \frac{a}{b}. \tag{9.1}$$

Now the exact quotient is a non-terminating number of which the first six decimal places are $0{\cdot}095159\ldots$. Hence the computed value of c is $0{\cdot}0952$, and the computational equation is

$$c \equiv \frac{a}{b} + \epsilon \tag{9.2}$$

$$\epsilon = 0{\cdot}0000\ 40\ldots. \tag{9.3}$$

The rounding error ϵ will clearly satisfy

$$|\epsilon| \leqslant \tfrac{1}{2}2^{-t}(\tfrac{1}{2}10^{-t}) \tag{9.4}$$

in all cases. As with (8.5) note that in (9.2), $\frac{a}{b}$ denotes the *exact quotient* and c the *computed quotient*. All arithmetic operations in equation (9.2) are exact mathematical operations so that we can manipulate them using the ordinary rules of arithmetic. Multiplying (9.2) by b we have

$$\begin{aligned} bc &\equiv a + b\epsilon \\ |bc - a| &\equiv |b\epsilon| \\ &\leqslant |b|\tfrac{1}{2}2^{-t}. \end{aligned} \tag{9.5}$$

The bound we have for the difference between bc and a is never greater than $\frac{1}{2}2^{-t}$ because b is a fixed-point binary number, but this bound diminishes with b. This trivial result is often important in error analysis.

FIXED-POINT ACCUMULATION OF INNER-PRODUCTS

10. Nearly all digital computers produce, in the first instance, the exact $2t$-digit product of two t-digit numbers and on many computers advantage is taken of this fact to obtain more accurate results. A very common computational requirement is the calculation of an inner-product s, defined by

$$s = \sum_{1}^{n} a_i b_i . \tag{10.1}$$

If each product in this expression is rounded separately, then the corresponding computational equation is

$$\left.\begin{aligned} s &\equiv \sum_{1}^{n} a_i b_i + \epsilon \\ |\epsilon| &\leqslant \tfrac{1}{2} n 2^{-t}. \end{aligned}\right\} \tag{10.2}$$

However, on most desk computers, the inner-product is accumulated exactly to the full $2t$ figures, and this exact result is obtained whether one wants it or not. We may say that a double-precision result is obtained *although the time taken by the computation is essentially that which we associate with single-precision work.* A similar facility is provided on a number of automatic digital computers. If we can take advantage of this facility, then the computational equation corresponding to (10.1) is

$$\left.\begin{aligned} s &\equiv \sum_{1}^{n} a_i b_i + \epsilon \\ |\epsilon| &\leqslant \tfrac{1}{2} 2^{-t}. \end{aligned}\right\} \tag{10.3}$$

The bound for the rounding error is reduced by the factor n.

Computers which are provided with this feature have an accumulator which can store a $2t$-digit number. Furthermore, the division facility of such computers is quite commonly designed to take advantage of the existence of the double-precision accumulator. The division facility provides for the division of a single-precision divisor into the number in the double-precision accumulator to give a correctly rounded single-precision quotient. When the dividend is a t-digit number it is extended to $2t$-digit form by the addition of t zeros. Again it may be remarked that all desk computers have this feature.

11. Consider now the computation of d, defined by the mathematical equation

$$d = \sum_{1}^{n} a_i b_i / c . \tag{11.1}$$

If both facilities are provided then $\sum_1^n a_i b_i$ may be computed exactly and c may be divided into this double-precision number. The computational equation corresponding to (11.1) is therefore

$$\left.\begin{aligned} d &\equiv \left(\sum_1^n a_i b_i / c\right) + \epsilon \\ |\epsilon| &\leqslant \tfrac{1}{2} 2^{-t} \end{aligned}\right\} \tag{11.2}$$

and hence

$$\left|cd - \sum_1^n a_i b_i\right| \equiv |c\epsilon| \leqslant \tfrac{1}{2}|c| 2^{-t}. \tag{11.3}$$

The bound given in (11.3) is directly proportional to $|c|$ and if this is much smaller than unity, the bound is much smaller than $\frac{1}{2}2^{-t}$.

12. The importance of these facilities may be illustrated by a simple example. Let d be defined mathematically by

$$d = (0{\cdot}6325 \times 0{\cdot}4126 - 0{\cdot}3127 \times 0{\cdot}8313)/0{\cdot}0013.$$

If both facilities are provided, the numerator is given exactly as $0{\cdot}2609\ 6950 - 0{\cdot}2599\ 4751 = 0{\cdot}0010\ 2199$, and we have

$$0{\cdot}0010\ 2199/0{\cdot}0013 = 0{\cdot}7861\ 4 \ldots .$$

Hence the computed d is $0{\cdot}7861$ with error $0{\cdot}0000\ 4 \ldots$. Moreover, the computed d satisfies

$$d \times 0{\cdot}0013 - (0{\cdot}6325 \times 0{\cdot}4126 - 0{\cdot}3127 \times 0{\cdot}8313) \equiv -0{\cdot}0000\ 0006.$$

If neither facility is provided then the computed numerator is given by

$$0{\cdot}2610 - 0{\cdot}2599 = 0{\cdot}0011$$

and since $0{\cdot}0011/0{\cdot}0013 = 0{\cdot}8461\ 5 \ldots$, the computed value of d is $0{\cdot}8462$. This has an error of $0{\cdot}0600\ 5 \ldots$, which *is more than 1,000 times as large.* The computed d satisfies

$$d \times 0{\cdot}0013 - (0{\cdot}6325 \times 0{\cdot}4126 - 0{\cdot}3127 \times 0{\cdot}8313) \equiv 0{\cdot}0000\ 7807,$$

and the error is again more than 1,000 times as large.

ROUNDING ERRORS IN FLOATING-POINT COMPUTATION

13. The details of the fundamental floating-point arithmetic operations differ a little from one computer to another. On a computer having a double-precision accumulator, the operations are as follows.

We denote the operands in each arithmetic operation by x_1 and x_2 where

$$x_1 = 2^{b_1} a_1 \text{ (or } 10^{b_1} a_1), \quad x_2 = 2^{b_2} a_2 \text{ (or } 10^{b_2} a_2) \tag{13.1}$$

and consider addition first. Suppose x_1 is the number with the larger modulus. Then the integer $(b_1 - b_2)$ is computed.

(i) If $b_1 - b_2 > t$ then x_2 is too small to have any effect as far as the first t significant digits of the sum are concerned, and we have

$$fl(x_1+x_2) \equiv x_1. \tag{13.2}$$

(ii) If $b_1 - b_2 \leqslant t$, then a_2 is divided by $2^{b_1-b_2}$ (or $10^{b_1-b_2}$) by shifting it b_1-b_2 places to the right. The sum $a_1 + 2^{b_2-b_1}a_2$ is then calculated exactly, and clearly requires less than $2t + 1$ digits for its representation. This sum is then multiplied by the appropriate power of 2 (or 10), using a left shift or a right shift, so that the resulting number lies in the range permitted for the mantissa of a floating-point number, and the index b_1 is adjusted to deal with this shift. Finally this $2t$-digit mantissa is rounded to t digits.

Note that since

$$|a_1| + 2^{b_2-b_1}|a_2| \leqslant 1 + 1 \tag{13.3}$$

the maximum right shift that can be required is one position. On the other hand considerable cancellation may take place, so that a left shift of up to t places may be necessary. If x_2 has the larger modulus, then the roles of x_2 and x_1 are interchanged.

14. The process may be illustrated by a number of simple examples of addition of numbers in 4-digit floating-point decimal arithmetic.

(i) $$10^{-6}(0\cdot3127) + 10^{4}(0\cdot4153).$$

The index of x_1 is 10 less than that of x_2. Hence the floating-point sum is $10^4(0\cdot4153)$.

(ii) $$10^{4}(0\cdot6314) + 10^{1}(0\cdot3865).$$

a_2 is shifted 3 places to the right and the addition takes place in 8-figure arithmetic

$$\begin{array}{r} 10^4 \times 0\cdot6314\ 0000 \\ +\ 10^4 \times 0\cdot0003\ 8650 \\ \hline 10^4 \times 0\cdot6317\ 8650 \\ \hline \end{array}$$

The exact sum is rounded to give $10^4(0\cdot6318)$.

(iii) $$10^{4}(0\cdot7418) + 10^{4}(0\cdot6158).$$

The addition takes place in the form

$$\begin{array}{r} 10^4 \times 0\cdot7418\ 0000 \\ +\ 10^4 \times 0\cdot6158\ 0000 \\ \hline 10^4 \times 1\cdot3576\ 0000 = 10^5 \times 0\cdot1357\ 6000. \\ \hline \end{array}$$

The exact sum is rounded to give $10^5(0\cdot1358)$.

(iv) $10^{-4}(0\cdot7617) + 10^{-4}(-0\cdot7613)$.

$$\begin{array}{rl} & 10^{-4} \times 0\cdot7617\ 0000 \\ - & 10^{-4} \times 0\cdot7613\ 0000 \\ \hline & 10^{-4} \times 0\cdot0004\ 0000 = 10^{-7}(0\cdot4000). \end{array}$$

Severe cancellation took place *and the computed sum is exact.*

(v) $10^{-4}(0\cdot1005) + 10^{-5}(-0\cdot9963)$.

$$\begin{array}{rl} & 10^{-4} \times 0\cdot1005\ 0000 \\ - & 10^{-4} \times 0\cdot0996\ 3000 \\ \hline & 10^{-4} \times 0\cdot0008\ 7000 = 10^{-7}(0\cdot8700). \end{array}$$

Cancellation again took place and *the computed sum is exact.*

It is evident from the examples, that the computed value is always that which would be obtained by calculating the exact sum, normalizing it so that it lies in the permitted range, and then rounding it to t places. If the normalized exact sum is $2^{b_3}a_3$ (or $10^{b_3}a_3$) then it is evident that the modulus of the error is bounded by $2^{b_3} \times \frac{1}{2}2^{-t}$ (or $10^{b_3} \times \frac{1}{2}10^{-t}$). We shall occasionally use this form of the error bound, but more frequently we shall be concerned with the relative error. Now the modulus of the exact sum lies between $\frac{1}{2}2^{b_3}$ and 2^{b_3} (or $\frac{1}{10}\ 10^{b_3}$ and 10^{b_3}) and hence we have

$$fl(x_1+x_2) \equiv (x_1+x_2)(1+\epsilon) \tag{14.1}$$

$$|\epsilon| \leqslant 2^{-t} \text{ (binary)} \tag{14.2}$$

$$|\epsilon| \leqslant \tfrac{1}{2}10^{1-t} \text{ (decimal)}. \tag{14.3}$$

We see that decimal computation is somewhat less satisfactory than binary.

We have not yet considered the case when either x_1 or x_2 is zero. If x_2 is zero then $fl(x_1+x_2) \equiv x_1$ and if x_1 is zero $fl(x_1+x_2) \equiv x_2$ so that no rounding error is involved, and (14.1) holds with $\epsilon = 0$. The computed sum therefore always has a low relative error. *In all cases when cancellation takes place,* $\epsilon = 0$. We stress this point, since cancellation is commonly associated with a loss of precision. (c.f. section 17 in which we discuss computers in which cancellation in addition and subtraction do not always correspond to $\epsilon = 0$.)

We may express our result in the form— *the computed sum of* x_1 *and* x_2 *is the exact sum of two numbers* $x_1(1+\epsilon)$ *and* $x_2(1+\epsilon)$, *for some value of* ϵ *satisfying* $|\epsilon| \leqslant 2^{-t}$ (or $\frac{1}{2}10^{1-t}$).

The results for subtraction are precisely analogous with those for addition.

Multiplication

15. The product of x_1 and x_2 is determined as follows.

The exponents b_1 and b_2 are added together to give b_3, and the exact $2t$-digit product of a_1 and a_2 is computed. This product satisfies

$$\tfrac{1}{4} \leqslant |a_1 a_2| \leqslant 1 \tag{15.1}$$

$$\text{or } \tfrac{1}{100} \leqslant |a_1 a_2| \leqslant 1 \tag{15.2}$$

and is therefore normalized if necessary by a shift to the left, the exponent being adjusted accordingly. The resulting $2t$-digit product is then rounded to give the t-digit mantissa of the computed product. If either x_1 or x_2 (or both) is zero then the computed product is taken to be zero.

As examples we have:—

(i) $10^{-4}(0{\cdot}8132) \times 10^{6}(0{\cdot}6135) = 10^{2}(0{\cdot}4988\ 9820)$.

Hence the computed product is $10^{2}(0{\cdot}4989)$.

(ii) $10^{-3}(0{\cdot}1213) \times 10^{-4}(0{\cdot}1714) = 10^{-7}(0{\cdot}0207\ 9082)$

$$= 10^{-8}(0{\cdot}2079\ 0820).$$

Hence the computed product is $10^{-8}(0{\cdot}2079)$.

As with addition or subtraction, the computed product is obtained by rounding the exact product to t places and hence

$$fl(x_1 \times x_2) \equiv x_1 x_2(1+\epsilon) \tag{15.3}$$

$$|\epsilon| \leqslant 2^{-t} \text{ (binary)} \tag{15.4}$$

$$\text{or } |\epsilon| \leqslant \tfrac{1}{2}10^{1-t} \text{ (decimal)}. \tag{15.5}$$

We may express this in the form— *the computed product is the exact product of* $x_1(1+\epsilon)$ *and* x_2, *or of* x_1 *and* $x_2(1+\epsilon)$, *or of* $x_1(1+\epsilon)^{1/2}$ *and* $x_2(1+\epsilon)^{1/2}$, *where* $|\epsilon| \leqslant 2^{-t}$ (*or* $\frac{1}{2}10^{1-t}$). We are free to associate the factor $(1+\epsilon)$ with x_1 or x_2, or to share it between them in any way which may prove convenient.

Division

16. Division is a more satisfactory process in floating-point than fixed-point; the only case which is not permissible is division by zero. The quotient of x_1 divided by x_2 is determined as follows.

The exponent b_2 is subtracted from b_1 to give b_3. a_1 is placed in the t most significant digits of the double-length accumulator and zero in the t least significant digits. If $|a_1| > |a_2|$ then the number in the accumulator is shifted one place to the right and b_3 is increased by 1. The number in the accumulator is then divided by a_2 to give a correctly rounded t-digit quotient. This must have a modulus lying between $\frac{1}{2}$ and 1 (or $\frac{1}{10}$ and 1) and hence is in normalized form.

Examples

(i) $10^{-6}(0\cdot9137) \div 10^{-2}(0\cdot1312)$.

We convert this to $10^{-4}(0\cdot9137\ 0000) \div 0\cdot1312$ and then to $10^{-3}(0\cdot0913\ 7000) \div 0\cdot1312 = 10^{-3}(0\cdot6964\ 17\ldots)$. The computed quotient is therefore $10^{-3}(0\cdot6964)$.

(ii) $10^{4}(0\cdot1235) \div 10^{-6}(0\cdot9872)$.

This is converted to

$$10^{10}(0\cdot1235\ 0000) \div 0\cdot9872 = 10^{10}(0\cdot1251\ 01\ldots).$$

The computed quotient is therefore $10^{10}(0\cdot1251)$.

If x_1 is zero and x_2 is not zero then $fl(x_1/x_2) \equiv 0$. Hence in all cases, except when $x_2 = 0$, we have

$$fl(x_1/x_2) \equiv (x_1/x_2)(1+\epsilon) \tag{16.1}$$

$$|\epsilon| \leqslant 2^{-t} \text{ (binary)} \tag{16.2}$$

$$|\epsilon| \leqslant \tfrac{1}{2}10^{1-t} \text{ (decimal)}. \tag{16.3}$$

We may express this in the form— *the computed quotient of x_1 and x_2 is the exact quotient of $x_1(1+\epsilon)$ and x_2, or of x_1 and $x_2/(1+\epsilon)$, for some ϵ satisfying $|\epsilon| \leqslant 2^{-t}$ (or $\frac{1}{2}10^{1-t}$).*

ROUND-OFF WITH SINGLE-PRECISION ACCUMULATOR

17. On a computer with a double-precision accumulator the computed sum or difference of two floating-point numbers always has a low relative error. On a computer without such an accumulator, the floating-point operations are usually a little less satisfactory.

The steps by which addition is performed will probably best be illustrated by reconsidering the examples of section 14.

In examples (i), (iii) and (iv) there is no essential difference, since no use is made of the second half of a double-precision accumulator in any case.

In example (ii) $0\cdot3865$ is shifted 3 places to the right *and then rounded to four decimal places.* Thus the addition takes place in the form

$$10^{4} \times 0\cdot6314 + 10^{4} \times 0\cdot0004 = 10^{4}(0\cdot6318).$$

The computed sum is again the same.

In example (v) $0\cdot9963$ is shifted one place to the right *and then rounded.* The addition takes place in the form

$$\begin{aligned} 10^{-4} \times 0\cdot1005 - 10^{-4} \times 0\cdot0996 &= 10^{-4}(0\cdot0009) \\ &= 10^{-7}(0\cdot9000). \end{aligned}$$

The computed sum differs from that obtained previously; *moreover it now has quite a high relative error whereas it was previously exact.* We can no longer write

$$fl(x_1+x_2) \equiv (x_1+x_2)(1+\epsilon) \tag{17.1}$$

with a value of ϵ which is of order 2^{-t}. However, we shall show that it is still true in all cases that

$$fl(x_1+x_2) \equiv x_1(1+\epsilon_1) + x_2(1+\epsilon_2) \tag{17.2}$$

where ϵ_1 and ϵ_2 are of order 2^{-t}, but are in general different.

There is an extra case (vi), which is different from any which occurs when we have a double-precision accumulator.

(vi) $$10^2(0\cdot9622) + 10^1(0\cdot3926).$$

With a double-precision accumulator we proceed thus

$$10^2(0\cdot9622\ 0000) + 10^2(0\cdot0392\ 6000) = 10^2(1\cdot0014\ 6000)$$

giving $10^3(0\cdot1001)$ as the computed sum.

On the other hand, with a single-precision accumulator we have

$$10^2(0\cdot9622) + 10^2(0\cdot0393) = 10^2(1\cdot0015)$$

giving $10^3(0\cdot1002)$ as the computed sum. Note that on an automatic computer a 5 in the discarded position is always rounded up. As a result the error made in the first round-off affects the second round-off.

18. We may now prove the assertion made in connexion with equation (17.2). For in cases (i), (ii), (iii), (iv) the assertion is true with $\epsilon_1 = \epsilon_2 = \epsilon$. In case (v) only one rounding error is made, that occurring when the shifted a_2 is rounded. Since cancellation takes place in the addition, no further rounding can be required. The rounding error is therefore bounded in modulus by $\frac{1}{2}2^{-t} \times 2^{b_1}$ (or $\frac{1}{2}10^{-t} \times 10^{b_1}$) and hence we have

$$fl(x_1+x_2) \equiv x_1(1+\epsilon) + x_2 \tag{18.1}$$

$$|\epsilon| \leqslant 2^{-t} \tag{18.2}$$

$$|\epsilon| \leqslant \tfrac{1}{2}10^{1-t}. \tag{18.3}$$

Similarly if the role of x_1 and x_2 are interchanged we have

$$fl(x_1+x_2) \equiv x_1 + x_2(1+\epsilon). \tag{18.4}$$

In case (vi) two separate rounding errors η_1 and η_2 are made. If b is the index associated with the computed sum then

$$|\eta_1| \leqslant 2^b\tfrac{1}{2}2^{-1-t}(\text{or } 10^b\tfrac{1}{2}10^{-1-t}) \tag{18.5}$$

$$|\eta_2| \leqslant 2^b\tfrac{1}{2}2^{-t}(\text{or } 10^b\tfrac{1}{2}10^{-t}). \tag{18.6}$$

Now the second error will not arise unless the exact value of $x_1 + x_2$ requires the exponent b and hence

$|x_1 + x_2| \geqslant \frac{1}{2}2^b(\text{or } \frac{1}{10}10^b)$, so that we have

$$|\eta_1 + \eta_2| \leqslant |\eta_1| + |\eta_2| \leqslant (1\tfrac{1}{2})2^{-t}|x_1 + x_2| \text{ (binary)} \tag{18.7}$$

$$\leqslant (0\cdot55)10^{1-t}|x_1 + x_2| \text{ (decimal)}. \tag{18.8}$$

Hence in this case

$$fl(x_1+x_2) \equiv (x_1+x_2)(1+\epsilon) \tag{18.9}$$

$$|\epsilon| \leqslant (1\tfrac{1}{2})2^{-t} \text{ (binary)} \tag{18.10}$$

$$|\epsilon| \leqslant (0{\cdot}55)10^{1-t} \text{ (decimal)}. \tag{18.11}$$

In all cases therefore (17.2) is true with

$$|\epsilon_1|, \ |\epsilon_2| \leqslant (1\tfrac{1}{2})2^{-t} \text{ (or } (0{\cdot}55)10^{1-t}). \tag{18.12}$$

19. Restriction to the use of a single-precision accumulator affects multiplication and division less than addition and subtraction, and the computed values always have a low relative error. The major difference then is that we may need different values of ϵ_1 and ϵ_2 in the error bounds for addition and subtraction. This will not affect our error analysis in any important detail and error bounds obtained for a computer using a single-precision accumulator will, as a rule, be no more than $1\frac{1}{2}$ (or $1\frac{1}{10}$) times those for a computer with a double-precision accumulator.

There are a number of round-off procedures in use and some of them are somewhat less accurate than either of those we have just analysed. As far as multiplication and division are concerned one is unlikely to encounter a procedure which gives an error which is greater than 1 in the least significant digit of the correctly normalized and rounded result, so that we have in any case

$$\left.\begin{aligned} fl(x_1 \times x_2) &\equiv x_1x_2(1+\epsilon) \\ fl(x_1 \div x_2) &\equiv \frac{x_1}{x_2}(1+\epsilon) \end{aligned}\right\} \quad |\epsilon| \leqslant 2^{1-t}. \tag{19.1}$$

Addition and subtraction admit of a rather greater variety of round-off procedures but a little reflection will convince the reader that no reasonably designed procedure could give an error which was greater than 1 in the least significant figure of x_1, plus 1 in the least significant figure of x_2, plus 1 in the least significant figure of x_1+x_2. Hence the total error could not possibly be greater than

$$2^{1-t}(|x_1| + |x_2| + |x_1 + x_2|)$$

and we have

$$\left.\begin{aligned} fl(x_1 \pm x_2) &\equiv x_1(1+\epsilon_1) + x_2(1+\epsilon_2) \\ |\epsilon_1| &\leqslant 4\ (2^{-t}) \quad |\epsilon_2| \leqslant 4\ (2^{-t}). \end{aligned}\right\} \tag{19.2}$$

In fact no computer known to the author produces errors which are as great as those given by the limits in (19.2). Nonetheless these bounds are only 4 times as great as those given for the most accurate procedure.

COMPARISON OF FIXED-POINT AND FLOATING-POINT COMPUTATION

20. The main points of difference between fixed-point and floating-point computation are the following.

For a given number of digits in the word, fixed-point is potentially more precise than floating-point because of the number of digits allotted to the exponent in the latter.

Much more careful preliminary analysis is needed for fixed-point computation than floating-point in order to ensure that numbers do not pass outside the permitted range. Often the natural flow of the computation must be modified in fixed-point computation by the introduction of appropriate scale factors. Quite commonly, in order to avoid an excessive use of scale factors, numbers in fixed-point computation are scaled so that they are much smaller than unity. In this way one may sacrifice more digits than would be required by the exponent in floating-point computation. If scale factors have to be introduced at very frequent intervals in fixed-point computation then it is little more than *ad hoc* floating-point computation. In matrix work, however, scale factors usually play a small part, and fixed-point work is here often significantly superior to standard floating-point.

21. Although many fixed-point computers have the facilities for the automatic exact accumulation of an inner-product and division into a double-precision dividend, corresponding facilities are as yet uncommon in connexion with floating-point arithmetic.

On ACE, the automatic computer used at the National Physical Laboratory, floating-point operations are effected by subroutines. A set of these, *having almost exactly the same speed as conventional single-precision subroutines*, has been designed to perform the following operations.

(i) *Addition and subtraction.* The subroutines accept numbers with single-precision or double-precision mantissae and produce a result having a double-precision mantissa.

(ii) *Division.* The subroutine accepts a double-precision dividend and a single-precision divisor, giving a single-precision quotient.

(iii) *Multiplication.* The subroutine accepts single-precision factors and gives a double-precision product.

These subroutines combine the accuracy associated with fixed-point accumulation of inner-products and the convenience of floating-point computation. It is worth noting that automatic floating-point facilities of this type (in addition to the standard type) are to be provided on at least one commercial computer.

However, these facilities do not permit inner-products to be accumulated *exactly*. For example

$$10^4(0\cdot3125) \times 10^5(0\cdot4167) + 10^2(0\cdot4132) \times 10^4(0\cdot3176) =$$

$$10^9(0\cdot1302\ 1875) + 10^6(0\cdot1312\ 3232).$$

Even allowing an 8-digit mantissa, we must discard the last 3 figures of the second product, but the error in the computed value is of order 10 instead of 10^5. *The failure to give an exact inner-product arises only because of the greater precision permissible in the factors in floating-point.* In fixed-point this example would probably have been $10^9(0{\cdot}3125 \times 0{\cdot}4167 + 0{\cdot}0041 \times 0{\cdot}0318)$ and accuracy would already have been sacrificed in the factors of the second product.

22. The provision of the accumulation facility in connexion with floating-point computation avoids a difficulty which is always present in fixed-point work. In accumulating an inner-product in fixed-point, the danger of the sum exceeding capacity must be borne in mind constantly. Further, it may happen that a partial sum exceeds capacity but when the later terms are added it returns to the permissible range.

If we knew in advance that the final sum would be in the permitted range we could ignore the spill over because the final result would be correct,* but unfortunately this is not usually the position. These points are covered automatically in floating-point accumulation, and although in general some rounding errors are inevitable, they will be in the digit at the end of the second half of the register and therefore involve 2^{-2t} instead of 2^{-t}.

The problem of overflow in fixed-point accumulation may be overcome as follows. The order of inner-products that may be computed by a subroutine is usually restricted by considerations of storage. Suppose we can deal with inner-products of orders up to 2^k, where k will usually be appreciably smaller than t. Then if the subroutine is designed to accumulate $\sum_1^n 2^{-k}a_ib_i$, (that is, each double-precision contribution is shifted k places to the right before accumulation), no partial sum can exceed capacity because we have

$$\left|2^{-k}\sum_1^r a_ib_i\right| \leqslant 2^{-k}\sum_1^r |a_ib_i| \leqslant 2^{-k}\sum_1^n |a_ib_i| < 1 \quad (r \leqslant n).$$

The discarded figures are in positions $(2t-k+1)$ to $2t$ and therefore their loss will make only a small contribution to the final error. This technique is convenient on ACE because it has good facilities for dealing with two-word aggregates. The accumulation of $\sum_1^n 2^{-k}a_ib_i$ is provided automatically on at least one commercial computer, the integer k being a parameter of the corresponding instruction.

We have taken the view that the facility for accumulating fixed-point inner-products is sufficiently common to justify our assuming its availability whenever this proves advantageous in the analysis. On the other hand, floating-point accumulation of the type described in section 21 is rarely available (except of course by means of subroutines which are much slower than the standard floating-point facilities). Accordingly most of our floating-point analysis corresponds to standard floating-point computation. The

* There are some computers for which this is not true.

round-off rules are those of sections 13–16 but the bounds for any other rounding procedure will not be greater by a factor of more than 4. (c.f. section 19.)

From time to time we give the analysis appropriate to floating-point operations of the type described in section 21. We distinguish floating-point operations of this kind by using the notation fl_2, thus

$$fl_2(a_1+a_2+\ldots+a_n)$$

$$fl_2(a_1b_1+a_2b_2+\ldots+a_nb_n).$$

In both of these the sum is produced as a double-precision number in the first instance and then rounded to single-precision before it is used in subsequent calculation. *No multiplications or divisions by a double-precision number are involved.*

COMMON FLOATING-POINT OPERATIONS

23. The calculation of extended products, sums and inner-products is so important that we give here for future reference bounds for the errors. From now on, we restrict ourselves to binary computation. We consider first the extended product p_n, defined by

$$p_n = fl(x_1x_2\ldots x_n).$$

It is implicit in the use of the notation on the right that $x_1, x_2, \ldots, x_n$ are standard floating-point numbers and that the multiplications take place in the order in which they are written. The quantities p_r are defined recursively by the relations

$$p_1 = x_1 \tag{23.1}$$

$$p_r = fl(p_{r-1}x_r) \equiv p_{r-1}x_r(1+\epsilon_r) \tag{23.2}$$

$$|\epsilon_r| \leqslant 2^{-t}. \tag{23.3}$$

Hence we have

$$p_n \equiv x_1x_2\ldots x_n(1+\epsilon_2)(1+\epsilon_3)\ldots(1+\epsilon_n) \tag{23.4}$$

where each ϵ_r satisfies (23.3). Equation (23.4) implies that

$$fl(x_1x_2\ldots x_n) \equiv x_1x_2\ldots x_n(1+E) \tag{23.5}$$

$$(1-2^{-t})^{n-1} \leqslant 1+E \leqslant (1+2^{-t})^{n-1}. \tag{23.6}$$

Provided $n-1$ is appreciably smaller than 2^t, which will almost invariably be the case, the computed product has a low relative error. Although the computed value will depend on the order in which the multiplications are performed, the bounds (23.6) are independent of the order.

24. Much the same result may be obtained for extended sequences of multiplications and divisions. In fact we have

$$fl(x_1x_2\ldots x_m/y_1y_2\ldots y_n) \equiv (x_1x_2\ldots x_m/y_1y_2\ldots y_n)(1+E) \tag{24.1}$$

$$(1-2^{-t})^{m+n-1} \leqslant 1 + E \leqslant (1+2^{-t})^{m+n-1} \tag{24.2}$$

and although the computed value will depend on the order in which the multiplications and divisions are performed, the bounds are again independent of that order. The computed value has a low relative error for any reasonable values of m and n.

25. For extended sequences of additions it is no longer true that the computed value necessarily has a low relative error even though this was true for the sum of *two* numbers with the more accurate round-off rule. In fact there is no useful bound for

$$|1 - fl(x_1+x_2+\ldots+x_n)/(x_1+x_2+\ldots+x_n)| \tag{25.1}$$

since the floating-point sum may be zero when the true sum is not, and vice versa. (Note that we use $fl(x_1+x_2+\ldots+x_n)$ to denote the computed sum when additions take place in the order in which they are written.)

It is still true however, that

$$s_n = fl(x_1+x_2+\ldots+x_n) \equiv x_1' + x_2' + \ldots + x_n'$$

where each x_i' differs from x_i by a factor which is close to unity. We define the quantities s_r recursively by the relations

$$s_1 = x_1 \tag{25.2}$$

$$s_r \equiv fl(s_{r-1}+x_r) \equiv (s_{r-1}+x_r)(1+\epsilon_r) \tag{25.3}$$

$$|\epsilon_r| \leqslant 2^{-t}. \tag{25.4}$$

Combining these equations, we have

$$s_n \equiv x_1(1+\eta_1) + x_2(1+\eta_2) + \ldots + x_n(1+\eta_n) \tag{25.5}$$

$$1 + \eta_1 = (1+\epsilon_2)(1+\epsilon_3)\ldots(1+\epsilon_n) \tag{25.6}$$

$$1 + \eta_r = (1+\epsilon_r)(1+\epsilon_{r+1})\ldots(1+\epsilon_n) \quad (r = 2, \ldots, n). \tag{25.7}$$

$$(1-2^{-t})^{n-1} \leqslant 1 + \eta_1 \leqslant (1+2^{-t})^{n-1} \tag{25.8}$$

$$(1-2^{-t})^{n+1-r} \leqslant 1 + \eta_r \leqslant (1+2^{-t})^{n+1-r} \quad (r = 2, \ldots, n). \tag{25.9}$$

Note that x_1 and x_2 have the same factor, which is not surprising since they play identical roles. It is inconvenient that the factor associated with x_1 is not defined by the same formula as the others and it is often simpler to write

$$(1-2^{-t})^{n+1-r} \leqslant 1 + \eta_r \leqslant (1+2^{-t})^{n+1-r} \quad (r = 1, \ldots, n) \tag{25.10}$$

which is true *a fortiori*.

Note that the bounds themselves are dependent on the order of summation. The *upper bound* for the error is smallest if the terms are added in order of increasing absolute magnitude since then the largest factor $(1+\eta_1)$ is associated with the smallest x_i. Although this gives the smallest upper bound it does not necessarily give the smallest error in practice. Consider for example the sum $10^4(0\cdot1025) + 10^3(-0\cdot9123) + 10^2(-0\cdot9663) + 10(-0\cdot9315)$. If we add them in the natural order we have

$$s_2 = 10^3(0\cdot1127), \quad s_3 = 10^2(0\cdot1607), \quad s_4 = 10(0\cdot6755)$$

and there are no rounding errors at any stage. If we add the numbers in the opposite order then we have

$$s_2 = 10^3(-0\cdot1059), \quad s_3 = 10^4(-0\cdot1018), \quad s_4 = 10(0\cdot7000)$$

and there are rounding errors at each of the first two stages.

It will be appreciated that this example is rather special. The more normal case is illustrated by the following example. Consider the sum

$$10^{-3}(0\cdot4462) + 10^{-3}(0\cdot6412) + 10^{-3}(0\cdot2413) + 10^0(0\cdot1234).$$

If the additions take place in the order in which the terms are given, the computed sum is $10^0(0\cdot1247)$ and the errors made in the three additions are $10^{-6}(-0\cdot4)$, $10^{-6}(-0\cdot3)$ and $10^{-4}(-0\cdot28)$. If they are added in the reverse order the computed sum is $10^0(0\cdot1246)$ and the errors in the three additions are $10^{-4}(-0\cdot413)$, $10^{-4}(-0\cdot412)$ and $10^{-4}(-0\cdot462)$. Adding the terms of smallest modulus first ensures that the rounding errors made in the early stages are small. Clearly if we include a large number of terms of order 10^{-3} in a sum of the above type then the advantage of adding the small terms first can be very considerable.

26. We now consider the error made when an inner-product is computed in floating-point. We write

$$s_n = fl(a_1b_1 + a_2b_2 + \ldots + a_nb_n). \tag{26.1}$$

The notation implies that a_i and b_i are standard floating-point numbers. The products are computed first and then added in the order in which they are written. We define the quantities s_r and t_r recursively by the relations

$$t_r = fl(a_rb_r) \tag{26.2}$$

$$s_1 = t_1 \qquad s_r = fl(s_{r-1} + t_r). \tag{26.3}$$

From these relations we deduce

$$t_r \equiv a_rb_r(1+\xi_r) \qquad (|\xi_r| \leqslant 2^{-t}) \tag{26.4}$$

$$s_r \equiv (s_{r-1} + t_r)(1+\eta_r) \quad (|\eta_r| \leqslant 2^{-t}) \tag{26.5}$$

and hence

$$s_n \equiv a_1b_1(1+\epsilon_1) + a_2b_2(1+\epsilon_2) + \ldots + a_nb_n(1+\epsilon_n) \tag{26.6}$$

$$1 + \epsilon_1 = (1+\xi_1)(1+\eta_2)\ldots(1+\eta_n) \tag{26.7}$$

$$1 + \epsilon_r = (1+\xi_r)(1+\eta_r)\ldots(1+\eta_n) \quad (r = 2, \ldots, n). \tag{26.8}$$

Hence

$$(1-2^{-t})^n \leqslant 1+\epsilon_1 \leqslant (1+2^{-t})^n \tag{26.9}$$

$$(1-2^{-t})^{n-r+2} \leqslant 1+\epsilon_r \leqslant (1+2^{-t})^{n-r+2} \quad (r = 2, \ldots, n). \tag{26.10}$$

To avoid a special formula for the bounds on ϵ_1 in a different way from the other ϵ_r we may write *a fortiori*

$$(1-2^{-t})^{n-r+2} \leqslant (1+\epsilon_r) \leqslant (1+2^{-t})^{n-r+2} \quad (r = 1, \ldots, n). \tag{26.11}$$

We may simplify the expression $(1 \pm 2^{-t})^r$ which occurs in the bounds. Normally r will be much smaller than 2^t and if

$$r2^{-t} < 0{\cdot}1 \tag{26.12}$$

then

$$(1+2^{-t})^r < 1 + (1{\cdot}06)r2^{-t} \tag{26.13}$$

$$(1-2^{-t})^r > 1 - (1{\cdot}06)r2^{-t}. \tag{26.14}$$

Usually in our error analyses it will be necessary for some other reason to restrict r at least as severely as in (26.12). For any modern automatic computer, t is such that (26.12) is in any case a very mild restriction. At various stages throughout this book we have used the results expressed by (26.13) and (26.14) without explicitly referring to the restriction (26.12). *For compactness of presentation we define t_1 by the relation*

$$2^{-t_1} = (1{\cdot}06)2^{-t} \tag{26.15}$$

so that

$$\begin{aligned} t_1 &= t - \log_2(1{\cdot}06) \\ &= t - 0{\cdot}08406. \end{aligned} \tag{26.16}$$

This enables us to replace inequalities of the form

$$(1-2^{-t})^r \leqslant 1+\epsilon \leqslant (1+2^{-t})^r \tag{26.17}$$

by the simpler inequality

$$|\epsilon| < r2^{-t_1}. \tag{26.18}$$

MORE PRECISE BOUNDS

27. The bounds we have obtained up to the present are almost attainable.

For example, consider the sum $fl(x_1+x_2+\ldots+x_{2^r})$ with $x_1 = 1$; $x_2 = 1 - 2^{-t}$; x_3 to $x_4 = 1 - 2^{1-t}$; x_5 to $x_8 = 1 - 2^{2-t}$; $\ldots$; $x_{2^{r-1}+1}$ to $x_{2^r} = 1 - 2^{r-1-t}$.

The error in the first addition is 2^{-t}

The error in each of the next 2 additions is 2^{1-t}

The error in each of the next 4 additions is 2^{2-t}

. . . . . .

The error in each of the next 2^{r-1} additions is 2^{r-1-t}

The total error is therefore

$$2^{-t}(1+2^2+2^4+\ldots+2^{2r-2}) = 2^{-t}\tfrac{1}{3}(4^r-1).$$

The upper bound we have obtained for the error from (25.5), (25.10), (26.13) and (26.14) is approximately

$$1\cdot06\,[2^r.2^{-t}+(2^r-1)2^{-t}+\ldots+2.2^{-t}+2^{-t}] = 1\cdot06\left[2^{-t}\frac{2^r(2^r+1)}{2}\right].$$

Hence the true error is approximately $\frac{1}{3}2^{-t}.2^{2r}$ and the error bound is approximately $(0\cdot53)\,2^{-t}.2^{2r}$. This is not a severe overestimate. However, in order to approach the upper bound as closely as this, not only must each error take its maximum value, but all the terms must be almost equal.

28. When additional information is given about the terms of a sum, then we can sometimes obtain a better upper bound for the error. Suppose for example we know that

$$\sum_1^n |x_r| < 1. \tag{28.1}$$

Now we have

$$fl\left(\sum_1^n x_r\right) \equiv \sum_1^n x_r(1+\epsilon_r) \qquad \begin{aligned} |\epsilon_1| &\leqslant (n-1)2^{-t_1} \\ |\epsilon_r| &\leqslant (n+1-r)2^{-t_1} \end{aligned} \tag{28.2}$$

where for convenience we write $fl\left(\sum_1^n x_r\right)$ for $fl(x_1+x_2+\ldots+x_r)$.

Hence

$$\left|fl\left(\sum_1^n x_r\right)-\sum_1^n x_r\right| \leqslant \sum_1^n |x_r|\,|\epsilon_r| \tag{28.3}$$

and we have

$$\sum_1^n |x_r|\,|\epsilon_r| \leqslant 2^{-t_1}(\Sigma_n+\Sigma_{n-1}+\ldots+\Sigma_2) \tag{28.4}$$

where

$$\Sigma_r = |x_1|+|x_2|+\ldots+|x_r| < 1. \tag{28.5}$$

Hence

$$\left|fl\left(\sum_1^n x_r\right)-\sum_1^n x_r\right| < (n-1)2^{-t_1}. \tag{28.6}$$

We now show that if

$$\sum_1^n |x_i| < 1-\tfrac{1}{2}(n-1)2^{-t_1} \tag{28.7}$$

then

$$|fl(x_1+x_2+\ldots+x_n)| < 1 \tag{28.8}$$

and

$$|fl(x_1+x_2+\ldots+x_n)-(x_1+x_2+\ldots+x_n)| < \tfrac{1}{2}(n-1)2^{-t_1}. \tag{28.9}$$

In other words, if the sum of the absolute values of the x_i is rather less than unity, then we can obtain a bound for the error in the computed sum which is half that obtained using the general formula.

The proof is by induction. Assume that it is true for values of r up to n. We then show that it is true for $n+1$. We are given

$$|x_1| + |x_2| + \ldots + |x_{n+1}| < 1 - \tfrac{1}{2}n2^{-t_1}. \tag{28.10}$$

It is convenient to write

$$s_r = fl(x_1+x_2+\ldots+x_r) \qquad p_r = (x_1+x_2+\ldots+x_r). \tag{28.11}$$

From (28.10) we have, *a fortiori*

$$|x_1| + |x_2| + \ldots + |x_n| < 1 - \tfrac{1}{2}(n-1)2^{-t_1} \tag{28.12}$$

and hence by hypothesis

$$|s_n - p_n| < \tfrac{1}{2}(n-1)2^{-t_1}. \tag{28.13}$$

From its definition

$$s_{n+1} = fl(s_n + x_{n+1}) \tag{28.14}$$

and

$$\begin{aligned} |s_n + x_{n+1}| &\leqslant |s_n| + |x_{n+1}| \\ &\leqslant |p_n| + \tfrac{1}{2}(n-1)2^{-t_1} + |x_{n+1}| \\ &\leqslant |x_1| + |x_2| + \ldots + |x_{n+1}| + \tfrac{1}{2}(n-1)2^{-t_1} \\ &< 1 - \tfrac{1}{2}2^{-t_1}. \end{aligned} \tag{28.15}$$

From the relations (28.14) and (28.15) we know that

$$|s_{n+1} - (s_n + x_{n+1})| = |fl(s_n + x_{n+1}) - (s_n + x_{n+1})| \leqslant \tfrac{1}{2}2^{-t} \tag{28.16}$$

and hence

$$\begin{aligned} |s_{n+1} - p_{n+1}| &= |s_{n+1} - p_n - x_{n+1}| \\ &= |s_{n+1} - (s_n + x_{n+1}) + s_n - p_n| \\ &\leqslant |s_{n+1} - (s_n + x_{n+1})| + |s_n - p_n| \\ &< \tfrac{1}{2}2^{-t} + \tfrac{1}{2}(n-1)2^{-t_1} = \tfrac{1}{2}n2^{-t_1}. \end{aligned} \tag{28.17}$$

This gives

$$\begin{aligned} |fl(x_1+x_2+\ldots+x_{n+1})| = |s_{n+1}| &< |p_{n+1}| + \tfrac{1}{2}n2^{-t_1} \\ &\leqslant |x_1| + |x_2| + \ldots + |x_{n+1}| + \tfrac{1}{2}n2^{-t_1} \\ &< 1 - \tfrac{1}{2}n2^{-t_1} + \tfrac{1}{2}n2^{-t_1} \\ &= 1. \end{aligned} \tag{28.18}$$

Relations (28.17) and (28.18) are the two required results. The result is true for $n = 2$ and hence is true in general. It is a best possible result, for if we take

$$x_1 = \tfrac{1}{2}, \quad x_2 = x_3 = \ldots = x_n = 2^{-t-1}$$

then in each addition the increment is converted to 2^{-t} by the round-off, and the total error is $\frac{1}{2}(n-1)2^{-t}$.

A trivial deduction from this result is that if

$$\sum_1^n |x_i y_i| < 1 - \tfrac{1}{2}(n+1)2^{-t_1} \tag{28.19}$$

then

$$|fl(x_1 y_1 \ldots x_n y_n) - (x_1 y_1 \ldots x_n y_n)| < \tfrac{1}{2}(n+1)2^{-t_1}. \tag{28.20}$$

This follows since the additional errors coming from the multiplications must have a total value of less than 2^{-t}.

29. There is another simple computation for which it is worthwhile to have a slightly sharper bound than is obtained by an immediate application of our general results. We show that if a, m and b are floating-point numbers and

$$|a + mb| < 1 - \tfrac{1}{2}2^{-t} \tag{29.1}$$

$$|mb| < 1 \tag{29.2}$$

then

$$|fl(a+mb) - (a+mb)| \leqslant 2^{-t}. \tag{29.3}$$

For we have

$$c = fl(mb) \equiv mb + \epsilon_1 \qquad (|\epsilon_1| \leqslant \tfrac{1}{2}2^{-t}) \tag{29.4}$$

because mb has an exponent of zero or less. Further we have

$$|a + c| = |a + mb + \epsilon_1| \leqslant |a + mb| + |\epsilon_1| < 1 \tag{29.5}$$

and hence

$$fl(a+mb) = fl(a+c) \equiv a + c + \epsilon_2 \qquad (|\epsilon_2| \leqslant \tfrac{1}{2}2^{-t}) \tag{29.6}$$

because $a + c$ has an exponent of zero or less. This gives

$$fl(a+mb) = a + mb + \epsilon_1 + \epsilon_2 \tag{29.7}$$

$$\begin{aligned} |fl(a+mb) - (a+mb)| &\leqslant |\epsilon_1| + |\epsilon_2| \\ &\leqslant 2^{-t}. \end{aligned} \tag{29.8}$$

We also deduce that $|fl(a+mb)| \leqslant 1$ since it is obtained by rounding $(a+c)$ which is certainly less than unity in absolute value.

It is at first sight surprising that we guarantee that $|fl(a+mb)| \leqslant 1$ although we had only $|a+mb| < 1 - \tfrac{1}{2}2^{-t}$ and the bound for the error is 2^{-t}. This is because we are dealing with numbers with a finite number of digits. The bound (29.8) is the best possible, as may be seen from the decimal example

$$10^{-2}(0\cdot3150) + 10^{0}(0\cdot6247) \times 10^{0}(0\cdot5000)$$

$$fl(mb) = \text{Rounded } (0\cdot31235) = 0\cdot3124$$

$$fl[10^{-2}(0\cdot3150) + 10^{0}(0\cdot3124)] = 10^{0}(0\cdot3156).$$

The exact value is $0\cdot3155$.

That $fl(a+mb)$ may take the value unity is illustrated by the example

$$10^{-1}(0\cdot4048) + 10^{0}(0\cdot9607) \times 10^{0}(0\cdot9987)$$

$$fl(mb) = \text{Rounded } (0\cdot9594\ 5109) = 0\cdot9595$$

$$fl[10^{-1}(0\cdot4048) + 10^{0}(0\cdot9595)] = 10^{1}(0\cdot1000).$$

The exact value is $10^{0}(0\cdot9999\ 3109)$.

30. In sections 27–29 we have used the more accurate bounds which express the error in terms of the index associated with the exact result. It will be seen that the gains are not spectacular, seldom amounting to more than a factor $\frac{1}{2}$. Even this modest gain has required a considerably more detailed analysis. In our view detailed analysis of this kind obscures the essential simplicity of an argument to little purpose, so that except on rare occasions, we content ourselves with the simpler bounds.

FLOATING-POINT ACCUMULATION OF SUMS AND INNER-PRODUCTS

31. We now consider briefly the errors made when floating-point accumulation is available. Consider first

$$fl_2(a_1+a_2+\ldots+a_n).$$

Here each a_i may have a single-precision or a double-precision mantissa. The addition is performed in the first instance retaining double-precision mantissae in each partial sum, and on completion of all additions the sum is rounded to single-precision. It is reasonable to assume that we have only a double-precision accumulator (not quadruple) so that the rounding procedure during the accumulation of the double-precision sum is that of section 17 rather than that of sections 13–16. As in (25.5) the double-precision sum is therefore

$$a_1(1+\epsilon_1) + a_2(1+\epsilon_2) + \ldots + a_n(1+\epsilon_n)$$

where from (18.12), (25.8) and (25.9)

$$(1-\tfrac{3}{2}2^{-2t})^{n-1} \leqslant 1+\epsilon_1 \leqslant (1+\tfrac{3}{2}2^{-2t})^{n-1}$$

$$(1-\tfrac{3}{2}2^{-2t})^{n+1-r} \leqslant 1+\epsilon_r \leqslant (1+\tfrac{3}{2}2^{-2t})^{n+1-r} \quad (r=2,\ldots,n). \quad (31.1)$$

This is then rounded to single-precision, so that we have finally

$$fl_2(a_1+a_2+\ldots+a_n) \equiv [a_1(1+\epsilon_1) + a_2(1+\epsilon_2) + \ldots + a_n(1+\epsilon_n)](1+\epsilon)$$

$$1-2^{-t} \leqslant 1+\epsilon \leqslant 1+2^{-t}. \quad (31.2)$$

The factor associated with a_r is $(1+\epsilon_r)(1+\epsilon)$ and since 2^{-2t} will usually be very small compared with 2^{-t}, this factor will be very close to $(1+\epsilon)$ for all r.

We turn now to the inner-product

$$fl_2(a_1b_1+a_2b_2+\ldots+a_nb_n)$$

where a_i and b_i are standard single-precision numbers. We have in much the same way

$$fl_2(a_1b_1+a_2b_2+\ldots+a_nb_n) \equiv [a_1b_1(1+\epsilon_1) + a_2b_2(1+\epsilon_2)+ \ldots \\ \ldots + a_nb_n(1+\epsilon_n)](1+\epsilon) \quad (31.3)$$

where

$$(1-\tfrac{3}{2}2^{-2t})^n \leqslant 1 + \epsilon_1 \leqslant (1+\tfrac{3}{2}2^{-2t})^n$$

$$(1-\tfrac{3}{2}2^{-2t})^{n+2-r} \leqslant 1 + \epsilon_r \leqslant (1+\tfrac{3}{2}2^{-2t})^{n+2-r} \quad (r = 2, \ldots, n)$$

$$1 - 2^{-t} \leqslant 1 + \epsilon \leqslant 1 + 2^{-t}. \quad (31.4)$$

The accuracy of the operation $fl_2\left(\dfrac{a_1b_1 + a_2b_2 + \ldots + a_nb_n}{c}\right)$ where c is a single-precision number, is particularly satisfactory. The numerator is accumulated with a double-precision mantissa and the mantissa of c is then divided into this. Hence we have

$$fl_2\left(\frac{a_1b_1+a_2b_2+\ldots+a_nb_n}{c}\right) \equiv \frac{a_1b_1(1+\epsilon_1)+a_2b_2(1+\epsilon_2)+\ldots+a_nb_n(1+\epsilon_n)}{c/(1+\epsilon)}$$

where the ϵ_i and ϵ satisfy the same inequalities as in (31.4). Note that the factors associated with the a_ib_i now involve only 2^{-2t}.

32. It should be emphasized that we still cannot guarantee that an accumulated sum or inner-product has a low *relative* error. Consider for example

$$10^1(0\cdot1002) \times 10^1(0\cdot1003) + 10^0(0\cdot9999) \times 10^0(-0\cdot9995) \\ + 10^{-1}(0\cdot2000) \times 10^0(-0\cdot2803)$$

using an 8-digit register in the accumulation. The individual products are

$$10^1(0\cdot1005\ 0060); \quad 10^0(-0\cdot9994\ 0005); \quad 10^{-2}(-0\cdot5606\ 0000).$$

When adding the first two products, the second must be converted to $10^1(-0\cdot0999\ 4001)$ giving a sum $10^{-2}(0\cdot5605\ 9000)$. Adding the third product therefore gives $10^{-6}(-0\cdot1000)$. The correct sum is $10^{-7}(-0\cdot5000)$.

However, we have from (31.3)

$$fl_2(a_1b_1+a_2b_2+\ldots+a_nb_n) - (a_1b_1+a_2b_2+\ldots+a_nb_n)(1+\epsilon) \\ \equiv [a_1b_1\epsilon_1 + a_2b_2\epsilon_2 + \ldots + a_nb_n\epsilon_n](1+\epsilon) \quad (32.1)$$

and each term in the square brackets on the right has a factor of order 2^{-2t}.

Again it is convenient to introduce somewhat simpler inequalities to replace

$$(1-\tfrac{3}{2}2^{-2t})^r \leqslant 1 + \epsilon \leqslant (1+\tfrac{3}{2}2^{-2t})^r. \quad (32.2)$$

If r is restricted accordingly to the inequality

$$\tfrac{3}{2}r2^{-2t} < 0{\cdot}1 \tag{32.3}$$

then

$$(1+\tfrac{3}{2}2^{-2t})^r < 1 + \tfrac{3}{2}r(1{\cdot}06)2^{-2t} \tag{32.4}$$

$$(1-\tfrac{3}{2}2^{-2t})^r > 1 - \tfrac{3}{2}r(1{\cdot}06)2^{-2t}. \tag{32.5}$$

We introduce t_2 defined by

$$(1{\cdot}06)2^{-2t} = 2^{-2t_2} \tag{32.6}$$

or

$$2t_2 = 2t - \log_2(1{\cdot}06) \tag{32.7}$$

so that (32.2) may be replaced by

$$|\epsilon| < \tfrac{3}{2}r2^{-2t_2} \tag{32.8}$$

where t_2 is only slightly different from t.

In this notation (32.1) becomes

$$fl_2(a_1b_1+a_2b_2+\ldots+a_nb_n) - (a_1b_1+a_2b_2+\ldots+a_nb_n)(1+\epsilon)$$
$$\equiv (a_1b_1\epsilon_1+a_2b_2\epsilon_2+\ldots+a_nb_n\epsilon_n)(1+\epsilon) \tag{32.9}$$

where

$$\left.\begin{aligned} |\epsilon| &\leqslant 2^{-t} \\ |\epsilon_1| &< \tfrac{3}{2}n2^{-2t_2} \\ |\epsilon_r| &< \tfrac{3}{2}(n+2-r)2^{-2t_2}. \end{aligned}\right\} \tag{32.10}$$

The factor $(1{\cdot}06)$ introduced in (32.4) and (32.5) is sufficiently generous that the extra factor $1+\epsilon$ on the right-hand side of (32.9) is already covered by the use of t_2, and hence we have finally,

$$fl_2(a_1b_1+a_2b_2+\ldots+a_nb_n) - (a_1b_1+a_2b_2+\ldots+a_nb_n)(1+\epsilon)$$
$$\equiv (a_1b_1\epsilon_1+a_2b_2\epsilon_2+\ldots+a_nb_n\epsilon_n) \tag{32.11}$$

where the inequalities (32.10) are still adequate. We have a similar result for $fl_2(a_1+a_2+\ldots+a_n)$. Again we have used (32.11) and (32.10) throughout the book without explicitly referring to the restriction (32.3).

STATISTICAL ERROR BOUNDS

33. All the bounds given so far have been rigorous upper bounds and though they have not usually been sharp, they have been within a factor of 2 or 3 of the least upper bound. However, each individual rounding error in a single arithmetic operation may take a value lying between $\pm\frac{1}{2}$ times the last figure retained and in general we may expect that the rounding errors in a computation will be more or less randomly distributed in this range.

Consider now, for example, our expression for the error in a floating-point multiplication. We have written

$$fl(x_1 \times x_2) = x_1x_2(1+\epsilon) \quad (|\epsilon| \leqslant 2^{-t}).$$

For the bound 2^{-t} to be reached, not only must the digits that were discarded in the mantissa of the product take their maximum value 2^{-1-t}, but the rounded mantissa must have its minimum value $\frac{1}{2}$. It is reasonable to suppose that a more realistic error bound for an extended product is given by

$$fl(x_1x_2\ldots x_n) = (x_1x_2\ldots x_n)(1+\epsilon) \quad (|\epsilon| < n^{1/2}2^{-t}) \qquad (33.1)$$

when n is large.

In order that the total error for an extended sum should approach the upper bound given in section 25 not only must each individual error have its maximum value but the actual distribution of the elements $x_1, x_2, \ldots, x_n$ must be very special. We shall not make any serious attempt to give statistical error bounds in this book, not even those based on simple assumptions concerning the distribution of errors. Occasionally, however, we shall make crude assessments such as those embodied in (33.1) in order to indicate the order of magnitude of the error that can be expected in practice.

BLOCK-FLOATING VECTORS AND MATRICES

34. In fixed-point computation involving vectors and matrices it is common to use a method of representation which, to some extent, combines the virtues of fixed and of floating-point.

In this scheme a single scale factor, in the form of a power of 2 (or 10), is associated with all components of a vector or a matrix. This factor is usually chosen so that the modulus of the largest component lies between $\frac{1}{2}$ and 1 ($\frac{1}{10}$ and 1). Such a vector is called a *standardized block-floating vector*. Thus, for example in (34.1), a is a *standardized block-floating* vector and A is a *standardized block-floating* matrix

$$a = 10^{-4}\begin{bmatrix} +0{\cdot}0013 \\ -0{\cdot}2763 \\ +0{\cdot}0002 \\ +0{\cdot}0013 \end{bmatrix}$$

$$A = 10^{5}\begin{bmatrix} +0{\cdot}0067 & +0{\cdot}2175 & +0{\cdot}6135 & +0{\cdot}2173 \\ -0{\cdot}1235 & +0{\cdot}3135 & -0{\cdot}3127 & -0{\cdot}0123 \\ -0{\cdot}0167 & -0{\cdot}0004 & +0{\cdot}0035 & +0{\cdot}0031 \\ -0{\cdot}0004 & +0{\cdot}2635 & -0{\cdot}3653 & -0{\cdot}0000 \end{bmatrix}. \qquad (34.1)$$

Sometimes the columns of a matrix differ so much in magnitude that each column is represented as a block-floating vector. Similarly we may represent each row by a block-floating vector. Finally we could have a scale factor associated with each row and each column.

Often at intermediate stages in a computation a block-floating vector is computed which does not have any component of modulus as large as $\frac{1}{2}$. Such a vector is called a *non-standardized* block-floating vector and is illustrated by a in (34.2)

$$a = 10^{-3} \begin{bmatrix} -0{\cdot}0023 \\ +0{\cdot}0124 \\ +0{\cdot}0003 \\ -0{\cdot}0021 \end{bmatrix}. \tag{34.2}$$

In some situations the scale factor associated with a vector is of no importance; this is usually true, for example, of an eigenvector. We define a *normalized block-floating vector* to be a standardized block-floating vector having the scale factor 2^0 (or 10^0). The scale factor is usually omitted in such a vector.

The advantage of this scheme is that only one word is allocated to the index in a block-floating vector and that each individual element is allocated a full word for its storage, no digits being wasted on the exponent as they would be in floating-point computation. The element of largest modulus always has the full number of significant figures and although the smaller elements have fewer significant figures this is usually quite satisfactory in practice.

The individual elements of a block-floating vector may be represented by single-precision or multiple-precision numbers, but in any case, one word suffices for the index. This mode of operation is particularly satisfactory on a computer which has the facility for accumulating inner-products; particularly if we can conveniently accumulate $2^{-k}\sum a_i b_i$ as a double-precision number (section 22). We can then choose k so that spill-over cannot occur. For example, when computing Ab with A and b in single-precision block-floating form we can accumulate 2^{-k} times each of the elements of Ab, and then finally normalize the vector to single-precision block-floating form.

On ACE this is our preferred mode of operation and we shall sometimes give the analysis appropriate to it. It must be admitted, however, that on some computers block-floating single-precision computation with accumulation of inner-products is little faster than true double-precision computation. We shall use the following abbreviations:

s.p.b.f. = single-precision block-floating

d.p.b.f. = double-precision block-floating.

FUNDAMENTAL LIMITATIONS OF t-DIGIT COMPUTATION

35. If we restrict ourselves rigidly to the use of numbers with t binary digits, then there is in general a strict limit to the accuracy attainable in any computation. We may describe any computation as the determination of a set of numbers, the answers, which are functions of a set of parameters, the data. For example, if we are inverting a matrix A, then the elements a_{ij} are the parameters and the elements of A^{-1} are the answers.

Now even if the data is known exactly it may not be representable by numbers with t binary digits. If we restrict ourselves to t-digit numbers then we must content ourselves with *t-digit approximations* to the data, thereby restricting our accuracy even before we start the computation. Quite frequently in matrix work we shall require the inverse of a matrix A, which has itself been computed by previous operations. If, for example, A is defined by $A = BC$ then, even if the elements of B and C are t-digit numbers, the elements of A will in general require $2t$ digits for their exact representation.

If the data is derived from physical observations it will inevitably be subject to the errors inherent in such observations. The problem presented to the computer will then be an approximation to the true problem. The word length t on most computers will usually be such that the accuracy of the physical observations is far less than 1 part in 2^t, so that the errors in the observations will be far more important then those which may result from the representation of the data by numbers with t binary digits.

ILL-CONDITIONED PROBLEMS

36. We see therefore that, except in the cases when the parameters are defined exactly and are exactly representable by numbers with t digits, we start at best with what may be called a *t-digit approximation* to the true problem. Accordingly in a complete error analysis we cannot avoid considering the effect on the solutions of perturbations in the parameters. If very small relative perturbations in the parameters make comparatively large relative errors in the solutions, then we say that the problem is *ill-conditioned.* Clearly ill-conditioned problems are very unsatisfactory because if we wish to be certain of a prescribed number of figures in the solutions we must determine accurately many more figures in the parameters. This is particularly undesirable and may indeed be impossible, if the parameters are derived from physical observations.

It should be emphasized that it is the susceptibility of *the required solution* to changes in the parameters which determines the condition of a problem. Thus it is meaningless to state that a matrix A is ill-conditioned. For example, it can be ill-conditioned with respect to the calculation of its inverse or the calculation of its eigenvalues or eigenvectors, but in general, ill-condition with respect to one of these will not imply ill-condition with respect to any other.

A good illustration of the importance of this concept is provided by the solution of the equations

$$Ax = b.$$

We may be concerned with such a set of equations in a situation in which the computed solution x is to be an approximation to an eigenvector of A. Now since we are not interested in arbitrary multipliers of an eigenvector, we are concerned only with the ratio of elements of x. It may well happen that small perturbations in A and b make very small changes in the ratios

of the components of x, while making very large changes in their absolute values. The computing problem in question will therefore be *well-conditioned* though if we had been concerned with the absolute magnitude of x it would have been ill-conditioned.

CONDITION NUMBERS

37. If we denote the parameters of a problem by $a_1, a_2, \ldots, a_m$ and the elements of the solution by $x_1, x_2, \ldots, x_n$ then the mn quantities k_{rs}, defined by

$$k_{rs} = \frac{\partial x_r}{\partial a_s} \quad (r = 1, \ldots, n, s = 1, \ldots, m) \tag{37.1}$$

give us complete information on the sensitivity of the solution with respect to perturbations in the parameters. For most problems these quantities k_{rs} will be too numerous to be of any practical value and we shall be forced to resort to a rather cruder estimate of the *condition* of the problem. For example, we may be able to prove that

$$[\sum \delta x_i^2]^{1/2} \leqslant K[\sum \delta a_i^2]^{1/2} \tag{37.2}$$

either for all values of δa_i or perhaps for all sufficiently small δa_i. Alternatively we may be able to prove that

$$|\delta x_j| \leqslant K_j[\sum \delta a_i^2]^{1/2} \tag{37.3}$$

with a different value K_j for each j. We shall refer to K or K_j as *condition numbers with respect to the computation of all x_i's or of individual x_i's* respectively. We have avoided framing a precise definition of condition numbers so that we may use the term freely. The condition numbers we have just given refer to absolute changes, but we shall sometimes be more interested in condition numbers defined with respect to relative changes. For these we may sometimes have condition numbers K_i defined by relations of the form

$$\left|\frac{\delta x_j}{x_j}\right| \leqslant K_j\left[\sum \left(\frac{\delta a_i}{a_i}\right)^2\right]^{1/2}. \tag{37.4}$$

However we cannot expect that there will always be useful condition numbers of this kind. If an x_j is zero, the left hand side of (37.4) is not even defined. More commonly we can establish a relation of the form

$$\frac{|\delta x_j|}{[\sum x_j^2]^{1/2}} \leqslant K_j\left[\frac{\sum \delta a_i^2}{\sum a_i^2}\right]^{1/2}. \tag{37.5}$$

38. Except when the parameters of a problem are known exactly and can be represented exactly by t-digit numbers, we effectively start our computation with perturbed parameters a_i' satisfying at best

$$|a_i' - a_i| \leqslant \tfrac{1}{2}2^{-t} \text{ in fixed-point} \tag{38.1}$$

$$\left|\frac{a_i' - a_i}{a_i}\right| \leqslant 2^{-t} \text{ in floating-point}. \tag{38.2}$$

If error bounds of the form (37.3) can be given then the error δx_j in the solution x_j will satisfy the relation

$$|\delta x_j| \leqslant \tfrac{1}{2} K_j m^{1/2} 2^{-t}$$

with similar results for other types of error bounds. We must regard errors of this order of magnitude as inherent in t-digit computation and they arise even when we use very stable methods of computation. If K_j is approximately 2^{k_j} then we are likely to have errors of the order of magnitude of $2^{k_j - t}$. If k_j is large, such errors may be unacceptable; in this case it is virtually certain that we must use computation of a higher precision. Note that the presence of the factor K_j is a fundamental weakness in the computing problem. It will, in general, be quite independent of t.

ROUNDING ERRORS IN THE COMPUTATION

39. We will be concerned mainly with computations which involve a very large number of arithmetic operations and in general, rounding errors will be made at each step. The central problem in error analysis is to give bounds for the cumulative effect of these rounding errors on the solutions.

Frequently it will prove possible to organize our analysis in such a way that a computed element x_j can be shown to be the exact solution of a perturbation of the original problem in which each a_i is replaced by an a_i', and that bounds can be given for $|a_i' - a_i|$ for each i. Occasionally we shall be able to assert that all n of the computed x_j constitute the exact solution of the *same* perturbed problem, but more often our bounds for the perturbations in the a_i will be different for each x_j.

In either case, however, this method has the advantage of leading to a consideration of the effect of perturbations in the parameters, and this is a problem which must usually be solved in any case because of the consideration of sections 35–37.

It is useful to consider what bounds we can reasonably expect for $|a_i' - a_i|$ and to this end we consider a specific example. Suppose we wish to calculate in floating-point arithmetic the zeros of the polynomial $p(x)$ defined by

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0. \tag{39.1}$$

In some methods it is convenient to have the coefficient of a_n equal to unity, so we might start by dividing by a_n. In general the quotients will not be exactly representable by t-digit numbers and the transformed polynomial will be $q(x)$, defined by

$$q(x) = x^n + \left(\frac{a_{n-1}}{a_n}\right)(1+\epsilon_{n-1})x^{n-1} + \ldots + \left(\frac{a_0}{a_n}\right)(1+\epsilon_0)$$

$$|\epsilon_r| \leqslant 2^{-t}.$$

Even if we calculate the zeros of $q(x)$ exactly, they are the zeros of

$$a_n x^n + a_{n-1}(1+\epsilon_{n-1})x^{n-1} + \ldots + a_0(1+\epsilon_0).$$

and we will therefore have the zeros of some t-figure approximation to the original polynomial. Now in the course of computing the zeros of a high-degree polynomial, we are likely to subject it to a *series* of transformations and it is hardly to be expected that their combined effect will be less harmful than the single trivial transformation we have just considered. However, if each of the final computed zeros were itself the exact solution of some t-figure approximation to $p(x)$, it would be reasonable to claim that the rounding errors had not 'accumulated' at all.

This does not imply that the solution would be accurate. Its accuracy would depend on the *condition of that zero with respect to perturbations in the coefficients of the polynomial.* If the corresponding condition number were very large, then the computed zero might be very inaccurate. In this case however, it would be unreasonable to say that the effect of the rounding errors had 'built up'. Indeed it seems reasonable to assert that the computed solution has the maximum accuracy we can expect from t-digit computation.

40. More frequently we would find that the most that could be proved was that each computed zero was the exact solution of some polynomial

$$a'_n x^n + \ldots + a'_0 \tag{40.1}$$

with some bounds for the $|a'_r - a_r|$ which were functions of r and n. Suppose for example we could prove that

$$|(a'_r - a_r)/a_r| \leqslant nr2^{-t}. \tag{40.2}$$

As an example let us take $n = 32$, so that we have
a fortiori

$$|(a'_r - a_r)/a_r| \leqslant 2^{10-t}. \tag{40.3}$$

In this case we could say that every computed zero was the exact solution of some $(t-10)$-digit approximation to the given polynomial. Note that if the coefficients a_r of the original polynomial required the full t digits for their representation, then the bounds for the errors in the solution resulting from the rounding errors made during the computation on a t-digit computer would be the same as the bounds for the errors which would arise immediately on transferring the polynomial to a $(t-10)$-digit computer. In as far as our bound (40.2) is likely to be a strict upper bound, we might expect statistical considerations to reduce the factor nr to $(nr)^{1/2}$, in which case the correct comparison would be with a $(t-5)$-digit computer.

In this example it would be reasonable to say that there had been some *accumulation* of rounding error. The effect of this accumulation has been to introduce the factor 2^{10} into the extreme upper bound for the error in a computed zero. We shall see in Chapter 2 that very large values of the condition numbers K_j are quite common even in polynomials of such a modest degree as 32, and that the K_j are potentially more damaging than the factor 2^{10}.

41. The considerations of sections 36–37 apply to almost any computing problem and it seems highly improbable that any method of solving any problem which is restricted to the use of t-digit arithmetic will, in general, do better than give the solution to a t-digit approximation to the original problem. Indeed if a very large number of elementary operations were involved it would be surprising if a method produced answers which were as good as this.

If we are working on a computer on which inner-products can be accumulated, then there may be stages in the work when we obtain results which are correct to $2t$ binary figures. *There is always the possibility that these stages may be of vital importance to the accuracy of the final answers, and when this is true we may obtain results which are considerably more accurate than those corresponding to t-digit approximations of the original problem.* A particularly favourable situation is that in which an accumulated inner-product of $2t$ digits is divided by a t-digit divisor.

Occasionally problems are encountered in which the parameters are exact numbers requiring few digits for their representation and in which all the numbers computed at intermediate stages are also of this type. For such a problem our computed answers will, of course, be exact. Even if rounding errors do occur at some stages of the computation, the fact that part of it is performed without error may lead to answers of exceptional accuracy. It is obvious that such examples can shed no light on our general problem, yet it is surprising how often such numerical examples are displayed and the high accuracy of the results used as a recommendation for a method which may be quite inaccurate when applied to an example in which rounding errors occur throughout.

Additional comments

The inequalities satisfied by the rounding errors in the basic fixed-point arithmetic operations were given by von Neumann and Goldstine in their paper on the inversion of matrices [**18**]. The notation and general lines of treatment of that paper were used by several subsequent writers; of particular importance was their use by Givens in his error analysis of the Givens' method for the calculation of the eigenvalues of a real symmetric matrix [**8**].

The inequalities satisfied by the errors in the basic floating-point arithmetic operations, roughly in the form given here, were first used by the author in 1957. After a period of development they were presented in 1960 in a paper on error analysis [**29**] which included several applications. The analysis in that paper was specifically for the rounding procedures described in sections 13–16. With these procedures the result of each of the basic operations is a number having a low relative error. This has led some to suppose that the analysis is essentially different for other rounding procedures. Our object here has been to emphasize that the error analysis is substantially the same for all the usual rounding procedures which are used in floating-point arithmetic.

The idea of a *backward* error analysis, was to some extent implicit in the papers of von Neumann and Goldstine [**18**] and Turing [**23**]. It was described explicitly in Givens' paper [**8**] in the section on the calculation of the eigenvalues of a tri-diagonal matrix by the Sturm-sequence process. The error analysis in that paper seems not to have attracted as much attention as it deserved, possibly because it was not published in a readily available journal. Backward analysis has been used extensively by the author for the treatment of algebraic processes and has the advantage of suggesting automatically a convenient basis for comparison with the computed values.

The term *condition number* seems first to have been used by Turing in his paper on rounding errors in matrix processes [**23**], though the term 'ill-condition' had been in common use among numerical analysts for some considerable time before this. It is important that the concept of the condition of a computing problem should be clearly understood, and that errors which are the result of inherent instability of a given problem should not be confused with those resulting from a genuine accumulation of rounding errors or from the fundamental instability of a computing technique.

Block-floating computation was used by desk computers long before the advent of modern high speed computers, though it was not necessary on a desk computer to adhere so rigidly to the procedure we have described. It was extensively used on the Pilot ACE at the National Physical Laboratory and later on DEUCE and ACE. It is possible that with the development of automatic floating-point facilities, and particularly of floating-point accumulation of inner-products, it will be less important in the future than it has been hitherto. On computers which have no automatic floating-point operations it is extremely effective and the appropriate error analysis is particularly instructive.

2

COMPUTATIONS INVOLVING POLYNOMIALS

EVALUATION OF POWER SERIES

1. In general, the evaluation of a polynomial is more conveniently performed in floating-point, since the numbers which arise are likely to cover a very wide range. However, fixed-point work is sometimes used in connexion with the evaluation of transcendental functions which have been represented by truncated power series. We do not wish to discuss the general problem of approximating transcendental functions and we consider only a very simple example in order to illustrate the main considerations when polynomials are used. The series we discuss is unlikely to be used in practice.

FIXED-POINT REPRESENTATION

2. We consider the calculation of e^x for values of x in the range $-4 \leqslant x \leqslant 4$ using a fixed number of terms of the power series, and for comparison we give the analyses appropriate to fixed-point and floating-point computation. We consider fixed-point computation first and assume we have a binary computer for which $t = 48$.

The maximum value of e^x in the given range is e^4 which is approximately 55, so that we work with the function $2^{-6}e^x$. The error made in truncating the power series at the nth term is bounded by a, where

$$a \leqslant 2^{-6}\left[\frac{4^{n+1}}{(n+1)!} + \frac{4^{n+2}}{(n+2)!} + \ldots\right] < 2^{-6}\frac{4^{n+1}}{(n+1)!}\Big/\left(1 - \frac{4}{n+2}\right). \quad (2.1)$$

For $n = 28$ we have $a < 2^{-49}$, so that the neglected terms are negligible to 48 binary places. If we denote the exact sum of the first 29 terms by $s(x)$ then

$$\left|2^{-6}e^x - s(x)\right| < 2^{-49} \quad (|x| \leqslant 4). \quad (2.2)$$

Since we must work with arguments in the range ± 1, we introduce y defined by

$$x = 4y \quad (2.3)$$

and work with $s(x)$ in the form

$$a_0 + a_1 y + \ldots + a_{28}y^{28} \quad (2.4)$$

where

$$a_r = 2^{-6}4^r/r!. \quad (2.5)$$

The coefficients are all less than unity, the largest being a_3 and a_4 which are $\frac{1}{6}$. Further we have

$$|a_r + a_{r+1}y + \ldots + a_{28}y^{28-r}| \leqslant \sum_0^{28} |a_r| < 2^{-6}e^4 + 2^{-49}$$
$$< 0{\cdot}9 \text{ (say)} \qquad (2.6)$$

so that there is no danger of the quantity on the left-hand side of (2.6) exceeding unity.

We must now represent each a_r by its 48-digit approximation $\bar{a}_r$ and we may write

$$\bar{a}_r = a_r + e_r \quad (|e_r| \leqslant 2^{-49}). \qquad (2.7)$$

Writing $\bar{s}(x)$ for the exact sum of the polynomial in $\bar{a}_r$ we have

$$|s(x) - \bar{s}(x)| = |\sum_0^{28} e_r y^r| \leqslant 2^{-49}(1 - |y|^{29})/(1 - |y|) \qquad (2.8)$$

where we take the multiplier of 2^{-49} to be 29 if $|y| = 1$.

Since the a_r are defined exactly, we could obtain the e_r as accurately as we please and then determine the maximum of the function $\sum_0^{28} e_r y^r$ in the y-range, but in practice we would not bother to do this unless the error bound was of great importance. Relations (2.2) and (2.8) show that

$$|2^{-6}e^x - \bar{s}(x)| \leqslant 30 \times 2^{-49}$$

for all x in the range.

3. We now turn to the errors made in evaluating the polynomial by nested multiplication. We define quantities $u_r(x)$ by the relations

$$u_{28}(x) = \bar{a}_{28} \qquad (3.1)$$

$$u_r(x) = fi[yu_{r+1}(x) + \bar{a}_r] \equiv yu_{r+1} + \bar{a}_r + f_r \qquad (3.2)$$

$$|f_r| \leqslant 2^{-49}. \qquad (3.3)$$

These relations show that $u_0(x)$, the computed value of the polynomial, *is the exact value of the polynomial with coefficients* $\bar{a}_r + f_r$. The error introduced during the calculation is of exactly the same type as that introduced by the original representation of the a_r to 48 binary places. We have in fact

$$|u_0(x) - \bar{s}(x)| = |\sum_0^{27} f_r y^r| \leqslant 2^{-49}(1 - |y|^{28})/(1 - |y|) \qquad (3.4)$$

giving as our final error bound

$$|u_0(x) - 2^{-6}e^x| < 2^{-49}\left[1 + \frac{1 - |y|^{28}}{1 - |y|} + \frac{1 - |y|^{29}}{1 - |y|}\right]. \qquad (3.5)$$

The extreme upper bound of the error in e^x for all x in the range is given by

$$|2^6 u_0(x) - e^x| < 58 \times 2^{-43}. \qquad (3.6)$$

For the relative error we have

$$\left|\frac{2^6 u_0(x)}{e^x} - 1\right| < e^{-x}2^{-43}\left[1 + \frac{1 - |y|^{28}}{1 - |y|} + \frac{1 - |y|^{29}}{1 - |y|}\right] \tag{3.7}$$

which takes its greatest value $58e^4 \times 2^{-43}$, when $x = -4$. This absolute error bound for e^x is the same as that for e^{-x} so that the bound for the relative error of the latter function is worse by the factor e^{2x}. For negative values of x considerable cancellation takes places in the summation.

FLOATING-POINT REPRESENTATION

4. We now consider floating-point representation and for the moment we shall assume that the series is truncated at the same term and that the mantissa has 48 binary digits; we comment on this later. There is now no need to introduce the factor 2^{-6} and we work directly with x and the series $p(x)$ defined by

$$p(x) = b_0 + b_1 x + \ldots + b_{28}x^{28} \tag{4.1}$$

$$b_r = \frac{1}{r!}. \tag{4.2}$$

We have

$$|e^x - p(x)| < 2^{-43}. \tag{4.3}$$

The polynomial $p(x)$ is replaced by $\bar{p}(x)$ defined by

$$\bar{p}(x) = \bar{b}_0 + \bar{b}_1 x + \ldots + \bar{b}_{28}x^{28} \tag{4.4}$$

where

$$\bar{b}_r = b_r(1+e_r) \quad (|e_r| \leqslant 2^{-48}). \tag{4.5}$$

Since the b_r decrease rapidly the bounds for the errors in the $\bar{b}_r$ also decrease rapidly. We may contrast this with the fixed-point series when the bound for the error in every coefficient is the same. The exact sum $\bar{p}(x)$, satisfies

$$|\bar{p}(x) - p(x)| \leqslant \sum_0^{28} b_r e_r |x|^r \leqslant 2^{-48}e^{|x|}. \tag{4.6}$$

Turning now to the calculation of the polynomial, we define $u_r(x)$ by the relations

$$u_{28}(x) = \bar{b}_{28} \tag{4.7}$$

$$\begin{aligned} u_r(x) &= fl[xu_{r+1}(x) + \bar{b}_r] \\ &= xu_{r+1}(x)(1+f_r)(1+g_r) + \bar{b}_r(1+g_r) \end{aligned} \tag{4.8}$$

$$|f_r| \leqslant 2^{-48} \qquad |g_r| \leqslant 2^{-48}. \tag{4.9}$$

Combining these equations and remembering (4.5) we have

$$u_0(x) \equiv b_0(1+E_0) + b_1(1+E_1)x + \ldots + b_{28}(1+E_{28})x^{28} \tag{4.10}$$

where

$$(1-2^{-48})^{2r+2} \leqslant 1+E_r \leqslant (1+2^{-48})^{2r+2} \tag{4.11}$$

$$(1-2^{-48})^{58} \leqslant (1-2^{-48})^{57} \leqslant 1+E_{28} \leqslant (1+2^{-48})^{57}$$

$$\leqslant (1+2^{-48})^{58}. \tag{4.12}$$

Now we certainly have

$$|E_r| < (1\cdot 01)(2r+2)2^{-48} \tag{4.13}$$

and hence

$$|u_0(x) - p(x)| < (1\cdot 01)2^{-48}[2b_0 + 4b_1|x| + \ldots + 58b_{28}|x|^{28}]$$

$$< (2\cdot 02)2^{-48}[e^{|x|} + |x|e^{|x|}], \tag{4.14}$$

giving

$$|e^x - u_0(x)| < 2^{-43} + (2\cdot 02)2^{-48}[e^{|x|} + |x|e^{|x|}]. \tag{4.15}$$

It is evident from (4.15) that for small values of x it is worthwhile to replace the term 2^{-43} in the bound by the appropriate bound involving x. We replace (4.3) by

$$|e^x - p(x)| \leqslant \frac{|x^{29}|}{29!}\left[1 + \frac{|x|}{30} + \frac{|x^2|}{30.31}\cdots\right] \leqslant \frac{7}{6}\frac{|x|^{29}}{29!} \leqslant 2^{-43}\left|\frac{x}{4}\right|^{29}. \tag{4.16}$$

This term decreases rapidly with $|x|$ and is always less than the term $(2\cdot 02)2^{-48}e^{|x|}(1+|x|)$. There is little point therefore in having additional terms in the series.

The error bound takes its maximum values at $x = \pm 4$ at both of which it is $2^{-43} + (10\cdot 1)2^{-48}e^4$ compared with 58×2^{-43} for fixed-point computation. The floating-point error bound is about one third of that for fixed-point. Again the relative error at $x = -4$ is higher than that at $x = +4$ by the factor e^8.

It should be remembered, however, that we have assumed that the mantissa of a floating-point number has the same number of digits as a fixed-point number. This will usually not be so; usually the number of digits allocated to the exponent will be such that fixed-point computation gives the greater accuracy.

CALCULATION OF ZEROS OF FUNCTIONS DEFINED BY POWER SERIES

5. The results we have just derived can be used to show that it is comparatively simple to calculate very accurately the real zeros of transcendental functions defined by power series and to find error bounds for the computed zeros. The first 50 zeros each of $J_0(x)$, $J_{1/2}(x)$, $J_1(x)$, . . . , $J_6(x)$ were computed to 15 decimal places on DEUCE using fixed-point arithmetic by the method of successive linear interpolation. High-precision arithmetic was of course required for the later zeros. Rigorous bounds were determined as follows. Each zero a, was determined to somewhat higher

precision than required and the function $J_n(x)$ was then evaluated at $(a+h)$ and $(a-h)$ for some suitable h. This h was chosen to be as small as possible consistent with the following requirements

(i) The computed values of $J_n(a+h)$ and $J_n(a-h)$ were of opposite signs.

(ii) The computed $J_n(a+h)$ and $J_n(a-h)$ were of sufficiently large absolute magnitude that the errors made in their computation could not have affected their signs.

Although much more powerful methods were subsequently developed for computing the zeros on DEUCE, the method we have just described was quite practical and had the advantage of being very general and simple to programme. Although it was necessary to use coefficients of multiple precision, double-precision *arguments* were adequate for all requirements. (On ACE single-precision arguments would have been adequate.)

POLYNOMIALS WITH ARBITRARY COEFFICIENTS

6. Experience with polynomials derived by truncating power series may mislead one into thinking that the use of high order polynomials does not lead to computational difficulties. However it must be appreciated that truncated power series are not typical of polynomials in general. They have the special feature that the terms decrease rapidly in size for values of x in the range for which they are appropriate. Further, one commonly knows a great deal about the behaviour of the transcendental function which is being approximated. Thus in the computation concerning the Bessel functions, rough approximations to the required zeros could be computed by a great variety of methods.

A tendency to underestimate the difficulties involved in working with general polynomials is perhaps a consequence of one's experience in classical analysis. There it is natural to regard a polynomial as a very desirable function since it is bounded in any finite region and has derivatives of all orders. In numerical work, however, polynomials having coefficients which are more or less arbitrary are tiresome to deal with by entirely automatic procedures. The remainder of this chapter is devoted to the classical problem of computing the zeros of a polynomial.

CONDITION OF A POLYNOMIAL WITH RESPECT TO THE COMPUTATION OF ITS ZEROS

7. We consider the determination of the n zeros of the polynomial

$$f(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_1 z + a_0 \qquad (7.1)$$

where it is assumed that the coefficients are the primary data. We have remarked in Chapter 1 that one of the most important characteristics of any computing problem is the sensitivity of the solutions with respect to changes in the data and we consider that problem first.

Let the zeros of the given polynomial be $z_1, z_2, \ldots, z_n$ and assume for the moment that z_r is an isolated zero; the other zeros may be of any multiplicity. We consider now the zero of $f(z) + \epsilon g(z)$ where

$$g(z) = b_n z^n + b_{n-1} z^{n-1} + \ldots + b_0 . \tag{7.2}$$

By the general theory of algebraic functions [**10**] we know that one zero $z(\epsilon)$ of $f(z) + \epsilon g(z)$ is given by

$$z_r(\epsilon) = z_r + \sum_1^\infty p_k \epsilon^k = z_r + h \text{ (say)} \tag{7.3}$$

where the series in ϵ is certainly convergent for sufficiently small ϵ. We have therefore

$$\sum_0^n a_i (z_r + h)^i + \epsilon \sum_0^n b_i (z_r + h)^i = 0 \tag{7.4}$$

and we may write this in the form

$$\sum_0^n c_k h^k + \epsilon \sum_0^n d_k h^k = 0 \tag{7.5}$$

where

$$c_k = \frac{1}{k!} f^{(k)}(z_r) \qquad d_k = \frac{1}{k!} g^{(k)}(z_r) . \tag{7.6}$$

Since the power series for h is convergent for sufficiently small ϵ, the power series on the left of equation (7.5) must vanish identically. Equating the coefficient of ϵ to zero and remembering that $c_0 = f(z_r) = 0$, we have

$$c_1 p_1 + d_0 = 0$$

giving

$$p_1 = -g(z_r)/f'(z_r) . \tag{7.7}$$

Hence

$$\left| z_r(\epsilon) - z_r + \epsilon \frac{g(z_r)}{f'(z_r)} \right| < K\epsilon^2 \tag{7.8}$$

for sufficiently small ϵ.

If z_r is a zero of multiplicity m, then again from the theory of algebraic functions [**10**] we know that there are m zeros of $f(z) + \epsilon g(z)$ defined by

$$z_r(\epsilon) = z_r + \sum_1^\infty p_k \epsilon^{k/m} = z_r + h , \tag{7.9}$$

one zero corresponding to each interpretation of $\epsilon^{1/m}$, the series being convergent for sufficiently small ϵ. Since

$$f(z_r) = f'(z_r) = \ldots = f^{m-1}(z_r) = 0$$

we have

$$\sum_m^n c_k h^k + \epsilon \sum_0^n d_k h^k = 0 . \tag{7.10}$$

Substituting for h from (7.9), each term in (7.10) becomes a power series in $\epsilon^{1/m}$ which is convergent for all sufficiently small ϵ. Equating the coefficient of $(\epsilon^{1/m})^m$ to zero gives

$$c_m p_1^m + d_0 = 0$$

$$p_1 = \left[-\frac{d_0}{c_m}\right]^{1/m} = \left[-\frac{m!g(z_r)}{f^m(z_r)}\right]^{1/m} \tag{7.11}$$

and we have

$$\left|z_r(\epsilon) - (z_r + p_1\epsilon^{1/m})\right| < K\epsilon^{2/m} \tag{7.12}$$

for sufficiently small ϵ. Relations (7.8) and (7.12) are the fundamental results on which the subsequent analysis is based.

Although the m coincident zeros are perturbed by amounts of moduli δ such that

$$|\delta| \sim |p_1\epsilon^{1/m}| \quad (\epsilon \to 0) \tag{7.13}$$

the perturbation in the sum of the m zeros is of order ϵ. For taking the m different values of $z_r(\epsilon)$ corresponding to the m interpretations of $\epsilon^{1/m}$ in (7.9) we have

$$\sum z_r(\epsilon) = m[z_r + p_m\epsilon + p_{2m}\epsilon^2 + \ldots]. \tag{7.14}$$

Similarly every integral symmetric function of the perturbed zeros is a polynomial in ϵ.

The most important case is that of a double zero. We have then

$$p_1 = [-2g(z_r)/f''(z_r)]^{1/2} \tag{7.15}$$

and (7.10) becomes

$$c_2[p_1\epsilon^{1/2} + p_2\epsilon + \ldots]^2 + c_3[p_1\epsilon^{1/2} + p_2\epsilon + \ldots]^3 + \ldots$$
$$+ \epsilon\{d_0 + d_1[p_1\epsilon^{1/2} + p_2\epsilon + \ldots] + \ldots\} = 0. \tag{7.16}$$

Equating the coefficient of $\epsilon^{3/2}$ to zero, we have

$$2p_1p_2c_2 + c_3p_1^3 + d_1p_1 = 0$$

$$p_2 = -\left[\frac{c_3p_1^2 + d_1}{2c_2}\right]$$

$$= \frac{f'''(z_r)\,g(z_r) - 3f''(z_r)\,g'(z_r)}{3[f''(z_r)]^2}. \tag{7.17}$$

The two perturbed zeros, α and β, may be expressed in the form

$$\left.\begin{aligned}\alpha &= z_r + p_1\epsilon^{1/2} + p_2\epsilon + \ldots\\ \beta &= z_r - p_1\epsilon^{1/2} + p_2\epsilon + \ldots\end{aligned}\right\} \tag{7.18}$$

so that the corresponding quadratic factor is

$$z^2 - (2z_r + 2p_2\epsilon + \ldots)z + [z_r^2 + (2p_2z_r - p_1^2)\epsilon + \ldots]. \tag{7.19}$$

If we are interested in the coefficients of the quadratic factor rather than the individual zeros, then a double zero would not in general imply that our problem was ill-conditioned, since the relevant perturbations are of order ϵ rather than $\epsilon^{1/2}$. Similar remarks apply for zeros of higher multiplicities.

In equations (7.7), (7.8), (7.15) and (7.17) we have tacitly assumed that $g(z_r) \neq 0$. When $g(z_r) = 0$, and z_r is a zero of $g(z)$ of multiplicity s, the equation for the perturbed zeros reduces to

$$(z-z_r)^m p(z) + \epsilon(z-z_r)^s q(z) = 0 \tag{7.20}$$

where

$$p(z_r),\ q(z_r) \neq 0.$$

z_r is therefore a zero of the perturbed polynomial of multiplicity $\min(m, s)$. We may say that the perturbations are peculiarly correlated. Correlated perturbations are of great importance in connexion with the solution of the eigenvalue problem via the explicit computation of the characteristic polynomial.

SOME TYPICAL DISTRIBUTIONS OF ZEROS

8. Since multiple zeros are necessarily ill-conditioned, we would naturally expect very close zeros, particularly clusters of such zeros, to be ill-conditioned. However we must not allow a preoccupation with the sensitivity of multiple zeros to blind us to the fact that zeros which could not be described as pathologically close, may be extremely ill-conditioned. We have seen that

$$z_r(\epsilon) - z_r \sim -\epsilon \frac{g(z_r)}{f'(z_r)} \quad (\epsilon \to 0) \tag{8.1}$$

but it is possible for the coefficient of ϵ to be very large nonetheless. We consider now some typical distributions and for comparison we take polynomials of degree 20 in each case. It will be interesting to see the effect of perturbations in the individual coefficients, and for a perturbation ϵ in the coefficient a_k we take $b_k = 1$, $b_i = 0$ $(i \neq k)$ in the results of section 7. For convenience we assume that $a_n = 1$.

LINEAR DISTRIBUTIONS OF ZEROS

9. Consider first the distribution 1, 2, 3, . . . , 20. For sufficiently small ϵ, (7.8) shows that the perturbation δz_r in the rth zero corresponding to a perturbation ϵ in the coefficient a_k satisfies the relation

$$\begin{aligned} \delta z_r &\sim -\epsilon \frac{(z_r)^k}{f'(z_r)} \\ &= \pm \epsilon \frac{r^k}{(20-r)!\,(r-1)!} = \pm A\epsilon \text{ (say)}. \end{aligned} \tag{9.1}$$

A is clearly a maximum when $k = 19$. Keeping k fixed at 19 and varying r we find that A takes its largest value, $\dfrac{16^{19}}{4!\,15!} \doteqdot 0{\cdot}24 \times 10^{10}$, when $r = 16$ and its smallest value, $\dfrac{1}{19!} \doteqdot 0{\cdot}82 \times 10^{-17}$ when $r = 1$. Hence most of the larger zeros are very sensitive to changes in the coefficients of the higher powers of z while the smallest zero is very insensitive.

In making this assessment, we have considered the effect of the same error in each coefficient. From the point of view of practical applications, this is rather unrealistic. We are more likely to have coefficients which all have the same *relative* error. If the polynomial is computed using t-digit floating-point arithmetic, then even if we obtain correctly rounded coefficients the error in a_k will lie in the range $\pm 2^{-t}a_k$ and we therefore consider the effect of such perturbations. We have

$$\delta z_r \sim \pm 2^{-t}a_k \frac{r^k}{(20-r)!\,(r-1)!}. \tag{9.2}$$

An assessment of the effect of the perturbations requires a knowledge of the order of magnitude of the coefficients of the polynomial, and these are shown below.

$$\begin{aligned} &z^{20} + 10^3z^{19} + 10^5z^{18} + 10^7z^{17} + 10^8z^{16} + 10^{10}z^{15} \\ &+ 10^{11}z^{14} + 10^{12}z^{13} + 10^{13}z^{12} + 10^{14}z^{11} + 10^{16}z^{10} \\ &+ 10^{17}z^9 + 10^{17}z^8 + 10^{18}z^7 + 10^{19}z^6 + 10^{19}z^5 + 10^{19}z^4 \\ &+ 10^{20}z^3 + 10^{20}z^2 + 10^{19}z + 10^{19}. \end{aligned} \tag{9.3}$$

The maximum value of δz_r now occurs when $r = 16$ and $k = 15$ for which we have

$$\delta z_r \sim 2^{-t}10^{10}\frac{16^{15}}{4!\,15!} \doteqdot 2^{-t}(3{\cdot}7 \times 10^{14}). \tag{9.4}$$

For single-precision computation on ACE t is 48, a comparatively long word, and yet we have

$$\delta z_r \doteqdot 1{\cdot}3. \tag{9.5}$$

This shows that we are outside the range when the perturbation in the 16th zero is approximately equal to the first term in its expansion.

The sensitivity of the zeros is illustrated in Table 1. We give there accurate zeros of $(z-1)(z-2)\ldots(z-20) + \epsilon z^{19}$ when ϵ takes the values -2^{-23} and -2^{-55}. It will be seen that the smaller zeros are very insensitive to the perturbations while the zeros from 10 to 20 are very sensitive.

TABLE 1

Accurate zeros of $(z-1)(z-2)\ldots(z-19)(z-20) - 2^{-23}z^{19}$

1·00000 0000	10·09526 6145 ± 0·64350 0904i
2·00000 0000	11·79363 3881 ± 1·65232 9728i
3·00000 0000	13·99235 8137 ± 2·51883 0070i
4·00000 0000	16·73073 7466 ± 2·81262 4894i
4·99999 9928	19·50243 9400 ± 1·94033 0347i
6·00000 6944	Note that 5 pairs of zeros have become complex. Changes are so great that linearized perturbation theory does not apply.
6·99969 7234	
8·00726 7603	
8·91725 0249	
20·84690 8101	

Accurate zeros of $(z-1)(z-2)\ldots(z-19)(z-20) - 2^{-55}z^{19}$

1·00000 0000	6·00000 0000	10·99999 9999	16·00000 0067
2·00000 0000	7·00000 0000	12·00000 0006	16·99999 9947
3·00000 0000	8·00000 0000	12·99999 9983	18·00000 0028
4·00000 0000	9·00000 0000	14·00000 0037	18·99999 9991
5·00000 0000	10·00000 0000	14·99999 9941	20·00000 0001

Changes are now accurately predicted by linearized perturbation theory.

For comparison we now consider the linear distribution of zeros $(k+1)$, $(k+2)$, . . . , $(k+20)$. We have

$$\frac{\partial z_r}{\partial a_s} = \pm \frac{(k+r)^s}{(20-r)!\,(r-1)!}. \tag{9.6}$$

For $k = 20$, $s = 19$, $r = 15$, this gives

$$\frac{\partial z_{15}}{\partial a_{19}} = \frac{(35)^{19}}{5!\,14!} \fallingdotseq 0{\cdot}21 \times 10^{17}. \tag{9.7}$$

It is evident that the sensitivity becomes more marked as k increases and if we consider a change of 1 part in 2^t in the coefficients this effect is even more pronounced.

For $k = -10$, however, the polynomial is much better-conditioned. The worst zero is the 18th for which we have

$$\frac{\partial z_{18}}{\partial a_{19}} = \frac{(8)^{19}}{2!\,17!} \fallingdotseq 0{\cdot}20 \times 10^{3}. \tag{9.8}$$

10. As a second example we consider a polynomial with zeros 2^{-1}, $2^{-2}, \ldots, 2^{-20}$, in geometric progression. The coefficients of this polynomial vary enormously in size; their orders of magnitude are indicated below.

$$z^{20} + z^{19} + 2^{-1}z^{18} + 2^{-4}z^{17} + 2^{-8}z^{16} + 2^{-13}z^{15} + 2^{-19}z^{14}$$
$$+ 2^{-26}z^{13} + 2^{-34}z^{12} + 2^{-43}z^{11} + 2^{-53}z^{10} + 2^{-64}z^{9} + 2^{-76}z^{8}$$
$$+ 2^{-89}z^{7} + 2^{-103}z^{6} + 2^{-118}z^{5} + 2^{-134}z^{4}$$
$$+ 2^{-151}z^{3} + 2^{-169}z^{2} + 2^{-189}z + 2^{-209}. \tag{10.1}$$

Clearly it will make a considerable difference whether we consider perturbations of fixed *absolute* or *relative* magnitude in each of the coefficients. We frame our analysis so that we can deal with both of them. We have

$$\delta z_r \sim -\delta a_k 2^{-kr}/PQ \tag{10.2}$$

where

$$P = (2^{-r}-2^{-1})(2^{-r}-2^{-2}) \ldots (2^{-r}-2^{-r+1}) \tag{10.3}$$

and

$$Q = (2^{-r}-2^{-r-1})(2^{-r}-2^{-r-2}) \ldots (2^{-r}-2^{-20}). \tag{10.4}$$

We have

$$P = (-1)^{r-1}\frac{1}{2^{\frac{1}{2}r(r-1)}}[(1-2^{-r+1})(1-2^{-r+2}) \ldots (1-2^{-1})] \tag{10.5}$$

$$Q = \frac{1}{2^{r(20-r)}}[(1-2^{-1})(1-2^{-2}) \ldots (1-2^{-20+r})]. \tag{10.6}$$

The expressions in square brackets are convergents to the infinite product

$$(1-2^{-1})(1-2^{-2})(1-2^{-3}) \ldots$$

and quite a crude inequality shows that they lie between $\frac{1}{2}$ and $\frac{1}{4}$, so that

$$\tfrac{1}{4}2^{-\frac{1}{2}r(39-r)} > |PQ| > \tfrac{1}{16}2^{-\frac{1}{2}r(39-r)} \tag{10.7}$$

and

$$|\delta z_r| < 16|\delta a_k|2^{\frac{1}{2}r(39-r-2k)}. \tag{10.8}$$

Because of the great disparity in size of the zeros it is more reasonable to consider the relative change in a zero and we have

$$\left|\frac{\delta z_r}{z_r}\right| < 16|\delta a_k|2^{\frac{1}{2}r(41-r-2k)}. \tag{10.9}$$

For a fixed value of k this takes its maximum value when $r = 20 - k$ so that

$$\left|\frac{\delta z_r}{z_r}\right| < 16|\delta a_k|2^{\frac{1}{2}(20-k)(21-k)} \text{ for all } k. \tag{10.10}$$

From the order of magnitude of the a_k given in (10.1) it is evident that

$$|a_k| < 4\cdot 2^{-\frac{1}{2}(20-k)(21-k)}. \tag{10.11}$$

Hence we have

$$\left|\frac{\delta z_r}{z_r}\right| < 64\left|\frac{\delta a_k}{a_k}\right| |a_k| 2^{\frac{1}{2}(20-k)(21-k)}$$

$$< 64\left|\frac{\delta a_k}{a_k}\right| . \qquad (10.12)$$

The relation (10.12) shows that small relative changes in the coefficients cannot lead to large relative changes in any of the zeros. If we take $|\delta a_k/a_k| = 2^{-t}$ we have

$$\left|\frac{\delta z_r}{z_r}\right| < 2^{6-t} \qquad (10.13)$$

so that not more than 6 significant binary figures can be affected in any zero. It may readily be verified that the smaller zeros are the more stable. Although the absolute difference between the smaller zeros is small, so that one might be tempted to regard these zeros as 'close', the ratio of each of these zeros to the next is 2 and it is this ratio which is important. If we were to add a twenty-first zero $2^{-20}(1+\eta)$, where η were small, then the zeros 2^{-20} and $2^{-20}(1+\eta)$ would be ill-conditioned. In Table 2 we show the effect of a perturbation 2^{-31} in a_{19}.

TABLE 2

Accurate zeros of $(z-2^{-1})(z-2^{-2})\ldots(z-2^{-20}) - 2^{-31}z^{19}$

0·50000 0001	0·48828 1250 × 10^{-3}
0·24999 9998	0·24414 0625 × 10^{-3}
0·12500 0000	0·12207 0313 × 10^{-3}
0·62499 9999 × 10^{-1}	0·61035 1563 × 10^{-4}
0·31250 0000 × 10^{-1}	0·30517 5781 × 10^{-4}
0·15625 0000 × 10^{-1}	0·15258 7891 × 10^{-4}
0·78125 0000 × 10^{-2}	0·76293 9453 × 10^{-5}
0·39062 5000 × 10^{-2}	0·38146 9727 × 10^{-5}
0·19531 2500 × 10^{-2}	0·19073 4863 × 10^{-5}
0·97656 2500 × 10^{-3}	0·95367 4316 × 10^{-6}

Note that to 9 significant decimals only 3 of the zeros are changed and these by only one or two in the 9th significant figure.

If we consider absolute changes of 2^{-t} in the coefficients, the situation is very different, and the extreme sensitivity of some of the zeros can be seen immediately from (10.2) and (10.7) as follows. Before the perturbation we have

$$z_1 z_2 \ldots z_{20} = 2^{-210}. \qquad (10.14)$$

If the last coefficient is changed from 2^{-210} to $(2^{-210}+2^{-48})$ then the perturbed zeros z_r' satisfy

$$z_1'z_2'\ldots z_{20}' = 2^{-210}+2^{-48} = (2^{162}+1)z_1z_2\ldots z_{20}. \quad (10.15)$$

From this relation it is obvious that for at least one value of r

$$\left|\frac{z_r'}{z_r}\right| \geqslant (2^{162}+1)^{1/20} > 2^8. \quad (10.16)$$

The significance of this result will become apparent later (section 24).

CHEBYSHEV POLYNOMIAL

11. Finally we consider the polynomial $z^{20}-1$, which has the zeros $e^{i2\pi r/20}$ $(r = 0, 1, \ldots, 19)$, and the Chebyshev polynomial $T_{20}(z)$ defined by

$$T_{20}(z) = \cos(20\cos^{-1}z), \quad (11.1)$$

which has the zeros $\cos\dfrac{(2r+1)\pi}{40}$ $(r = 0, 1, \ldots, 19)$. The former zeros are uniformly distributed round the unit circle while the latter are the projections on the real axis of the points $e^{i(2r+1)\pi/40}$ $(r = 0, 1, \ldots, 19)$ which are uniformly distributed on the upper unit semi-circle.

For the polynomial $z^{20}-1$ we have

$$\delta z_r \sim -\frac{(\delta a_k)z_r{}^k}{f'(z_r)} = -\frac{(\delta a_k)z_r{}^k}{20z_r{}^{19}} \quad (11.2)$$

giving

$$|\delta z_r| \sim \frac{|\delta a_k|}{20}. \quad (11.3)$$

This shows that all the zeros are very well-conditioned, a change in any coefficient making a considerably smaller change in each of the zeros.

The polynomial $T_{20}(z)$ is

$$5\ 24288z^{20} - 26\ 21440z^{18} + 55\ 70560z^{16} - 65\ 53600z^{14} + 46\ 59200z^{12}$$
$$- 20\ 50048z^{10} + 5\ 49120z^8 - 84480z^6 + 6600z^4 - 200z^2 + 1. \quad (11.4)$$

When divided through by the coefficient of z^{20}, the modulus of the maximum coefficient is 12·5 so that none of the coefficients is large. We have

$$\delta z_r \sim -\left(\frac{\delta a_k}{a_k}\right)\frac{a_k\cos^k\dfrac{(2r+1)\pi}{40}}{\prod\limits_{s\neq r}\left[\cos\dfrac{(2r+1)\pi}{40}-\cos\dfrac{(2s+1)\pi}{40}\right]}. \quad (11.5)$$

From the geometrical picture it is clear that the zeros nearest to ± 1 are 'closest' together, and since for these $\cos\dfrac{(2r+1)\pi}{40}$ is near to unity we will expect them to be the worse-conditioned. When $r = 1$ the denominator of the right hand side of (11.5) is approximately $-1\cdot 6(10^{-4})$ so that we have

$$\delta z_1 \sim \left(\frac{\delta a_k}{a_k}\right)\frac{a_k \cos^k \frac{3\pi}{40}}{(1\cdot 6)10^{-4}}$$

$$\doteqdot \left(\frac{\delta a_{14}}{a_{14}}\right)(5\cdot 3)10^4 \quad (\text{when } k = 14). \tag{11.6}$$

The zeros near the ends of the interval $(-1, +1)$ are therefore moderately ill-conditioned, but the central zeros are well-conditioned.

These few examples show that the variation in the condition of the zeros of polynomials is very marked. Extreme ill-condition may exist even for polynomials having zeros which, at first sight, might appear to be quite well separated. The example of a geometrical distribution shows that it is the ratio of neighbouring zeros which is the decisive factor and not their absolute distances. The possession of several zeros having ratios which are fairly close to unity, though by no means pathologically close, always implies ill-conditioning of those zeros. This is well-illustrated by the linear distribution 1, 2, . . . , 20.

Only one of the distributions we considered corresponded to complex zeros and in this example all zeros were very well-conditioned. It is perhaps worth noting that if we take polynomials of fixed order and compare random distributions of zeros in the unit circle with random distributions in the interval $(-1, +1)$ then the former will, in general, be much the better-conditioned. In our experience polynomials with complex zeros which have arisen in practice have had quite well-conditioned zeros.

SIGNIFICANCE OF THE CONDITION OF THE ZEROS OF POLYNOMIALS

12. Although there are numerous physical and mathematical problems which reduce to the calculation of the zeros of polynomials, the coefficients of the polynomials are seldom the primary data. Most commonly the explicit polynomial form is derived from some other form by means of a preliminary computation. We may illustrate this by considering the solution of a set of simultaneous first-order differential equations in n unknowns represented by

$$A\dot{x} = Bx \tag{12.1}$$

where A and B are $n \times n$ matrices. The solutions of these equations are generally of the form $x = ae^{\lambda t}$ where the λ are the roots of

$$\det(A\lambda - B) = 0. \tag{12.2}$$

This is a polynomial equation and we might decide to produce the coefficients of the explicit polynomial and then use one of the techniques for calculating its zeros. Now suppose the roots of (12.2) are comparatively insensitive to changes in the elements of A and B. (Note that if the elements of A and B are the primary data, only those figures in the zeros which are not affected by possible errors in A and B are of any significance.) It may well happen that in spite of this, the zeros of the explicit polynomial are very sensitive to errors in its coefficients. If, for example, the zeros have a distribution of the type 1, 2, . . . , 20, the explicit polynomial is bound to be very ill-conditioned.

Unless we have some previous knowledge of the distribution of zeros, which is very unlikely, then we *must* compute the coefficients of the polynomial to high accuracy to guard against the possibility of its being ill-conditioned.

It may be objected that if the elements of A and B are known only to 8 figures (say), then the exact polynomial corresponding to the *given* A and B will differ from the *true* polynomial corresponding to the correct A and B in the ninth significant figure and therefore no purpose is served in computing it to double-precision, which on ACE for example is 28 decimals. This argument is however fallacious. The zeros of the *exact* polynomial corresponding to the given A and B are obviously the zeros of the given $\det(\lambda A - B)$. Hence by computing this polynomial with sufficient accuracy we can ensure that its zeros are as close as we please to those of the given $\det(A - \lambda B)$. Consequently if, for example, the errors in the given A and B affect only the 7th figure of the zeros, we can be certain of obtaining 7 relevant figures from the polynomial. *The errors in the accurately computed polynomial compared with the polynomial corresponding to the correct A and B will be in the 8th figure, but they will be so correlated that the zeros differ only in the 7th figure.*

13. Not only must the explicit polynomial be computed to high accuracy but, in general, the same high accuracy must be used in any later transformations that may be performed. As a simple example, the polynomial derived from $\det(\lambda A - B)$ will not have the coefficient of λ^n equal to unity. If the method of solution requires this, the division by the coefficient of λ^n must be performed to high accuracy.

As a more subtle example, suppose the zeros of a polynomial are approximately 1, 2, . . . , 20. Then the 15th zero, for example, is very ill-conditioned. If we transform the variable from z to z' given by $z' = z - 15$, then the corresponding zero of the transformed equation is very well-conditioned. The transformation of variables must be carried out to high precision, but by using a shift of origin of exactly 15 (i.e. a small integer) we can make it much simpler to perform the transformation accurately. By similar shifts of origin we can make each zero well-conditioned in turn, and in each case, it will simplify the computation if the shift is by an amount which can be represented by a few digits rather than one requiring the full number of digits which are used in the coefficients.

After computing each of the transformed equations we can dispense with high precision representation since the relevant zero will no longer be very sensitive to small relative changes in the coefficients.

DETERMINATION OF THE ZEROS

14. In the previous section we have been primarily concerned with the effect of errors in the coefficients of the polynomial itself. We now consider the effect of rounding errors made during the course of the process by means of which the zeros are determined. We cannot possibly consider all the various processes which have been devised for computing the zeros and we shall concentrate on iterative methods.

In order to gain some insight into the effect of rounding errors, we shall consider first one of the simplest of all methods. Suppose we have a polynomial of degree n, with real coefficients, having the zeros $z_1, z_2, \ldots, z_n$. Let z_k be an isolated real zero. The other zeros may all be real or there may be some complex conjugate pairs, and they may be of any multiplicity. We shall assume that z_k is not too ill-conditioned for the precision of computation we are using, though the precise meaning of this remark will become apparent only during the course of the argument.

The technique we are going to analyse is known as the *method of bisection*. Suppose we are given two values a and b such that

$$b > a, \qquad f(a) < 0, \qquad f(b) > 0. \tag{14.1}$$

Then, ignoring rounding errors, we may determine a sequence of pairs of values x_r, y_r defined by the following relations:—

$$\left.\begin{aligned} &x_0 = a, \quad y_0 = b, \quad f(x_r) \leqslant 0, \quad f(y_r) > 0. \\ \text{If } &f\left(\frac{x_r + y_r}{2}\right) \leqslant 0 \text{ then } x_{r+1} = \tfrac{1}{2}(x_r + y_r), \quad y_{r+1} = y_r. \\ \text{If } &f\left(\frac{x_r + y_r}{2}\right) > 0 \text{ then } x_{r+1} = x_r, \quad y_{r+1} = \tfrac{1}{2}(x_r + y_r). \end{aligned}\right\} \tag{14.2}$$

There is clearly at least one zero in each of the (x_r, y_r) intervals and the width of the rth interval is $2^{-r}(b-a)$. A zero may therefore be located with any required precision.

15. Consider now the limitations placed on the effectiveness of this process by the use of t-digit floating-point arithmetic. We assume that two values a and b are given and that z_k is the only zero in the interval (a, b).

If x is one of the centre points then the polynomial is evaluated in the following steps:

$$\left.\begin{aligned} &s_n(x) \equiv a_n \\ &s_r(x) \equiv fl[xs_{r+1}(x) + a_r] \quad (r = n-1, n-2, \ldots, 0). \end{aligned}\right\} \tag{15.1}$$

As in section 4, we see that the computed value $s_0(x)$ is given by

$$s_0(x) \equiv a_n(1+E_n)x^n + a_{n-1}(1+E_{n-1})x^{n-1} + \ldots + a_0(1+E_0) \qquad (15.2)$$

where

$$(1-2^{-t})^{2r+2} \leqslant 1 + E_r \leqslant (1+2^{-t})^{2r+2} \quad (r<n)$$

$$(1-2^{-t})^{2n+2} \leqslant (1-2^{-t})^{2n+1} \leqslant 1+E_n \leqslant (1+2^{-t})^{2n+1} < (1+2^{-t})^{2n+2}. \qquad (15.3)$$

Each computed value of $s_0(x)$ then, is the exact value, for that value of x, of a polynomial with coefficients $a_r(1+E_r)$ for some set of E_r satisfying equations (15.3). Now if $z_i(E)$ denotes the set of zeros of the perturbed polynomial for the appropriate value of $E_1, E_2, \ldots, E_n$ then, corresponding to all the possible sets of values of E_r, each $z_i(E)$ will lie in some closed domain containing z_i since $E_r = 0$ $(r = 0, \ldots, n)$ is a permissible set of values. Some real zeros may become complex conjugate pairs for some permissible perturbations, but we assume that $z_k(E)$ remains an isolated real zero. (This now qualifies our requirement that z_k should not be too ill-conditioned for the precision of computation.) The closed domain containing z_k is therefore a section of the real axis including z_k as an interior point.

This situation is illustrated in Fig. 1 where z_5 is the appropriate zero. The zeros z_3, z_4 constitute a double zero which becomes a complex conjugate pair for some perturbations; z_6 and z_7 are a conjugate pair which

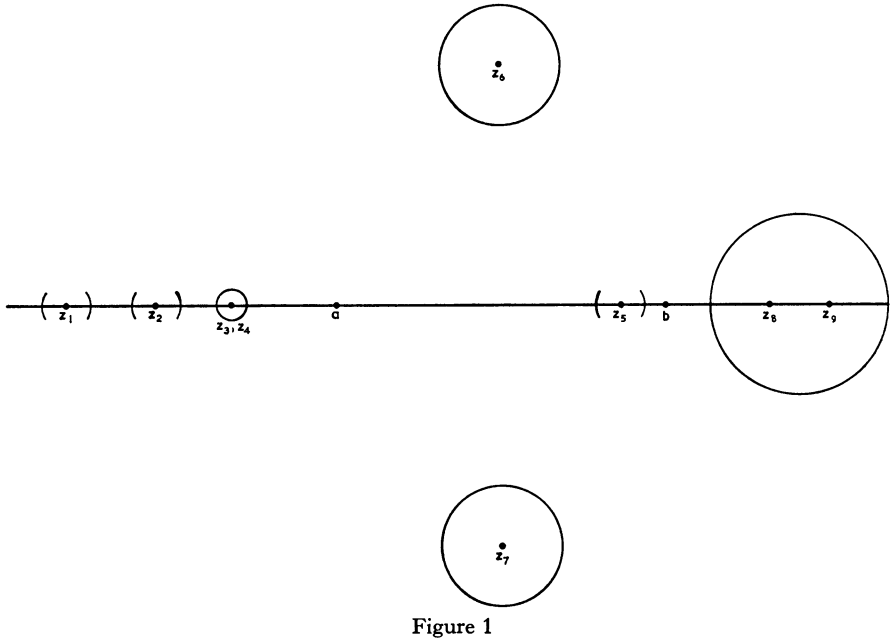

Figure 1

remain a conjugate pair, z_8 and z_9 are two real ill-conditioned zeros which sometimes become a complex pair, while z_1 and z_2 are real zeros which remain real. It is clear that the computed values of $f(a)$ and $f(b)$ will be negative and positive respectively whatever perturbations E_r may correspond to their computation. We denote the width of the z_k interval by ρ_k.

16. In the practical application of the bisection process the correct decision will certainly be made about the sign of $f(x)$, provided x does not lie in the interval surrounding z_k. If a bisection point lies inside this interval an incorrect decision might be made. The history of the practical procedure must therefore be as follows:

Either:

(i) The correct decision is made every time, in which case the final interval really will contain z_k. If we make m steps, the zero is correctly located in an interval of width $2^{-m}(b-a)$.

Or:

(ii) There is at least one incorrect decision. The first incorrect decision must occur when the bisection point is in the z_k interval. If (x_r, y_r) is the interval resulting from this decision, then either one or both of the end points lie in the z_k interval. We now show that at all subsequent steps either one or both of x_i and y_i lie in the z_k interval. The argument is as follows.

Either:

(*a*) x_r and y_r both lie in the z_k interval. In this case all subsequent end points must clearly do so.

Or:

(*b*) Only one of x_r and y_r lies in the z_k interval. We assume, typically, that it is y_r. Let A be the mid-point of (x_r, y_r). If A is outside the z_k interval (Fig. 2), then the next decision will be correct and $(x_{r+1}, y_{r+1}) = (A, y_r)$.

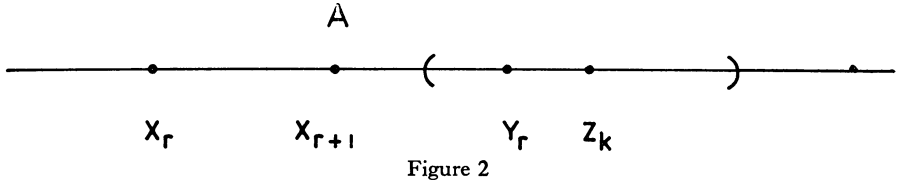

Figure 2

Hence the next interval has one end point in the z_k interval. If A is inside the z_k interval (Fig. 3) then we must have $(x_{r+1}, y_{r+1}) = (x_r, A)$ or (A, y_r).

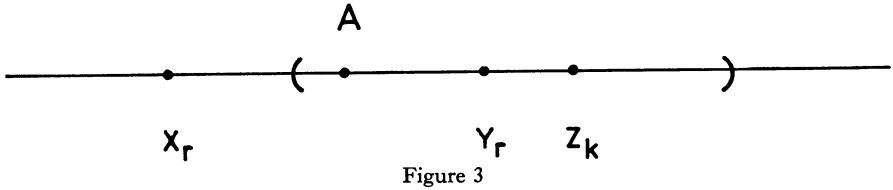

Figure 3

We have now shown that when one of the points x_r, y_r lies in the z_k interval then one or both of x_{r+1}, y_{r+1} lies in the z_k interval and the required result is therefore established.

17. In all cases the mid-point of (x_m, y_m) cannot be further from z_k than $\rho_k + 2^{-m-1}(b-a)$. Clearly the number of effective stages of bisection is limited by the number of digits we are using in the computation; we can

continue until $2^{-m}(b-a)$ is of the order of $2^{-t}z_k$. The fundamental limitation is therefore the size of ρ_k, and an extreme upper bound for this is the maximum perturbation in z_k caused by perturbations of up to $(2r+2)a_r2^{-t}$ in a_r [c.f. (15.3)].

We consider now what this means for a polynomial of degree 127 (say), for which we have $2r+2 \leqslant 256 = 2^8$. Perhaps the most instructive way of interpreting the result is that, if it was originally necessary to round the coefficients in order to represent them to t digits, *then if the subsequent computation were performed using $t+8$ digits, the error resulting from rounding errors in the computation would not be greater than that resulting from the initial rounding of the coefficients.* In fact the perturbations $(2r+2)a_r2^{-t}$ represent extreme upper bounds and $(2r+2)^{\frac{1}{2}}\,a_r2^{-t}$ would perhaps be more realistic. Hence we would expect 4 binary *guarding* figures to be adequate. We shall show later that even this is often a considerable over-estimate (section 23).

Even for a polynomial of comparatively high degree the effect of the 'accumulation' of rounding errors is quite slight. Our results show that the *sensitivity* of the zero to individual rounding errors is the more important consideration.

ITERATIVE METHODS

18. It may well be felt that the success of bisection is related to the fact that decisions depend only on the sign of the computed function, but this is not so. The limiting accuracy attainable with computation of a given precision is much the same for several iterative methods in which the evaluation of the polynomial for a sequence of approximations to a zero plays a dominant role. As typical of such methods we may consider *Newton's process.*

We assume the reader is already familiar with this process and give a brief description merely in order to introduce our notation. From an initial approximation z_k+h_0 to the zero z_k, a sequence of further approximations z_k+h_r is determined by means of the relations

$$\begin{aligned} z_k + h_{r+1} &= (z_k+h_r) - f(z_k+h_r)/f'(z_k+h_r) \\ &= z_k + [\tfrac{1}{2}h_r^2 f''(z_k) + \ldots]/[f'(z_k) + \ldots] \end{aligned} \qquad (18.1)$$

where the omitted terms in the numerator and denominator involve higher powers of h_r. If z_k is an isolated zero, so that $f'(z_k) \neq 0$, then we have

$$h_{r+1} \sim \frac{f''(z_k)}{2f'(z_k)}\,h_r^2 \qquad (h_r \to 0). \qquad (18.2)$$

Notice that the coefficient of h_r^2 may be quite large if z_k is an ill-conditioned zero and h_r must then be very small before the relation (18.2) comes into evidence. However, for exact computation the process must ultimately converge very rapidly.

19. We now consider the effect of rounding errors on the determination of a zero z_k. As before we must assume that z_k is not too ill-conditioned for the precision of the computation. We assume further that we are starting from an initial value for which the exact Newton process does indeed converge to z_k; we shall not be concerned therefore with the problem of convergence in the large. Our final assumption is that, for all values of z with which we shall be concerned, the relative error in the computed $f'(z)$ is much smaller than the relative error in $f(z)$. For values of z in the neighbourhood of a zero this is a reasonable assumption.

We shall now use $z_k+\bar{h}_r$ to denote the successive computed approximations and z_k+h_{r+1} to denote the approximation which would be obtained by performing the Newton process exactly with $z_k+\bar{h}_r$. We denote the computed values of $f(z_k+\bar{h}_r)$ and $f'(z_k+\bar{h}_r)$ by $\bar{f}_r$ and $\bar{f}'_r$, and the true value of $f(z_r+\bar{h}_r)$ and $f'(z_k+\bar{h}_r)$ by f_r and f'_r. From (18.2) we know that for sufficiently small $\bar{h}_r$ we have

$$|h_{r+1}| < A\bar{h}_r^2 \tag{19.1}$$

and we assume that $\bar{h}_r$ is in the range

$$A|\bar{h}_r| < 0{\cdot}1 \text{ (say)}. \tag{19.2}$$

It is evident that the exact Newton process converges very rapidly.

Consider now the computation of $f(z_k+\bar{h}_r)$. We have seen (section 15) that the computed value is the exact value of a polynomial with coefficients $a_i(1+E_i)$ and if $z_i(E)$ are the zeros of this polynomial

$$\bar{f}_r \equiv a_n(1+E_n)\prod_{i=1}^{n}[z_k+\bar{h}_r-z_i(E)] \tag{19.3}$$

$$f_r \equiv a_n\prod_{i=1}^{n}[z_k+\bar{h}_r-z_i] \tag{19.4}$$

giving

$$\bar{f}_r/f_r = (1+E_n)\frac{\bar{h}_r+z_k-z_k(E)}{\bar{h}_r}\prod_{i\neq k}\left(\frac{z_k+\bar{h}_r-z_i(E)}{z_k+\bar{h}_r-z_i}\right). \tag{19.5}$$

We are interested in the region for which the computed $\bar{h}_{r+1}$ is reasonably close to the true h_{r+1}. Provided z_k is sufficiently isolated from all the other zeros the final product in (19.5) is close to unity for all $\bar{h}_r$ in which we are interested. Hence $\bar{f}_r/f_r$ is close to unity as long as the relevant $\bar{h}_r$ are such that $|\bar{h}_r|$ is appreciably greater than $|z_k-z_k(E)|$. We have already assumed that $\bar{f}'_r$ is a very good approximation to f'_r throughout the relevant region and hence we may write

$$\frac{\bar{f}_r}{\bar{f}'_r} = (1+B_r)\frac{f_r}{f'_r} \tag{19.6}$$

knowing that B_r is largely determined by $\bar{f}_r/f_r$ and so will be small until $|h_r|$ approaches $|z_k - z_k(E)|$ in magnitude. We have

$$z_k + \bar{h}_{r+1} \equiv z_k + \bar{h}_r - \frac{\bar{f}_r}{\bar{f}'_r}$$

$$= z_k + \bar{h}_r - (1+B_r)\frac{f_r}{f'_r} \tag{19.7}$$

$$\bar{h}_{r+1} = \bar{h}_r - \frac{f_r}{f'_r} - B_r\frac{f_r}{f'_r}$$

$$= h_{r+1} - B_r(\bar{h}_r - h_{r+1}), \tag{19.8}$$

and hence

$$\begin{aligned} |\bar{h}_{r+1}| &\leqslant (1 + |B_r|)h_{r+1} + |B_r|\,|\bar{h}_r| \\ &< (1 + |B_r|)A\bar{h}_r^2 + |B_r|\,|\bar{h}_r| \\ &< [(0\cdot1)(1 + |B_r|) + |B_r|]\,|\bar{h}_r|. \end{aligned} \tag{19.9}$$

It is obvious that the rounding errors may ultimately destroy quadratic convergence, but as long as $|B_r| < 0\cdot1$ (say), we have

$$|\bar{h}_{r+1}| < 0\cdot21\,|\bar{h}_r| \tag{19.10}$$

so that convergence continues to be quite satisfactory. In fact the error certainly continues to diminish until $|B_r|$ reaches $0\cdot8$.

It is evident from the argument used in deriving (19.6) that *convergence ceases in practice when the error made in computing f_r is of a comparable order of magnitude to its true value, so that the computed value has no correct significant figure.* Now the true value of $f(z_k+\bar{h}_r)$ is approximately $f'(z_k)\bar{h}_r$ while the error made in evaluating the function is approximately $\sum_{i=0}^{n} a_iE_iz_k^i$, from (15.2). Hence we must have for the limiting accuracy

$$\sum_{i=0}^{n} a_iE_iz_k^i/f'(z_k)\bar{h}_r = O(1). \tag{19.11}$$

This shows that $z_k + \bar{h}_r$ must lie in an interval of much the same width as ρ_k in section 15. We would expect the limiting accuracy in practice to be appreciably better than that obtained by taking the upper bound of the numerator of the expression in (19.11) because of the statistical distribution of the errors.

SIMPLE EXAMPLES

20. After reaching the limiting accuracy with $z_k + \bar{h}_r$ we will obtain a sequence of approximations, $z_k + \bar{h}_m$ all of much the same accuracy. The approximations will in fact be representable by $z_k + C\bar{h}_r$ where $|C|$ is of the order of unity. For ill-conditioned polynomials of low degree it is quite common for the *computed* value of the polynomial to be zero throughout a whole interval containing the zero. For example, the polynomial

$$z^2 - 2\cdot028\ 8888z + 1\cdot028\ 7690 \tag{20.1}$$

has the zeros $1\cdot032\,5673\ldots$ and $0\cdot9963\,2146\ldots$. Using eight-decimal floating-point computation, the polynomial has the computed value zero for any z in the range

$$1\cdot032\ 5660 \leqslant z \leqslant 1\cdot032\ 5687 \tag{20.2}$$

while the polynomial (z^2-2z+1) takes the computed value zero for all z in the range

$$0\cdot9999\ 2930 \leqslant z \leqslant 1\cdot0000\ 707. \tag{20.3}$$

The magnitudes of these intervals are what one would have expected from considering the zeros of $\sum a_i(1+E_i)z^i$.

Note that in these two cases the error in each computed value of the polynomial is precisely equal and opposite to its true value for any value of z in the appropriate interval, thus providing a striking illustration of the conclusion reached in the last section.

For large-order polynomials it is uncommon for the computed value to be zero. The behaviour of successive iterates is well illustrated by the convergent to the zero $z = 12$ obtained by working in binary floating-point arithmetic with $t = 46$, or about 14 decimals, with the polynomial $(z-1)(z-2)\ldots(z-12)$. After reaching the limiting accuracy the errors in successive convergents were

$$10^{-8}(-4\cdot46\ldots);\quad 10^{-8}(+5\cdot95\ldots);\quad 10^{-8}(+3\cdot35\ldots);$$
$$10^{-8}(-1\cdot12\ldots);\quad 10^{-8}(+1\cdot86\ldots);\quad \ldots\ .$$

It is perhaps worth remarking that if we start with an approximation to a zero which is appreciably more accurate than the limiting accuracy which we have just described, a single iteration will usually spoil this very good approximation and produce one with an error which is typical of the limiting accuracy. This will not be true, however, if the polynomial has integer coefficients and a zero which requires very few digits for its representation. If this exact zero is used as an approximation, then usually no rounding errors will be made in the evaluation of the polynomial, that is, we are effectively working to an infinite number of figures, and hence subsequent iterates remain constant at that exact value. Comments which we make from time to time will usually not be true of such special examples, and we will not refer to this phenomenon again.

POLYNOMIAL DEFLATION

21. The analysis of the previous section is appropriate on the assumption that we are given approximations to each of the zeros z in turn which are sufficiently accurate to ensure that Newton's process really does converge to the appropriate zero and not wander away to some other zero.

Iterative methods are commonly used in a rather different way on fast automatic computers. Iteration is started with a value which is more or less arbitrary and, after a few iterations in which the approximations 'hunt', a value sufficiently near to one of the zeros is reached and the process

then 'homes' on to that zero*. When an approximate zero, α, has been determined in this way, the polynomial is divided by $(z-\alpha)$ and iteration is continued with the quotient polynomial. Proceeding in this way, all zeros are ultimately determined and the danger of converging twice to the same zero is avoided.

However, there is a danger that the zeros of the quotient polynomials may gradually diverge from those of the original polynomial, so that the zeros found at a late stage in the process may be very inaccurate. At first sight the danger appears to be quite considerable. Suppose we have, for example, a cubic polynomial with zeros 1, 1 and 2. Working in t-digit arithmetic, convergence to a zero, $z = 1$, will stop when $z-1$ is of order $2^{-t/2}$. If we accept such an approximation, then all coefficients of the deflated polynomial will differ from the true deflated polynomial, $(z-1)(z-2)$, in the last $\frac{t}{2}$ figures. Now $z = 2$ is a well-conditioned zero. Is there not a danger that the zero of the computed deflated polynomial will differ from 2 in the $\frac{t}{2}$th significant figure?

ANALYSIS OF ERRORS INHERENT IN DEFLATION

22. Before attempting to assess the effect of the practical deflation process, it is important to decide what is the best we could reasonably expect from it. We have already seen that when we iterate in the original polynomial, then for a given precision of computation the accuracy of determination of a zero is limited by its condition. The *accumulation* of rounding errors plays little part in limiting the accuracy. *An ill-conditioned zero will be determined inaccurately even when we iterate with the original polynomial.*

It would be unreasonable to expect deflation to improve this situation and we would say that deflation had made no contribution to the error if each computed zero were as accurate as that which could be obtained by iterating for that zero in the original polynomial, *however inaccurate this might be*. We would describe deflation as very stable if the computed zeros were only slightly less accurate than this. With this in mind we now consider the errors resulting from one stage of deflation.

Suppose Newton's process has reached the limiting accuracy and α has been accepted as a zero. We assume that it represents an approximation to the zero z_1. To obtain the deflated polynomial we must divide $f(z)$ by $z-\alpha$. Usually the process is performed in floating-point arithmetic, and if we denote the coefficient of z^r in the deflated polynomial by s_{r+1} we have

$$\left.\begin{aligned} s_n &\equiv a_n \\ s_r &\equiv fl(\alpha s_{r+1}+a_r) \quad (r = n-1, n-2, \ldots, 0). \end{aligned}\right\} \tag{22.1}$$

* We shall not concern ourselves here with cases in which the approximations oscillate indefinitely.

The justification for calling the coefficients s_r is immediately apparent if we compare equations (22.1) with equations (15.1). The coefficients s_r are precisely the partial sums which are computed when evaluating $f(\alpha)$.

In section 15 we expressed $s_r(\alpha)$ as the exact partial sum corresponding to a polynomial with perturbed coefficients. However these perturbed coefficients were different for each s_r. There we were interested only in s_0 so that this was not a disadvantage. Now, however, it is more convenient to write

$$s_r \equiv fl(\alpha s_{r+1} + a_r) \equiv \alpha s_{r+1} + a'_r \quad (r = n-1, n-2, \ldots, 0) \tag{22.2}$$

this equation being the definition of a'_r.

Before estimating $a'_r - a_r$ we attempt to relate the exact zeros of the *computed deflated polynomial* $s(z)$ defined by

$$s(z) = s_n z^{n-1} + s_{n-1} z^{n-2} + \ldots + s_1 \tag{22.3}$$

with those of $a'_n z^n + a'_{n-1} z^{n-1} + \ldots + a'_0$. We have

$$\begin{aligned}(z-\alpha)s(z) &\equiv s_n z^n + (s_{n-1} - \alpha s_n) z^{n-1} + (s_{n-2} - \alpha s_{n-1}) z^{n-2} \\ &\quad + \ldots + (s_1 - \alpha s_2) z - \alpha s_1 \\ &= a_n z^n + a'_{n-1} z^{n-1} + a'_{n-2} z^{n-2} + \ldots + a'_1 z - \alpha s_1 . \end{aligned} \tag{22.4}$$

The exact zeros of the computed polynomial $s(z)$ are therefore the exact zeros of the polynomial $\bar{f}(z)$ on the right of (22.4) except for $z = \alpha$. The coefficients of this polynomial are all in a satisfactory form except for the constant term. We obtain a more appropriate expression for this as follows. We have

$$s_0 = \alpha s_1 + a'_0 \tag{22.5}$$

from (22.2) with $r = 0$. Hence

$$-\alpha s_1 = a'_0 - s_0 . \tag{22.6}$$

Now s_0 is the computed value of $f(\alpha)$ and if α were an exact zero and the computation were performed exactly, then s_0 would be zero. The form $a'_0 - s_0$ is therefore an appropriate one for the constant coefficient. Now we know that for limiting accuracy, the error in the computed value of $f(\alpha)$ is of the same order of magnitude as its true value. *Hence the computed value s_0 is of the same order of magnitude as the error*, and this can be expressed in the form

$$a_n E_n \alpha^n + a_{n-1} E_{n-1} \alpha^{n-1} + \ldots + a_0 E_0 \tag{22.7}$$

where the E_i lie in the usual ranges. We may therefore write

$$\bar{f}(z) = a_n z^n + a'_{n-1} z^{n-1} + \ldots + a'_1 z + a'_0 - s_0 \tag{22.8}$$

$$|s_0| < K_1 |a_n E_n \alpha^n + a_{n-1} E_{n-1} \alpha^{n-1} + \ldots + a_0 E_0| \tag{22.9}$$

where K_1 is of order unity. We are interested in the zeros $z'_2, z'_3, \ldots, z'_n$ of $\bar{f}(z)$ which correspond to the zeros $z_2, z_3, \ldots, z_n$ of $f(z)$. In particular we wish to compare the errors $z_i - z'_i$ which can be attributed to the deflation

process with the errors which would have arisen in these zeros if we had iterated in the original polynomial. If we denote by z_i'' the zeros which would be obtained by iterating in the original polynomial we have

$$|z_i'' - z_i| \leqslant K_2 |a_n E_n' z_i^n + a_{n-1} E_{n-1}' z_i^{n-1} + \ldots + a_0 E_0'| / |f'(z_i)| \quad (22.10)$$

where K_2 is of order unity, for some set of E_i' satisfying the usual relations. The bounds of E_i and E_i' are the same.

23. Now we may separate the error in z_i' into two parts.

(i) *That coming from the error* $-s_0$ *in the constant term.* This gives rise to an error, the order of magnitude of which is

$$K |a_n E_n \alpha^n + a_{n-1} E_{n-1} \alpha^{n-1} + \ldots + a_0 E_0| / |f'(z_i)| . \quad (23.1)$$

Clearly if $|\alpha| < |z_i|$ then the probable value of this in the statistical sense is less than the probable value of the right-hand side of (22.10). If $|\alpha| \ll |z_i|$ we might well expect (23.1) to be much the smaller.

On the other hand if $|\alpha| \gg |z_i|$ we have a very dangerous situation. If for example $|\alpha| = 1$ and $|z_i| = 10^{-3}$ then the element $a_r E_r \alpha^r$ may be larger than $a_r E_r' z_i^r$ by the factor 10^{3r}. Even for a polynomial of quite moderate order it appears that the error introduced by deflation may then be of a much greater order of magnitude than that inherent in iterating for z_i in the original polynomial.

(ii) *That arising from the presence of the coefficients* a_r' *instead of* a_r. The difference $a_r' - a_r$ is the error made in computing $fl(\alpha s_{r+1} + a_r)$ from the computed s_{r+1}. For arbitrary α, s_{r+1} and a_r we have

$$fl(\alpha s_{r+1} + a_r) = [\alpha s_{r+1}(1+\epsilon_1) + a_r](1+\epsilon_2) . \quad (23.2)$$

Hence

$$fl(\alpha s_{r+1} + a_r) - (\alpha s_{r+1} + a_r) = \alpha s_{r+1}[(1+\epsilon_1)(1+\epsilon_2) - 1] + a_r \epsilon_2$$

$$|fl(\alpha s_{r+1} + a_r) - (\alpha s_{r+1} + a_r)| \leqslant (2|\alpha s_{r+1}| + |a_r|) 2^{-t} . \quad (23.3)$$

If

$$|\alpha s_{r+1}| < |a_r| \quad (23.4)$$

this gives

$$|fl(\alpha s_{r+1} + a_r) - (\alpha s_{r+1} + a_r)| < 3|a_r| 2^{-t} \quad (23.5)$$

or

$$fl(\alpha s_{r+1} + a_r) \equiv \alpha s_{r+1} + a_r(1+E_r) \equiv \alpha s_{r+1} + a_r'$$

$$|E_r| < 3(2^{-t}) .$$

Hence when the relation (23.4) holds for all r, the errors arising from the presence of the coefficients a_r' instead of a_r are bounded by those arising from perturbations of up to 3 parts in 2^t in the original coefficients. (Note that this analysis applies also to perturbations corresponding to iteration in the original polynomial with values of z in the neighbourhood of α, and justifies our remark in section 17 that the effective perturbations are often much smaller than the bounds $a_r E_r$ we have given hitherto.)

Now we cannot ensure that the relation (23.4) will be satisfied at all stages, particularly as some of the coefficients a_r might well be zero. Clearly, however, if α is an approximation to z_1, the zero of smallest modulus, then this will tend to keep $a'_r - a_r$ to its lowest level, in that it will make it least likely that $|\alpha s_{r+1}|$ is greater than $|a_r|$. A few simple examples should make this point clear.

EXAMPLES OF DEFLATION

24. Consider first the polynomial

$$x^4 - 6{\cdot}7980x^3 + 2{\cdot}9948x^2 - 0{\cdot}04368\,6x + 0{\cdot}00008\,9248$$

having the zeros

$$0{\cdot}00245\,32\ldots;\ 0{\cdot}01257\,6\ldots;\ 0{\cdot}45732\ldots;\ 6{\cdot}3256\ldots. \qquad (24.1)$$

This has very well-conditioned zeros with respect to small relative changes in the coefficients but not with respect to small absolute changes.

Suppose we accept the zero $0{\cdot}00245\,32$ and deflate, using 5-decimal floating-point arithmetic. The computation involved in the deflation may be written in the form

$$\begin{array}{llll} 1 - 6{\cdot}7980 & + 2{\cdot}9948 & - 0{\cdot}04368\,6 & + 0{\cdot}00008\,9248 \\ \quad - 0{\cdot}00245\,32 & + 0{\cdot}01667\,07206 & - 0{\cdot}00730\,58749\,2 & + 0{\cdot}00008\,9247416 \\ \hline 1 - 6{\cdot}7955 & + 2{\cdot}9781 & - 0{\cdot}03638\,0 & \end{array} \qquad (24.2)$$

using the usual method of detached coefficients. We have entered the exact values of αs_{r+1} in row 2, though we have used only the values rounded to five significant decimals. This makes it obvious that the computation has been performed exactly with the perturbed polynomial

$$x^4 - 6{\cdot}79795\,32x^3 + 2{\cdot}99477\,07206x^2 - 0{\cdot}04368\,58749\,2x + 0{\cdot}00008\,92474\,16. \qquad (24.3)$$

Further, this polynomial leaves the remainder zero exactly when divided by $x - 0{\cdot}00245\,32$ and gives exactly the computed quotient polynomial. The perturbed polynomial (24.3) differs from the original only in the 5th significant figure.

Note that each element of αs_{r+1} in row 2 is much smaller than the corresponding element of a_r in row 1 and this has ensured that the rounding error made in computing $fl(\alpha s_{r+1} + a_r)$ is bounded by 1 in the last significant figure of a_r. Moreover the last term in row 2 is very close to the constant term in row 1. It should be appreciated that the perturbed polynomial (24.3) is exactly equal to the computed quotient polynomial (24.2) multiplied by $x - 0{\cdot}00245\,32$. The zeros of the quotient polynomial are $0{\cdot}01257\,6$, $0{\cdot}45732$ and $6{\cdot}3256$ to five figures, and are therefore quite unharmed by the deflation.

If the zero 6·3256 is accepted first, the deflation proceeds as follows:

1	− 6·7980	+ 2·9948	− 0·04368 6	+ 0·00008 9248
(unrounded)	− 6·3256	+ 2·98821 344	− 0·04174 896	+ 0·01225 26872
(rounded)	− 6·3256	+ 2·9882	− 0·04174 9	
1	− 0·4724	+ 0·00660 00	− 0·00193 7	(24.4)

Again it is obvious that the computation has been performed exactly with the polynomial

$$x^4 - 6{\cdot}7980x^3 + 2{\cdot}99481\,344x^2 - 0{\cdot}04368\,596x + 0{\cdot}01225\,26872 \quad (24.5)$$

and that this polynomial gives exactly the computed quotient polynomial (24.4).

Note now that the perturbed polynomial disagrees completely with the original in its constant term. This error rises from the fact that computed s_0 differs from zero by a quantity which is much larger than the constant term, in spite of the fact that the accepted zero is correct to 5 figures. It is obvious that at least one of the zeros of (24.5) and hence of the computed deflated polynomial (24.4) must differ substantially from the corresponding zero of the original polynomial.

25. A less trivial computation illustrating the same point was performed on DEUCE. The zeros of $(z-2^{-1})(z-2^{-2})\ldots(z-2^{-20}) - 2^{-31}z^{19}$ were found by iteration and deflation, two distinct computations being performed. The first time the zeros were extracted in increasing order and all were found to be accurate, deflation making errors which did not even effect the last figure of each zero. In the second computation the largest zero was found first, and after each deflation the zero which had just been accepted was used as the first approximation to a zero of the deflated polynomial. The errors resulting from the deflation were now so great that sixteen of the twenty zeros bore no relation to the true zeros. The computed zeros are given in Table 3.

TABLE 3

Zeros of $(x-2^{-1})(x-2^{-2})\ldots(x-2^{-20}) - 2^{-31}x^{19}$ found by iteration and deflation using 18 decimals. The largest zero was found first.

+ 0·50000 0002	+ 0·05766 8540 ± 0·01565 2764i
+ 0·24999 9998	− 0·02420 5717 ± 0·03723 1856i
+ 0·12499 9919	+ 0·00961 0341 ± 0·04736 8347i
+ 0·06614 1329	− 0·04193 0285 ± 0·00849 0280i
	− 0·03579 2108 ± 0·02440 4253i
	− 0·00847 7427 ± 0·04528 1360i
	+ 0·02799 3309 ± 0·04293 8042i
	+ 0·04456 2246 ± 0·03210 7627i

Our analysis enables us to predict both these results with complete confidence. We consider, for simplicity, the polynomial without the perturbation $2^{-31}x^{19}$. This perturbation is, in any case, irrelevant to our present argument. If an initial guess is chosen such that approximations converge towards the zero 2^{-20}, then because this is well-conditioned, the limiting accuracy is very high and we will converge to a value $2^{-20}(1+\epsilon)$ where ϵ is of order 2^{-t}. The first few steps in the deflation process are shown in (25.1); we have given only the relevant power of two, since this is adequate to establish our point. It is evident that the elements in row 2 are all of a smaller order of magnitude than those in row 1. In other words $|\alpha s_{r+1}| \ll |a_r|$, so that the error made in computing s_r is bounded by one in the last significant figure in a_r.

$$\begin{array}{cccccccccc}
2^0 & -2^0 & 2^{-1} & -2^{-4} & 2^{-8} & -2^{-13} & 2^{-19} & -2^{-26} & 2^{-34} & -2^{-43}\ldots \\
 & -2^{-20} & 2^{-20} & -2^{-21} & 2^{-24} & -2^{-28} & 2^{-33} & -2^{-39} & 2^{-46} & -2^{-54}\ldots \\
\hline
2^0 & -2^{-0} & 2^{-1} & -2^{-4} & 2^{-8} & -2^{-13} & 2^{-19} & -2^{-26} & 2^{-34} & -2^{-43}\ldots
\end{array} \qquad (25.1)$$

We have seen from (22.6) that the effective perturbation in a_0 is augmented by s_0. This is the computed value of the polynomial corresponding to the accepted zero. Because α is so small, s_0 is far smaller than a_0, as may be seen from (22.7) and the orders of magnitude of the a_r. Hence the perturbations corresponding to deflation with the accepted zero $2^{-20}(1+\epsilon)$ are all bounded by one in the last significant figure of each a_r and we know that all zeros are little affected by such perturbations.

When an approximation to the zero 2^{-1} is accepted first, the situation is very different. Suppose the accepted zero is $2^{-1}+2^{-40}$, a very good approximation indeed. We now show that even if no rounding errors were made in the deflation process, the quotient polynomial $q(x)$ would have some zeros which were entirely different from those of the original polynomial $f(x)$.

We may write

$$f(x) = (x-2^{-1}-2^{-40})q(x) + r \qquad (25.2)$$

where $q(x)$ is the polynomial corresponding to exact division by $(x-2^{-1}-2^{-40})$. Hence we have

$$\begin{aligned} r &= f(2^{-1}+2^{-40}) \\ &> 2^{-40}(2^{-1}-2^{-2})(2^{-1}-2^{-3})\ldots(2^{-1}-2^{-20}) \\ &> 2^{-61} \end{aligned} \qquad (25.3)$$

(using quite a crude approximation). The zeros of $q(x)$ are therefore those of $f(x)-r$. Now the constant term in $f(x)$ is $+2^{-210}$ while the constant term in $f(x)-r$ is less than $2^{-210}-2^{-61}$. Since the constant term is the product of the zeros, $f(x)-r$ cannot have zeros of the form $2^{-k}(1+\epsilon_k)$ with small ϵ_k.

It will readily be verified that in the deflation process αs_{r+1} is of a much higher order of magnitude than a_r for most values of r. Hence the error made in computing $fl(a_r+\alpha s_{r+1})$ is of the order of $2^{-t}\alpha s_{r+1}$ and is much greater than $2^{-t}a_r$.

DEFLATION OF ILL-CONDITIONED POLYNOMIALS

26. The two previous examples show that it is vital to accept small zeros first in a polynomial whose zeros are widely dispersed. This is true even when, as in our examples, all the zeros are well-conditioned. We now consider a simple polynomial $f(x)$ given by

$$f(x) = x^3 - 4x^2 + 5x - 2 = (x-1)^2(x-2) \qquad (26.1)$$

having a double zero. This zero is of course ill-conditioned while the zero at $x = 2$ is very well-conditioned. Suppose we start with an approximation which gives convergence towards the double zero. Working with ten-decimal floating-point arithmetic the limiting accuracy will be attained when there is still an error of order 10^{-5}. We might, for example, accept the value $1\cdot00000\ 6723$. The computed quotient $s(x)$ is given by

$$s(x) = x^2 - 2\cdot99999\ 3277x + 1\cdot99998\ 6554 \qquad (26.2)$$

and $\bar{f}(x)$, the exact product $(x-1\cdot00000\ 6723)s(x)$, is

$$x^3 - 4\cdot00000\ 0000x^2 + 4\cdot99999\ 99999\ 54801\ 271x - 1\cdot99999\ 99999\ 09602\ 542\,. \qquad (26.3)$$

The coefficients of $\bar{f}(x)$ are perturbed very little from those of $f(x)$ in spite of the low accuracy of the accepted zero. The zeros of $s(x)$ are those of $\bar{f}(x)$ other than $1\cdot00000\ 6723$. Since the zero at $x = 2$ is well-conditioned, $\bar{f}(x)$,and hence $s(x)$, must have a zero which is very close to 2. In fact the errors introduced by the deflation, expressed as perturbations in the a_r, are certainly no larger than those which correspond to iteration in the original polynomial.

The other zero of $s(x)$ corresponds to the double zero. The computed zeros obtained by deflation are $1\cdot00000\ 6723$, $0\cdot99999\ 3277$ and $2\cdot00000\ 0000$ to ten figures. The well-conditioned zero is found accurately in spite of the poor value accepted for the first zero. The two values given for the double zero have equal and opposite errors so that their sum is correct to working accuracy.

Not only was the well-conditioned zero unharmed but the accuracy obtained in the second of the pair corresponding to the double zero, is no poorer than that obtained by iteration in the original polynomial. Again the result is forecast by the error analysis. We have seen in section 23 that the size of $a_r'-a_r$ is decided only by the error made in computing $fl(\alpha s_{r+1}+a_r)$ from computed s_{r+1}, and does not depend upon the accuracy of α. The remaining perturbation is that of $-s_0$ in the constant term. Now s_0 is the computed value of $f(\alpha)$ and is equal to its true value plus the error made in computing it. The latter corresponds to perturbations in each a_r

which are certainly bounded by $a_r E_r \alpha^r$, while the former is $a_n \Pi(\alpha - x_i)$. The limiting accuracy is reached when these are of the same order of magnitude.

Notice that when α is an approximation to a double zero, two factors in $a_n \Pi(\alpha - x_i)$ are small, so that its value is much smaller than might be expected having regard to the accuracy of α. Similar remarks apply to zeros of higher multiplicity.

This example illustrates the point that the best order in which to find the zeros is almost independent of their relative conditions. Provided we find the smaller zeros first, each zero is determined with an accuracy which is dependent primarily on its condition, and not on the accuracy of the zeros which precede it in the deflation process. In the last examples there was no great disparity in the size of the zeros and little harm would have resulted from finding the largest zero first. However one should not press this too far. For the polynomial $(x-1)^3(x-2)^3(x-3)^3(x-4)$ for example, it is important not to find the zero at $x = 4$ first although the ratio of the greatest to the smallest zero is quite modest.

27. As a more impressive example illustrating the same point we give the results obtained on DEUCE with a polynomial of degree 16 having zeros of widely differing conditions. The polynomial is

$$\begin{aligned} &12501\ 62561x^{16} + 3854\ 55882x^{15} + 8459\ 47696x^{14} + 2407\ 75148x^{13} \\ &+ 2479\ 26664x^{12} + 642\ 49356x^{11} + 410\ 18752x^{10} + 94\ 90840x^9 \\ &+ 41\ 78260x^8 + 8\ 37860x^7 + 2\ 67232x^6 + 44184x^5 + 10416x^4 \\ &+ 1288x^3 + 224x^2 + 16x + 2. \end{aligned} \tag{27.1}$$

The zeros of this polynomial obtained using the binary equivalent of 18-decimal floating-point arithmetic are given in Table 4. These zeros were found in the order given, the polynomial being deflated after the computation of each zero. The first incorrect figure in each zero is underlined. It will be seen that the last computed zero is correct to 17 decimals although it was found after deflations using zeros which were correct to only 9 decimals.

TABLE 4

$-$ 0·01869	49953	44575	74	$\pm$	0·25304	56818	77087 20i
$-$ 0·00232	09446	09917	65	$\pm$	0·29258	37451	03485 46i
$-$ 0·00049	14536	35956	63	$\pm$	0·30418	23930	23125 81i
$-$ 0·00014	26406	86229	90	$\pm$	0·30861	21214	90245 78i
$-$ 0·00004	71327	17211	44	$\pm$	0·31066	18480	81534 84i
$-$ 0·00001	48358	66297	06	$\pm$	0·31169	63042	80926 06i
$-$ 0·00000	30543	16902	26	$\pm$	0·31219	69686	05743 56i
$-$ 0·13244	72469	90246	19	$\pm$	0·13600	55079	51377 64i

In this example the smallest zero was found last, but since the zeros are all of much the same order of magnitude this was not harmful.

28. As far as iteration in the original polynomial is concerned our main result is that the limiting accuracy attainable with computation of a given precision does not fall far short of the best we could reasonably expect. In a later section we discuss another common method, but experience suggests that it is highly unlikely that any other method will attain a much greater accuracy working to the same precision.

When the deflation process is included, the situation is not quite so satisfactory. If the zeros have a wide dispersion the acceptance of an approximation to one of the larger zeros before finding the smaller may be quite fatal to the accuracy of the latter. A method of choosing each initial approximation so as to guarantee convergence to the smallest zero of the current polynomial would be of great value.

To complete our arguments we should consider the effect of several successive deflations. We can do this by examining the appropriate number of stages of backward analysis, but this becomes cumbersome. Usually if the polynomial is fairly ill-conditioned to start with, the condition of the successive deflated polynomials shows a steady improvement. This is true, for example, of the polynomial having the zeros 1, 2, 3, . . . , 20. If the condition of a zero is not deteriorating, the effect of the rounding errors in the successive stages of deflation cannot be worse than additive provided zeros are found in increasing order.

29. It *is* possible, however, for the condition to deteriorate somewhat. This is true, for example, of the polynomial $z^{20} - 1$ if the zeros $e^{ir\pi/10}$ are found in the order corresponding to $r = r_0, r_0 + 1, \ldots$. The initial condition of all the zeros is extremely satisfactory as we saw in section 11. Suppose the zeros corresponding to $r = 1, 2, \ldots, 9$ have been found. The remaining eleven zeros are those in the lower half of the complex plane and the central zero of this group is at $z = -i$. Now the perturbation δ, of this zero with respect to a change δa_r in the rth coefficient of the deflated polynomial is given approximately by

$$|\delta| = |\delta a_r| \, |(-i)^r| \Big/ \left(2\sin\frac{\pi}{20}\right)^2 \left(2\sin\frac{2\pi}{20}\right)^2 \ldots \left(2\sin\frac{5\pi}{20}\right)^2, \quad (29.1)$$

where the factors in the denominator are the distances of this central zero from the other zeros. In the original polynomial the perturbation of the same zero was given by

$$|\delta| = |\delta a_r| \, |(-i)^r| \Big/ \left(2\sin\frac{\pi}{20}\right)^2 \left(2\sin\frac{2\pi}{20}\right)^2 \ldots \left(2\sin\frac{9\pi}{20}\right)^2 \left(2\sin\frac{10\pi}{20}\right). \quad (29.2)$$

The factors in the denominator of (29.2) which are absent in the denominator of (29.1) all have values between $2\sin\frac{\pi}{4}$ and $2\sin\frac{\pi}{2}$. Hence the

factor associated with $|\delta a_r|$ in (29.1) is considerably bigger than that in (29.2). Moreover the coefficients of the deflated polynomial are quite appreciably larger than those of the original polynomial, which are 0 or 1. Hence the equivalent perturbations δa_r associated with iteration in the deflated polynomial are greater than those associated with iteration in the original.

The original polynomial is so well-conditioned that this worsening of condition is not very important. Moreover, if the zeros are found more or less at random round the unit circle, this deterioration does not usually occur. Experience leads one to conjecture that a severe worsening of the condition of a polynomial of any reasonable order is most unlikely.

PURIFICATION IN THE ORIGINAL POLYNOMIAL

30. When all the zeros have been obtained by deflation, we can use the computed values as initial approximations for iteration in the original polynomial, thus obtaining the *limiting* accuracy for the precision of computation that is being used. If zeros have been found strictly in increasing order and the polynomial was of such a type that no deterioration of condition occurred during deflation, no substantial improvement is effected by this purification process. Thus in the example of Table 4 the ill-conditioned zeros were improved by only one binary figure. Almost all the inaccuracy in each of the zeros was attributable to its condition and the errors in the deflations made negligible contributions.

Nevertheless purification in the original polynomial is worth while in all cases. There seems to be no simple method of ensuring that Newton's process determines the zeros in increasing order, and purification gives some protection against the acceptance of spurious values resulting from finding the zeros in the wrong order. It also gives some improvement in cases where the accuracy falls appreciably below the limiting accuracy owing to deterioration of the condition.

The objection is sometimes raised that when iteration is performed in the original polynomial the accuracy of some of the initial approximations may be such that the iterates for a particular zero converge to some other zero and the latter is then found more than once. In practice this happens only when the approximation is very poor, and in this case it is better that such an inadequacy should be exposed.

It might be felt that it would be advantageous at each stage to purify the zero by iteration in the original polynomial before deflating. Although such iteration might be performed in order to make certain that the computed zero has not departed too far from the true zero of the original polynomial, *the associated deflation process should not be performed using the purified value.* The first example of section 26 makes this quite clear. When we have accepted a poor approximation to an ill-conditioned zero, the deflated polynomial obtained by dividing out the corresponding factor has errors which are related so as to leave well-conditioned zeros undisturbed.

31. We have concentrated our discussion on Newton's iterative method because this is the simplest, but there are several other iterative methods for which much the same results apply. Among these we may mention two cubically convergent methods. The first [**19**] is based on the formula

$$z_{r+1} = z_r - \frac{f(z_r)}{f'(z_r)} - \frac{\{f(z_r)\}^2\{f''(z_r)\}}{2\{f'(z_r)\}^3} \tag{31.1}$$

and the second, due to Laguerre [**19**], on the formulae

$$z_{r+1} = z_r - \frac{nf(z_r)}{f'(z_r) \pm \{H(z_r)\}^{1/2}} \tag{31.2}$$

$$H(z_r) = (n-1)^2\{f'(z_r)\}^2 - n(n-1)f(z_r)f''(z_r). \tag{31.3}$$

The fundamental limitation on the accuracy attainable with these methods is determined by our ability to evaluate $f(z_r)$ accurately in the neighbourhood of a zero. The limiting accuracy is therefore almost exactly the same as for Newton's method. Which of the methods we prefer in practice, depends on their convergence in the large and freedom from exceptional cases for which homing on a zero does not take place. Although of fundamental importance, such topics are not our main concern in this book.

Laguerre's method has a decided advantage, at least as far as polynomials with real zeros are concerned. Starting from $z_0 = a$, the successive application of (31.2) gives two sequences which converge to the nearest zero greater than a and the nearest zero less than a respectively. We therefore have some control over the order in which zeros are found and this is valuable in connexion with the deflation process.

32. The method of Bairstow [**19**] may also be analysed by similar techniques. This method is concerned with the calculation of the real quadratic factors of polynomials with real coefficients. From an approximate quadratic factor $x^2 - px - l$ an improved factor is computed as follows. Let the result of dividing $f(x)$ twice in succession by $x^2 - px - l$ be denoted by

$$\left.\begin{aligned} f(x) &= (x^2-px-l)q(x) + q_1x + (q_0-pq_1) \\ q(x) &= (x^2-px-l)T(x) + T_1x + (T_0-pT_1)\,. \end{aligned}\right\} \tag{32.1}$$

Then $x^2 - (p+\delta p)x - (l+\delta l)$ is the improved factor where

$$\left.\begin{aligned} D\delta p &= T_1q_0 - T_0q_1 \qquad & D\delta l &= Mq_1 - T_0q_0 \\ M &= lT_1 + pT_0 & D &= T_0^2 - MT_1\,. \end{aligned}\right\} \tag{32.2}$$

Bairstow's method is quadratically convergent and is somewhat more convenient in practice than that of Newton. The fundamental limitation on its accuracy is determined by our ability to evaluate q_0 and q_1 as x^2-px-l tends to a quadratic factor.

If we write

$$q(x) = q_n x^{n-2} + q_{n-1}x^{n-3} + \ldots + q_2 \tag{32.3}$$

then we have

$$\left.\begin{aligned} q_n &= a_n \qquad q_{n-1} = pq_n + a_{n-1} \\ q_r &= pq_{r+1} + lq_{r+2} + a_r \quad (r = n-2, n-3, \ldots, 0). \end{aligned}\right\} \tag{32.4}$$

We may compare this three-term recurrence relation giving the q_r with the two-term relation giving the s_r in equation (15.1). In practice q_r is given in terms of computed q_{r+1} and q_{r+2} by the relation

$$\begin{aligned} q_r &\equiv fl(pq_{r+1} + lq_{r+2} + a_r) \\ &\equiv pq_{r+1} + lq_{r+2} + a_r' \end{aligned} \tag{32.5}$$

where this is to be regarded as the definition of a_r'. The computation of q_1 and q_0 is therefore exact for a polynomial with the perturbed coefficient a_r'.

We shall not carry out the analysis in detail but from considerations analogous to those of section 23 it may be shown that, if we use deflation, it will be essential to find first those factors having the smaller values of p and l.

The methods we have just mentioned all involve the computation of the derivative of the polynomial or its equivalent. There is another class of iterative methods in which the computation of the derivative is avoided. Typical of these is the method of successive linear interpolation in which successive iterates are given by

$$x_{r+1} = \frac{x_r f(x_{r-1}) - x_{r-1} f(x_r)}{f(x_{r-1}) - f(x_r)}. \tag{32.6}$$

The fundamental limitation on the accuracy of these methods is also the ability to compute $f(x_r)$ in the neighbourhood of the zero and the limiting accuracy is much the same as that of methods using the derivative. However, the recognition of the limiting accuracy is somewhat more tiresome; the methods have the disadvantage that after attaining the limiting accuracy for some value of x_r, the next iterate may move well away from the true value. This contrasts with the earlier methods for which an x_r of limiting accuracy is almost invariably succeeded by an infinite sequence of x_i, all of which are also of limiting accuracy.

THE ROOT-SQUARING PROCESS

33. It might well be felt that any of the other better known methods of computing zeros, will obtain values of an accuracy comparable with what we have called the limiting accuracy for iteration, provided we use floating-point arithmetic. This is far from true, however. Indeed we have already shown that if we include deflation, the iterative method itself may fall catastrophically short of this, even when accepted values are correct to working accuracy and all zeros are well-conditioned.

However, deflation has always had a rather poor reputation (in our opinion, undeservedly), and it is perhaps more convincing to show that one of the alternatives to iteration may give approximations of far less than the limiting accuracy. One of the most popular methods of computing zeros is that of root-squaring [1], [19], and we now consider its numerical stability.

Suppose $z_1, z_2, \ldots, z_n$ are the zeros of the polynomial

$$f(z) = a_0 + a_1 z + \ldots + a_n z^n. \tag{33.1}$$

Then the polynomial with zeros $-z_1^2, -z_2^2, \ldots, -z_n^2$ is given by

$$g(z) = b_0 + b_1 z + \ldots + b_n z^n \tag{33.2}$$

where

$$b_r = a_r^2 - 2a_{r-1}a_{r+1} + 2a_{r-2}a_{r+2} - \ldots \quad (r = 0, 1, 2, \ldots, n). \tag{33.3}$$

The application of this transformation m times in succession gives a polynomial with zeros $-z_1^M, -z_2^M, \ldots, -z_n^M$ where $M = 2^m$, and for sufficiently large m the polynomial has coefficients from which the moduli of the z_i may readily be determined.

We shall not be concerned with the practical details of the process for extracting the zeros but with the answer to the question 'How are the zeros of the polynomial affected by computing the b_r using floating-point arithmetic?'

As usual, we attempt a backward analysis first. It will be convenient to refer to the polynomial $g(z)$ as the *squared polynomial corresponding to* $f(z)$. Clearly there is some modified polynomial $\bar{f}(z)$ having coefficients $\bar{a}_r$ for which the computed polynomial is the exact corresponding squared polynomial. Indeed, if $y_1, y_2, \ldots, y_n$ are the exact zeros of computed $g(x)$ then $\bar{f}(x)$ is the polynomial having the zeros $-y_1^{\frac{1}{2}}, -y_2^{\frac{1}{2}}, \ldots, -y_n^{\frac{1}{2}}$. If we take the appropriate square roots, then obviously as the word length tends to infinity the $\bar{a}_r$ tend to the a_r.

34. We may express each $\bar{a}_r$ in the form

$$\bar{a}_r = a_r(1 + F_r) \tag{34.1}$$

and it is our object to find bounds for F_r and in particular to compare them with the bounds for the E_r given by equation (15.3). The b_r are the computed coefficients and we have

$$\begin{aligned} b_r &= fl(a_r^2 - 2a_{r-1}a_{r+1} + 2a_{r-2}a_{r+2} - \ldots) \\ &\equiv a_r^2(1 + E_{r1}) - 2a_{r-1}a_{r+1}(1 + E_{r2}) + 2a_{r-2}a_{r+2}(1 + E_{r3}) - \ldots \end{aligned} \tag{34.2}$$

where the E_{ri} have the usual bounds associated with inner-products. By hypothesis the computed b_r correspond *exactly* to the polynomial with coefficients $a_r(1 + F_r)$ and hence

$$\begin{aligned} b_r \equiv\ & a_r^2(1 + F_r)^2 - 2a_{r-1}a_{r+1}(1 + F_{r-1})(1 + F_{r+1}) \\ & + 2a_{r-2}a_{r+2}(1 + F_{r-2})(1 + F_{r+2}) \ldots . \end{aligned} \tag{34.3}$$

Equations (34.2) and (34.3) provide a set of $n+1$ simultaneous non-linear equations expressing the F_r in terms of the E_{ri}. This contrasts with the situation which we had in the backward analysis of the evaluation of $f(x)$ in the iteration process. There the perturbations were determined quite simply, the variables occurring separately.

If the process is to be of much value we expect the F_r to be small. Ignoring products of the F_r for the moment, we obtain a set of linear algebraic equations in F_r; the typical equation being

$$2a_r^2F_r - 2a_{r-1}a_{r+1}(F_{r-1}+F_{r+1}) + 2a_{r-2}a_{r+2}(F_{r-2}+F_{r+2}) - \ldots$$

$$= a_r^2E_{r1} - 2a_{r-1}a_{r+1}E_{r2} + 2a_{r-2}a_{r+2}E_{r3} - \ldots . \quad (34.4)$$

The matrix of the coefficients of F_r is, typically for $n = 4$, of the form

$$\begin{bmatrix} 2a_0^2 & & & & & \\ -2a_0a_2 & 2a_1^2 & -2a_2a_0 & & & \\ -2a_0a_4 & -2a_1a_3 & 2a_2^2 & -2a_3a_1 & -2a_4a_0 & \\ & & -2a_2a_4 & 2a_3^2 & -2a_4a_2 & \\ & & & & 2a_4^2 & \end{bmatrix}. \quad (34.5)$$

The condition of this matrix of coefficients is obviously of vital importance; if it is ill-conditioned then the solutions F_r may well be quite large. In fact we can see at once that the matrix can be singular, for the rth column is a multiple of a_r, and hence is null if a_r is zero. This is not necessarily serious, because if a_r is zero we are not interested in F_r and this may turn out to be the only large element in the solution. Nevertheless it is not a promising state of affairs and it suggests that there may be polynomials such that the errors made in one stage of root-squaring are equivalent to large perturbations in the original coefficients.

FORWARD ERROR ANALYSIS OF ROOT-SQUARING

35. We turn to a forward analysis which in this case appears to be more illuminating. We now denote the computed coefficients of the squared polynomial by $\bar{b}_r$ and the exact coefficients by b_r. Not only have we to compare b_r with $\bar{b}_r$ but we have also to take into account the difference between the conditions of the zeros of the squared polynomial and those of the original. If the squared polynomial is much better conditioned we can tolerate much larger relative errors in the b_r than in the a_r.

We consider first a very ill-conditioned polynomial having the zeros

$$-(a+\theta_1), -(a+\theta_2), \ldots, -(a+\theta_n)$$

where

$$2^{-t} \ll |\theta_i| \ll a \quad (i = 1, 2, \ldots, n). \quad (35.1)$$

In fact we shall assume that t is so large that the zeros of the computed squared polynomial are still very close to $-(a+\theta_r)^2$. Now the original polynomial is approximately $(z+a)^n$ and hence we have

$$a_r \doteq {}^nC_r a^{n-r} \tag{35.2}$$

while the exact squared polynomial is approximately $(z+a^2)^n$, so that

$$b_r \doteq {}^nC_r a^{2n-2r} = a_r a^{n-r}. \tag{35.3}$$

The perturbation δ_r in the zero, $-(a+\theta_r)$, resulting from a perturbation $a_s\epsilon$ in a_s is given by

$$|\delta_r| \doteq \left|\frac{(a+\theta_r)^s a_s \epsilon}{\prod_{i\neq r}(\theta_r-\theta_i)}\right| \doteq \left|\frac{a^s a_s \epsilon}{\prod_{i\neq r}(\theta_r-\theta_i)}\right| \tag{35.4}$$

while the perturbation η_r in the zero, $-(a+\theta_r)^2$, resulting from a perturbation $b_s\epsilon$ in b_s is given by

$$\begin{aligned}|\eta_r| &\doteq \left|\frac{(a+\theta_r)^{2s} b_s \epsilon}{\prod_{i\neq r}[(a+\theta_r)^2-(a+\theta_i)^2]}\right| \\ &\doteq \left|\frac{a^{2s+1-n} b_s \epsilon}{2^{n-1}\prod_{i\neq r}(\theta_r-\theta_i)}\right|.\end{aligned} \tag{35.5}$$

The corresponding zero of the original polynomial is

$$-[(a+\theta_r)^2-\eta_r]^{1/2} \doteq -\left[a+\theta_r-\frac{\eta_r}{2a}\right]. \tag{35.6}$$

If we denote the perturbation in this derived zero of the original polynomial by δ_r' we have

$$\begin{aligned}\left|\frac{\delta_r'}{\delta_r}\right| &= \left|\frac{\eta_r}{2a\delta_r}\right| = \frac{b_s}{2^n a^{n-s} a_s} \\ &= \frac{1}{2^n}.\end{aligned} \tag{35.7}$$

This shows that we can tolerate relative errors in b_s which are 2^n times as large as those which can be tolerated in a_s. (Note that this is only true if both errors are small enough for the linear theory to apply.)

This result holds even if all the θ_i are zero, for we have then from (7.12):

$$|\delta_r| \doteq |(a_s a^s \epsilon)^{1/n}| \tag{35.8}$$

$$|\eta_r| \doteq |(b_s a^{2s} \epsilon)^{1/n}| \tag{35.9}$$

$$|\delta_r'| \doteq \left|\frac{(b_s a^{2s}\epsilon)^{1/n}}{2a}\right| = \left|\left(a_s a^s \frac{\epsilon}{2^n}\right)^{1/n}\right|. \tag{35.10}$$

Hence again we can tolerate relative errors in the b_s which are 2^n times as large as those in the a_s.

36. In fact the relative error in the computed coefficients $\bar{b}_s$ *is* high for ill-conditioned polynomials of the type we have just discussed. For the polynomial with zeros $a(1+\theta_i)$ we have

$$\bar{b}_s = fl(a_s^2 - 2a_{s-1}a_{s+1} + 2a_{s-2}a_{s+2} - \ldots) \tag{36.1}$$

and we know that $a_s \doteq {}^nC_s\, a^{n-s}$ and $b_s \doteq {}^nC_s\, a^{2n-2s}$. Now, as in (34.2), the errors in the calculation of the inner-product are bounded by

$$|a_s^2 E_{s1}| + |2a_{s-1}a_{s+1}E_{s2}| + \ldots \tag{36.2}$$

where the E_{si} are the usual bounds. The orders of magnitude of the elements in (36.2) are

$${}^nC_s\,{}^nC_s\, a^{2n-2s}|E_{s1}| + 2\,{}^nC_{s-1}\,{}^nC_{s+1}\,a^{2n-2s}|E_{s2}| + \ldots . \tag{36.3}$$

Considerable cancellation takes place in the computation of $\bar{b}_s$ since although the first component is of order $({}^nC_s)^2 a^{2n-2s}$, the final sum is of order ${}^nC_s\, a^{2n-2s}$. The order of magnitude of the relative error in $\bar{b}_s$ is likely to be approximately ${}^nC_s|E_{s1}|$. This takes its greatest value for $s = \dfrac{n}{2}$ and we have, using (26.6) and (26.9) of Chapter 1,

$$\frac{\bar{b}_{n/2}}{b_{n/2}} = 1 + e \quad \text{where} \quad |e| = O({}^nC_{n/2}\, n2^{-t}) \tag{36.4}$$

since the inner-product is then of order $\dfrac{n}{2}$. Hence we have

$$|e| = O\left[2^n\left(\frac{2n}{\pi}\right)^{1/2} 2^{-t}\right] \text{ for large } n. \tag{36.5}$$

Now we have shown that the errors made when iterating in the original polynomial are equivalent to relative errors in the a_r and we had

$$\bar{a}_r = (1+e); \qquad e = O(n2^{-t}). \tag{36.6}$$

Hence, although (35.7) shows that the conditions of the zeros of the squared polynomial are better by factors which are of order 2^n, (36.5) shows that the relative error in the computed coefficients is about 2^n times as great as that corresponding to iteration.

When root-squaring is applied to an ill-conditioned polynomial it is necessary then to use higher precision to deal with the cancellation that takes place in the computation of the $\bar{b}_r$. *We should not regard this as a weakness of the method since the number of extra figures required is strictly comparable with the number which would be required when iterating for a zero in the original polynomial.* Ill-conditioning is bound to make itself felt in some way, and this is merely the way in which it reveals itself in root-squaring. After a few stages of root-squaring the zeros become well separated and cancellation ceases.

In fact, we now show that if we are able to accumulate inner-products in floating-point, then root-squaring deals more effectively with ill-conditioning than iteration in the original polynomial. (Note that this is not a contradiction of our assertion about the fundamental limitations of t-digit arithmetic. In accumulating the inner-product we are effectively performing $2t$-digit arithmetic and there is always the chance that this may be turned to advantage.) In accumulating floating-point we have, from (31.3) of Chapter 1,

$$\begin{aligned} \bar{b}_s &\equiv fl_2(a_s^2 - 2a_{s-1}a_{s+1} + 2a_{s-2}a_{s+2} \ldots) \\ &\equiv [a_s^2(1+E_1) - 2a_{s-1}a_{s+1}(1+E_2) - \ldots](1+G_1) \qquad (36.7) \\ |E_r| &= O(2^{-2t}) \qquad |G_1| \leqslant 2^{-t}. \qquad (36.8) \end{aligned}$$

Hence from (32.9) and (32.10) of Chapter 1,

$$\bar{b}_s = b_s(1+G_1) + e \qquad (36.9)$$

$$|e| < (|a_s^2| + |2a_{s-1}a_{s+1}| + \ldots)(\tfrac{3}{2}n2^{-2t_2}(1+2^{-t}). \qquad (36.10)$$

Probably the most severe cancellation occurs when the polynomial has n almost coincident zeros and even in this case it is evident that if t is appreciably greater than n, e will be unimportant compared with b_sG_1. *The $\bar{b}_s$ will therefore have errors which are bounded by one part in 2^t. Since we have shown that the condition number is improved by a factor of 2^n, the accuracy attainable working with the squared polynomial is much higher than that attainable with the original.*

The user of root-squaring has little to fear then from ill-conditioned equations, and for the worst types of zero-clusters we may actually gain substantially if we can accumulate inner-products. We cannot obtain a corresponding gain from accumulating inner-products when iterating in the original polynomial because the main limitation on the accuracy is the computation of the s_r defined to be $fl(\alpha s_{r+1} + a_r)$ as in (22.1). Each s_r must be rounded before computing the next, because otherwise the multiplication of double-precision numbers would be involved. The use of the double-precision αs_{r+1} in the addition is advantageous only when cancellation takes place, and this is rare.

NUMERICAL EXAMPLE

37. We may illustrate these points by considering a typical example of an ill-conditioned polynomial. We take the distribution of zeros -1, -2, -3, . . . , -20. The order of magnitude of the coefficients a_s and b_s to the nearest power of 2 are displayed in Table 5.

TABLE 5

s	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$\log_2 a_s$	1	8	15	21	26	31	36	40	44	47	51	54	56	59	61	62	63	64	64	63	62
$\log_2 b_s$	1	12	22	32	41	50	58	66	73	80	86	93	98	104	109	113	117	120	122	123	123

We examine now the computation of the b_i; for b_{15}, for example, we have

$$b_{15} = a_{15}^2 - 2a_{16}a_{14} + 2a_{17}a_{13} - 2a_{18}a_{12} + 2a_{19}a_{11} - 2a_{20}a_{10} \quad (37.1)$$

and we know that $b_{15} \doteq 2^{50}$.

Now the first two terms in (37.1) are approximately 2^{62} and 2^{63} respectively, after which they decrease. There is a cancellation of some 13 binary figures in the computation of b_{15}. The relative error in b_{15} is therefore likely to be of the order of 2^{13-t}. It will be seen that in all cases the maximum term in the expression for b_s is either a_s^2 or $2a_{s+1}a_{s-1}$ and the greatest cancellation occurs in the computation of b_{10} for which the relative error is likely to be of the order of 2^{16-t}. Cancellation is therefore almost as severe as in the case of 20 pathologically close zeros, (36.4) and (36.5).

In spite of this cancellation, the preservation of these zeros by root-squaring does not compare unfavourably with that corresponding to iteration in the original polynomial. We have already seen in (9.4) that the zero at $z = -16$ is one of the most sensitive. A fair comparison is between a perturbation $2^{-t}a_s$ in a_s and $2^{-t}(a_s^2 + 2a_{s+1}a_{s-1} + \ldots)$ in b_s. The comparison is at its least favourable for root-squaring when $s = 13$. We denote the perturbation corresponding to $2^{-t}a_s$ by p and that corresponding to the perturbation in b_s by q. Then from Table 5, replacing a_{13} by 2^{40} and $(a_{13}^2 + 2a_{14}a_{13} + \ldots)$ by 2^{81}

$$p \doteq 2^{-t}2^{40}\frac{(16)^{13}}{4!\,15!} \quad (37.2)$$

$$q \doteq 2^{-t}2^{81}\frac{(16)^{26}16.32}{4!\,36!}. \quad (37.3)$$

From (35.6), q corresponds to an error q' given by

$$q' \doteq 2^{-t}2^{81}\frac{(16)^{26}16.32}{4!\,36!\,2.16}$$

$$= 2^{-t}2^{81}\frac{(16)^{27}}{4!\,36!}. \quad (37.4)$$

Hence

$$\frac{q'}{p} \doteq 2^{41}\frac{16^{14}15!}{36!} \doteq 0{\cdot}56. \quad (37.5)$$

The effect of the high relative error is almost exactly balanced by the improvement of the condition. Whether root-squaring or iteration is the more effective will depend mainly on whether the particular values assumed by the rounding errors in the one or the other happen to be more favourable.

Since the maximum cancellation is 16 binary digits, if we can accumulate floating-point inner-products, the error caused by the cancellation will be quite negligible provided t is appreciably greater than 16. This will be true on almost any modern computer. Hence the relative error in the computed b_s will be bounded by one part in 2^t. We will therefore gain

the full advantage from the improvement in the condition and iteration in the computed squared polynomial will give about 16 more correct binary figures in the zeros than iteration in the original!

DETERIORATION OF CONDITION

38. There remains the possibility that a well-conditioned polynomial may lead to an ill-conditioned squared polynomial. We now show that there are indeed such polynomials. Consider the polynomial $f(z) = z^2 - 1$. This has two distinct well-conditioned zeros at $z = \pm 1$. The squared polynomial on the other hand is $z^2 + 2z + 1$ and this has coincident zeros. A severe worsening of the condition has taken place.

A simple example will illustrate the loss of accuracy that takes place in one stage of root-squaring with such a polynomial. Consider the polynomial

$$0{\cdot}93254\,613x^2 - 10^{-8}(0{\cdot}12346\,723)x - 0{\cdot}87654\,321. \qquad (38.1)$$

The corresponding, squared polynomial, computed using 8 decimals is

$$0{\cdot}86964\,228x^2 + 1{\cdot}63483\,40x + 0{\cdot}76832\,800. \qquad (38.2)$$

Virtually no cancellation takes place. The zeros of the computed squared polynomial are $-0{\cdot}94017\,094$ and $-0{\cdot}93972\,156$ corresponding to $0{\cdot}96962\,413$ and $-0{\cdot}96939\,237$* in the original. The true zeros of the original are $\pm\,0{\cdot}96950\,824$.

39. As a more convincing example we take, not a specially constructed polynomial, but one which was used to illustrate the root-squaring process [**19**]. The polynomial is

$$\begin{aligned} &2{\cdot}03253\,121x^{16} + 3{\cdot}43560\,48x^{15} + 25{\cdot}17830\,48x^{14} + 37{\cdot}65109\,6x^{13} \\ &+ 128{\cdot}21874\,8x^{12} + 166{\cdot}44768x^{11} + 345{\cdot}07256x^{10} + 378{\cdot}908x^9 \\ &+ 524{\cdot}327x^8 + 468{\cdot}88x^7 + 443{\cdot}576x^6 + 304{\cdot}08x^5 + 190{\cdot}68x^4 \\ &+ 89{\cdot}6x^3 + 32{\cdot}8x^2 + 8x + 1. \end{aligned} \qquad (39.1)$$

The zeros of this polynomial to 8 decimal places are

$$\begin{aligned} &-0{\cdot}29350\,453 \pm 0{\cdot}14349\,930i \\ &-0{\cdot}22447\,006 \pm 0{\cdot}45092\,796i \\ &-0{\cdot}14762\,378 \pm 0{\cdot}77175\,720i \\ &-0{\cdot}09003\,999 \pm 1{\cdot}06119\,206i \\ &-0{\cdot}05086\,444 \pm 1{\cdot}29691\,128i \\ &-0{\cdot}02566\,871 \pm 1{\cdot}47437\,714i \\ &-0{\cdot}01049\,355 \pm 1{\cdot}59629\,550i \\ &-0{\cdot}00248\,920 \pm 1{\cdot}66712\,036i \end{aligned} \qquad (39.2)$$

* The signs to be associated with these values are indeterminate.

It is not a particularly ill-conditioned polynomial, especially from the point of view of a computer such as ACE which has the equivalent of a 14-decimal digit word. It has been solved by several different iterative methods on ACE and DEUCE using deflation, and in all cases even the least accurate zero had $t-7$ correct binary figures when working to a precision of t binary figures. Thus only a little more than 2 decimals was lost, and examination of the condition of the zeros given above would show that this is the minimum loss that can be expected.

The polynomial obtained after three stages of root-squaring using effectively ten-decimal floating-point arithmetic without accumulation as carried out on a desk computer, is

$$\begin{aligned}
&10^{2}(2\cdot 91217\ 3603)x^{16} + 10^{4}(7\cdot 74010\ 338)x^{15} + 10^{6}(8\cdot 57690\ 8)x^{14}\\
&+ 10^{8}(5\cdot 14935\ 1)x^{13} + 10^{10}(1\cdot 82373\ 6)x^{12} + 10^{11}(3\cdot 89619\ 5)x^{11}\\
&+ 10^{12}(4\cdot 94648)x^{10} + 10^{13}(3\cdot 55854)x^{9} + 10^{14}(1\cdot 33744\ 2)x^{8}\\
&+ 10^{14}(2\cdot 22644\ 7)x^{7} + 10^{14}(1\cdot 68660\ 7)x^{6} + 10^{12}(6\cdot 21763\ 0)x^{5}\\
&+ 10^{12}(3\cdot 40519\ 8222)x^{4} - 10^{10}(2\cdot 50193\ 6135)x^{3} + 10^{7}(6\cdot 49492\ 8487)x^{2}\\
&- 10^{4}(1\cdot 39578\ 86)x + 1\cdot 0. \qquad (39.3)
\end{aligned}$$

Now consider the 8th pair of zeros given in (39.2). These are very well separated. The corresponding zeros of the first squared polynomial are $2\cdot 77928\ 4 \pm 0\cdot 00830\ 0i$ and these are very close indeed. Indeed, most of the complex pairs have moved very much closer together. The first squared polynomial is much worse-conditioned than the original and moreover considerable cancellation occurs in calculating the coefficients. This cancellation gradually dies out in the later squarings but is still far from negligible in the third squaring which is exhibited above.

It is not surprising to find then that the rounding errors have caused some of the zeros of the third squared polynomial to differ very substantially from those corresponding to the original polynomial. In fact accurate zeros of the computed squared polynomial (39.3) are

$$\begin{aligned}
10^{-3}(0\cdot 11413\ 304) &\pm 10^{-4}(0\cdot 61810\ 124)i\\
10^{-2}(0\cdot 35254\ 760) &\pm 10^{-2}(0\cdot 21783\ 512)i\\
10^{-2}(-0\cdot 85387\ 199) &\pm 10^{0}(0\cdot 14505\ 105)i\\
10^{1}(-0\cdot 12898\ 826) &\pm 10^{1}(0\cdot 10370\ 309)i\\
10^{1}(-0\cdot 76596\ 691) &\pm 10^{1}(0\cdot 24842\ 925)i\\
10^{2}(-0\cdot 22141\ 752) &\pm 10^{1}(0\cdot 31225\ 083)i\\
10^{2}(-0\cdot 40136\ 605) \quad & 10^{2}(-0\cdot 52572\ 3869)\\
10^{2}(-0\cdot 46837\ 460) \quad & 10^{2}(-0\cdot 64045\ 531)
\end{aligned}\Bigg\}\ \text{Real zeros} \qquad (39.4)$$

Four of the zeros have become real; the zero of the original polynomial 'corresponding' to the last zero is obtained by taking that 8th root of $10^2(0\cdot64045\,531)$ which is a pure imaginary. This gives $1\cdot68194\,24i$ compared with the true value of $-0\cdot00248\,920+1\cdot66712\,036i$. For computation of the same precision Bairstow's method with deflation gave almost 8 correct decimals in this zero (without purification in the original).

As a more extreme example of the same phenomenon we may take the polynomial (27.1). This polynomial is, to be sure, quite ill-conditioned initially, the worst zeros having condition numbers of order 10^9. However it is obvious that the first squared polynomial is far more ill-conditioned. The ill-condition of the original polynomial arises from the fact that there are clusters of zeros near $\pm 0\cdot3i$; but after squaring, these two clusters move together, giving a far more severe cluster near $-0\cdot1$. Olver states [**19**] that root-squaring failed to determine the zeros even when working to 20 decimals. Using Bairstow and deflation on this polynomial with 18-digit decimal computation the *least* accurate zero was obtained correct to 9 decimals.

GENERAL COMMENTS ON THE COMPUTATION OF ZEROS OF POLYNOMIALS

40. The danger of a severe worsening of the condition of a polynomial as a result of root-squaring is, in our opinion, the most serious weakness in the method. We have considered in detail the case when some of the zeros are almost pure imaginaries. This is likely to be a very common situation in the consideration of problems of damped mechanical and electrical vibrations. There we are likely to be interested in the regions of critical stability and these occur when at least one of the zeros becomes pure imaginary.

However, any pair of zeros having phases which are close to $\pm\frac{\pi}{2^m}$, for some value of m, will become an almost coincident pair of zeros at some stage. The same remark applies to zeros of almost equal modulus having a phase separation of $\frac{\pi}{2^m}$. For example, the polynomial with zeros ± 1, ± 2, . . . , ± 10 becomes a polynomial having 10 double roots after one application of root-squaring. It is perhaps worth mentioning that if the coefficients of the original polynomial do not require many figures for their representations, several stages of root-squaring may be performed without rounding errors. This can be very important in practice. With iterative methods the arguments will inevitably be such that rounding errors are involved.

No universally satisfactory method has so far been proposed for computing zeros and it is perhaps premature to exclude root-squaring from consideration. In our experience iterative methods with deflation followed by purification have proved very successful, but for the general problem it does not seem to be easy to devise a method of ensuring that convergence

will always take place starting from rather crude approximations. Further, if we use deflation, we need to have a device for ensuring that the zeros are found in increasing order of magnitude, though there are other methods of eliminating computed zeros which avoid this necessity. The method of D. H. Lehmer [**16**] is, in many respects, the most promising of recent proposals though it is, perhaps, too slow for some purposes.

In attempting to devise a procedure for an automatic computer which will find zeros to a prescribed accuracy, it should be appreciated that computation of a very high precision may sometimes be necessary. An essential feature of any such programme would be some device for assessing the condition of each zero and adjusting the precision of computation accordingly. Most automatic procedures have aimed rather lower than this and usually are designed to use at most double- or treble-precision arithmetic.

Additional comments

Discussions of the sensitivity of the zeros of polynomials to perturbations in the coefficients have tended to show a preoccupation with multiple zeros, to the neglect of the general problem of ill-conditioning. One of the more satisfactory treatments of this problem was in a paper by Olver on 'The evaluation of zeros of high-degree polynomials' [19], though even there the comments could well have been more emphatic. Notwithstanding the triviality of the determination of the sensitivity of the zeros of, for example, the polynomial $\prod_{r=1}^{20}(x-r)$, it is probably true to say that most numerical analysts are shocked when first brought face to face with the result.

Many of the standard numerical methods which have been developed for the computation of the eigenvalues of a matrix lead to an explicit determination of the characteristic polynomial. Although expositions of such methods often included a discussion of the accuracy of the computed polynomial coefficients, the relevance of this to the accuracy of the eigenvalues was usually ignored.

Our main object in the chapter has been to focus attention on the severe inherent limitations of all numerical methods for finding the zeros of polynomials and to show that several of the simpler iterative methods, *provided they are well started*, give almost the best possible results for a prescribed precision in the computation.

The errors resulting from polynomial deflation are discussed in detail partly because this process gives a simple illustration of the power of backward analysis and partly because misunderstanding concerning this process is quite common. Its use as an illustration should not be taken to imply that we are wholeheartedly in favour of polynomial deflation. On general grounds of stability there is much to be said for alternative methods which enable one to work directly with the original polynomial throughout.

Typical of these methods is the following. Suppose zeros $z_1, z_2, \ldots, z_r$ of the polynomial $f(z)$ have been accepted. Then if

$$f(z) = g(z)\prod_{i=1}^{r}(z-z_i)$$

we have

$$f'(z)/f(z) = g'(z)/g(z) + \sum_{i=1}^{r}\frac{1}{z-z_i}.$$

Hence $g'(z)/g(z)$ may be computed from $f'(z)$ and $f(z)$ and the accepted zeros, and we can use Newton's method without an explicit deflation. The accuracy of the later zeros is then completely unaffected by errors in the earlier zero; indeed, even the acceptance of a completely erroneous zero does no harm.

The discussion of the problem of cancellation in the root-squaring process has much in common with that given by Olver [19], but here the emphasis is very different. It would be natural from Olver's treatment to conclude that this cancellation is a specific weakness of root-squaring, whereas from our standpoint this is not true. The main weakness of root-squaring, that of a possible severe worsening of the condition in the early stages, seems to have attracted little attention. It is perhaps worth mentioning that the ultimate aim of the root-squaring process is to produce a very well-conditioned polynomial.

We wish to draw the attention of the reader to the paper 'A machine method for solving polynomial equations' by D. H. Lehmer [16]. The technique described there should provide a reliable automatic procedure for locating the zeros and its use in conjunction with one of the standard iterative procedures should ensure that the latter is well-started. It has the further advantage of finding the zeros roughly in increasing order of absolute magnitude so that polynomial deflation may be used with greater safety than with most methods.

3

MATRIX COMPUTATIONS

INTRODUCTION

1. The two central problems in the matrix field are the solution of linear equations, including as a special case the inversion of matrices, and the calculation of the eigensystem of a matrix. The literature dealing with these topics is very extensive and we do not wish to duplicate it in this book. We have therefore selected a few typical examples which illustrate the techniques, and have concentrated on the more practical aspects.

Rigorous error analyses of matrix processes are commonly regarded as beyond the range of all but a few specialists. In our opinion this is far from true. The difficulties involved in such error analyses are conceptual rather than mathematical. The inexperienced worker in this field is inclined to attempt a comparison between the numbers actually obtained in the practical realisation of an algorithm and those which would have been obtained by exact computation throughout. Such a comparison is seldom profitable. Usually it is very difficult to carry it through, and in any case it often leads to the conclusion that the numbers arising in the course of the practical computation can be almost arbitrarily different from those arising in an exact computation.

The choice of a good basis of comparison is an essential feature of a successful analysis, and in this respect the 'backward' technique has distinct advantages, though there are algorithms for which forward and backward analyses are equally satisfactory.

Most of the early analyses were for fixed-point computation, probably because the first automatic computers did not have floating-point facilities. This seems to have given the impression that rigorous floating-point error analyses are comparatively difficult. This is by no means the case; on the contrary, in our experience, floating-point error analyses have proved to be considerably more straightforward than the corresponding fixed-point analyses. Quite a number of floating-point analyses consist essentially of applications of the results given in Chapter 1, sections 24–26 and 31–32 for $fl(x_1y_1+\ldots+x_ny_n)$ and $fl_2(x_1y_1+\ldots+x_ny_n)$, and the corresponding results for minor modifications such as $fl(x_1y_1+\ldots+x_ny_n+z)$, for example. The reader should familiarise himself with these results so that he will be capable of making simple extensions of them purely automatically.

2. It is useful to have a single number which gives an overall assessment of the size of a vector or a matrix and plays the same role as the modulus in the case of a complex number. For this purpose we shall use certain functions of the elements of a vector and matrix which are called *norms*.

The norm of a vector x will be denoted by $\|x\|$ and the norms we use will all satisfy the relations

$$\|x\| > 0 \text{ unless } x = 0 \tag{2.1}$$

$$\|kx\| = |k|\,\|x\| \text{ where } k \text{ is a complex number} \tag{2.2}$$

$$\|x + y\| \leqslant \|x\| + \|y\|. \tag{2.3}$$

From (2.3) we have

$$\left.\begin{aligned} \|x - y\| &\geqslant \|x\| - \|y\| \\ \|x - y\| &\geqslant \|y\| - \|x\| \end{aligned}\right\} \tag{2.4}$$

one of which is always trivial.

There are three vector norms in common use. They are defined by

$$\|x\|_p = (|x_1|^p + |x_2|^p + \ldots + |x_n|^p)^{\frac{1}{p}} \quad (p = 1, 2, \infty), \tag{2.5}$$

where $\|x\|_\infty$ is interpreted as $\max |x_i|$. The norm $\|x\|_2$ is the *length* of the vector x as normally understood. If θ is the angle between x and y then

$$x^T y = \|x\|_2 \|y\|_2 \cos\theta. \tag{2.6}$$

Similarly the norm of a matrix A will be denoted by $\|A\|$ and the norms we use will satisfy the relations

$$\|A\| > 0 \text{ unless } A = 0. \tag{2.7}$$

$$\|kA\| = |k|\,\|A\| \text{ where } k \text{ is a complex number} \tag{2.8}$$

$$\|A + B\| \leqslant \|A\| + \|B\| \tag{2.9}$$

$$\|AB\| \leqslant \|A\|\,\|B\|. \tag{2.10}$$

With each vector norm we can associate a corresponding matrix norm defined by

$$\|A\| = \max \|Ax\|/\|x\| \quad (x \neq 0) \tag{2.11}$$

and it may be verified that this quantity satisfies the relations (2.7) to (2.10). From (2.2) it is obvious that this is equivalent to

$$\|A\| = \max \|Ax\| \quad (\|x\| = 1). \tag{2.12}$$

This matrix norm is said to be the matrix norm *subordinate* to the vector norm. Corresponding to the three vector norms of (2.5) we have the matrix norms $\|A\|_p$. The reader should verify that

$$\|A\|_1 = \max_j \sum_i |a_{ij}| \tag{2.13}$$

$$\|A\|_\infty = \max_i \sum_j |a_{ij}| \tag{2.14}$$

$$\|A\|_2 = (\text{maximum eigenvalue of } A^H A)^{1/2} \tag{2.15}$$

where A^H is the complex conjugate transpose of A. The third of these norms $\|A\|_2$, is sometimes called the *spectral* norm. In particular (2.15) shows that if A is unitary, that is if A^HA is I, then $\|A\|_2 = 1$.

From their definitions it is evident that a vector norm and the subordinate matrix norm satisfy

$$\|Ax\| \leqslant \|A\| \|x\|, \tag{2.16}$$

and from (2.12), remembering that $\|Ax\|$ is continuous and the surface defined by $\|x\| = 1$ is closed, we see that there is always a vector for which equality holds. It is also obvious that $\|I\| = 1$ for any subordinate matrix norm.

Any matrix and vector norms for which (2.16) holds are said to be *consistent* or *compatible*. The existence of the subordinate norms ensures that there is always a matrix norm consistent with a given vector norm. A vector norm consistent with any given matrix norm may be defined by

$$\|x\| = \|x, 0, 0, \ldots, 0\|. \tag{2.17}$$

There is another important matrix norm which is consistent with the vector norm $\|x\|_2$. This is the norm $\|A\|_E$ defined by

$$\|A\|_E = \Big(\sum_i \sum_j |a_{ij}|^2\Big)^{1/2} \tag{2.18}$$

usually called the *Euclidean* norm or the *Schur* norm. The reader should verify that this is a norm and that it is consistent with $\|x\|_2$. That $\|A\|_E$ cannot be subordinate to any vector norm is obvious from the fact that $\|I\|_E = n^{1/2}$. This last property is rather undesirable and the Euclidean norm is unsatisfactory accordingly. We might say that in general it is rather larger than necessary. In fact it may be shown that

$$\|A\|_2 \leqslant \|A\|_E \leqslant n^{1/2}\|A\|_2 \tag{2.19}$$

both limits being attainable.

In strictly mathematical work the Euclidean norm is out of favour and the matrix norm $\|A\|_2$ is used in conjunction with $\|x\|_2$. However, for practical work and error analysis, $\|A\|_E$ has advantages over $\|A\|_2$.

These are:

(i) The Euclidean norm of $|A|$ is the same as that of A (here $|A|$ denotes the matrix with elements $|a_{ij}|$).

(ii) It is easy to compute, while the determination of $\|A\|_2$ is a major undertaking.

These two advantages are shared by $\|A\|_1$ and $\|A\|_\infty$, but the most we can say about $\||A|\|_2$ is

$$\||A|\|_2 \leqslant \||A|\|_E = \|A\|_E \leqslant n^{1/2}\|A\|_2. \tag{2.20}$$

In spite of the second of the inequalities (2.19), $\|A\|_E$ is often a surprisingly good approximation to $\|A\|_2$.

An important property of the matrix norms we have introduced is that each of them provides an upper bound for the moduli of the eigenvalues. For if λ is any eigenvalue of A and x is a corresponding eigenvector we have

$$Ax = \lambda x. \tag{2.21}$$

Hence using consistent matrix and vector norms we have

$$\|A\|\,\|x\| \geqslant \|Ax\| = \|\lambda x\| = |\lambda|\,\|x\|, \tag{2.22}$$

giving

$$\|A\| \geqslant |\lambda|. \tag{2.23}$$

From this result and (2.15) we can establish a relation between the 1, 2 and ∞ norms of A. We have in fact

$$\begin{aligned} \|A\|_2^2 &= \text{max eigenvalue of } A^H A \\ &\leqslant \|A^H A\|_\infty \\ &\leqslant \|A^H\|_\infty \|A\|_\infty \\ &= \|A\|_1 \|A\|_\infty. \end{aligned} \tag{2.24}$$

ERROR ANALYSIS OF SIMPLE MATRIX OPERATIONS

3. Consider first the multiplication of a matrix A by a scalar k. If the computed matrix be denoted by B, we have, for floating-point computation

$$\begin{aligned} b_{ij} &\equiv fl(k \times a_{ij}) \\ &\equiv ka_{ij}(1+\epsilon_{ij}) \qquad (|\epsilon_{ij}| \leqslant 2^{-t}). \end{aligned} \tag{3.1}$$

Hence

$$\begin{aligned} B - kA &\equiv ka_{ij}\epsilon_{ij} \\ \|B - kA\|_E &\leqslant |k|2^{-t}\|A\|_E. \end{aligned} \tag{3.2}$$

The corresponding results involving the spectral norm are weaker: the best we can obtain is

$$\|B - kA\|_2 \leqslant |k|2^{-t}\|A\|_E \tag{3.3}$$

or

$$\|B - kA\|_2 \leqslant |k|2^{-t}n^{1/2}\|A\|_2. \tag{3.4}$$

However, if all the elements of A are of the same sign we have $\| |A| \|_2 = \|A\|_2$ and the factor $n^{1/2}$ may be omitted. For the other norms we have

$$\|B - kA\|_p \leqslant |k|2^{-t}\|A\|_p \quad (p = 1, \infty). \tag{3.5}$$

Equation (3.2) may be written

$$\frac{\|B - kA\|_E}{|k|\,\|A\|_E} \leqslant 2^{-t} \tag{3.6}$$

in which the left-hand side may be thought of as a relative error.

In fixed-point computation, if $|k| \leqslant 1$, we have

$$b_{ij} = fi(k \times a_{ij}) \equiv ka_{ij} + \epsilon_{ij} \quad (|\epsilon_{ij}| \leqslant 2^{-t-1}). \tag{3.7}$$

Hence, from (2.18),

$$\|B - kA\|_E \leqslant n2^{-t-1}. \tag{3.8}$$

Note that if the elements of A are appreciably smaller than unity, we may have a high relative error in the computed matrix. Again the best we can achieve in spectral norms is

$$\|B - kA\|_2 \leqslant n2^{-t-1} \tag{3.9}$$

which is weaker.

MATRIX MULTIPLICATION

4. We consider first the computation of Ax where A is a $n \times n$ matrix. We may write

$$\begin{aligned} y_i &= fl(a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n) \\ &= a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n + e_i \end{aligned} \tag{4.1}$$

where from Chapter 1, section 26

$$|e_i| \leqslant 2^{-t_1}[n|a_{i1}||x_1| + n|a_{i2}||x_2| + (n-1)|a_{i3}||x_3| + \ldots + 2|a_{in}||x_n|]. \tag{4.2}$$

Hence

$$y \equiv Ax + e$$

where

$$|e| \leqslant 2^{-t_1}|A|\,D\,|x|; \qquad D = \begin{bmatrix} n & & & \\ & n & & \\ & & n-1 & \\ & & & \ddots \\ & & & & 2 \end{bmatrix}. \tag{4.3}$$

Here again $|A|$ denotes the matrix with elements $|a_{ij}|$ and an inequality in matrix or vector form applies separately to corresponding elements on each side.

This gives

$$\|e\|_2 = \||e|\|_2 \leqslant 2^{-t_1}n\|A\|_E\,\|x\|_2. \tag{4.4}$$

For matrix multiplication in floating-point we have similarly

$$C = fl(AB) = AB + E \tag{4.5}$$

$$\|E\|_E \leqslant 2^{-t_1}n\|A\|_E\|B\|_E. \tag{4.6}$$

Since $\|A\|_E\ 3\|B\|_E$ may be very much greater than $\|AB\|_E$ the computed C may well have a very high relative error.

If we can accumulate inner-products in floating-point, then writing

$$C = fl_2(AB) = AB + E \tag{4.7}$$

we see immediately from the bounds given in Chapter 1, section 32 for an accumulated floating-point inner-product that

$$|C - AB| \leqslant 2^{-t}|AB| + \tfrac{3}{2}2^{-t_2}|A|\,D\,|B| \tag{4.8}$$

where D is as in (4.3). Hence

$$\|C - AB\|_E = \||\,|C - AB|\,\|_E \leqslant 2^{-t}\|AB\|_E + \tfrac{3}{2}2^{-2t_2}n\|A\|_E\,\|B\|_E. \tag{4.9}$$

Unless quite exceptional cancellation takes place, the second term on the right will be negligible compared with the first. For matrices of any reasonable order we will then have effectively

$$\frac{\|C - AB\|_E}{\|AB\|_E} \leqslant 2^{-t}, \tag{4.10}$$

showing that the computed product has a very low relative error. However it is *possible* for t or more figures to be lost as a result of cancellation in the computation of each element of C, and then (4.10) is by no means true.

If B is A^T then the diagonal elements are given by

$$c_{ii} = fl_2(a_{i1}a_{i1} + a_{i2}a_{i2} + \ldots + a_{in}a_{in}) \tag{4.11}$$

and all contributions are positive. There is then no cancellation in the computation of the diagonal elements and we have

$$\begin{aligned}\|AA^T\|_E &\geqslant \left[(AA^T)_{11}^2 + (AA^T)_{22}^2 + \ldots + (AA^T)_{nn}^2\right]^{1/2} \\ &\geqslant \left[\frac{1}{n}\{(AA^T)_{11} + (AA^T)_{22} + \ldots + (AA^T)_{nn}\}^2\right]^{1/2} \\ &= n^{-1/2}\|A\|_E^2. \end{aligned} \tag{4.12}$$

Equation (4.9) now gives

$$\begin{aligned}\|C - AA^T\|_E &\leqslant 2^{-t}\|AA^T\|_E + \tfrac{3}{2}2^{-2t_2}n\|A\|_E^2 \\ &\leqslant 2^{-t}\|AA^T\|_E + \tfrac{3}{2}2^{-2t_2}n^{3/2}\|AA^T\|_E \\ &= 2^{-t}\|AA^T\|_E(1 + \tfrac{3}{2}2^{-t_2}n^{3/2}). \end{aligned} \tag{4.13}$$

Hence unless n is very large we have effectively

$$\frac{\|C - AA^T\|_E}{\|AA^T\|_E} \leqslant 2^{-t} \tag{4.14}$$

without any proviso about cancellation.

5. We shall define a matrix A as being an infinity row standardized single-precision block-floating matrix if we have $A = 2^a B$ where B has each component expressed as a fixed-point number and

$$\tfrac{1}{2} \leqslant \|B\|_\infty \leqslant 1. \tag{5.1}$$

We shall use the abbreviation i.s.s.p.b.f. matrix.

If P and Q are two i.s.s.p.b.f. matrices such that

$$P = 2^p R; \qquad Q = 2^q S, \tag{5.2}$$

then the components of RS may be accumulated exactly without overspill to give a non-standardized double-precision matrix. This exact product may then be reduced to an i.s.s.p.b.f. matrix by multiplying each component by the same power of 2 and rounding to t places. We have

$$\begin{aligned} T &\equiv \text{b.f.}\,(P \times Q) \\ &\equiv PQ + E \qquad (5.3) \\ \|E\|_\infty &\leqslant n2^{-t}\|PQ\|_\infty \qquad (5.4) \end{aligned}$$

where PQ has n columns.

Similarly if A is an i.s.s.p.b.f. matrix and b is a standardized s.p.b.f. vector then

$$\begin{aligned} c &\equiv \text{b.f.}\,(A \times b) \\ &\equiv Ab + e \qquad (5.5) \\ \|e\|_\infty &\leqslant 2^{-t}\|Ab\|_\infty. \qquad (5.6) \end{aligned}$$

MATRICES WHICH ARE NOT INFINITY ROW STANDARDIZED

6. Sometimes we shall work with standardized s.p.b.f. matrices of the form $2^a B$ where B satisfies

$$\tfrac{1}{2} \leqslant \max |b_{ij}| < 1 \tag{6.1}$$

rather than (5.1). If b is a standardized s.p.b.f. vector of the form $2^c d$ where

$$\tfrac{1}{2} \leqslant \|d\|_\infty < 1 \tag{6.2}$$

then it may not be possible to compute Bd exactly as a d.p.b.f. vector without overspill. If it *is* known that overspill will not occur then we *can* form Bd exactly and then standardize it and round it. The computed vector will then satisfy (5.5) and (5.6). This situation often exists in the solution of linear equations when residuals are being computed.

Otherwise it may be necessary to accumulate $2^{-k}Bd$ (where k is chosen so that $2^k > n \geqslant 2^{k-1}$) rounding off each contribution to each component. We therefore obtain $2^{-k}Bd + f$ where

$$\|f\|_\infty \leqslant \tfrac{1}{2}n2^{-2t}. \tag{6.3}$$

Writing this result in the form $2^{-k}(Bd + 2^k f)$ we see that the relative error ρ, before standardization is given by

$$\rho = \frac{\|2^k f\|_\infty}{\|Bd\|_\infty} \leqslant \frac{n^2 2^{-2t}}{\|Bd\|_\infty} \quad \text{(since } 2^k \leqslant 2n\text{)}. \tag{6.4}$$

Provided $\|Bd\|_\infty \geqslant n^2 2^{-t}$ the error at this stage is less than 1 part in 2^t. Standardization then introduces a further error that is bounded by 1 part in 2^t. Hence the final computed s.p.b.f. vector will have a relative error which is almost as small as we had before.

As $\|Bd\|$ decreases past $n^2 2^{-t}$ the maximum relative error increases, and since we may have $Bd = 0$, it can become arbitrarily large. Note however that when $\|Bd\|_\infty$ is less than 2^{-t} no error is made in the standardization and the infinity norm of the absolute error, $2^{a+c+k}\|f\|_\infty$ remains bounded by $n^2 2^{-2t} \times 2^{a+c}$.

We shall frequently produce a non-standardized d.p.b.f. vector v which we then round to a s.p.b.f. vector $\bar{v}$. If the first t digits in each component are zero then no errors are introduced when standardizing. Otherwise we have

$$\left.\begin{aligned} \bar{v} &= v + e \\ \|e\|_\infty &\leqslant 2^{-t}\|v\|_\infty . \end{aligned}\right\} \tag{6.5}$$

In all cases we have

$$1 - 2^{-t} \leqslant \frac{\|\bar{v}\|_\infty}{\|v\|_\infty} \leqslant 1 + 2^{-t}. \tag{6.6}$$

Similar considerations hold when we consider the addition of two standardized s.p.b.f. vectors a and b. The computed sum c satisfies

$$\begin{aligned} c &\equiv \text{b.f.}(a+b) \\ &\equiv a + b + e. \end{aligned} \tag{6.7}$$

$$\|e\|_\infty \leqslant 2^{-t}\max(\|a\|_\infty, \|b\|_\infty) \text{ (if no overspill)} \tag{6.8}$$

$$\|e\|_\infty \leqslant 2^{-t}(\|a\|_\infty + \|b\|_\infty) \text{ (in all cases)}. \tag{6.9}$$

If we are permitted to form a d.p.b.f. sum as an intermediate stage and then standardize and round, we have

$$\|e\| \leqslant 2^{-t}\|a + b\|_\infty \tag{6.10}$$

and the result always has a low relative error even when cancellation takes place.

ORTHOGONALIZATION OF VECTORS

7. Given two independent vectors x and y, then a vector z may be constructed which lies in the space spanned by x and y and is orthogonal to x. If we write

$$z = y - \alpha x \tag{7.1}$$

then

$$0 = x^T z = x^T y - \alpha x^T x \tag{7.2}$$

$$\alpha = x^T y / x^T x. \tag{7.3}$$

The division is always possible since x^Tx is positive. If x and y are truly independent then z is not null.

In practice the situation is not quite as simple as this. By way of illustration we consider single-precision block-floating decimal computation and we assume that both x and y are standardized vectors having largest components of absolute magnitude lying between $0\cdot1$ and $1\cdot0$. Now if y is not an exact multiple of x the two vectors are certainly *mathematically* independent, yet if we apply the above process, the computed z may be quite unsatisfactory.

When working with t figures we would regard x and z as orthogonal if the angle θ between them is such that

$$\theta = \frac{\pi}{2} + \epsilon; \qquad \epsilon = O(2^{-t}). \tag{7.4}$$

The precise meaning of the second of the equations (7.4) has been left a little vague. This is intentional because the permissable range of ϵ will depend to some extent on the nature of the computation which is being used.

The unsatisfactory nature of the computed z arises from the fact that, whether we use floating-point or fixed-point computation, the practical realization of (7.1) to (7.3) ensures only that

$$x^Tz = O(2^{-t}). \tag{7.5}$$

From (2.6) we see that this will not imply that z is almost orthogonal to x *unless* $\|z\|_2$ *is of order unity.*

NUMERICAL EXAMPLE

8. A simple example computed in s.p.b.f. will make this point clear. We take

$$x = \begin{bmatrix} 0\cdot213625 \\ 0\cdot314317 \\ 0\cdot412135 \end{bmatrix}; \quad y = \begin{bmatrix} 0\cdot174681 \\ 0\cdot257023 \\ 0\cdot336951 \end{bmatrix}. \tag{8.1}$$

We have

$$fi(x^Ty) = 0\cdot256972; \qquad fi(x^Tx) = 0\cdot314286, \tag{8.2}$$

where inner-products were accumulated, giving

$$\alpha = 0\cdot817637. \tag{8.3}$$

The corresponding computed z is given by

$$\begin{aligned} z &= \text{b.f.}(y - \alpha z) \\ &= 10^{-4}\begin{bmatrix} 0\cdot132959 \\ 0\cdot257911 \\ -0\cdot258250 \end{bmatrix} \end{aligned} \tag{8.4}$$

and we have

$$fi(x^T z) = 10^{-6}(0\cdot 303531) \tag{8.5}$$

verifying (7.5). On the other hand we have

$$\cos\theta = 10^{-1}(0\cdot 139\ldots); \qquad \theta = \frac{\pi}{2} - \epsilon \tag{8.6}$$

$$\epsilon = O(10^{-2}) \tag{8.7}$$

so that x and z are not orthogonal to working accuracy.

9. It is natural to ask 'what can be done?' about this situation. It must be appreciated that unless x and y are 'exact' vectors they do not accurately define a two-space. The vector y is quite close to a multiple of x. If the components of both x and y are rounded values, the direction orthogonal to x in the two-space is not well determined; alterations of up to $\frac{1}{2}10^{-6}$ in the components of x and y make angular changes of order 10^{-2} in the direction orthogonal to x.

If x and y are exact vectors we can obviously determine the true vector z as accurately as we please by working to a sufficiently high precision. In the example of section 8 there is a loss of 4 decimals as a result of cancellation, and if we wish to compute a z which is orthogonal to x to k significant figures we must be prepared to work to $k+4$ figures. Note that this loss of 4 figures is not known *a priori*. The case when x and y are exact is not very subtle and is of little practical importance.

We continue now with the case when x and y have rounded components. Suppose we take the computed z obtained as in (8.4) and orthogonalize it again with respect to x. It will be convenient to rename z and α of section 8 and call them z_1 and α_1, so that

$$z_1 = \text{b.f.}\,(y - \alpha_1 x). \tag{9.1}$$

The reorthogonalized vector z_2 will be obtained from the relation

$$z_2 = \text{b.f.}\,(z_1 - \alpha_2 x) \tag{9.2}$$

where the computation is again performed in block-floating arithmetic. We have

$$\alpha_2 = \text{b.f.}\left(\frac{x^T z_1}{x^T x}\right). \tag{9.3}$$

Now our previous computation has already ensured that $x^T z_1 = O(10^{-t})$ and hence α_2 will itself be of order 10^{-t}. For our numerical example we have from (8.2) and (8.5)

$$\alpha_2 = 10^{-6}(0\cdot 965780) \tag{9.4}$$

$$z_2 = 10^{-4}\begin{bmatrix} 0\cdot 130896 \\ 0\cdot 254875 \\ -0\cdot 262230 \end{bmatrix} \tag{9.5}$$

and

$$fi(x^T z_2) = 10^{-11}(0\cdot 42325). \tag{9.6}$$

The vector z_2 is indeed truly orthogonal to x to working accuracy. If we denote the angle between x and z_2 by θ_2 then it will readily be verified that

$$\theta_2 = \frac{\pi}{2} + \epsilon_2; \qquad \epsilon_2 = O(10^{-6}). \tag{9.7}$$

Note that the components of z_1 and z_2 differ only by quantities of order 10^{-6} as could have been forecast by a trivial error analysis. However, this amounts to a relative correction of about 1 part in 100. It is convenient to introduce the normalized vectors w_1 and w_2 corresponding to z_1 and z_2. We have

$$w_1 = \begin{bmatrix} 0{\cdot}132959 \\ 0{\cdot}257911 \\ -0{\cdot}258250 \end{bmatrix}; \qquad w_2 = \begin{bmatrix} 0{\cdot}130896 \\ 0{\cdot}254875 \\ -0{\cdot}262230 \end{bmatrix}. \tag{9.8}$$

no rounding errors being involved in this 'normalization'.

Now the vectors z_2 and w_2 are by no means the vectors which would have been obtained by doing the orthogonalization process accurately, though the accurate w does in fact agree with both w_1 and w_2 in the first two decimals. Since x and y are not exact, there is in any case no virtue in obtaining the 'true' w. For the true w corresponding to the given x and y differs from the true w corresponding to the exact (but unknown) x and y in the third significant decimal.

It might well be asked whether the computed w_2 has any advantages at all over the computed w_1. This question cannot be answered except in the context of the computation in which the w is to be used. The facts about w_2 are the following:

(i) It is a normalized vector.

(ii) It is orthogonal to x 'to working accuracy'.

(iii) It satisfies the relation

$$10^{-4}w_2 = z_2 = y - \alpha_1 x + e \tag{9.9}$$

$$e_i = O(10^{-6}). \tag{9.10}$$

In (9.9) we have used α_1 and not $\alpha_1+\alpha_2$, which might have seemed more natural. However α_2 is smaller than α_1 by a factor of order 10^{-6} and hence $\alpha_2 x$ may be included in the error vector e. The number $\alpha_1 + \alpha_2$ is inconvenient to use since it requires 12 decimals for its representation.

GENERAL CASE

10. The analysis we have given is quite general. If x and y are normalized s.p.b.f. vectors in t-digit binary arithmetic, and the computed s.p.b.f. vector z_1 has an exponent of $-k$ $(k < t)$ then the angle θ_1, between x and z_1, will satisfy

$$\theta_1 = \frac{\pi}{2} + \epsilon_1; \qquad \epsilon_1 = O(2^{k-t}). \tag{10.1}$$

The components of the vector z_2 will differ from z_1 by quantities of order 2^{-t} but z_2 will make an angle θ_2 with x satisfying

$$\theta_2 = \frac{\pi}{2} + \epsilon_2; \qquad \epsilon_2 = O(2^{-t}) \tag{10.2}$$

and we shall have

$$z_2 = 2^{-k} w_2 = y - \alpha_1 x + e \tag{10.3}$$

$$e_i = O(2^{-t}). \tag{10.4}$$

If $k \geqslant t$ then all figures in z_1 are spurious. In this case if we continue with z_1 and derive z_2 it is just conceivable that there will be a further loss of figures, so that z_2 itself will require orthogonalization with respect to x. In practice the need for more than two reorthogonalization stages is so improbable as to be scarcely worthy of consideration. In any case, if $k \geqslant t$ we may take w to be any normalized s.p.b.f. vector which is orthogonal to x to working accuracy. Such a vector will satisfy

$$2^{-t} w = y - \alpha_1 x + e \tag{10.5}$$

$$e_i = O(2^{-t}). \tag{10.6}$$

In practice we have to decide whether an orthogonal vector w, satisfying (10.3) and (10.4) (or (10.5) and (10.6)) is of relevance to our computing problem. In many contexts the answer will be that it is; this may seem surprising since the computed orthogonal vector will differ in its last k significant figures from the vector which would be obtained from x and y using exact computation. If $k \geqslant t$ the computed vector will not agree with the exact vector at all.

It would be wrong to think of the need for reorthogonalization as being the result of the 'cumulative' effect of rounding errors. Indeed, in block-floating work there is virtually no accumulation of errors. The quantities $x^T x$ and $x^T y$ are computed exactly and then rounded, so that the computed α_1, which is in standard floating-point form, will have a relative error of at most 3 parts in 2^t. Only one further rounding error is made in computing each component of z_1 and if cancellation occurs in all components this error is bounded by $\frac{1}{2}2^{-t-k}$ where the maximum element of z is between 2^{-k} and 2^{-k-1}.

Much the same result applies to computation in standard floating-point. Here accumulation of rounding errors can play a slightly more important role, but its effect is still overshadowed by that of cancellation. *When no cancellation occurs, reorthogonalization is pointless since the rounding errors made in the reorthogonalization merely reinforce those made in the original orthogonalization.*

These results may seem trivial when presented in connexion with a simple example. However, a failure to distinguish clearly between the effect of the accumulation of rounding errors and that of cancellation, and to

appreciate the full significance of reorthogonalization, was for a long time the main obstacle to the understanding of Lanczos' method for the eigenvalue problem [15].

We have considered the case of two vectors only. By a similar process we may generate a set of n orthogonal vectors $z_1, z_2, \ldots, z_n$, from n independent vectors $x_1, x_2, \ldots, x_n$ where

$$z_1 = x_1$$

$$z_r = x_r - \alpha_{r1}z_1 - \alpha_{r2}z_2 - \ldots - \alpha_{r,r-1}z_{r-1}.$$

Here again we may reorthogonalize each z_r with respect to all previous z_i before proceeding to the computation of z_{r+1}.

SOLUTION OF EQUATIONS AND MATRIX INVERSION

11. We consider first the sensitivity of the solution of the set of equations

$$Ax = b \tag{11.1}$$

to variations in A and b. As far as the variation in b is concerned, the problem is trivial. If

$$A(x+h) = b + k \tag{11.2}$$

then

$$Ah = k \tag{11.3}$$

$$h = A^{-1}k \tag{11.4}$$

giving

$$\|h\| = \|A^{-1}k\| \leqslant \|A^{-1}\|\,\|k\|$$

$$\|h\|/\|k\| \leqslant \|A^{-1}\|. \tag{11.5}$$

For the relative change $\|h\|/\|x\|$ we proceed as follows. From (11.1) we have

$$\|b\| = \|Ax\| \leqslant \|A\|\,\|x\| \tag{11.6}$$

$$\|x\| \geqslant \|b\|\,\|A\|^{-1}. \tag{11.7}$$

Hence

$$\frac{\|h\|}{\|x\|} \leqslant \frac{\|A^{-1}\|\,\|k\|}{\|A\|^{-1}\|b\|} = \frac{\|A\|\,\|A^{-1}\|\,\|k\|}{\|b\|} \tag{11.8}$$

showing that $\|A\|\,\|A^{-1}\|$ may be regarded as a condition number. The condition number in most common use is $\|A\|_2\,\|A^{-1}\|_2$; this is usually denoted by $\kappa(A)$ and is called the *spectral condition number*.

When $\|A\|\,\|A^{-1}\|$ is very large this result will be very pessimistic for most right-hand sides b and perturbations h. Most b will be such that

$$\|x\| \gg \|b\|\,\|A\|^{-1}.$$

However there are always some b and k for which (11.8) is realistic.

12. Turning now to the effect of changes in A we consider

$$(A+E)(x+h) = b \tag{12.1}$$

giving

$$(A+E)h = -Ex. \tag{12.2}$$

Even if A is not singular (and we naturally assume this) $A+E$ may be singular if E is not restricted. Writing

$$A+E = A(I+A^{-1}E) \tag{12.3}$$

it is evident that $A+E$ is non-singular provided $I+A^{-1}E$ is non-singular. Now if λ_i is any eigenvalue of $A^{-1}E$ we have from (2.23)

$$\|A^{-1}E\| \geqslant \lambda_i, \tag{12.4}$$

and since the eigenvalues of $I + A^{-1}E$ are $1+\lambda_i$, this matrix is non-singular if

$$\|A^{-1}E\| < 1. \tag{12.5}$$

Assuming this condition to be satisfied and writing

$$A^{-1}E = F \tag{12.6}$$

we have

$$\begin{aligned} h &= -(A+E)^{-1}Ex \\ &= -(I+F)^{-1}A^{-1}Ex. \end{aligned} \tag{12.7}$$

Writing

$$(I+F)^{-1} = G, \tag{12.8}$$

$$I = G + FG \tag{12.9}$$

and

$$1 \geqslant \|G\| - \|F\|\,\|G\| \tag{12.10}$$

for any of our norms except the Euclidean. Hence, since $\|F\| < 1$,

$$\|G\| \leqslant \frac{1}{1-\|F\|}. \tag{12.11}$$

Equation (12.7) now gives

$$h = -GA^{-1}Ex \tag{12.12}$$

$$\begin{aligned} \|h\| &\leqslant \frac{\|A^{-1}E\|\,\|x\|}{1-\|A^{-1}E\|} \\ &\leqslant \frac{\|A^{-1}\|\,\|E\|\,\|x\|}{1-\|A^{-1}\|\,\|E\|} \end{aligned} \tag{12.13}$$

provided

$$\|A^{-1}\|\,\|E\| < 1. \tag{12.14}$$

Our proof does not apply to the Euclidean norm, since we took $\|I\| = 1$ in (12.10), but since the result is true for the 2-norm it is true *a fortiori* for the Euclidean.

Relation (12.13) may be written in the form

$$\frac{\|h\|}{\|x\|} \leqslant \frac{\|A\|\,\|A^{-1}\|\dfrac{\|E\|}{\|A\|}}{1 - \|A\|\,\|A^{-1}\|\dfrac{\|E\|}{\|A\|}}. \tag{12.15}$$

This expresses the relative error in x in terms of the relative error $\|E\|/\|A\|$ in A. Again we see that $\|A\|\,\|A^{-1}\|$ is the decisive quantity.

ROUNDING OF MATRIX OF COEFFICIENTS

13. We can apply the result of section 12 to determine the effect made by rounding the coefficients of A to t significant digits. We have then

$$|e_{ij}| \leqslant 2^{-t}|a_{ij}| \tag{13.1}$$

$$\|E\|_E \leqslant 2^{-t}\|A\|_E. \tag{13.2}$$

Hence from (12.14)

$$\frac{\|h\|_2}{\|x\|_2} \leqslant \frac{2^{-t}\|A\|_E\,\|A^{-1}\|_E}{1 - 2^{-t}\|A\|_E\,\|A^{-1}\|_E}. \tag{13.3}$$

Unless $2^{-t}\|A\|_E\,\|A^{-1}\|_E$ is appreciably less than unity we cannot guarantee that $\|h\|_2/\|x\|_2$ is at all small.

In terms of the 2-norm we have from (2.20)

$$\|E\|_2 \leqslant 2^{-t}n^{1/2}\|A\|_2 \tag{13.4}$$

giving

$$\frac{\|h\|_2}{\|x\|_2} \leqslant \frac{2^{-t}\kappa(A)n^{1/2}}{1 - 2^{-t}\kappa(A)n^{1/2}}. \tag{13.5}$$

We must now have

$$2^{-t}\kappa(A)n^{1/2} \ll 1 \tag{13.6}$$

if the error h is to be appreciably less than x.

This result would suggest that when using t-digit floating-point arithmetic we will not *in general* be able to obtain even an approximate solution to a set of equations for which $\kappa(A) \geqslant n^{-1/2}2^t$ and this is indeed true. In fact it is by no means easy to 'recognise' an accurate inverse if κ exceeds this bound.

Suppose X is the matrix obtained by rounding the exact inverse A^{-1} to t significant figures. Then as in (13.2) and (13.4)

$$X = A^{-1} + E \tag{13.7}$$

where

$$\left.\begin{aligned} \|E\|_E &\leqslant 2^{-t}\|A^{-1}\|_E \\ \|E\|_2 &\leqslant 2^{-t}n^{1/2}\|A^{-1}\|_2. \end{aligned}\right\} \tag{13.8}$$

Hence

$$\begin{aligned} AX &= A(A^{-1} + E) \\ &= I + AE \end{aligned} \tag{13.9}$$

and we have, for example,

$$\begin{aligned} \|AE\|_2 &\leqslant \|A\|_2 \|E\|_2 \\ &\leqslant \|A\|_2 2^{-t} n^{1/2} \|A^{-1}\|_2 \\ &= 2^{-t} n^{1/2} \kappa(A). \end{aligned} \tag{13.10}$$

If $\kappa(A) \geqslant n^{-1/2} 2^t$, then AE may have a norm which is greater than unity and AX will not even be approximately equal to I in spite of the fact that X is the 'correct' inverse to t figures.

ERROR ANALYSIS OF GAUSSIAN ELIMINATION

14. We have already given a very detailed error analysis of Gaussian elimination in [**30**] and we do not wish to repeat the discussion in full. However, there are a number of practical consequences of the results which we obtained which are not well known, and in order to give a self-contained presentation we give a brief account of the earlier analysis.

The solution of a set of equations and the inversion of a matrix by Gaussian elimination are based on the triangularization of a matrix. If we denote the original set of equations by

$$A^{(1)}x = b^{(1)} \tag{14.1}$$

then $n-1$ equivalent sets

$$A^{(r)}x = b^{(r)} \quad (r = 2, \ldots, n) \tag{14.2}$$

are produced, the matrix $A^{(n)}$ of the final set being of upper triangular form. The form of the matrix $A^{(r)}$ is sufficiently illustrated by displaying $A^{(3)}$ when $n = 5$.

$$A^{(3)} = \begin{bmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{bmatrix}. \tag{14.3}$$

In general $A^{(r)}$ is already of triangular form as regards its first r rows and it has a square matrix of non-zero elements in the bottom right-hand corner. This square matrix is of order $n+1-r$. The matrix $A^{(r+1)}$ is derived from $A^{(r)}$ by subtracting a multiple m_{ir} of the rth row from the ith row for values of i from $r+1$ to n. The multipliers m_{ir} are defined by

$$m_{ir} = a_{ir}^{(r)} / a_{rr}^{(r)}. \tag{14.4}$$

The rth row of $A^{(r)}$ is called the rth *pivotal row*, and $a_{rr}^{(r)}$ is called the rth *pivot*.

We ignore the operations on the right-hand side $b^{(r)}$ for the moment, since the production of $A^{(n)}$ is of value in a wider context than the solution of equations. It is the essential process involved in the evaluation of $\det(A^{(1)})$ and of $(A^{(1)})^{-1}$.

15. We give first an analysis which is appropriate for any form of computation, fixed-point or floating-point. The history of the (i, j) element differs according as $i \leqslant j$ or $i > j$.

(i) $i \leqslant j$

The element is modified in each transformation until $A^{(i)}$ is obtained, after which it remains constant. We may write

$$\left.\begin{aligned} a_{ij}^{(2)} &\equiv a_{ij}^{(1)} - m_{i1}\, a_{1j}^{(1)} + \epsilon_{ij}^{(2)} \\ a_{ij}^{(3)} &\equiv a_{ij}^{(2)} - m_{i2}\, a_{2j}^{(2)} + \epsilon_{ij}^{(3)} \\ &- - - - - - - - - - - - - \\ a_{ij}^{(i)} &\equiv a_{ij}^{(i-1)} - m_{i,\,i-1}\, a_{i-1,j}^{(i-1)} + \epsilon_{ij}^{(i)} \end{aligned}\right\} \tag{15.1}$$

where all $a_{ij}^{(k)}$ and m_{ik} refer to computed values, and $\epsilon_{ij}^{(k)}$ is the difference between the accepted $a_{ij}^{(k)}$ and the exact value which would be obtained using the computed $a_{ij}^{(k-1)}$, $m_{i,\,k-1}$ and $a_{k-1,j}^{(k-1)}$. The bounds for the $\epsilon_{ij}^{(k)}$ depend on the type of arithmetic. Summing equations (15.1) we have

$$a_{ij}^{(i)} \equiv a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)} - m_{i2}a_{2j}^{(2)} - \ldots - m_{i,\,i-1}a_{i-1,j}^{(i-1)} + e_{ij} \tag{15.2}$$

where

$$e_{ij} = \epsilon_{ij}^{(2)} + \epsilon_{ij}^{(3)} + \ldots + \epsilon_{ij}^{(i)}. \tag{15.3}$$

Note that no products of $\epsilon_{ij}^{(k)}$ arise and that only elements of pivotal rows, m_{ik}'s and an element of the original matrix occur in (15.2).

(ii) $i > j$

The element is modified as in (i) until $A^{(j)}$ is obtained; $a_{ij}^{(j)}$ is then used to compute m_{ij}, and $a_{ij}^{(j+1)}$ to $a_{ij}^{(n)}$ are all taken to be exactly equal to zero. The computed multiplier m_{ij} satisfies

$$m_{ij} = \frac{a_{ij}^{(j)}}{a_{jj}^{(j)}} + \eta_{ij} \tag{15.4}$$

where η_{ij} is the rounding error in the division. Equation (15.4) may be written in the form

$$0 = a_{ij}^{(j)} - m_{ij}a_{jj}^{(j)} + \epsilon_{ij}^{(j+1)} \tag{15.5}$$

where

$$\epsilon_{ij}^{(j+1)} = a_{jj}^{(j)}\eta_{ij}. \tag{15.6}$$

The equations are therefore

$$\left.\begin{aligned} a_{ij}^{(2)} &\equiv a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)} + \epsilon_{ij}^{(2)} \\ a_{ij}^{(3)} &\equiv a_{ij}^{(2)} - m_{i2}a_{2j}^{(2)} + \epsilon_{ij}^{(3)} \\ &- - - - - - - - - - - - \\ a_{jj}^{(j)} &\equiv a_{ij}^{(j-1)} - m_{i,\,j-1}a_{j-1,j}^{(j-1)} + \epsilon_{ij}^{(j)} \\ 0 &\equiv a_{ij}^{(j)} - m_{ij}a_{jj}^{(j)} + \epsilon_{ij}^{(j+1)}. \end{aligned}\right\} \tag{15.7}$$

Summing, we have

$$0 \equiv a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)} - m_{i2}a_{2j}^{(2)} - \ldots - m_{ij}a_{jj}^{(j)} + e_{ij} \tag{15.8}$$

where

$$e_{ij} = \epsilon_{ij}^{(2)} + \epsilon_{ij}^{(3)} + \ldots + \epsilon_{ij}^{(j+1)}. \tag{15.9}$$

If we take the terms involving m_{ik} to the left-hand side in equations (15.2) and (15.8) the set of n^2 equations are the equivalent of the single matrix equation

$$LU \equiv A^{(1)} + E \tag{15.10}$$

where L is the lower triangular matrix defined, typically when $n = 4$, by

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{21} & 1 & 0 & 0 \\ m_{31} & m_{32} & 1 & 0 \\ m_{41} & m_{42} & m_{43} & 1 \end{bmatrix} \tag{15.11}$$

and U is the upper triangular matrix defined by

$$U = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & a_{34}^{(3)} \\ 0 & 0 & 0 & a_{44}^{(4)} \end{bmatrix}. \tag{15.12}$$

The (i, j) element of E is defined by (15.3) or (15.9) according as $i \leqslant j$ or $i > j$. If $i \leqslant j$ it is given by the sum of $i-1$ rounding errors and if $i > j$ it is given by the sum of j rounding errors of which the jth comes from the division in which m_{ij} is computed.

FLOATING-POINT BOUNDS

16. Our result shows that the computed L and U correspond to the exact triangularization of $A+E$. We wish therefore to obtain bounds for E. Now in floating-point, the computed $a_{ij}^{(k)}$ is defined by

$$\begin{aligned} a_{ij}^{(k)} &\equiv fl(a_{ij}^{(k-1)} - m_{i,\,k-1}a_{k-1,j}^{(k-1)}) \\ &\equiv [a_{ij}^{(k-1)} - m_{i,\,k-1}a_{k-1,j}^{(k-1)}(1+\epsilon_1)](1+\epsilon_2). \end{aligned} \tag{16.1}$$

Hence

$$\begin{aligned} \epsilon_{ij}^{(k)} &= a_{ij}^{(k)} - (a_{ij}^{(k-1)} - m_{i,\,k-1}a_{k-1,j}^{(k-1)}) \\ &= a_{ij}^{(k)} - \left(\frac{a_{ij}^{(k)}}{1+\epsilon_2} + m_{i,\,k-1}a_{k-1,j}^{(k-1)}\epsilon_1\right) \\ &= \frac{a_{ij}^{(k)}\epsilon_2}{1+\epsilon_2} - m_{i,\,k-1}a_{k-1,j}^{(k-1)}\epsilon_1, \end{aligned} \tag{16.2}$$

the result holding for all relevant $\epsilon_{ij}^{(k)}$ except $\epsilon_{ij}^{(j+1)}$. It is clear that if we are to obtain a satisfactory bound for the $\epsilon_{ij}^{(k)}$ we will need good bounds for the m_{ik} and the $a_{ij}^{(k)}$.

If the computation is carried out exactly as described, the multipliers may be almost arbitrarily large. Indeed if an $a_{jj}^{(j)}$ is zero the process breaks down, as can be seen from (15.4). In practice it is usual when dealing with *general* matrices to select the pivotal rows (and sometimes the pivots) so as to ensure that

$$|m_{ij}| \leqslant 1. \tag{16.3}$$

There are two main ways in which this is done.

(i) The columns may be eliminated in the natural order, but at the rth stage, the pivotal row is taken to be that one of the remaining $n+1-r$ rows which has the element of largest modulus in column r. This we have called *partial* pivoting.

(ii) At the rth stage we may select as pivot the element of largest modulus in the *whole* of the remaining $n+1-r$ square array. This we have called *complete* pivoting.

If we use partial pivoting, then it is immediately evident that if all elements of $A^{(1)}$ satisfy

$$|a_{ij}^{(1)}| \leqslant a, \tag{16.4}$$

then all elements of $A^{(r)}$ satisfy

$$|a_{ij}^{(r)}| \leqslant 2^{r-1}a. \tag{16.5}$$

There are rather special matrices for which the upper bound is attained [**30**], so that the final pivot is $2^{n-1}a$, but in practice such growth is very rare. It is very uncommon for any element to attain a value as great as $8a$ and if the matrix is at all ill-conditioned, it is far more likely that the elements of successive $A^{(k)}$ will decrease. Moreover there are several important classes of matrices for which it is known *a priori* that the growth is restricted to a much more modest factor [**30**].

If we use complete pivoting, then it has been shown [**30**] that

$$|a_{rr}^{(r)}| < r^{1/2}\left(2^1 3^{1/2} 4^{1/3} \ldots r^{\frac{1}{r-1}}\right)^{1/2} a \tag{16.6}$$

so that the maximum growth is far more severely restricted than in (16.5). The proof of (16.6) shows that the bound cannot be attained and suggests that it is a severe overestimate. No matrix has yet been discovered for which $|a_{rr}^{(r)}| > ra$.

We shall leave this point for the moment, and merely assume that some form of pivoting has been used, so that

$$|m_{ij}| \leqslant 1. \tag{16.7}$$

We shall denote the maximum element in any $|A^{(r)}|$ by g. There is no real loss of generality in assuming

$$|a_{ij}^{(i)}| \leqslant 1 \tag{16.8}$$

since this may be achieved by scaling (without rounding error). Whatever form of pivoting is used, (15.10) is true provided $A^{(1)}$ is used to denote the original matrix with its rows (and columns, if we are using complete pivoting) suitably permuted.

From (16.2) we have

$$|\epsilon_{ij}^{(k)}| \leqslant \frac{g2^{-t}}{1-2^{-t}} + g2^{-t}$$

$$< (2{\cdot}01)g2^{-t}\,(\text{say}). \tag{16.9}$$

This applies to all $\epsilon_{ij}^{(k)}$ except $\epsilon_{ij}^{(j+1)}$ $(i > j)$. For these we must use (15.4) and (15.6). We have

$$m_{ij} = fl\left(\frac{a_{ij}^{(j)}}{a_{jj}^{(j)}}\right) = \frac{a_{ij}^{(j)}}{a_{jj}^{(j)}}(1+\epsilon) \tag{16.10}$$

$$\eta_{ij} = \frac{a_{ij}^{(j)}}{a_{jj}^{(j)}}\,\epsilon\,. \tag{16.11}$$

$$|\epsilon_{ij}^{(j+1)}| = |a_{jj}^{(j)}\frac{a_{ij}^{(j)}}{a_{jj}^{(j)}}\epsilon| = |a_{ij}^{(j)}\epsilon| \leqslant g2^{-t} < (2{\cdot}01)g2^{-t} \tag{16.12}$$

so that we need not give this element special treatment.
Combining (15.3), (15.9), (16.9) and (16.12), we have

$$|E| \leqslant (2{\cdot}01)g2^{-t}\begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ 1 & 1 & 1 & \ldots & 1 & 1 \\ 1 & 2 & 2 & \ldots & 2 & 2 \\ 1 & 2 & 3 & \ldots & 3 & 3 \\ \hline 1 & 2 & 3 & \ldots & (n-1) & (n-1) \end{bmatrix}. \tag{16.13}$$

It is our contention that in practice, using pivoting, g is almost invariably of order unity and is quite unimportant. If the sizes of the elements of successive $A^{(k)}$ show a downward trend, the bound for E will be much too great. If there is a loss of 1 binary digit per stage, for example, then

$$|\epsilon_{ij}^{(k)}| \leqslant (2{\cdot}01)2^{-t}2^{1-k} \tag{16.14}$$

and

$$|e_{ij}| \leqslant (2{\cdot}01)2^{-t}(1 + 2^{-1} + 2^{-2} + \ldots + 2^{1-j}) \leqslant (2{\cdot}01)2^{-t}2 \tag{16.15}$$

For very ill-conditioned matrices a loss of more than one binary digit per stage may well occur.

GAUSSIAN ELIMINATION IN FIXED-POINT

17. For fixed-point work some form of pivoting is essential in order to ensure that the $|m_{ij}|$ are bounded by unity. If the original matrix is scaled so that all $a_{ij}^{(k)}$ satisfy

$$|a_{ij}^{(k)}| \leqslant 1 \tag{17.1}$$

then all $\epsilon_{ij}^{(k)}$ satisfy

$$|\epsilon_{ij}^{(k)}| \leqslant \tfrac{1}{2}2^{-t}. \tag{17.2}$$

If we scale originally so that

$$|a_{ij}^{(1)}| \leqslant \tfrac{1}{8} \tag{17.3}$$

it is very uncommon for the condition (17.1) to be violated. The bound for $|E|$ is then as in (16.10) except that the factor $(2\cdot 01)g2^{-t}$ is replaced by $\frac{1}{2}2^{-t}$. Note that if the sizes of the elements of successive $A^{(k)}$ show a downward trend we do not have a gain of the type discussed for floating-point at the end of the last section.

DETERMINANT EVALUATION

18. The determinant of $A^{(1)}+E$ is equal to that of $A^{(n)}$ (apart from the sign, which depends on the number of interchanges). We have therefore

$$|\det(A^{(1)}+E)| = |a_{11}^{(1)}a_{22}^{(2)} \ldots a_{nn}^{(n)}|. \tag{18.1}$$

Floating-point computation is inevitable at this stage and we have for the computed determinant

$$\begin{aligned} |\det| &= |fl(a_{11}^{(1)}a_{22}^{(2)} \ldots a_{nn}^{(n)})| \\ &= |a_{11}^{(1)}a_{22}^{(2)} \ldots a_{nn}^{(n)}|(1+\epsilon) \qquad (18.2) \\ (1-2^{-t})^{n-1} &\leqslant 1+\epsilon \leqslant (1+2^{-t})^{n-1}. \qquad (18.3) \end{aligned}$$

The factor $1+\epsilon$ is comparatively unimportant and apart from this, the computed determinant is the exact determinant of $A^{(1)}+E$. If the eigenvalues of $A^{(1)}$ are $\lambda_1, \lambda_2, \ldots, \lambda_n$ and those of $A^{(1)}+E$ are $\lambda_1', \lambda_2', \ldots, \lambda_n'$ then

$$\text{exact } \det(A^{(1)}) = \Pi\lambda_i \tag{18.4}$$

$$\text{computed } \det(A^{(1)}) = (1+\epsilon)\Pi\lambda_i'. \tag{18.5}$$

This result is important in the determination of the eigenvalues of a matrix A from computed values of $\det(A-\lambda I)$.

SOLUTION OF A TRIANGULAR SET OF EQUATIONS USING STANDARD FLOATING-POINT ARITHMETIC

19. The Gaussian algorithm expresses the set of equations

$$Ax = b \tag{19.1}$$

in the form

$$LUx = b \tag{19.2}$$

where L is unit lower triangular and U is upper triangular. The computed L and U satisfy $LU = A + E$ and hence if we solve (19.2) without further rounding error we obtain the solution of the system

$$(A+E)x = b. \tag{19.3}$$

Equation (19.2) may be solved in the two steps

$$Ly = b \tag{19.4}$$

$$Ux = y, \tag{19.5}$$

each of which requires the solution of a set of equations having a triangular matrix of coefficients. We therefore consider the errors made in solving such sets of equations. The fact that L is *unit*-triangular is not of much importance and the analysis is essentially the same for both upper and lower triangles. We consider therefore the solution of a lower triangular set, where the matrix of coefficients does not necessarily have a unit diagonal.

We write the equations in the form

$$\left.\begin{aligned} l_{11}x_1 &= b_1 \\ l_{21}x_1 + l_{22}x_2 &= b_2 \\ l_{31}x_1 + l_{32}x_2 + l_{33}x_3 &= b_3 \\ - - - - - - - - - - - - - - - - - \\ l_{n1}x_1 + l_{n2}x_2 + l_{n3}x_3 + \ldots + l_{nn}x_n &= b_n. \end{aligned}\right\} \tag{19.6}$$

The variables x_1, x_2, ... are determined in succession from equations 1, 2 ... respectively. At the time when x_r is computed from the rth equation, $x_1, x_2, \ldots, x_{r-1}$ have already been computed.

In floating-point we have

$$x_r \equiv fl\left(\frac{-l_{r1}x_1 - l_{r2}x_2 - \ldots - l_{r,r-1}x_{r-1} + b_r}{l_{rr}}\right) \tag{19.7}$$

giving

$$x_r \equiv \frac{-l_{r1}x_1(1+E_{r1}) - l_{r2}x_2(1+E_{r2}) - \ldots - l_{r,r-1}x_{r-1}(1+E_{r,r-1}) + b_r(1+\epsilon_r)}{l_{rr}(1+\eta_r)} \tag{19.8}$$

where certainly

$$|\eta_r| \leqslant 2^{-t_1} \qquad (|\epsilon_r| \leqslant 2^{-t_1}) \tag{19.9}$$

$$|E_{ri}| \leqslant (r+2-i)2^{-t_1}. \tag{19.10}$$

This follows from an obvious extension of the results of section 26 of Chapter 1, the last product term being replaced by the simple term b_r. The factor $1+\eta_r$ covers the error introduced in the division by b_{rr}.

It is more convenient to express (19.8) in the form

$$x_r \equiv \frac{-l_{r1}x_1\dfrac{1+E_{r1}}{1+\epsilon_r} - l_{r2}x_2\dfrac{1+E_{r2}}{1+\epsilon_r} - \ldots - l_{r,r-1}x_{r-1}\dfrac{1+E_{r,r-1}}{1+\epsilon_r} + b_r}{l_{rr}\dfrac{1+\eta_r}{1+\epsilon_r}}. \tag{19.11}$$

If we write

$$(1+E_{ri})/(1+\epsilon_r) = 1 + F_{ri} \tag{19.12}$$

$$\frac{1+\eta_r}{1+\epsilon_r} = 1 + F_{rr}, \tag{19.13}$$

equation (19.11) may be expressed in the form

$$l_{r1}x_1(1+F_{r1}) + l_{r2}x_2(1+F_{r2}) + \ldots + l_{rr}x_r(1+F_{rr}) \equiv b_r. \tag{19.14}$$

Since $-2^{-t} \leqslant \eta_r \leqslant 2^{-t}$ and $-2^{-t} \leqslant \epsilon_r \leqslant 2^{-t}$ we have

$$|F_{rr}| \leqslant 2 \times 2^{-t}. \tag{19.15}$$

The bound for F_{ri} is interesting. If we have the more accurate type of floating-point (sections 13 to 16 of Chapter 1), then (26.7) and (26.8) of Chapter 1 show that the factor $1+\epsilon_r$ occurs explicitly in each of the $1+E_{ri}$ and hence

$$|F_{ri}| \leqslant (r+1-i)2^{-t}. \tag{19.16}$$

With the less accurate form of arithmetic (sections 17 to 19 of Chapter 1), the factor $1+\epsilon_r$ is different from any of the $r+2-i$ factors of $1+E_{ri}$ and we have

$$|F_{ri}| \leqslant (r+3-i)2^{-t}. \tag{19.17}$$

Since we have been assuming the former or ACE-type floating-point throughout, we take the bound (19.16), though the overall result is not materially affected in this case.

Equations (19.14), (19.15) and (19.16) show that the computed vector is the exact solution of

$$(L+\delta L)x = b \tag{19.18}$$

where δL is bounded as is shown for $n = 5$ in (19.19):

$$|\delta L| \leqslant 2^{-t_1}\begin{bmatrix} |l_{11}| & & & & \\ 2|l_{21}| & 2|l_{22}| & & & \\ 3|l_{31}| & 2|l_{32}| & 2|l_{33}| & & \\ 4|l_{41}| & 3|l_{42}| & 2|l_{43}| & 2|l_{44}| & \\ 5|l_{51}| & 4|l_{52}| & 3|l_{53}| & 2|l_{54}| & 2|l_{55}| \end{bmatrix}. \tag{19.19}$$

Note that the matrix δL is a function of the computed vector x but the bound for $|\delta L|$ is independent of x.

If $|l_{ij}| \leqslant g$ then

$$\|\delta L\|_\infty \leqslant \tfrac{1}{2}(n^2+n+2)g2^{-t_1} \tag{19.20}$$

$$\|\delta L\|_1 \leqslant \tfrac{1}{2}(n^2+n)g2^{-t_1} \tag{19.21}$$

$$\|\delta L\|_2 \leqslant \|\delta L\|_E = \left[(3n-3) + \sum_{r-1}^{n} r(n-r+1)^2\right]^{1/2} g2^{-t_1}$$

$$< \frac{1}{\sqrt{12}}(n+2)^2 g2^{-t_1}. \tag{19.22}$$

Note also that

$$\|\delta L\| \leqslant n2^{-t_1}\|L\| \tag{19.23}$$

for the 1, ∞ and E norms.

ACCURACY OF COMPUTED SOLUTION

20. If we assume that L and b are scaled so that all elements are of modulus less than unity, then we may take $g = 1$ and

$$b - Lx = \delta Lx \tag{20.1}$$

$$\|b - Lx\|_\infty \leqslant \tfrac{1}{2}(n^2+n+2)2^{-t_1}\|x\|_\infty. \tag{20.2}$$

The left-hand side of (20.1) is the residual vector and (20.2) gives a bound for the maximum element of the residual vector in terms of the maximum element of x. Notice that the bound for this residual is directly dependent upon $\|x\|_\infty$ itself. If $\|x\|_\infty$ is of order unity, the residual vector is necessarily very small whether or not x is an accurate solution.

The factor $\frac{1}{2}(n^2+n+2)$ is hardly likely to be attained, and in general even the replacement of this factor by its square root still gives an overestimate of the true residual. This is not surprising when one remembers that the upper bounds of the factors $1+E_{ri}$ in the inner-products can be attained only in very special circumstances.

By way of comparison we consider now the residual vector which corresponds to the vector obtained by rounding the *exact* solution of

$$Lx = b \tag{20.3}$$

to t binary figures. Let us denote the exact solution by z and the rounded solution by u so that

$$u_i = z_i(1+\epsilon_i) \quad (|\epsilon_i| \leqslant 2^{-t}). \tag{20.4}$$

Hence

$$b - Lu = -Lw \text{ where } w_i = z_i\epsilon_i. \tag{20.5}$$

With this solution we can guarantee only that

$$\|b - Lu\|_\infty \leqslant n2^{-t}\|z\|_\infty. \tag{20.6}$$

Provided the computed solution x of the last section is not hopelessly inaccurate $\|x\|_\infty$ and $\|z\|_\infty$ will be approximately equal, and the bound we have for the residual corresponding to x is only $\frac{1}{2}n$ times as large as the bound corresponding to the correctly rounded solution.

Applying the result of (12.15) and using (19.23) we have

$$\frac{\|x-z\|}{\|z\|} \leqslant \frac{\|L\|\,\|L^{-1}\|\dfrac{\|\delta L\|}{\|L\|}}{1-\|L\|\,\|L^{-1}\|\dfrac{\|\delta L\|}{\|L\|}}$$

$$\leqslant \frac{n2^{-t_1}\|L\|\,\|L^{-1}\|}{1-n2^{-t_1}\|L\|\,\|L^{-1}\|} \tag{20.7}$$

for any norm except the 2-norm. As is to be expected, the condition number $\|L\|\,\|L^{-1}\|$ is of fundamental importance in this bound. *We can guarantee that the error is small compared with the true solution only if*

$$n2^{-t_1}\|L\|\,\|L^{-1}\| \ll 1 \tag{20.8}$$

in which case the right-hand side of (20.7) *is approximately* $n2^{-t_1}\|L\|\,\|L^{-1}\|$. *However, the remarkable thing is that in practice, for very large classes of matrices the relative error is not at all dependent on* $\|L\|\,\|L^{-1}\|$. We return to this important point later.

SOLUTION OF TRIANGULAR SET OF EQUATIONS WITH FLOATING-POINT ACCUMULATION OF INNER-PRODUCTS

21. Although the result we have already obtained is by no means unsatisfactory, that corresponding to floating-point accumulation is considerably better. For this we have

$$x_r \equiv fl_2\left(\frac{-l_{r1}x_1 - l_{r2}x_2 - \ldots - l_{r,r-1}x_{r-1} + b_r}{l_{rr}}\right) \tag{21.1}$$

and the computed solution satisfies

$$(L+\delta L)x = b \tag{21.2}$$

where $|\delta L|$ is bounded as shown in (21.3):

$$|\delta L| \leqslant 2^{-2t_1}\frac{3}{2}\begin{bmatrix} |l_{11}| & & & & \\ 4|l_{21}| & |l_{22}| & & & \\ 5|l_{31}| & 4|l_{32}| & |l_{33}| & & \\ 6|l_{41}| & 5|l_{42}| & 4|l_{43}| & |l_{44}| & \\ 7|l_{51}| & 6|l_{52}| & 5|l_{53}| & 4|l_{54}| & |l_{55}| \end{bmatrix} + 2^{-t_1}\begin{bmatrix} |l_{11}| & & & & \\ & |l_{22}| & & & \\ & & |l_{33}| & & \\ & & & |l_{44}| & \\ & & & & |l_{55}| \end{bmatrix}. \tag{21.3}$$

This follows from a simple extension of the result of Chapter 1, equation (32.11), analogous to that we have just used in section 19. The result corresponding to (20.2) is now

$$\|b - Lx\|_\infty \leqslant 2^{-t_1}[1 + \tfrac{3}{2}(n^2+5n-4)2^{-2t_2+t_1}]\,\|x\|_\infty \qquad (21.4)$$

and if $n^2 2^{-t} \ll 1$ the second term in the bracket is negligible. *Notice that the maximum value of the residual in this case is actually smaller than the expected value corresponding to the correctly rounded solution.* This should not surprise us since the relation (21.1) *guarantees* that the computed solution gives a very small residual.

We have treated the fixed-point case in some detail in **[30]** and shall not repeat the discussion here. Scaling is usually necessary in the 'fixed-point' solution of triangular equations and the discussion is somewhat lengthy.

INVERSION OF A TRIANGULAR MATRIX

22. The inversion of a triangular matrix L is a special case of the solution of linear equations. We have only to solve the set of equations

$$Lx = b \qquad (22.1)$$

where the right-hand side is to be taken to be each of the columns in turn of the identity matrix. If we denote the rth column of the identity matrix by e_r then, when solving $Lx = e_r$, our computed solution x_r satisfies

$$(L + \delta L_r)x_r = e_r. \qquad (22.2)$$

Now it is obvious that x_r has its first $r-1$ components equal to zero and that in solving $Lx = e_r$ we use only the lower triangular matrix in the bottom right-hand corner of L. Hence $|\delta L_r|$ is bounded as shown in (22.3) for the case $r = 3$, $n = 6$,

$$|\delta L_3| \leqslant 2^{-t_1}\begin{bmatrix} 0 & & & & & \\ 0 & 0 & & & & \\ 0 & 0 & |l_{33}| & & & \\ 0 & 0 & 2|l_{43}| & 2|l_{44}| & & \\ 0 & 0 & 3|l_{53}| & 2|l_{54}| & 2|l_{55}| & \\ 0 & 0 & 4|l_{63}| & 3|l_{64}| & 2|l_{65}| & 2|l_{66}| \end{bmatrix} = K_r \text{ (say)}. \qquad (22.3)$$

It does not appear to be easy to take advantage of these bounds for the individual δL_r, but we obviously have

$$|\delta L_r| \leqslant K_1 \quad (r = 1, \ldots, n) \qquad (22.4)$$

where K_1 is the matrix of the form given on the right-hand side of (19.19). This is a gross overestimate as far as some of the $|\delta L_r|$ are concerned.

The complete inverse satisfies the equation

$$I - LX = F \qquad (22.5)$$

where

$$|F| \leqslant K_1|X|. \tag{22.6}$$

Hence for any except the 2-norm

$$\|F\| \leqslant \|K_1\|\,\|X\|, \tag{22.7}$$

since

$$\||X|\| = \|X\| \tag{22.8}$$

except for the 2-norm.

From (19.23)

$$\|F\| \leqslant n2^{-t_1}\|L\|\,\|X\|. \tag{22.9}$$

This enables us to give a bound for the error in the inverse in terms of the condition number of L. From (22.5)

$$L^{-1} - X = L^{-1}F \tag{22.10}$$

$$\|X\| - \|L^{-1}\| \leqslant \|L^{-1} - X\| = \|L^{-1}F\| \leqslant \|L^{-1}\|n2^{-t_1}\|L\|\,\|X\| \tag{22.11}$$

$$\|X\|(1 - n2^{-t_1}\|L\|\,\|L^{-1}\|) \leqslant \|L^{-1}\|. \tag{22.12}$$

Similarly

$$\|X\|(1 + n2^{-t_1}\|L\|\,\|L^{-1}\|) \geqslant \|L^{-1}\|. \tag{22.13}$$

Hence from (22.9), (22.10) and (22.12)

$$\|L^{-1} - X\| \leqslant \frac{\|L^{-1}\|n2^{-t_1}\|L\|\,\|L^{-1}\|}{1 - n2^{-t_1}\|L\|\,\|L^{-1}\|}$$

$$\frac{\|L^{-1} - X\|}{\|L^{-1}\|} \leqslant \frac{n2^{-t_1}\|L\|\,\|L^{-1}\|}{1 - n2^{-t_1}\|L\|\,\|L^{-1}\|}. \tag{22.14}$$

It will be noted that this expression is of precisely the same form as (20.7). The bound for the relative error is approximately $n2^{-t}\|L\|\,\|L^{-1}\|$, provided this is appreciably less than unity.

HIGH ACCURACY OF SOLUTIONS OF TRIANGULAR EQUATIONS

23. In practice one almost invariably finds that if L is ill-conditioned, so that $\|L\|\,\|L^{-1}\| \gg 1$, then the computed solution of $Lx = b$ (or the computed inverse) is far more accurate than (20.7) or (22.14) would suggest. A simple example will show how this may come about. Consider the equations

$$\begin{aligned}
10^{-4}(0{\cdot}9143)x_1 &= 0{\cdot}6524\\
(0{\cdot}8762)x_1 + 10^{-4}(0{\cdot}7156)x_2 &= 0{\cdot}3127\\
(0{\cdot}7943)x_1 + (0{\cdot}8143)x_2 + 10^{-4}(0{\cdot}9504)x_3 &= 0{\cdot}4186\\
(0{\cdot}8017)x_1 + (0{\cdot}6123)x_2 + (0{\cdot}7165)x_3 + 10^{-4}(0{\cdot}7123)x_4 &= 0{\cdot}7853
\end{aligned} \tag{23.1}$$

The computed solution using 4-decimal floating-point compution is

$$x^T = 10^4(0\cdot7136);\ 10^8(-0\cdot8738);\ 10^{12}(0\cdot7485);\ 10^{16}(-0\cdot7528). \quad (23.2)$$

Clearly $\|L\| = O(1)$ and $\|L^{-1}\| = O(10^{16})$ so that the condition number is of order 10^{16}. Hence $n10^{-4}\|L\|\,\|L^{-1}\| = O(10^{12})$, and (20.7) and (22.14) give quite unrealistic bounds. The correct solution of the equations (23.1) rounded to four decimals is

$$10^4(0\cdot7136);\ 10^8(-0\cdot8736);\ 10^{12}(0\cdot7485);\ 10^{16}(0\cdot7528). \quad (23.3)$$

Hence for the relative error we have

$$\frac{\|x - L^{-1}b\|}{\|L^{-1}b\|} = O(10^{-4}) \quad (23.4)$$

and the bound (20.7) is completely misleading.

It might be felt that our example was rather special, but we have shown in [**30**] that for a very wide class of matrices

$$\frac{\|x - L^{-1}b\|}{\|L^{-1}b\|} \leqslant f(n)2^{-t} \quad (23.5)$$

where $f(n)$ is a simple function of n which depends on the type of arithmetic used in computing x but which does not involve $\|L\|\,\|L^{-1}\|$. We may write

$$x - L^{-1}b = s \quad (23.6)$$

$$\|s\| \leqslant f(n)2^{-t}\|L^{-1}b\|. \quad (23.7)$$

24. A similar result applies to the inversion of triangular matrices. In practice we have for the computed inverse

$$X - L^{-1} = S \quad (24.1)$$

$$\|S\| \leqslant f(n)2^{-t}\|L^{-1}\| \quad (24.2)$$

so that *the computed inverse almost invariably has a low relative error.* From (24.1) we have

$$LX - I = LS; \qquad \|LX - I\| \leqslant \|L\|\,\|S\| \leqslant f(n)2^{-t}\|L\|\,\|L^{-1}\| \quad (24.3)$$

$$XL - I = SL; \qquad \|XL - I\| \leqslant \|S\|\,\|L\| \leqslant f(n)2^{-t}\|L\|\,\|L^{-1}\|. \quad (24.4)$$

The reader should study these last two results and make certain that he appreciates their full significance. *Whenever they are true they guarantee that X will be equally good as a left-hand or right-hand inverse.* Our fundamental result of section 22 guaranteed only that

$$\|LX - I\| \leqslant n2^{-t}\|L\|\,\|X\| \leqslant \frac{n2^{-t}\|L\|\,\|L^{-1}\|}{1 - n2^{-t}\|L\|\,\|L^{-1}\|} \quad (24.5)$$

and gave no hint of a corresponding bound for $\|XL - I\|$. In fact all we can guarantee rigorously is that

$$XL - I = L^{-1}FL. \tag{24.6}$$

$$\|XL - I\| \leqslant \frac{\|L^{-1}\| n2^{-t} \|L\| \, \|L^{-1}\| \, \|L\|}{1 - n2^{-t}\|L\| \, \|L^{-1}\|}$$

$$= \frac{n2^{-t}(\|L\| \, \|L^{-1}\|)^2}{1 - n2^{-t}\|L\| \, \|L^{-1}\|}. \tag{24.7}$$

Since the explicit bounds may obscure this result we express it in the following simple form—

If

$$LX - I = F \quad \textit{giving} \quad \|LX - I\| = \|F\| \tag{24.8}$$

then

$$XL - I = L^{-1}FL \quad \textit{giving} \quad \|XL - I\| \leqslant \|L^{-1}\| \, \|L\| \, \|F\|. \tag{24.9}$$

The residual of X as a left-hand inverse may be larger than the residual as a right-hand inverse by a factor as great as $\|L\| \, \|L^{-1}\|$. The method we have described for inverting a triangular matrix automatically ensures that $LX - I$ is as small as we can expect, having regard to the size of the elements of X. We are asserting that the computed X is *almost invariably* of such a nature that $XL - I$ is equally small.

SOLUTION OF A GENERAL SET OF EQUATIONS

25. We have already reduced the solution of

$$Ax = b \tag{25.1}$$

to that of

$$LUx = b \tag{25.2}$$

where the computed L and U satisfy

$$LU \equiv A + E. \tag{25.3}$$

The computed solution is obtained by solving two triangular sets of equations and in practice we obtain

$$(L + \delta L)y \equiv b \tag{25.4}$$

$$(U + \delta U)x \equiv y \tag{25.5}$$

where δL and δU satisfy the bounds we obtained in the last few sections. Hence x satisfies

$$(L + \delta L)(U + \delta U)x \equiv (L + \delta L)y \equiv b \tag{25.6}$$

$$(A + E + L\delta U + \delta L U + \delta L \delta U)x \equiv b, \tag{25.7}$$

showing that x is the exact solution of a perturbed set of equations. The perturbation of the matrix A is a function of x but the bound we have obtained is not.

If pivoting has been used, then $|l_{ij}| \leqslant 1$ for all i, j and if g is the maximum element of any $|A^{(r)}|$ it is also a bound for the elements of U. For floating-point computation we have from (16.10) and (19.19)

$$\|E\|_\infty \leqslant (2\cdot 01)g2^{-t_1}(\tfrac{1}{2}n+1)(n-1) \tag{25.8}$$

$$\|\delta L\|_\infty \leqslant \tfrac{1}{2}(n^2+n+2)2^{-t_1} \tag{25.9}$$

$$\|\delta U\|_\infty \leqslant \tfrac{1}{2}g(n^2+n+2)2^{-t_1} \tag{25.10}$$

$$\|L\|_\infty \leqslant n \tag{25.11}$$

$$\|U\|_\infty \leqslant gn. \tag{25.12}$$

Hence, writing (25.7) in the form

$$(A+K)x \equiv b, \tag{25.13}$$

we have a bound for $\|K\|_\infty$ which for sufficiently large n is effectively

$$g2^{-t_1}(2\cdot 005n^2 + n^3 + \tfrac{1}{4}n^4 2^{-t_1}). \tag{25.14}$$

The last term in the brackets is unimportant if $n2^{-t} \ll 1$. The corresponding bound for the residual is therefore effectively

$$\|b - Ax\|_\infty \leqslant g2^{-t_1}(2\cdot 005n^2 + n^3)\|x\|_\infty. \tag{25.15}$$

The dominant term in this expression is $g2^{-t}n^3\|x\|_\infty$ and this comes from errors in the triangular solutions. We have already remarked (section 20) that the bound in the triangular solution may only be attained in exceptional circumstances, so that even the replacement of the factor $2\cdot 005n^2 + n^3$ by its square root gives a somewhat higher value than we would normally expect to obtain in practice. In our experience $g2^{-t_1}n\|x\|_\infty$ is seldom exceeded.

26. We now turn to the *accuracy* of the computed solution as distinct from the size of the residual. Since we have

$$LU = (A+E), \tag{26.1}$$

even if we solve

$$LUx = b \tag{26.2}$$

exactly, using the computed L and U, we will have the solution of $(A+E)x = b$ and not that of $Ax = b$.

Now we have already remarked that the accuracy of the computed solution of an ill-conditioned triangular set of equations is usually much greater than can be gauged from the size of the residuals. We are now solving in succession the two triangular sets of equations

$$Ly = b \text{ and } Ux = y. \tag{26.3}$$

If it is still true that the computed x is 'very close' to the exact solution of $LUx = b$ (*corresponding, of course, to the computed L and U*) then this means that the final computed x is close to the exact solution of $(A+E)x = b$, and the main error in our computed solution is that arising from E.

We will make the assumption for the moment that when we attempt to solve *any* triangular set of equations

$$Lx = b \tag{26.4}$$

the computed x satisfies

$$\frac{\|x - L^{-1}b\|}{\|L^{-1}b\|} \leqslant f(n)2^{-t}, \tag{26.5}$$

as discussed in section 23. This has been rigorously demonstrated only for certain special types of triangular matrices but has been observed in practice to be almost universally true. Under this assumption we shall compare the computed solution of $LUx = b$ with $(LU)^{-1}b$.

We have for the y and x of (25.4) and (25.5) under our assumption

$$y - L^{-1}b = e, \qquad \|e\| \leqslant f(n)2^{-t}\|L^{-1}b\| \tag{26.6}$$

$$x - U^{-1}y = f, \qquad \|f\| \leqslant f(n)2^{-t}\|U^{-1}y\|. \tag{26.7}$$

Hence

$$x = U^{-1}L^{-1}b + U^{-1}e + f \tag{26.8}$$

giving

$$\begin{aligned}\|x - (LU)^{-1}b\| &= \|U^{-1}e + f\| \\ &\leqslant \|U^{-1}\|\,\|e\| + \|f\| \\ &\leqslant \|U^{-1}\|f(n)2^{-t}\|L^{-1}b\| + f(n)2^{-t}\|U^{-1}\|\,\|y\| \\ &\leqslant f(n)2^{-t}\|U^{-1}\|\,\|L^{-1}b\| + f(n)2^{-t}\|U^{-1}\|\|(1 + f(n)2^{-t})\,\|L^{-1}b\| \\ &\doteq f(n)2^{-t}(2 + f(n)2^{-t})\,\|U^{-1}\|\,\|L^{-1}b\|. \end{aligned} \tag{26.9}$$

This shows that if $\|(LU)^{-1}b\|$ *and* $\|U^{-1}\|\,\|L^{-1}b\|$ *are of the same order of magnitude then*

$$\frac{\|x - (LU)^{-1}b\|}{\|(LU)^{-1}b\|} = g(n)2^{-t} \tag{26.10}$$

for some $g(n)$ which, like $f(n)$, is a simple function of n. In practice, when pivoting has been used this extra requirement is usually satisfied when $\|(LU)^{-1}b\|$ is of the same order of magnitude as $\|(LU)^{-1}\|\,\|b\|$, that is, when b is a right-hand side which fully reflects the ill-condition of LU. For such right-hand sides it is the error E which is the limiting factor in the accuracy of the final computed solution x.

Although the above argument is of a somewhat tendentious nature its importance in practice cannot be too strongly stressed. The example of sections 29 and 30 provides illustrations.

INVERSION OF A GENERAL MATRIX

27. The result of the last section may be applied to the inversion of a matrix. In fact we have to solve

$$LUx_r = e_r \quad (r = 1, \ldots, n) \tag{27.1}$$

for the n vectors x_r $(r = 1, \ldots, n)$. The x_r are the columns of the computed inverse. Each x_r will satisfy an equation of the form

$$(A + E + L\delta U_r + \delta L_r U + \delta L_r \delta U_r)x_1 \equiv e_r \tag{27.2}$$

where the perturbations are different for each value of r, but the upper bounds we have obtained apply equally to all of them. Hence, denoting the computed solution by X, we have

$$AX - I = F \tag{27.3}$$

where

$$\|F\|_\infty \leqslant \|X\|_\infty g2^{-t}(2\cdot 005n^2 + n^3 + \tfrac{1}{4}n^4 2^{-t}). \tag{27.4}$$

Much the same result applies for the 1-norm or the E-norm but for the 2-norm the result is a little weaker. In the bounds for the contribution from the triangular solutions we have expressions of the form $\| |\delta L| \| |X| \|$, and although our bound was explicitly for $\| |\delta L| \|$, we must replace $\| |X| \|_2$ by $n^{1/2}\|X\|_2$. The n^3 term in (27.4) therefore becomes $n^{7/2}$ if we use the 2-norm. This difficulty did not arise in the previous section because $\| |x| \|_2 = \|x\|_2$ for a vector.

LEFT-HANDED AND RIGHT-HANDED INVERSES

28. The bound we have obtained is for $AX - I$ and hence assesses X as a right-hand inverse. From the equation $AX - I = F$ we can only guarantee that $\|XA - I\| \leqslant \|A\| \|A^{-1}\| \|F\|$. It is natural to ask whether in practice the left-handed residual really is larger by a factor of order $\|A\| \|A^{-1}\|$ where A is ill-conditioned. The answer is that in general it is not. *Using the method we have described* one finds almost invariably that $AX - I$ and $XA - I$ are of much the same order of magnitude even when A is ill-conditioned. (See examples of sections 29 and 30.)

This last result could have been deduced from the corresponding result for the inversion of triangular matrices. If no error were made in the triangular inversions the computed inverse would be $(A + E)^{-1}$. The residual matrices F and G corresponding to $(A+E)^{-1}$ are given by

$$I - (A+E)^{-1}A \tag{28.1}$$

$$I - A(A+E)^{-1} \tag{28.2}$$

giving

$$F = I - (I+A^{-1}E)^{-1} \tag{28.3}$$

$$G = I - (I+EA^{-1})^{-1}. \tag{28.4}$$

We conclude that

$$\|F\| \leqslant \frac{\|A^{-1}\| \|E\|}{1 - \|A^{-1}\| \|E\|} \tag{28.5}$$

$$\|G\| \leqslant \frac{\|A^{-1}\| \|E\|}{1 - \|A^{-1}\| \|E\|} \tag{28.6}$$

so that the bound for the left-hand residual would be the same as that for the right-hand residual.

Now we expect the errors made in inverting the triangular matrices to be such that the computed inverse X satisfies

$$\frac{\|X-(A+E)^{-1}\|}{\|(A+E)^{-1}\|} \leqslant 2^{-t}g(n) \tag{28.7}$$

where $g(n)$ is a simple function of n. For, corresponding to our remark at the end of section 26, we will expect this to be true whenever $\|(LU)^{-1}I\|$ is of the same order of magnitude as $\|(LU)^{-1}\|\,\|I\|$, but this is always true. At least one column of $(LU)^{-1}$ must reflect the ill-condition of LU! (Again see the example of section 29.) We can write

$$X = (A+E)^{-1} + Y \tag{28.8}$$

$$\|Y\| \leqslant 2^{-t}g(n)\|(A+E)^{-1}\|\,. \tag{28.9}$$

The left-hand and right-hand residuals are therefore $F - YA$ and $-AY + G$ and hence there is no reason to expect the left-hand residual to be greater than the right.

NUMERICAL EXAMPLE

29. A simple numerical example will illustrate the results we have just obtained and show how remarkable they are. We consider the solution of a set of three equations using 6-decimal floating-point arithmetic and also, at the same time, the inversion of the matrix of coefficients. Note that in Table 1, L and U always stand for the computed L and U whenever they are used.

TABLE 1

$$A = \begin{bmatrix} 0{\cdot}932165 & 0{\cdot}443126 & 0{\cdot}417632 \\ 0{\cdot}712345 & 0{\cdot}915312 & 0{\cdot}887652 \\ 0{\cdot}632165 & 0{\cdot}514217 & 0{\cdot}493909 \end{bmatrix} \qquad b_1 = \begin{bmatrix} 0{\cdot}876132 \\ 0{\cdot}815327 \\ 0{\cdot}912345 \end{bmatrix} \qquad b_2 = \begin{bmatrix} 0{\cdot}876132 \\ 0{\cdot}815327 \\ 0{\cdot}648206 \end{bmatrix}$$

$$L = \begin{bmatrix} 1{\cdot}000000 & & \\ 0{\cdot}764183 & 1{\cdot}000000 & \\ 0{\cdot}678169 & 0{\cdot}370573 & 1{\cdot}000000 \end{bmatrix} \qquad U = \begin{bmatrix} 0{\cdot}932165 & 0{\cdot}443126 & 0{\cdot}417632 \\ & 0{\cdot}576683 & 0{\cdot}568505 \\ & & 10^{-4}(0{\cdot}110000) \end{bmatrix}$$

$$LU \equiv \begin{bmatrix} 0{\cdot}932165 & 0{\cdot}443126 & 0{\cdot}417632 \\ 0{\cdot}712344\ 646195 & 0{\cdot}915312\ 356058 & 0{\cdot}887652\ 274656 \\ 0{\cdot}632165\ 405885 & 0{\cdot}514217\ 465653 & 0{\cdot}493908\ 679173 \end{bmatrix}$$

Observe $LU \equiv A + E$; $|E_{ij}| \leqslant \frac{1}{2}10^{-6}$.

$$\left.\begin{matrix}\text{Computed solution} \\ \text{of } Ly = b_1\end{matrix}\right\} = \begin{bmatrix} 0{\cdot}876132 \\ 0{\cdot}145802 \\ 0{\cdot}264149 \end{bmatrix} \qquad \left.\begin{matrix}\text{Computed solution} \\ \text{of } Ux = y\end{matrix}\right\} = \begin{bmatrix} 10^{3}(\ 0{\cdot}495702) \\ 10^{5}(-0{\cdot}236728) \\ 10^{5}(\ 0{\cdot}240135) \end{bmatrix}$$

$$\left.\begin{matrix}\text{Accurate solution} \\ \text{of } LUx = b_1\end{matrix}\right\} = \begin{bmatrix} 10^{3}(\ 0{\cdot}495688\ldots) \\ 10^{5}(-0{\cdot}236730\ldots) \\ 10^{5}(\ 0{\cdot}240135\ldots) \end{bmatrix}$$

Observe that computed solution of $LUx = b$ is such that $\|\text{error}\|/\|x\| = O(10^{-6})$.

$$A \times \text{computed solution} = \begin{bmatrix} 0{\cdot}848914\ 03 \\ 0{\cdot}744229\ 59 \\ 0{\cdot}893028\ 73 \end{bmatrix} \qquad \text{Residual vector} = \begin{bmatrix} 0{\cdot}027217\ 97 \\ 0{\cdot}071097\ 41 \\ 0{\cdot}019316\ 27 \end{bmatrix}$$

$$\left.\begin{matrix}\text{Accurate solution} \\ \text{of } Ax = b_1\end{matrix}\right\} = \begin{bmatrix} 10^3(\ 0{\cdot}464479) \\ 10^5(-0{\cdot}221797) \\ 10^5(\ 0{\cdot}224990) \end{bmatrix} \qquad \text{Residual vector} = \begin{bmatrix} 0{\cdot}004439\ 165 \\ 0{\cdot}009252\ 145 \\ 0{\cdot}005181\ 865 \end{bmatrix}$$

$X =$ *Computed inverse of* LU

$$\begin{bmatrix} 10^3(-0{\cdot}738281) & 10^3(-0{\cdot}695049) & 10^4(\ 0{\cdot}187349) \\ 10^5(\ 0{\cdot}353969) & 10^5(\ 0{\cdot}332125) & 10^5(-0{\cdot}896199) \\ 10^5(-0{\cdot}359075) & 10^5(-0{\cdot}336885) & 10^5(\ 0{\cdot}909091) \end{bmatrix}$$

$Y =$ *Accurate inverse of* LU

$$\begin{bmatrix} 10^3(-0{\cdot}738278) & 10^3(-0{\cdot}695073) & 10^4(\ 0{\cdot}187345) \\ 10^5(\ 0{\cdot}353970) & 10^5(\ 0{\cdot}332124) & 10^5(-0{\cdot}896199) \\ 10^5(-0{\cdot}359076) & 10^5(-0{\cdot}336885) & 10^5(\ 0{\cdot}909091) \end{bmatrix}$$

Observe $\|X - (LU)^{-1}\|/\|(LU)^{-1}\| = O(10^{-6})$.

$Z =$ *Accurate inverse of* A (correct to 6 significant figures)

$$\begin{bmatrix} 10^3(-0{\cdot}691606) & 10^3(-0{\cdot}651287) & 10^4(\ 0{\cdot}175529) \\ 10^5(\ 0{\cdot}331643) & 10^5(\ 0{\cdot}311177) & 10^5(-0{\cdot}839673) \\ 10^5(-0{\cdot}336427) & 10^5(-0{\cdot}315636) & 10^5(\ 0{\cdot}851751) \end{bmatrix}$$

AX (exact)

$$\begin{bmatrix} 0{\cdot}965961\ 035 & 0{\cdot}026291\ 915 & 0{\cdot}043249\ 650 \\ -0{\cdot}067636\ 145 & 1{\cdot}020718\ 095 & 0{\cdot}045758\ 450 \\ -0{\cdot}065098\ 565 & -0{\cdot}006885\ 085 & 1{\cdot}101359\ 450 \end{bmatrix}$$

XA (exact)

$$\begin{bmatrix} 1{\cdot}040417\ 580 & 0{\cdot}042210\ 636 & 0{\cdot}042166\ 870 \\ -0{\cdot}054482\ 500 & 1{\cdot}010391\ 100 & -0{\cdot}054998\ 300 \\ 0{\cdot}001931\ 500 & -0{\cdot}030482\ 300 & 1{\cdot}037229\ 900 \end{bmatrix}$$

Observe that $AX-I$ and $XA-I$ are of same order of magnitude and could scarcely be expected to be smaller.

$$\left.\begin{matrix}\text{Computed solution} \\ \text{of } Ly = b_2\end{matrix}\right\} = \begin{bmatrix} 0{\cdot}876132 \\ 0{\cdot}145802 \\ 10^{-4}(0{\cdot}100000) \end{bmatrix} \qquad \left.\begin{matrix}\text{Computed solution} \\ \text{of } Ux = y\end{matrix}\right\} = \begin{bmatrix} 0{\cdot}838436 \\ -0{\cdot}643371 \\ 0{\cdot}909091 \end{bmatrix}$$

$$\left.\begin{matrix}A \times \text{computed} \\ \text{solution}\end{matrix}\right\} = \begin{bmatrix} 0{\cdot}876131\ 768706 \\ 0{\cdot}815326\ 940000 \\ 0{\cdot}648205\ 815152 \end{bmatrix} \qquad \text{Residual vector} = 10^{-6} \begin{bmatrix} 0{\cdot}231294 \\ 0{\cdot}060000 \\ 0{\cdot}184848 \end{bmatrix}$$

$$\left.\begin{matrix}\text{Accurate solution} \\ \text{of } Ax = b_2\end{matrix}\right\} = \begin{bmatrix} 0{\cdot}838561 \\ -0{\cdot}649354 \\ 0{\cdot}915160 \end{bmatrix} \qquad \text{Residual vector} = 10^{-6} \begin{bmatrix} 0{\cdot}324919 \\ 0{\cdot}168583 \\ 0{\cdot}190813 \end{bmatrix}$$

The residual vector corresponding to the computed solution is of the same order of magnitude as that corresponding to the correctly rounded solution.

30. The condition number of the matrix is of order 10^5 because obviously $\|A\| = O(1)$ and $\|A^{-1}\| = O(10^5)$. Since 'accumulation' of rounding errors is of no importance for a matrix of order 3, we would expect the computed solution and the computed inverse to have a relative error of approximately $10^5 \times 10^{-6}$ and this is indeed so. As far as b_1 is concerned, almost the whole of the error comes from the error E in the triangular decomposition. In spite of the ill-condition of U (its condition number is also $O(10^5)$), the computed solution of $LUx = b_1$ has a relative error of order 10^{-6}, thus supporting our assertion of section 26.

The residual $b_1 - Ax$ is as small as we can expect having regard to the size of the elements of x. The last digit in the components of x is in the $0{\cdot}1$ position and we cannot expect residuals appreciably below $0{\cdot}1$. *In fact we show that the residuals of the correct solution rounded to* 6 *significant decimals are of much the same size as those corresponding to our computed solution.* Nevertheless, our computed solution has errors as large as 1500 in some of its elements. A vector having *randomly* distributed errors as large as this, would have residuals of order 1000, that is about 10^5 times as large, but the method of solution is such as to give related errors which give a small residual. Notice that the computed solution corresponding to the right-hand side b_1 and the true solution are almost exactly in the ratio $1{\cdot}06731$ to 1 which is the ratio of the correct to the computed u_{33}. (See also section 53.) This ratio would hold for any right-hand side for which $\|x\|/\|b\|$ was of the order of magnitude of $\|A^{-1}\|$. If A were even more ill-conditioned the ratio of corresponding elements of the correct and computed solutions would remain constant to even more figures.

Similar remarks apply to the computed inverse. Again it is E which 'spoils' the inverse; the computed inverse of LU has a very low relative error. The residual matrix $AX - I$ is as small as can be expected having regard to the digital positions of its end figures and this residual matrix is smaller by a factor of order 10^{-5} than that corresponding to a matrix having randomly distributed errors of the same order of magnitude as those in X. Again the residual matrix corresponding to the correct inverse rounded to 6 significant decimals is of much the same size as that corresponding to X. This was, of course predicted by our error analysis.

The residual matrix $XA - I$ is of the same order of magnitude as $AX - I$ and is not *larger by the factor* $\|A\|\,\|A^{-1}\|$. This does *not* follow from the straightforward analysis. It is a consequence of the extraordinary accuracy of the computed inverses of *most* triangular matrices. In contrast to the smallness of $AX - I$ which is automatically guaranteed by the method of solution, that of $XA - I$ is to some extent dependent on the nature of the matrices involved. If it were particularly important that $XA - I$ should be small in a practical application, then it would be safer to invert A^T and take its transpose. Again note that the computed inverse and the correct inverse are almost exactly proportional and the constant of proportionality is again $1{\cdot}06731$.

31. The first right-hand side b_1 is such that $\|(LU)^{-1}b_1\|$ is of the order of magnitude $\|U^{-1}\|\,\|L^{-1}\|$ which is itself of the order of magnitude of $\|(LU)^{-1}\|\,\|b_1\|$, but we have chosen b_2 so that $\|(LU)^{-1}b_2\|$ is far smaller than $\|(LU)^{-1}\|\,\|b_2\|$. In fact the computed solution is of order unity. Note that the relative error in the computed solution is still very high. The residual vector on the other hand is very small, as was to be expected since we have shown that the norm of the residual vector is directly proportional to the norm of the solution. A very small residual gives no guarantee of accuracy when the matrix is ill-conditioned. However the computed solution is the exact solution corresponding to a right-hand side b_2' which differs from the given b_2 by less than 1 unit in the last significant figure of each component. In the terminology of section 26, b_1 is a right-hand side which fully reflects the ill-condition of the matrix of coefficients, while b_2 does not reflect it at all.

We see that with some right-hand sides, the ill-condition of a matrix is not revealed by the 'size' of the solution vector or by the size of the residuals. In the case of inversion we cannot be deceived in this way. If $\|AX - I\|$ is small (in an absolute sense !) X has a low relative error. Further we saw in section 27 that if the computed inverse X of a normalized matrix A does not have a large norm then $\|AX - I\|$ is certain to be small. In our example we would have been warned of the ill-condition of A even if we had worked only with b_2 and had not computed the inverse. The small element u_{33} makes the ill-condition quite evident.

However, if we take the matrix $A - \lambda I$ of order 21, where

$$A = \begin{bmatrix} 10 & 1 & & & & & \\ 1 & 9 & 1 & & & & \\ & 1 & 8 & & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & \cdot \\ & & & & 1 & -9 & 1 \\ & & & & & 1 & -10 \end{bmatrix}, \quad \lambda = 10{\cdot}74619\ 42$$

and the right-hand side e_{21} (the last column of the unit matrix), and use partial pivoting, then there are no small pivots. If we work with 9 significant decimals, then the computed solution is of order unity and the residuals are of order 10^{-8}. There are therefore no obvious signs of ill-condition and yet $A - \lambda I$ is almost singular and the computed solution is incorrect even in its most significant figure. The calculation of the inverse reveals the ill-condition, as indeed it must. The use of *complete* pivoting also reveals ill-conditioning at the decomposition stage; this has invariably proved true in the author's experience and is not unexpected.

COMPACT METHODS OF TRIANGULAR DECOMPOSITION

31. We have seen that when A is ill-conditioned, the error E is the more important error as regards the accuracy of computed inverses. A method

which reduces E to a minimum would therefore have much to recommend it. Now Gaussian elimination produces triangular matrices which satisfy

$$LU = A \tag{31.1}$$

ignoring rounding errors. We can derive the elements of L and U directly from this matrix relation. If we equate corresponding elements in equation (31.1) then we have

$$l_{i1}u_{1j} + l_{i2}u_{2j} + \ldots + l_{i,i-1}u_{i-1,j} + u_{ij} = a_{ij} \quad (j \geqslant i) \tag{31.2}$$

$$l_{i1}u_{1j} + l_{i2}u_{2j} + \ldots + l_{ij}u_{jj} = a_{ij} \quad (i > j). \tag{31.3}$$

Equation (31.2) may be used to determine elements of U and equation (31.3) to determine elements of L. The elements are determined in the order

$$\left.\begin{array}{l} u_{11} \rightarrow u_{12} \rightarrow u_{13} \ldots \rightarrow u_{1n} \\ l_{21} \rightarrow u_{22} \rightarrow u_{23} \ldots \rightarrow u_{2n} \\ l_{31} \rightarrow l_{32} \rightarrow u_{33} \ldots \rightarrow u_{3n} \\ - - - - - - - - - - - - - \\ l_{n1} \rightarrow l_{n2} \rightarrow l_{n3} \ldots \rightarrow u_{nn}. \end{array}\right\} \tag{31.4}$$

The equation defining u_{ij} in terms of previously computed quantities is

$$u_{ij} = a_{ij} - l_{i1}u_{1j} - \ldots - l_{i,i-1}u_{i-1,j} \tag{31.5}$$

and that defining l_{ij} is

$$l_{ij} = (a_{ij} - l_{i1}u_{1j} - \ldots - l_{i,j-1})/u_{jj}. \tag{31.6}$$

If we accumulate inner-products and also divide into such an inner-product, only one rounding error is involved in each determination.

If the computation is performed as described, it is the equivalent of Gaussian elimination *without pivoting* and hence the l_{ij} and u_{ij} may become arbitrarily large. However the work can easily be re-organised so as to incur the benefits of partial pivoting, in which case, apart from rounding errors, all elements of L and U are identical with those in Gaussian elimination with partial pivoting.

TRIANGULAR DECOMPOSITION WITH PARTIAL PIVOTING

32. The equivalent of partial pivoting is achieved as follows.

There are n major steps, in the rth of which the rth column of L and the rth row of U are determined. We start the rth step with a configuration of elements like that shown inside the brackets in (32.1) for the case $n = 5$, $r = 3$.

$$\begin{array}{c} \\ s_3 \\ s_4 \\ s_5 \end{array} \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ l_{21} & u_{22} & u_{23} & u_{24} & u_{25} \\ l_{31} & l_{32} & a_{33} & a_{34} & a_{35} \\ l_{41} & l_{42} & a_{43} & a_{44} & a_{45} \\ l_{51} & l_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix}. \tag{32.1}$$

The elements a_{ij} in this configuration are unmodified elements of the original matrix, but row interchanges may have taken place as will become evident when we describe the rth stage. We first form the $n+1-r$ quantities $s_r, s_{r+1}, \ldots, s_n$ defined by

$$s_t = a_{tr} - l_{t1}u_{1r} - l_{t2}u_{2r} - \ldots - l_{t,r-1}u_{r-1,r} \quad (t = r, r+1, \ldots, n). \quad (32.2)$$

These are the numerators in (31.6) for all possible choices of the rth pivotal row. If the s_t of maximum modulus is s_{p_r} then the whole of row r and row p_r of the array (32.1) are interchanged including s_r and s_{p_r}. If we call the current element occupying any position by the same name as was used before the interchange, then we have

$$u_{rt} = a_{rt} - l_{r1}u_{1t} - l_{r2}u_{2t} - \ldots - l_{r,r-1}u_{r-1,t} \quad (t = r, r+1, \ldots, n) \quad (32.3)$$

and

$$l_{tr} = s_t/u_{rr} \quad (t = r+1, r+2, \ldots, n). \quad (32.4)$$

Obviously all l_{ij} satisfy $|l_{ij}| \leqslant 1$.

In order to achieve the full advantage of this direct determination of L and U, it is essential that some form of accumulation is used when computing the relevant inner-products. If we merely use standard floating-point then this technique is identical, even as regards round off, with Gaussian elimination with partial pivoting. Note that all comments we have made concerning the growth in size of elements of U apply equally well to this new technique.

If the matrix is scaled originally so that all elements of $|U|$ remain bounded by unity then corresponding to fixed-point accumulation we have

$$LU = A + E \quad (32.5)$$

where E is bounded as is shown in (32.6) for the case $n = 5$

$$|E| \leqslant \tfrac{1}{2}2^{-t}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ |u_{11}| & 1 & 1 & 1 & 1 \\ |u_{11}| & |u_{22}| & 1 & 1 & 1 \\ |u_{11}| & |u_{22}| & |u_{33}| & 1 & 1 \\ |u_{11}| & |u_{22}| & |u_{33}| & |u_{44}| & 1 \end{bmatrix}. \quad (32.6)$$

From our assumption we may replace each $|u_{rr}|$ by unity and hence we have

$$\|E\| < \frac{n}{2}2^{-t}. \quad (32.7)$$

However, the bound (32.6) emphasizes the lack of symmetry in our treatment of L and U. We have chosen each diagonal element of L to be unity but we could in fact take l_{rr} and u_{rr} to be any values for which

$$l_{rr}u_{rr} = a_{rr} - l_{r1}u_{1r} - l_{r2}u_{2r} - \ldots - l_{r,r-1}u_{r-1,r}. \quad (32.8)$$

Since the expression on the right may be negative we cannot, in general, take l_{rr} and u_{rr} to be equal. We can however take

$$l_{rr} = |u_{rr}|, \tag{32.9}$$

so that l_{rr} is positive and u_{rr} has the sign of the right-hand side of (32.8). We then have

$$LU = A + E \tag{32.10}$$

where, when $n = 4$

$$|E| \leqslant \tfrac{1}{2}2^{-t}\begin{bmatrix} 2|u_{11}| & |u_{11}| & |u_{11}| & |u_{11}| \\ |u_{11}| & 2|u_{22}| & |u_{22}| & |u_{22}| \\ |u_{11}| & |u_{22}| & 2|u_{33}| & |u_{33}| \\ |u_{11}| & |u_{22}| & |u_{33}| & 2|u_{44}| \end{bmatrix}. \tag{32.11}$$

This technique gives a more balanced error matrix and when A is ill-conditioned it is common for the later $|u_{rr}|$ to be much smaller than unity. The norm of the balanced error matrix may then be much smaller than $\frac{n}{2}2^{-t}$.

There is little virtue in having l_{rr} exactly equal to $|u_{rr}|$ and we may avoid the square roots by taking l_{rr} to be the power of two nearest to the square root of the modulus of the right-hand side of (32.8). This also has the effect of replacing division by u_{rr} by a shift operation.

POSITIVE DEFINITE MATRIX

33. When the matrix A is positive definite and symmetric then it is well known [**30**] that it can be expressed in the form LL^T where the elements of L are real. If we equate the diagonal elements in the equation

$$LL^T = A \tag{33.1}$$

we have

$$l_{i1}^2 + l_{i2}^2 + \ldots + l_{ii}^2 = a_{ii} \quad (i = 1, \ldots, n). \tag{33.2}$$

Hence if all elements of $|A|$ are bounded by unity then so, too, are all elements of L. This symmetric decomposition, which is due to Cholesky, can be performed in fixed-point, *and the reader should note that no pivoting is required.* It is precisely this decomposition that has been generalized in the previous section to cover unsymmetric matrices. The computed L satisfies

$$LL^T = A + E \tag{33.3}$$

where E is bounded as in (32.11), and is now, of course, symmetric. There is a danger that rounding errors may destroy positive definiteness; we can be certain that this is not true if $A+E$ is positive definite. From (12.5), this is true if $\|A^{-1}\|\,\|E\| < 1$ and therefore if

$$\|A^{-1}\| < 2^{t+1}/(n+1). \tag{33.4}$$

Similarly, in order to guarantee that all computed $|l_{ij}|$ are bounded by unity, we must have

$$|a_{ij}| \leqslant 1 - 2^{-t}. \tag{33.5}$$

(For a more detailed analysis see [**30**].)

The Cholesky decomposition therefore has a guaranteed stability even without pivoting. If A is symmetric but not positive definite we can still apply the method described at the end of section 32 and if we do not use pivoting it may readily be verified that U differs from L^T only as regards the signs of some of its rows. *However, we no longer have the guaranteed stability of the positive definite case and we may obtain arbitrarily poor inverses even for well-conditioned matrices if we do not use pivoting.* This may be verified for the very well-conditioned symmetric matrix

$$\begin{bmatrix} 0{\cdot}00000\ 003 & 0{\cdot}96512\ 714 \\ 0{\cdot}96512\ 714 & 0{\cdot}00000\ 003 \end{bmatrix}. \tag{33.6}$$

NUMERICAL EXAMPLE

34. In Table 2 we exhibit the inverse of an ill-conditioned matrix. This matrix is defined by

$$A = \frac{50 \times 10!}{10^8} \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \\ \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} & \frac{1}{10} \end{bmatrix}.$$

The scale factor is included so that the initial matrix can be represented without rounding error; multiplication by a constant factor does not affect she condition of the matrix.

TABLE 2

A

$$\begin{bmatrix} 0{\cdot}9072\ 0000 & 0{\cdot}6048\ 0000 & 0{\cdot}4536\ 0000 & 0{\cdot}3628\ 8000 & 0{\cdot}3024\ 0000 \\ 0{\cdot}6048\ 0000 & 0{\cdot}4536\ 0000 & 0{\cdot}3628\ 8000 & 0{\cdot}3024\ 0000 & 0{\cdot}2592\ 0000 \\ 0{\cdot}4536\ 0000 & 0{\cdot}3628\ 8000 & 0{\cdot}3024\ 0000 & 0{\cdot}2592\ 0000 & 0{\cdot}2268\ 0000 \\ 0{\cdot}3628\ 8000 & 0{\cdot}3024\ 0000 & 0{\cdot}2592\ 0000 & 0{\cdot}2268\ 0000 & 0{\cdot}2016\ 0000 \\ 0{\cdot}3024\ 0000 & 0{\cdot}2592\ 0000 & 0{\cdot}2268\ 0000 & 0{\cdot}2016\ 0000 & 0{\cdot}1814\ 4000 \end{bmatrix}$$

L

$$\begin{bmatrix} 0{\cdot}9524\ 7047 & & & & \\ 0{\cdot}6349\ 8032 & 0{\cdot}2244\ 9943 & & & \\ 0{\cdot}4762\ 3524 & 0{\cdot}2693\ 9933 & 0{\cdot}0549\ 9088 & & \\ 0{\cdot}3809\ 8819 & 0{\cdot}2693\ 9934 & 0{\cdot}0942\ 7011 & 0{\cdot}0136\ 0665 & \\ 0{\cdot}3174\ 9016 & 0{\cdot}2565\ 7079 & 0{\cdot}1178\ 3769 & 0{\cdot}0302\ 3706 & 0{\cdot}0033\ 8039 \end{bmatrix}$$

$10^8(A-LL^T)$ (upper half only, computed exactly)

$$\begin{bmatrix} 0{\cdot}3777\ 9791 & -0{\cdot}3831\ 1504 & -0{\cdot}2873\ 3628 & -0{\cdot}0393\ 7493 & -0{\cdot}1915\ 5752 \\ & -0{\cdot}0857\ 6273 & -0{\cdot}1117\ 8587 & -0{\cdot}1074\ 7970 & 0{\cdot}0496\ 6991 \\ & & 0{\cdot}0294\ 5191 & -0{\cdot}0106\ 8546 & 0{\cdot}0260\ 9237 \\ & & & 0{\cdot}0124\ 4537 & 0{\cdot}0018\ 6161 \\ & & & & 0{\cdot}0003\ 4185 \end{bmatrix}$$

L^{-1}

$$\begin{bmatrix} 1{\cdot}049\ 9013 & & & & \\ -\ 2{\cdot}969\ 5696 & 4{\cdot}454\ 3543 & & & \\ 5{\cdot}455\ 4511 & -\ 21{\cdot}82\ 1802 & 18{\cdot}18\ 4834 & & \\ -\ 8{\cdot}399\ 2677 & 62{\cdot}99\ 4461 & -125{\cdot}9\ 8886 & 73{\cdot}49\ 3476 & \\ 11{\cdot}73\ 9135 & -140{\cdot}8\ 6944 & 493{\cdot}0\ 4186 & -657{\cdot}3\ 8764 & 295{\cdot}8\ 2385 \end{bmatrix}$$

$X =$ *computed* A^{-1} (upper half)

$$\begin{bmatrix} 248{\cdot}0\ 3757 & -2315{\cdot}0680 & 6945{\cdot}3056 & -8334{\cdot}4536 & 3472{\cdot}7161 \\ & 2430\ 8{\cdot}534 & -7778\ 7{\cdot}957 & 9723\ 5{\cdot}511 & -4167\ 2{\cdot}540 \\ & & 2592\ 94{\cdot}16 & -3333\ 78{\cdot}98 & 1458\ 53{\cdot}54 \\ & & & 4375\ 59{\cdot}80 & -1944\ 70{\cdot}94 \\ & & & & 8751\ 1{\cdot}750 \end{bmatrix}$$

True inverse of A (correctly rounded)

$$\begin{bmatrix} 248{\cdot}0\ 1587 & -2314{\cdot}8148 & 6944{\cdot}4444 & -8333{\cdot}3333 & 3472{\cdot}2222 \\ & 2430\ 5{\cdot}556 & -7777\ 7{\cdot}778 & 9722\ 2{\cdot}222 & -4166\ 6{\cdot}667 \\ & & 2592\ 59{\cdot}26 & -3333\ 33{\cdot}33 & 1458\ 33{\cdot}33 \\ & & & 4375\ 00{\cdot}00 & -1944\ 44{\cdot}44 \\ & & & & 8750\ 0{\cdot}000 \end{bmatrix}$$

AX

$$\begin{bmatrix} 1{\cdot}00000\ 3536 & 0{\cdot}00051\ 408 & 0{\cdot}00205\ 632 & 0{\cdot}00338\ 688 & 0{\cdot}00009\ 072 \\ 0{\cdot}00001\ 8144 & 1{\cdot}00021\ 824 & 0{\cdot}00232\ 848 & 0{\cdot}00186\ 192 & 0{\cdot}00049\ 248 \\ 0{\cdot}00001\ 7712 & 0{\cdot}00015\ 552 & 1{\cdot}00202\ 4 & 0{\cdot}00149\ 472 & 0{\cdot}00045\ 576 \\ 0{\cdot}00001\ 10016 & 0{\cdot}00018\ 216 & 0{\cdot}00157\ 1328 & 1{\cdot}00152\ 4032 & 0{\cdot}00029\ 8368 \\ 0{\cdot}00000\ 9072 & 0{\cdot}00016\ 2 & 0{\cdot}00137\ 664 & 0{\cdot}00134\ 496 & 1{\cdot}00026\ 864 \end{bmatrix}$$

COMMENTS ON THE SOLUTION

35. This example illustrates most of the points made in the previous section. The computed L is such that LL^T is exceptionally close to A. This is to be expected because the diagonal elements of L are, in general, appreciably smaller than unity. Many of the elements of $A-LL^T$ are far less than one in the eighth decimal although only 8-decimal multiplication with accumulation was used.

Although L is ill-conditioned, virtually no error was made in computing its inverse: indeed, even if this part had been carried out *exactly* with the computed L, the resulting L^{-1} would have differed by at most one in the

last digit from the computed L^{-1}. The computed A^{-1} is therefore virtually the exact inverse of LL^T. We therefore benefit very substantially from the fact that $A - LL^T$ is so small.

The maximum error in any component of A^{-1} is 59·80 and occurs in the (4, 4) element. It is interesting to compare this with the error which would result from changing, for example, the (4, 4) element of A by 10^{-8}. This gives a change of approximately 1476 in the (4, 4) element of the inverse. *In other words, all the rounding errors made in carrying out the whole of the computation have only $\frac{1}{30}$th of the effect of a change of one in the eighth figure of one of the elements of A.*

Since the computed X is exactly symmetric we have $(AX-I)^T = XA - I$ in this case. Although individual errors in elements of X are as large as 50, these errors are so related that the residuals are smaller by a factor of about 10^5 than those corresponding to an approximate inverse having random errors of the same order of magnitude. The residuals are about as small as those corresponding to the correctly rounded inverse.

RESIDUAL CORRESPONDING TO BLOCK-FLOATING SOLUTION

36. The analysis of the previous section has shown the great advantage of using a direct form of triangular decomposition with the equivalent of partial pivoting and inner-product accumulation. We shall therefore assume in our future analysis that this had been done and that the solution has been completed using block-floating arithmetic. We shall assume that the matrix L is unit-triangular so that the close relation with Gaussian elimination is maintained.

We shall denote the computed solution of

$$Ax = b \tag{36.1}$$

by $2^k y$ where y is standardized as in (6.2). In **[30]** we showed that if all elements of U are bounded by unity then the computed LU satisfies

$$LU \equiv A + E \quad (|e_{ij}| \leqslant \tfrac{1}{2}2^{-t}) \tag{36.2}$$

$$(A+E)2^k y \equiv b + e \quad (|e_i| \leqslant \tfrac{1}{2}i2^{k-t}) \tag{36.3}$$

provided the whole of the back-substitution is performed using the same scale factor 2^k.

This implies that $2^k y$ satisfies

$$(A+E+E_1)2^k y \equiv b \tag{36.4}$$

where $E_1\, 2^k y$ is $-e$ in (36.3). Hence

$$\|E_1\|_\infty \leqslant n2^{-t} \tag{36.5}$$

and

$$\|E + E_1\|_\infty \leqslant \frac{3n}{2}\, 2^{-t}. \tag{36.6}$$

We may also write

$$A2^k y = b + f \tag{36.7}$$

$$\|f\|_\infty \leqslant \frac{3n}{2} \|2^k y\|_\infty 2^{-t}. \tag{36.8}$$

The bound for the error E_1 is twice as large as that for E, but, as we have shown in sections 26 and 30, E usually plays the more important part in determining the accuracy of $2^k y$. As far as the residual f is concerned, however, (36.8) shows that E_1 plays its full part. The above results will form the basis for the analysis of the next few sections which has not previously appeared elsewhere. *The infinity norm is used throughout sections* 37 *to* 45.

ITERATIVE REFINEMENT OF THE SOLUTION

37. If the matrix A is ill-conditioned then the computed solution $x^{(1)}$ of $Ax = b$ may not be sufficiently accurate. We could obtain a more accurate solution by working to a higher precision, but this may be impractical. In any case it is important to have some assessment of the accuracy of the computed solution.

We describe now a process for computing a sequence of vectors $x^{(1)}$, $x^{(2)}$, . . . , which, in certain circumstances will tend to the true solution, x. We assume that the first approximation $x^{(1)}$ has been obtained using the method for which the analysis of section 36 applies. We have therefore a unit lower triangular matrix L and an upper triangular matrix U such that

$$LU = A + E. \tag{37.1}$$

(A and b from now on refer to the form taken by the equations when their rows have been suitably permuted.)

We define vectors $r^{(s)}$ and $x^{(s)}$ by the relations

$$r^{(s)} = b - Ax^{(s)} \tag{37.2}$$

$$x^{(s+1)} = x^{(s)} + (LU)^{-1} r^{(s)} \tag{37.3}$$

so that $r^{(s)}$ is the residual vector corresponding to $x^{(s)}$. *If this sequence could be derived without further rounding error* then we would have

$$\begin{aligned} x^{(s+1)} &= x^{(s)} + (LU)^{-1}(b - Ax^{(s)}) && (37.4)\\ &= x^{(s)} + (LU)^{-1}(Ax - Ax^{(s)}) \\ x^{(s+1)} - x &= x^{(s)} - x + (LU)^{-1}A(x - x^{(s)}) \\ &= [I - (LU)^{-1}A](x^{(s)} - x) && (37.5)\\ &= [I - (LU)^{-1}A]^s (x^{(1)} - x). && (37.6) \end{aligned}$$

Similarly we have

$$r^{(s+1)} = A(x - x^{(s+1)}) = A[I - (LU)^{-1}A]^s (x - x^{(1)}) \tag{37.7}$$

and

$$r^{(s+1)} = [I - A(LU)^{-1}] r^{(s)}. \tag{37.8}$$

If $E = 0$, it is evident that $x^{(s)}$ becomes x and $r^{(s)}$ becomes zero in one iteration, as we would expect.

Equations (37.6) and (37.7) show that we can guarantee convergence if

$$[I - (A+E)^{-1}A]^s \to 0 \text{ as } s \to \infty . \tag{37.9}$$

If we write

$$(A+E)^{-1}A = I - F; \qquad A(A+E)^{-1} = I - G \tag{37.10}$$

then

$$\|F\|, \|G\| \leqslant \|E\| \|A^{-1}\|/(1 - \|E\| \|A^{-1}\|). \tag{37.11}$$

Convergence is assured, therefore, if

$$\|E\| \|A^{-1}\| < \tfrac{1}{2} \tag{37.12}$$

and hence, from the previous section, if

$$\|A^{-1}\| < 2^t/n. \tag{37.13}$$

We may have convergence even if $\|A^{-1}\|$ exceeds this bound, but only if E or b are specially related to A. That some such bound as (37.13) would be involved is obvious, since if $\|A^{-1}\|$ is too large $A+E$ may even be singular.

If (37.13) is satisfied, then we may write (37.12) as

$$\|E\| \|A^{-1}\| < 2^{-p} \quad (p > 1). \tag{37.14}$$

From (37.5), (37.8) and (37.11) it then follows that

$$\|x^{(s+1)} - x\| \leqslant [2^{-p}/(1-2^{-p})]\|x^{(s)} - x\| \tag{37.15}$$

$$\|r^{(s+1)}\| \leqslant [2^{-p}/(1-2^{-p})]\|r^{(s)}\| . \tag{37.16}$$

Both the error and the residual decrease by at least the factor $2^{-p}/(1-2^{-p})$ with each iteration, so that if p is appreciably greater than 2 we effectively gain at least p binary figures per iteration.

PRACTICAL PROCEDURE

38. We now turn to the practical realisation of this technique in block-floating arithmetic. The process will not work at all unless (37.13) is satisfied. In order to ensure a reasonably rapid rate of convergence of the exact iterative process let us assume that

$$\|A^{-1}\| < (0\cdot1)2^{t-1}/n. \tag{38.1}$$

Since, from (36.4) and (36.6), $x^{(1)}$ is the exact solution of

$$(A+E+E_1)x^{(1)} = b; \qquad \|E + E_1\|_\infty \leqslant \frac{3n}{2}2^{-t} \tag{38.2}$$

we have from (12.13)

$$\frac{\|x - x^{(1)}\|}{\|x\|} \leqslant \frac{\|A^{-1}\| \frac{3n}{2} 2^{-t}}{1 - \|A^{-1}\| \frac{3n}{2} 2^{-t}}$$

$$< \frac{0{\cdot}075}{1 - 0{\cdot}075} < 0{\cdot}082 \tag{38.3}$$

$$0{\cdot}918\|x\| \leqslant \|x^{(1)}\| \leqslant 1{\cdot}082\|x\|. \tag{38.4}$$

Hence $x^{(1)}$ is already of the same order of magnitude as x, and further we now show that the residual corresponding to $x^{(1)}$ will be much the same size as that corresponding to $\bar{x}$, the correctly rounded s.p.b.f. solution. In fact, we have the bounds

$$\|b - Ax^{(1)}\| \leqslant \frac{3n}{2} \|x^{(1)}\| 2^{-t} \tag{38.5}$$

and, from (20.6),

$$\|b - A\bar{x}\| \leqslant n\|\bar{x}\| 2^{-t} \tag{38.6}$$

and $\|x\|$, $\|x^{(1)}\|$ and $\|\bar{x}\|$ all agree within a factor $(1 \pm 0{\cdot}09)$. This means that $x^{(1)}$ has only to have a slightly more favourable set of rounding errors to give the smaller residual and that clearly the $r^{(s)}$ corresponding to the computed block-floating vectors cannot possibly decrease by a factor of $2^{-\mu}$ with each iteration as (37.16) suggests. They are bound to remain roughly at their initial value and are as likely to increase as to decrease.

Nonetheless, it is a simple matter to organize the computation so that *the computed $x^{(s)}$ tend to $\bar{x}$ at about the same rate as is suggested by the theoretical investigation of the previous section* as we now show. The practical procedure is as follows.

We denote the sth computed solution by $x^{(s)}$ and write

$$x^{(s)} = 2^k y^{(s)} \text{ and } \bar{x} = 2^k \bar{y}. \tag{38.7}$$

Assuming that the practical process does converge, the index k will remain constant and equal to that corresponding to $\bar{x}$. (If the maximum component of $\bar{y}$ is almost exactly $\frac{1}{2}$ or 1 the index k may vary by one from time to time, but this is not an important point.) The steps in an iteration are:

(i) Compute the exact d.p.b.f. vector $r^{(s)}$ defined by

$$r^{(s)} = b - Ax^{(s)}. \tag{38.8}$$

This will be a non-standardized vector, and if (38.1) is satisfied there will be considerable cancellation—(usually of almost t figures). In fact from (38.5) we know that certainly for $x^{(1)}$ the residual satisfies

$$\|r^{(1)}\| \leqslant \frac{3n}{2} \|x^{(1)}\| 2^{-t} \tag{38.9}$$

and we shall show that later $r^{(s)}$ satisfy a similar inequality.

(ii) Convert $r^{(s)}$ to a standardized s.p.b.f. vector $\bar{r}^{(s)}$.
This vector satisfies

$$\bar{r}^{(s)} \equiv r^{(s)} + g^{(s)} \tag{38.10}$$

$$\|g^{(s)}\| \leqslant 2^{-t}\|r^{(s)}\|. \tag{38.11}$$

(iii) Use L and U to solve $LUx = \bar{r}^{(s)}$. The computed solution $\delta^{(s)}$ satisfies

$$(A+E+E_1)\delta^{(s)} \equiv \bar{r}^{(s)} \tag{38.12}$$

where E_1 is different for each s but is uniformly bounded as in (36.5).

(iv) Compute $x^{(s+1)}$ by adding $\delta^{(s)}$ to $x^{(s)}$ in s.p.b.f. arithmetic. We have

$$\begin{aligned} x^{(s+1)} &\equiv \text{b.f.}(x^{(s)}+\delta^{(s)}) \\ &= x^{(s)} + \delta^{(s)} + h^{(s)} \end{aligned} \tag{38.13}$$

$$\|h^{(s)}\| \leqslant 2^{-t}\|x^{(s)}\| \tag{38.14}$$

provided $\|\delta^{(s)}\| \leqslant \|x^{(s)}\|$. (The proof subsequently justifies this assumption.)

ANALYSIS OF THE PRACTICAL PROCEDURE

39. Combining (38.10) to (38.13) we have

$$\begin{aligned} x^{(s+1)} &= x^{(s)} + (A+E+E_1)^{-1}(r^{(s)} + g^{(s)}) + h^{(s)} \\ &= x^{(s)} + (A+E+E_1)^{-1}(Ax - Ax^{(s)} + g^{(s)}) + h^{(s)} \end{aligned} \tag{39.1}$$

leading to

$$x^{(s+1)} - x = [I - (A+E+E_1)^{-1}A](x^{(s)} - x) + (A+E+E_1)^{-1}g^{(s)} + h^{(s)}. \tag{39.2}$$

Writing

$$I - (A+E+E_1)^{-1}A = P \tag{39.3}$$

and using (38.11), (38.8) and (38.14) we have

$$\begin{aligned} \|x^{(s+1)} - x\| &\leqslant \|P\|\,\|x^{(s)} - x\| + \|(A+E+E_1)^{-1}\|\,2^{-t}\|A\|\,\|x^{(s)} - x\| + 2^{-t}\|x^{(s)}\| \\ &\leqslant [\|P\| + 2^{-t}\|A\|\,\|(A+E+E_1)^{-1}\| + 2^{-t}]\|x^{(s)} - x\| + 2^{-t}\|x\| \\ &= \beta\|x^{(s)} - x\| + 2^{-t}\|x\| \text{ (say)}. \end{aligned} \tag{39.4}$$

Hence

$$\|x^{(s+1)} - x\| \leqslant \beta^s\|x^{(1)} - x\| + 2^{-t}\|x\|/(1-\beta). \tag{39.5}$$

Provided $\beta < 1$ the first term tends to zero. Now we have

$$\begin{aligned} \beta &= \|P\| + 2^{-t}\|A\|\,\|(A+E+E_1)^{-1}\| + 2^{-t} \\ &\leqslant \frac{\|A^{-1}\|\,\|E+E_1\| + n2^{-t}\|A^{-1}\|}{1 - \|A^{-1}\|\,\|E+E_1\|} + 2^{-t} \text{ from (37.11) and (39.3)} \\ &\leqslant \frac{\frac{5}{2}n2^{-t}\|A^{-1}\|}{1 - \frac{3}{2}n2^{-t}\|A^{-1}\|} + 2^{-t} \\ &< 0{\cdot}14 \text{ (say)}, \end{aligned} \tag{39.6}$$

from the assumptions (38.1) and (38.2). The norm of $x^{(s+1)}-x$ therefore certainly decreases steadily and at a rate which is limited primarily by $\frac{3}{2}n2^{-t}\|A^{-1}\|$. We have not shown that $x^{(s)}-x$ tends to zero however, and indeed it does not. Only t figures are available for the representation of $x^{(s)}$ and the most we can expect is that $x^{(s)}$ will ultimately become $\bar{x}$. This is covered by the term $2^{-t}\|x\|/(1-\beta)$ in relation (39.5).

We now show that if $\|A^{-1}\|$ is appreciably below the bound assumed in (38.1) then the probability is very high that all components of $x^{(s)}$ tend to those of $\bar{x}$, even as regards round-off.

The relations (39.5) and (39.6) guarantee in any case that $\|x^{(s)}-x\|$ will ultimately be less than $(1\cdot2)2^{-t}\|x\|$. Let us consider the effect of one more iteration after this stage is reached. From (38.12) and (38.10) the computed $\delta^{(s)}$ is given by

$$\begin{aligned}\delta^{(s)} &= (A+E+E_1)^{-1}(r^{(s)}+g^{(s)})\\ &= A^{-1}r^{(s)}+[(A+E+E_1)^{-1}-A^{-1}]r^{(s)}+(A+E+E_1)^{-1}g^{(s)}\\ &= (x-x^{(s)})+j\ \text{(say)}.\end{aligned}\tag{39.7}$$

Writing

$$\alpha = \|A^{-1}\|\,\|E+E_1\|\tag{39.8}$$

we have from (38.11), (37.11) and (39.3)

$$\begin{aligned}\|j\| &\leqslant \|[(A+E+E_1)^{-1}-A^{-1}]A(x-x^{(s)})\|+\|(A+E+E_1)^{-1}\|\,2^{-t}\,\|A(x-x^{(s)})\|\\ &\leqslant \frac{\alpha}{1-\alpha}\|x-x^{(s)}\|+\frac{n2^{-t}\|A^{-1}\|}{(1-\alpha)}\|x-x^{(s)}\|\\ &\leqslant \left[\frac{\frac{5n}{2}2^{-t}\|A^{-1}\|}{1-\frac{3n}{2}2^{-t}\|A^{-1}\|}\right](1\cdot2)2^{-t}\|x\|.\end{aligned}\tag{39.9}$$

Hence if $n2^{-t}\|A^{-1}\| \ll 1$

$$\|j\| \leqslant 2^{-t}\|x\| < 2^{-t-q}\|x\|\ \text{(say)},\tag{39.10}$$

and from (39.7)

$$x^{(s)}+\delta^{(s)} = x+j\tag{39.11}$$

showing that the left-hand side differs from x only in digital positions $t+q$ and beyond. Since $x^{(s+1)}$ is obtained by rounding $x^{(s)}+\delta^{(s)}$ it will be the correctly rounded $\bar{x}$ unless there is a very close round-off.

That the $r^{(s)}$ do not decrease is entirely a result of the roundings made when $\delta^{(s)}$ is added to $x^{(s)}$. The residual corresponding to $x^{(s)}+\delta^{(s)}$ *is* considerably smaller than the first residual, but the residual corresponding to $x^{(s+1)}$ contains the extra term $Ah^{(s)}$ and unfortunately $h^{(s)}$ satisfies only the inequality (38.14). Hence from (38.4)

$$\|Ah^{(s)}\| \leqslant n2^{-t}\|x^{(s)}\| \leqslant 1\cdot082n2^{-t}\|x\|.\tag{39.12}$$

Apart from some amelioration resulting from the statistical effect, (39.12) gives a true appraisal of the size of the $r^{(s)}$.

40. The analysis of the last section shows that, provided $n2^{-t}\|A^{-1}\|$ is below a critical value, the practical process really does work. Even if we use somewhat less accurate types of arithmetic, the performance is much the same as with the form of block-floating we have described. Although the upper bound for the residuals may be a little higher with some methods of computing, the improvement in accuracy is usually better than one might expect, primarily because the basic step in an iteration is the solution of triangular sets of equations and the computed solution of these equations tends to be closer to their true solution than the general analysis indicates. However we must always compute $r^{(s)}$ to high accuracy. *If we are using standard floating-point arithmetic for instance, $r^{(s)}$ must be computed in double-precision, or the error made in computing the residual will be comparable with its true value.*

When the iterative procedure is used in practice, we will not normally know $\|A^{-1}\|$. Suppose we have used the process described in the previous section and p more figures 'settle' with each iteration. Can we guarantee that all figures which have settled are correct, and indeed, when the final $x^{(s)}$ has been reached, that $x^{(s)} + \delta^{(s+1)}$ has $p+t$ correct figures? Unfortunately the answer to this question would appear to be that one cannot. If $\frac{5n}{2}\,2^{-t}\|A^{-1}\| < 2^{-p}$ we know that we must gain at least p figures per stage, but there remains the possibility that $\frac{5n}{2}\,2^{-t}\|A^{-1}\| > 1$ and in spite of this the successive iterates *appear* to be gaining in accuracy. That this should happen consistently is hardly credible and it is even more improbable that the same phenomenon should occur with several independent right-hand sides. In practice we have taken the view that unless the matrix of coefficients is very special, we can take the results obtained by iteration at their face value.

It is interesting to apply the iterative technique to the example at the end of section 30. We then find that the second iterate does not agree with the first iterate even in the most significant figure so that the inadequacy of the first iterate is exposed. Note that this really *does* guarantee that $\frac{5n}{2}\,\|A^{-1}\| > 2^t$ and the ill-condition of the matrix is rigorously established.

THE USE OF AN ESTIMATE FOR $\|A^{-1}\|$

41. It does not seem possible to obtain a rigorous estimate of the accuracy of a solution to a set of equations without some estimate for $\|A^{-1}\|$. If we are given a correct estimate of $\|A^{-1}\|$ of the form

$$2^{p-1} \leqslant \|A^{-1}\| < 2^p \tag{41.1}$$

then, provided $\frac{5n}{2}\,\|A^{-1}\|2^{-t} < 1$, we can be certain that the iterative method converges to the correctly rounded solution $\bar{x}$, with an error of less than one in the last digit of each component of the final $x^{(s)}$.

Now

$$\bar{x} = 2^k\bar{y} = 2^k(y+e) \tag{41.2}$$

where

$$\|e\| \leqslant \tfrac{1}{2}2^{-t}. \tag{41.3}$$

Hence if we compute the residual r exactly, as we can by accumulation, we have

$$\begin{aligned} r &= A(x-\bar{x}) \\ &= 2^kAe \end{aligned} \tag{41.4}$$

$$\|r\| \leqslant 2^k\frac{n}{2}2^{-t}. \tag{41.5}$$

Hence

$$x - \bar{x} = A^{-1}r \tag{41.6}$$

$$\|x-\bar{x}\| \leqslant \|A^{-1}\|2^k\frac{n}{2}2^{-t}. \tag{41.7}$$

This means that even if the given $\bar{x}$ *is* the correctly rounded solution we will not be in a position to demonstrate this from the residual alone. We shall merely be able to show that

$$\|x-\bar{x}\|/\|\bar{x}\| \leqslant n\|A^{-1}\|2^{-t} \tag{41.8}$$

which will be a very much weaker result when $\|A^{-1}\| \gg 1$. To achieve more than this we need *an approximate inverse and a bound for its error.*

ASSESSMENT OF A COMPUTED INVERSE

42. If we use block-floating arithmetic for inverting a matrix by the method described in section 36, then each column of the inverse will be obtained as a block-floating vector. If the rth column of the computed inverse be denoted by x_r and

$$x_r = 2^kry_r \tag{42.1}$$

then

$$(A+E+E_r)x_r = e_r \tag{42.2}$$

where

$$\|E\| \leqslant \frac{n}{2}2^{-t}; \qquad \|E_r\| \leqslant n2^{-t}*. \tag{42.3}$$

Hence the computed inverse X satisfies

$$I - AX = EX + F_1 \tag{42.4}$$

where F_1 is $(E_1x_1, E_2x_2, \ldots, E_nx_n)$ so that

$$\|F\| \leqslant (n^2+\tfrac{1}{2}n)2^{-t}\|X\| \tag{42.5}$$

---

* Note that although we have a uniform bound $n2^{-t}$ for $\|E_r\|$ we do not have $|E_r| \leqslant |K|$ with $\||K|\| \leqslant n2^{-t}$ for some K independent of r. This contrasts with the situation in section 27.

Hence

$$A^{-1} - X = A^{-1}F \tag{42.6}$$

$$(n^2+\tfrac{1}{2}n)2^{-t}\|A^{-1}\|\,\|X\| \geqslant \begin{cases}\|A^{-1}\| - \|X\| \\ \|X\| - \|A^{-1}\|.\end{cases} \tag{42.7}$$

Therefore

$$\frac{\|A^{-1}\|}{1+(n^2+\tfrac{1}{2}n)2^{-t}\|A^{-1}\|} \leqslant \|X\| \leqslant \frac{\|A^{-1}\|}{1-(n^2+\tfrac{1}{2}n)2^{-t}\|A^{-1}\|} \tag{42.8}$$

provided

$$(n^2+\tfrac{1}{2}n)2^{-t}\|A^{-1}\| < 1. \tag{42.9}$$

It does not seem possible to reduce the factor $n^2+\frac{1}{2}n$ which occurs in several of the above inequalities although in practice it is usually an exceptionally severe overestimate.

When X has been computed we know, even without computing F, that it will satisfy the relation (42.5). However, we can easily compute F exactly by accumulating inner-products so that we need not accept (42.8), which takes no account of the statistical distribution of errors. The computed F immediately provides us with a rigorous and, in general, realistic estimate of the accuracy of the computed X. In fact we have

$$\frac{\|X - A^{-1}\|}{\|A^{-1}\|} \leqslant \|F\|. \tag{42.10}$$

If we apply the same process to the inversion of A^T the computed inverse Y satisfies

$$I - A^T Y = G \tag{42.11}$$

$$\|G\| \leqslant (n^2+\tfrac{1}{2}n)2^{-t}\|Y\| \tag{42.12}$$

and hence

$$I - Y^T A = G^T. \tag{42.13}$$

We see that we have the same bound for Y^T as a left-hand inverse of A as we had for X as a right-hand inverse.

USE OF THE APPROXIMATE INVERSE TO SOLVE EQUATIONS

43. We now consider the use of the approximate inverse Y^T of the last section to solve the equations $Ax = b$. It is more natural to rename Y^T and G^T, and we call them X and F respectively. We have

$$I - XA = F; \qquad \|F\| \leqslant (n^2+\tfrac{1}{2}n)2^{-t}\|X\| \tag{43.1}$$

$$\|X\| \leqslant \|A^{-1}\|/[1-(n^2+\tfrac{1}{2}n)2^{-t}\|A^{-1}\|]. \tag{43.2}$$

We assume that

$$(n^2+\tfrac{1}{2}n)2^{-t}\|A^{-1}\| \leqslant 0{\cdot}1 \text{ (say)},$$

since otherwise X will not be even an approximate inverse. We have therefore

$$\|F\| \leqslant \left(\frac{2n^2+n}{1\cdot 8}\right) 2^{-t}\|A^{-1}\|. \tag{43.3}$$

As a first approximation to the solution we take $x^{(1)}$ defined by

$$\begin{aligned} x^{(1)} &= Xb \\ &= XAx \\ &= (I-F)x. \end{aligned} \tag{43.4}$$

For this approximation we have

$$\|x^{(1)} - x\| \leqslant \|F\|\,\|x\| \tag{43.5}$$

$$\|x^{(1)} - x\|/\|x\| \leqslant \left(\frac{2n^2+n}{1\cdot 8}\right) 2^{-t}\|A^{-1}\|. \tag{43.6}$$

Notice that the situation is far less satisfactory in theory if we have $I - AX = F$. However, in practice it does not usually make any difference when an elimination technique has been used, because usually $I-AX$ and $I-XA$ are of much the same size (c.f. section 30).

The relation (43.5) is interesting when A is ill-conditioned and b is a normalized right-hand side for which the corresponding x is of order unity. We have

$$\begin{aligned} \|A^{-1} - X\| &= \|FA^{-1}\| \\ &\leqslant \left(\frac{2n^2+n}{1\cdot 8}\right) 2^{-t}\|A^{-1}\|^2 \end{aligned} \tag{43.7}$$

and this is usually quite realistic at least as far as the factor $\|A^{-1}\|^2$ is concerned. If

$$\|A^{-1}\| = (2^k) \text{ (say)} \tag{43.8}$$

then

$$\|A^{-1} - X\| \leqslant \left(\frac{2n^2+n}{1\cdot 8}\right) 2^{2k-t} \tag{43.9}$$

so that X may have absolute errors of order $n^2 2^{2k-t}$. Yet Xb has the error Fx and if $\|x\|$ is of order unity

$$\begin{aligned} \|Fx\| &\leqslant \|F\| \\ &\leqslant \left(\frac{2n^2+n}{1\cdot 8}\right) 2^{k-t}. \end{aligned} \tag{43.10}$$

In other words, *the absolute errors in X will be so related that, in spite of the fact that b is of order unity, the errors in Xb will be far smaller than those in X.*

The bound (43.6) for the relative error in $x^{(1)}$ is roughly n times the bound (38.3) for the error in the solution obtained by solving $Ax = b$ directly. (From our remark of section 42 concerning the factor $n^2 + \frac{1}{2}n$ we shall expect the ratio to be more favourable in practice.)

However, the errors in $x^{(1)}$ are no longer always related so as to make $\|b - Ax^{(1)}\|$ a similar multiple of the norm of the residual corresponding to the correctly rounded solution. In fact we have

$$\begin{aligned} r^{(1)} &= b - Ax^{(1)} \\ &= (I - AX)b \\ &= AFx. \end{aligned} \tag{43.11}$$

In contrast to (38.6) this merely guarantees that

$$\begin{aligned} \|r^{(1)}\| &\leqslant \|A\|\,\|F\|\,\|x\| \\ &\leqslant \left(\frac{2n^3 + n^2}{1\cdot 8}\right) 2^{-t}\|A^{-1}\|\,\|x\|. \end{aligned} \tag{43.12}$$

On the other hand because $I - AX$ is usually of the same order of magnitude as $I - XA$ we have

$$r^{(1)} = (I - AX)b \tag{43.13}$$

$$\begin{aligned} \|r^{(1)}\| &\leqslant \|I - AX\|\,\|b\| \\ &= O\|I - XA\|\,\|b\| \\ &= \|F\|\,\|b\| \\ &\leqslant (n^2 + \tfrac{1}{2}n)2^{-t}\|A^{-1}\|\,\|b\|/[1 - (n^2 + \tfrac{1}{2}n)2^{-t}\|A^{-1}\|]. \end{aligned} \tag{43.14}$$

The reader should contrast the form of this result with (38.9). More significant than the presence of the factor $n^2 + \frac{1}{2}n$ in (43.14) (compared with $\dfrac{3n}{2}$ in (38.9)) is the factor $\|A^{-1}\|\,\|b\|$ where (38.9) has $\|x^{(1)}\|$.

If b is a right-hand side for which the solution truly reflects the condition of A, that is

$$\|A^{-1}b\| = O\|A^{-1}\|\,\|b\| \tag{43.15}$$

then, irrespective of the accuracy of $x^{(1)}$, the residual will be misleadingly small, just as it was in the iterative procedure of section 38. When $\|A^{-1}b\|$ is of the same order of magnitude as $\|b\|$, however, the residual (43.14) is a realistic index of the accuracy of the solution, whereas the method of solution of section 38 still provides misleadingly small residuals. These comments will be illustrated fully in section 45.

ITERATIVE PROCEDURE BASED ON USE OF THE APPROXIMATE INVERSE

44. Using the approximate inverse X of the previous section we may set up an iterative procedure for computing a sequence of approximate solutions $x^{(1)}, x^{(2)}, \ldots$, to the equation $Ax = b$, analogous to that presented in section 37 *et seq.*

The procedure is as follows.

(i) Compute $x^{(1)}$ defined by

$$x^{(1)} = \text{b.f. } (Xb). \tag{44.1}$$

(ii) Compute the exact residual $r^{(s)}$ defined by

$$r^{(s)} = b - Ax^{(s)}. \tag{44.2}$$

$r^{(s)}$ is a non-standardized d.p.b.f. vector.

(iii) Standardize $r^{(s)}$ and round it to give $\bar{r}^{(s)}$, a standardized s.p.b.f. vector.

(iv) Compute $\delta^{(s)}$ defined by

$$\delta^{(s)} = \text{b.f. } (X\bar{r}^{(s)}). \tag{44.3}$$

(v) Compute $x^{(s+1)}$ defined by

$$x^{(s+1)} = \text{b.f. } (x^{(s)} + \delta^{(s)}). \tag{44.4}$$

Ignoring rounding errors in the iterative process itself, we have

$$x^{(s+1)} = x^{(s)} + X(b - Ax^{(s)}) \tag{44.5}$$

$$\begin{aligned} x^{(s+1)} - x &= (I - XA)(x^{(s)} - x) \\ &= (I - XA)^s(x^{(1)} - x) \end{aligned} \tag{44.6}$$

$$\begin{aligned} r^{(s+1)} &= (I - AX)r^{(s)} \\ &= (I - AX)^s r^{(1)} \\ &= A(I - XA)^s(x^{(1)} - x). \end{aligned} \tag{44.7}$$

The behaviour of $x^{(s)}$ is not materially affected by rounding errors but the behaviour of $r^{(s)}$ in practice depends on $\|A^{-1}b\|/\|A^{-1}\|\,\|b\|$, as mentioned in the previous section. The variation in the behaviour of the residuals is illustrated in the example of the next section.

NUMERICAL EXAMPLE

45. The two iterative techniques of sections 38 and 44 are illustrated in Table 3 for the matrix of order three previously analysed in Table 1.

We exhibit the solution of

$$Ax = b$$

with two different right-hand sides b_1 and b_2, which are such that

$$\frac{\|A^{-1}b_1\|}{\|A^{-1}\|\,\|b_1\|} = O(1)$$

$$\frac{\|A^{-1}b_2\|}{\|A^{-1}\|\,\|b_2\|} = O\left(\frac{1}{\|A^{-1}\|}\right).$$

The computation has been performed in block-floating arithmetic using eight decimals throughout.

For the iterative technique of section 38, we find that with both b_1 and b_2 the accuracy of successive iterates improves by about three decimals per iteration. (The first computed solution corresponding to b_1 is 'exceptionally' accurate having regard to the fact that the condition number of A is of order 10^5. This is because the last pivot in U has an exceptionally favourable rounding. Its correct value is 0·00001 17405 . . . and the computed value is 0·00001 174. The same exceptional accuracy does not occur with b_2 because the computed solution of $Ly = b_2$ has its last component equal to 0·00001 075 and this does not have a favourable rounding since the true value is 0·00001 07444 . . ..)

With the right-hand side b_1 we find that the second and third iterates agree exactly, while with b_2 the third and fourth agree exactly. The fortuitous exceptional accuracy of the first iterate which occured with b_1 is maintained in subsequent iterates as we would expect.

As far as the residuals are concerned, we find that from the start they are as small as those corresponding to the final correct solution. With the right-hand side b_1 they remain at the level of 10^{-3} and with the right-hand side b_2 at the level of 10^{-8}.

Turning to the method of section 44, the matrix X was obtained by inverting A^T by triangular decomposition and then taking the transpose of the resulting matrix. This guaranteed that $I-XA$ was as small as possible, but as is usually the case, $I-AX$ was equally small.

Again with b_1 the second and third iterates were identical and with b_2, the third and fourth were identical. About three figures were gained at each stage (as long as this was possible with 8-digit numbers); the exceptional accuracy in the case of b_1 arose from the same cause as before, though here it was A^T which was triangularly decomposed and not A.

The behaviour of the residuals for b_1 is different now from that for b_2. The residuals corresponding to b_1 remain of much the same order of magnitude for each iterate; in fact those corresponding to the first iterate are actually slightly smaller than subsequent residuals. As far as b_2 is concerned though, the variation in the residuals reflects the growing accuracy of the solution. They change from about 10^{-3} to about 10^{-8}.

In all four cases we have performed one iteration after the solution has settled, this being necessary to show that the limiting accuracy has been reached. In each case the correction δ_r is in figures beyond the 8th but, of course, $x_r + \delta_r$ is correct up to the 11th significant figure. It can be seen that the δ_r given by the two methods for this last iteration agree in these extra three figures. To improve the solution further it would be necessary to add the δ_r to the x_r and retain more than 8 figures.

TABLE 3

A			b_1	b_2
0·93216 500	0·44312 600	0·41763 200	0·87613 200	0·87613 200
0·71234 500	0·91531 200	0·88765 200	0·81532 700	0·81532 700
0·63216 500	0·51421 700	0·49390 900	0·91234 500	0·64820 600

FIRST METHOD

$$L = \begin{bmatrix} 1{\cdot}00000\ 000 & & \\ 0{\cdot}76418\ 338 & 1{\cdot}00000\ 000 & \\ 0{\cdot}67816\ 856 & 0{\cdot}37057\ 286 & 1{\cdot}00000\ 000 \end{bmatrix} \qquad U = \begin{bmatrix} 0{\cdot}93216\ 500 & 0{\cdot}44312\ 600 & 0{\cdot}41763\ 200 \\ & 0{\cdot}57668\ 248 & 0{\cdot}56850\ 457 \\ & & 0{\cdot}00001\ 174 \end{bmatrix}$$

Results corresponding to b_1

1st iterate

$$\begin{bmatrix} 464{\cdot}499 \\ -22180{\cdot}655 \\ 22499{\cdot}979 \end{bmatrix}$$

2nd iterate

$$\begin{bmatrix} 464{\cdot}479 \\ -22179{\cdot}679 \\ 22498{\cdot}989 \end{bmatrix}$$

3rd iterate

$$\begin{bmatrix} 464{\cdot}479 \\ -22179{\cdot}679 \\ 22498{\cdot}989 \end{bmatrix}$$

1st residual

$$10^{-3}\begin{bmatrix} 0{\cdot}11946\ 7 \\ 0{\cdot}11689\ 7 \\ 0{\cdot}07888\ 9 \end{bmatrix}$$

2nd residual

$$10^{-3}\begin{bmatrix} -0{\cdot}27252\ 9 \\ -0{\cdot}20523\ 5 \\ -0{\cdot}18369\ 3 \end{bmatrix}$$

1st correction

$$\begin{bmatrix} -0{\cdot}02028\ 552 \\ 0{\cdot}97555\ 852 \\ -0{\cdot}98954\ 685 \end{bmatrix}$$

2nd correction

$$10^{-3}\begin{bmatrix} -0{\cdot}28462\ 289 \\ -0{\cdot}48430\ 146 \\ 0{\cdot}49659\ 284 \end{bmatrix}$$

which leaves the second iterate unaltered.

Results corresponding to b_2

1st iterate

$$\begin{bmatrix} 0{\cdot}83857\ 198 \\ -0{\cdot}64985\ 977 \\ 0{\cdot}91567\ 291 \end{bmatrix}$$

2nd iterate

$$\begin{bmatrix} 0{\cdot}83856\ 142 \\ 0{\cdot}64935\ 439 \\ 0{\cdot}91516\ 025 \end{bmatrix}$$

3rd iterate

$$\begin{bmatrix} 0{\cdot}83856\ 142 \\ -0{\cdot}64935\ 441 \\ 0{\cdot}91516\ 028 \end{bmatrix}$$

4th iterate

$$\begin{bmatrix} 0{\cdot}83856\ 142 \\ -0{\cdot}64935\ 441 \\ 0{\cdot}91516\ 028 \end{bmatrix}$$

1st residual

$$10^{-8}\begin{bmatrix} 0{\cdot}19552 \\ -0{\cdot}12021\ 8 \\ -0{\cdot}56918 \end{bmatrix}$$

2nd residual

$$10^{-8}\begin{bmatrix} 0{\cdot}18208\ 4 \\ 0{\cdot}44567\ 8 \\ 0{\cdot}23710\ 8 \end{bmatrix}$$

3rd residual

$$10^{-8}\begin{bmatrix} -0{\cdot}18456 \\ -0{\cdot}38665\ 4 \\ -0{\cdot}21618\ 5 \end{bmatrix}$$

1st correction

$$10^{-3}\begin{bmatrix} -0{\cdot}01056\ 048 \\ 0{\cdot}50538\ 087 \\ -0{\cdot}51265\ 549 \end{bmatrix}$$

2nd correction

$$10^{-7}\begin{bmatrix} -0{\cdot}00015\ 969 \\ -0{\cdot}21353\ 731 \\ 0{\cdot}27052\ 811 \end{bmatrix}$$

3rd correction

$$10^{-8}\begin{bmatrix} -0{\cdot}00161\ 451 \\ -0{\cdot}13453\ 335 \\ -0{\cdot}29557\ 070 \end{bmatrix}$$

which leaves the third iterate unaltered.

SECOND METHOD

Decomposition of A^T

$$L = \begin{bmatrix} 1{\cdot}00000\ 000 & & \\ 0{\cdot}47537\ 292 & 1{\cdot}00000\ 000 & \\ 0{\cdot}44802\ 369 & 0{\cdot}98581\ 903 & 1{\cdot}00000\ 000 \end{bmatrix} \qquad U = \begin{bmatrix} 0{\cdot}93216\ 500 & 0{\cdot}71234\ 500 & 0{\cdot}63216\ 500 \\ & 0{\cdot}57668\ 248 & 0{\cdot}21370\ 288 \\ & & 0{\cdot}00001\ 174 \end{bmatrix}$$

Computed inverse X of A from this L and U

$$\begin{bmatrix} -696{\cdot}636 & -651{\cdot}315 & 1755{\cdot}365 \\ 33165{\cdot}770 & 31119{\cdot}092 & -83970{\cdot}957 \\ -33644{\cdot}203 & -31564{\cdot}980 & 85178{\cdot}876 \end{bmatrix}$$

Results corresponding to b_1

1st iterate

$$\begin{bmatrix} 464{\cdot}499 \\ -22180{\cdot}654 \\ 22499{\cdot}978 \end{bmatrix}$$

2nd iterate

$$\begin{bmatrix} 464{\cdot}479 \\ -22179\ 679 \\ 22498{\cdot}989 \end{bmatrix}$$

3rd iterate

$$\begin{bmatrix} 464{\cdot}479 \\ -22179{\cdot}679 \\ 22498{\cdot}989 \end{bmatrix}$$

1st residual

$$10^{-4}\begin{bmatrix} 0{\cdot}93973 \\ 0{\cdot}89237 \\ 0{\cdot}58581 \end{bmatrix}$$

2nd residual

$$10^{-3}\begin{bmatrix} -0{\cdot}27252\ 9 \\ -0{\cdot}20523\ 5 \\ -0{\cdot}18369\ 3 \end{bmatrix}$$

1st correction

$$\begin{bmatrix} -0{\cdot}02028\ 547 \\ 0{\cdot}97455\ 868 \\ -0{\cdot}98854\ 707 \end{bmatrix}$$

2nd correction

$$10^{-3}\begin{bmatrix} -0{\cdot}28476\ 148 \\ -0{\cdot}48397\ 475 \\ 0{\cdot}49640\ 062 \end{bmatrix}$$

which leaves the second iterate unaltered.

Results corresponding to b_2

1st iterate

$$\begin{bmatrix} 0{\cdot}83898\ 823 \\ -0{\cdot}64982\ 842 \\ 0{\cdot}91518\ 520 \end{bmatrix}$$

2nd iterate

$$\begin{bmatrix} 0{\cdot}83856\ 130 \\ -0{\cdot}64935\ 394 \\ 0{\cdot}91515\ 988 \end{bmatrix}$$

3rd iterate

$$\begin{bmatrix} 0{\cdot}83856\ 142 \\ -0{\cdot}64935\ 441 \\ 0{\cdot}91516\ 028 \end{bmatrix}$$

4th iterate

$$\begin{bmatrix} 0{\cdot}83856\ 142 \\ -0{\cdot}64935\ 441 \\ 0{\cdot}91516\ 028 \end{bmatrix}$$

1st residual

$$10^{-3}\begin{bmatrix} -0{\cdot}19822\ 042 \\ 0{\cdot}10770\ 692 \\ -0{\cdot}03838\ 072 \end{bmatrix}$$

2nd residual

$$10^{-7}\begin{bmatrix} 0{\cdot}68797\ 78 \\ 0{\cdot}06479\ 02 \\ 0{\cdot}29579\ 56 \end{bmatrix}$$

3rd residual

$$10^{-8}\begin{bmatrix} -0{\cdot}18456\ 0 \\ -0{\cdot}38665\ 4 \\ -0{\cdot}21618\ 5 \end{bmatrix}$$

1st correction

$$10^{-3}\begin{bmatrix} -0{\cdot}42692\ 676 \\ 0{\cdot}47448\ 224 \\ -0{\cdot}02531\ 611 \end{bmatrix}$$

2nd correction

$$10^{-6}\begin{bmatrix} 0{\cdot}12002\ 006 \\ -0{\cdot}47139\ 340 \\ 0{\cdot}39706\ 039 \end{bmatrix}$$

3rd correction

$$10^{-8}\begin{bmatrix} -0{\cdot}00169\ 236 \\ -0{\cdot}13457\ 032 \\ -0{\cdot}29542\ 546 \end{bmatrix}$$

which leaves the third iterate unaltered.

SENSITIVITY OF THE EIGENVALUES OF A MATRIX

46. An exhaustive treatment of the eigenvalue problem is beyond the scope of this book, but two examples are given (in sections 46 to 54 and 55 to 58 respectively) which illustrate the nature of the error analysis associated with this problem.

We consider first the sensitivity of the eigenvalues of a matrix with respect to changes in its elements and we restrict ourselves to the case when A is similar to a diagonal matrix. We have then

$$P^{-1}AP = \operatorname{diag}(\lambda_i) \tag{46.1}$$

for some non-singular matrix P. The matrix P is not unique for clearly

$$\begin{aligned} D^{-1}P^{-1}APD &= D^{-1}\operatorname{diag}(\lambda_i)D \\ &= \operatorname{diag}(\lambda_i) \end{aligned} \tag{46.2}$$

where D is any non-singular diagonal matrix. If A has multiple eigenvalues we have an even wider choice for P.

Suppose λ is an eigenvalue of the matrix $A+E$. Then for some non-zero x we have

$$\begin{aligned} (A+E)x &= \lambda x \\ (\lambda I - A)x &= Ex. \end{aligned} \tag{46.3}$$

Now either

(i) λ is an eigenvalue of A

or

(ii) λ is not an eigenvalue of A. In this case we may proceed as follows. From (46.3)

$$P^{-1}(\lambda I - A)P(P^{-1}x) = (P^{-1}EP^{-1})P^{-1}x \tag{46.4}$$

giving

$$[\operatorname{diag}(\lambda - \lambda_i)]P^{-1}x = (P^{-1}EP)P^{-1}x. \tag{46.5}$$

Since λ is not equal to any λ_i, $\operatorname{diag}(\lambda - \lambda_i)$ is non-singular and we have

$$P^{-1}x = [\operatorname{diag}(\lambda - \lambda_i)^{-1}](P^{-1}EP)P^{-1}x \tag{46.6}$$

and hence

$$\begin{aligned} \|P^{-1}x\| &\leqslant \|\operatorname{diag}(\lambda - \lambda_i)^{-1}\| \, \|P^{-1}EP\| \, \|P^{-1}x\| \\ 1 &\leqslant \|\operatorname{diag}(\lambda - \lambda_i)^{-1}\| \, \|P^{-1}EP\|. \end{aligned} \tag{46.7}$$

For any of the p-norms of section 2 we have

$$\|\operatorname{diag}(\lambda - \lambda_i)^{-1}\|_p = \max \frac{1}{|\lambda - \lambda_i|} \tag{46.8}$$

and hence

$$\min |\lambda - \lambda_i| \leqslant \|P^{-1}\| \, \|E\| \, \|P\|. \tag{46.9}$$

Since in case (i) we have $\min |\lambda - \lambda_i| = 0$ the relation (46.9) is true in all cases. Every eigenvalue of $A+E$ therefore lies in at least one of the circular discs defined by (46.9). Application of (2.10) shows that $\|P\| \, \|P^{-1}\|$ can never be less than 1, which is therefore the best possible condition number for the eigenvalue problem. The relation (46.9) was first derived by Bauer and Fike [2].

The matrix P has its columns equal to n independent eigenvectors of A and (46.9) shows that it is the number $\|P\|\,\|P^{-1}\|$ which determines the overall sensitivity of the eigenvalues to perturbations of A. Since P is to some extent arbitrary, we may regard $\min \|P\|\,\|P^{-1}\|$ taken over all permissible P as a condition number. *Notice that* (46.9) *holds whether or not* $\|E\|$ *is small compared with* $\|A\|$.

If A is Hermitian, we may take P to be unitary and then, using the 2-norm, we have,

$$\min |\lambda - \lambda_i| \leqslant \|E\|_2 \tag{46.10}$$

since $\|P\|_2 = 1$ for a unitary matrix (section 2). The determination of the eigenvalues of an Hermitian matrix is therefore always a perfectly conditioned problem however poorly separated the eigenvalues may be.

The matrix of eigenvectors of any similarity transform, HAH^{-1}, of A, is given by HP. If H is a unitary matrix then $\|HP\|_2 = \|P\|_2$, showing that the condition of the eigensystem of a matrix is unaltered by a similarity transformation with a unitary matrix. When the matrix H of a transformation is not unitary then it may improve or worsen the condition of the matrix to which it is applied. If A is transformed to real diagonal form by H its condition *with respect to the eigenvalue problem* cannot have deteriorated since a real diagonal matrix is Hermitian and therefore perfectly conditioned.

There is much to be said for unitary transformations, even though they cannot actually improve the condition of the eigensystem. If the condition of A is bad, we are almost certain to suffer from this in any case, and protection against worsening of condition is very valuable. A unitary transformation has the additional advantage that it does not amplify errors made in a previous stage. Thus if we apply a sequence of unitary transformations and make errors at each stage, we have

$$A_{r+1} = R_r A_r R_r^H + E_r \tag{46.11}$$

where R_r is unitary and E_r is the error made in the current stage. Hence

$$A_f = P_1 A_1 P_1^H + (P_2 E_1 P_2^H + P_3 E_2 P_3^H + \ldots + P_{f-1} E_{f-2} P_{f-1}^H + E_{f-1}) \tag{46.12}$$

where $P_r = R_{f-1} R_{f-2} \ldots R_r$ and is unitary. The 2-norm of the term in brackets on the right of (46.12) is certainly bounded by the sum of the 2-norms of the E_r and hence the cumulative effect is merely additive.

We can obtain a little more detailed information about the perturbation of the eigenvalues if we make use of the concept of continuity. Consider the matrices $A + kE$ for values of k from 0 to 1, and let us denote its eigenvalues by $\lambda_i(k)$. When $k = 0$ we have $\lambda_i(k) = \lambda_i$ and for every value of k every eigenvalue of $A + kE$ lies in at least one of the circular discs

$$|\lambda_i - \lambda| \leqslant k\|P\|\,\|E\|\,\|P^{-1}\|. \tag{46.13}$$

If s of the circular discs (46.9) form a connected domain isolated from the others then, from the continuity of the $\lambda_i(k)$ with respect to k, exactly s of the eigenvalues of $A + E$ lie in this domain. In particular, if the circular disc centred on λ_i is isolated, then this disc contains just one eigenvalue of $A + E$.

47. Relation (46.9) gives us overall information on the sensitivity of the eigenvalues, but some eigenvalues may be much less sensitive than others. We now investigate the factors influencing the sensitivity of individual eigenvalues. Again we use the relation

$$P^{-1}(A+E)P = \operatorname{diag}(\lambda_i) + P^{-1}EP. \tag{47.1}$$

Now the columns of P are eigenvectors of A and the rows of P^{-1} are eigenvectors of A^T. We denote these vectors by u_i and v_i respectively and it is well known that they form a biorthogonal set of vectors. It is convenient to normalize the vectors so that

$$\|u_i\|_2 = \|v_i\|_2 = 1 \tag{47.2}$$

in which case we may take the columns of P to be u_i and the rows of P^{-1} to be $\left(\frac{1}{s_i}\right) v_i^T$ where

$$s_i = v_i^T u_i = \cos\theta_i, \tag{47.3}$$

θ_i being the angle between v_i and u_i.

Now the matrix on the right of (47.1) has the same eigenvalues as $A + E$. If we write

$$E = \epsilon B \quad \text{and} \quad P^{-1}EP = \epsilon C \tag{47.4}$$

then

$$c_{ij} = (v_i^T B u_j)/s_i. \tag{47.5}$$

We are therefore interested in the eigenvalues of

$$\operatorname{diag}(\lambda_i) + \epsilon C \tag{47.6}$$

and these are the same as those of

$$\operatorname{diag}(\lambda_i) + \epsilon DCD^{-1} \tag{47.7}$$

where D is any diagonal matrix. The following two theorems due to Gerschgorin [6] enable us to locate these eigenvalues.

Theorem 1. Every eigenvalue of a matrix G lies in at least one of the circular discs, centre g_{ii} and radius $\sum_{j\neq i} |g_{ij}|$.

Theorem 2. If k of the circular discs form a connected domain which is disjoint from the remaining discs, then precisely k eigenvalues lie in this connected domain. Theorem 2 is most frequently used with $k = 1$ to locate a single eigenvalue in one of the circular discs.

If we take $d_{ii} = k\epsilon$, $d_{jj} = 1\,(j\neq i)$ in (47.7) then the circular discs for $A + E$ are as follows.

One disc with centre $\lambda_i + \epsilon c_{ii}$ and radius r_i and $n-1$ discs with centres $\lambda_s + \epsilon c_{ss}$ and radii r_s $(s \neq i)$ where

$$r_i = k\epsilon^2 \sum_{j \neq i} |c_{ij}| \tag{47.8}$$

$$r_s = \epsilon \sum_{j \neq i,\, s} |c_{sj}| + \frac{1}{k} |c_{si}| \quad (s \neq i). \tag{47.9}$$

If λ_i is an isolated eigenvalue then we may choose k so that

$$\frac{1}{k} |c_{si}| < |\lambda_i - \lambda_s| \quad (s \neq i). \tag{47.10}$$

Then for sufficiently small ϵ the disc with radius r_i is isolated from the others and therefore contains just one eigenvalue. We have in fact

$$\lambda_i(\epsilon) \sim \lambda_i + \epsilon(v_i^T B u_i / s_i) \quad (\epsilon \to 0). \tag{47.11}$$

We may obviously introduce ϵ in (47.4) so that $\|B\|_2 = 1$ and then

$$\left| \frac{v_i^T B u_i}{s_i} \right| \leqslant \frac{1}{|s_i|} = \frac{1}{|\cos \theta_i|} \tag{47.12}$$

showing that the sensitivity of λ_i is primarily dependent on $\dfrac{1}{|s_i|}$. When A is symmetric we have $u_i = v_i$ and hence $\cos \theta_i = 1$, showing again that all the eigenvalues are well-conditioned.

Notice that if we transform A to RAR^T where R is orthogonal then u_i and v_i become Ru_i and Rv_i respectively and we have

$$\begin{aligned} (Rv_i)^T(Ru_i) &= v_i^T R^T R u_i \\ &= v_i^T u_i \end{aligned} \tag{47.13}$$

showing that *the sensitivity of each individual eigenvalue is invariant with respect to orthogonal similarity transformations.*

EXAMPLE OF ILL-CONDITIONED EIGENVALUES

48. Matrices may be constructed having well-separated eigenvalues which are very ill-conditioned. Consider the matrix A of order 20 defined by

$$\left.\begin{aligned} a_{ii} &= i \qquad a_{i,i+1} = 20 \\ a_{ij} &= 0 \text{ otherwise}. \end{aligned}\right\} \tag{48.1}$$

This is a triangular matrix and hence its characteristic equation is $\prod_{i=1}^{20} (i - \lambda) = 0$. If the element $a_{20,1}$ is changed from zero to ϵ, the characteristic equation becomes

$$\prod_{i=1}^{20} (i - \lambda) = 20^{19}\epsilon. \tag{48.2}$$

Hence

$$\frac{\partial \lambda_i}{\partial \epsilon} = \pm \left(\frac{20^{19}}{(20-i)!\,(i-1)!} \right). \tag{48.3}$$

This is a very large quantity for all relevant i, It takes its maximum value when $i = 10$ or 11 for which we have

$$\left| \frac{\partial \lambda_{10}}{\partial \epsilon} \right| = \left| \frac{\partial \lambda_{11}}{\partial \epsilon} \right| = \frac{20^{19}}{10!\ 9!} \doteqdot (0{\cdot}4)10^{12} \tag{48.4}$$

and its minimum when $i = 1$ or 20 for which

$$\left| \frac{\partial \lambda_{20}}{\partial \epsilon} \right| = \left| \frac{\partial \lambda_1}{\partial \epsilon} \right| = \frac{20^{19}}{19!} \doteqdot (0{\cdot}4)10^{8}. \tag{48.5}$$

It will readily be verified that each u_i is almost orthogonal to the corresponding v_i. We have, for example,

$$u_{10}^T = \left[1; \frac{9!}{8!20}; \frac{9!}{7!20^2}; \ldots; \frac{9!}{1!20^8}; \frac{9!}{20^9}; \quad 0; \quad 0; \ldots; \quad 0; 0 \right]$$

$$v_{10}^T = \left[0; \quad 0; \quad 0; \ldots; \quad 0; \frac{10!}{10^{10}}; \frac{-10!}{1!20^9}; \frac{10!}{2!20^8}; \ldots; \frac{-10!}{9!20}; 1 \right] \tag{48.6}$$

giving $v_{10}^T u_{10} = \dfrac{9!10!}{20^{19}}$, so that clearly $\cos\theta_{10} < \dfrac{9!10!}{20^{19}}$. (48.7)

Notice, however, that the eigenvalues of this matrix are not very sensitive to changes of the diagonal elements and those immediately above and below the diagonal. This follows from (47.11) and the form of the vectors u_i and v_i given in (48.6) and (48.7). If we compute the eigenvalues using a technique which deals explicitly with tri-diagonal matrices, then we might hope that each computed eigenvalue will be the exact eigenvalue of some matrix $A+E$ where E is also of tri-diagonal form. In this case we would obtain accurate eigenvalues in spite of the ill-condition of the matrix A taken as a whole. However, if we were to supply A as the data for a more general technique, then there is a danger that each computed eigenvalue will be the eigenvalue of some matrix $A + F$ where F is non-null in some of the more sensitive elements.

A POSTERIORI ESTIMATES FOR A COMPUTED EIGENVALUE AND EIGENVECTOR OF A REAL SYMMETRIC MATRIX

49. Suppose we are given an alleged eigenvalue λ and eigenvector x of a real symmetric matrix A. The vector η defined by

$$\eta = Ax - \lambda x \tag{49.1}$$

may be called the *residual vector corresponding to* x *and* λ. If λ and x were exact, then η would be null. We might therefore expect a norm of η to give us some information on the accuracy of λ and x, provided x is suitably normalized.

We assume that

$$\|x\|_2 = 1; \qquad \|\eta\|_2 = \epsilon. \tag{49.2}$$

Since A is symmetric there exists an orthogonal R such that

$$RAR^T = \operatorname{diag}(\lambda_i).$$

Hence (49.1) gives

$$[\operatorname{diag}(\lambda_i - \lambda)]Rx = R\eta. \tag{49.3}$$

Two cases arise:

(i) $\lambda = \lambda_i$ for some i.

(ii) $\lambda \neq \lambda_i$ for any i, in which case $\operatorname{diag}(\lambda_i - \lambda)$ is non-singular. Hence

$$Rx = [\operatorname{diag}(\lambda_i - \lambda)^{-1}]R\eta.$$

$$\|Rx\|_2 \leqslant \|\operatorname{diag}(\lambda_i - \lambda)^{-1}\|_2 \|R\eta\|_2$$

$$1 \leqslant \max |\lambda_i - \lambda|^{-1}\epsilon \tag{49.4}$$

$$\min |\lambda_i - \lambda| \leqslant \epsilon. \tag{49.5}$$

Hence (49.5) holds in either case, showing that there is at least one eigenvalue in the interval $\lambda \pm \epsilon$. There may, of course, be several.

We cannot deduce any useful information about x unless we have some information about the eigensystem in general. Suppose, however, we know that there are exactly s eigenvalues in the interval $\lambda \pm \epsilon$ and that the remaining $n-s$ are outside the interval $\lambda \pm a$ for some a. We may number the s eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_s$. If $u_1, u_2, \ldots, u_n$ form an orthogonal system of eigenvectors then we may write

$$x = \sum_1^n \alpha_i u_i \tag{49.6}$$

$$\sum_1^n \alpha_i^2 = 1. \tag{49.7}$$

Hence

$$\eta = Ax - \lambda x = \sum_1^n \alpha_i(\lambda_i - \lambda)u_i \tag{49.8}$$

$$\epsilon^2 = \sum_1^n \alpha_i^2(\lambda_i - \lambda)^2$$

$$\geqslant \sum_{s+1}^n \alpha_i^2(\lambda_i - \lambda)^2 \geqslant \sum_{s+1}^n \alpha_i^2 a^2. \tag{49.9}$$

We have therefore

$$\left(\sum_{s+1}^{n} \alpha_i^2\right)^{1/2} \leqslant \frac{\epsilon}{a} \tag{49.10}$$

$$\left(\sum_{1}^{s} \alpha_i^2\right)^{1/2} > \left(1 - \frac{\epsilon^2}{a^2}\right)^{1/2}. \tag{49.11}$$

If $a \gg \epsilon$ then x has negligible components of $u_{s+1}, \ldots, u_n$. The most important case is when $s = 1$, in which case

$$x = u_1 + f \tag{49.12}$$

where

$$\|f\|^2 \leqslant \frac{\epsilon^2}{a^2} + \left[1 - \left(1 - \frac{\epsilon^2}{a^2}\right)^{1/2}\right]^2$$

$$\doteq \frac{\epsilon^2}{a^2} \text{ if } a \gg \epsilon. \tag{49.13}$$

It is perhaps worth remarking that (49.1) is equivalent to

$$(A - \eta x^T)x = \lambda x. \tag{49.14}$$

The λ and x are therefore exact for the matrix $A - \eta x^T$ and we have

$$\|\eta x^T\|_2 = \epsilon. \tag{49.15}$$

As a standard for comparison we obtain a bound for the residual vector corresponding to $\bar{\lambda}$ and $\bar{x}$ obtained by rounding an exact eigenvalue λ and eigenvector x to t binary places. If we are working in fixed-point then

$$\bar{x} = x + f \qquad \|f\| \leqslant \tfrac{1}{2}n^{1/2}2^{-t} \tag{49.16}$$

$$\bar{\lambda} = \lambda + \epsilon \qquad |\epsilon| < \tfrac{1}{2}2^{-t}. \tag{49.17}$$

Hence

$$\eta = A\bar{x} - \bar{\lambda}\bar{x} = A(x+f) - (\lambda+\epsilon)(x+f)$$

$$= (A - \lambda I)f - \epsilon(x+f). \tag{49.18}$$

If $|a_{ij}| \leqslant 1$ then

$$\|\eta\|_2 \leqslant 2n\|f\|_2 + |\epsilon|(\|x\|_2 + \|f\|_2)$$

$$\leqslant n^{3/2}2^{-t} + \tfrac{1}{2}2^{-t}(1 + \tfrac{1}{2}n^{1/2}2^{-t})$$

$$\doteq n^{3/2}2^{-t}. \tag{49.19}$$

In practice we can scarcely expect any method to give a better result than this in general.

50. We now use the perturbation theory to assess the effectiveness of a method of computing the eigenvectors of a symmetric triple-diagonal matrix. There are several methods for determining the eigenvalues of such a matrix to high accuracy, so we assume that the eigenvalues are given. The mathematical basis of the method is that usually known as *inverse* iteration or *Wielandt* iteration [**24**]. Starting from an arbitrary normalized vector x_1 and an approximate eigenvalue λ, the sequence of vectors defined by

$$(A-\lambda I)x_{r+1} = x_r \tag{50.1}$$

is computed.

If we ignore rounding errors the process may be analysed very simply. We may expand x_1 in terms of a complete orthonormal system of eigenvectors $u_1, u_2, \ldots, u_n$ in the form

$$x_1 = \sum \alpha_i u_i . \tag{50.2}$$

We have then

$$x_{r+1} = \sum \alpha_i \left(\frac{1}{\lambda_i - \lambda}\right)^r u_i . \tag{50.3}$$

If λ is close to an isolated eigenvalue λ_1 then we may write

$$\lambda = \lambda_1 + \epsilon \tag{50.4}$$

$$\begin{aligned} x_{r+1} &= \frac{\alpha_1}{\epsilon^r} u_1 + \sum_2^n \alpha_i \left(\frac{1}{\lambda_i - \lambda}\right)^r u_i \\ &= \frac{1}{\epsilon^r}\left[\alpha_1 u_1 + \epsilon^r \sum_2^n \alpha_i \left(\frac{1}{\lambda_i - \lambda}\right)^r u_i\right]. \end{aligned} \tag{50.5}$$

If

$$|\lambda_i - \lambda| \gg \epsilon \quad (i = 2 \ldots n); \quad \alpha_1 \neq 0 , \tag{50.6}$$

then x_{r+1} is ultimately proportional to u_1. In the application we are proposing, if we assume $|a_{ij}| \leqslant 1$, then the given value of λ will have an error which is of order 2^{-t} if we use, for example, the bisection technique and the Sturm sequence property [**7**]. Hence, unless α_1 is pathologically small or λ is very close to some other λ_i, convergence to u_1 will be very rapid even if the initial conditions are unfavourable. Suppose for example

$$t = 48, \quad \epsilon = 2^{-48}, \quad \min_{i \neq 1} |\lambda_i - \lambda| = 2^{-20} \text{ and } \alpha_1 = 2^{-20} \tag{50.7}$$

so that the conditions are extremely unfavourable in that λ_1 is fairly poorly separated and the initial vector is quite deficient in u_1. After only two iterations we have

$$\begin{aligned} x_3 &= 2^{96}\left[2^{-20}u_1 + 2^{-96}\sum_2^n \alpha_i \left(\frac{1}{\lambda_i - \lambda}\right)^2 u_i\right] \\ &= 2^{76}u_1 + f_3 \end{aligned} \tag{50.8}$$

where

$$f_3 = \sum_2^n \alpha_i \left(\frac{1}{\lambda_i - \lambda}\right)^2 u_i \tag{50.9}$$

$$\|f_3\| \leqslant 2^{40} \left(\sum_2^n \alpha_i^2\right)^{1/2}$$

$$\leqslant 2^{40}. \tag{50.10}$$

Similarly after three iterations

$$x_4 = 2^{124} u_1 + f_4 \tag{50.11}$$

$$\|f_4\| \leqslant 2^{60} \tag{50.12}$$

and hence

$$x_4 = 2^{124}(u_1 + g_4) \tag{50.13}$$

$$\|g_4\| < 2^{-64}. \tag{50.14}$$

The normalized x_4 would therefore be correct to more than 64 binary places in spite of the unfavourable circumstances.

EFFECT OF ROUNDING ERRORS

51. We now consider the effect of the rounding errors made in one iteration. We denote the initial normalized vector by b so that the relevant iteration consists of solving the equations

$$(A - \lambda I)x = b. \tag{51.1}$$

The matrix $A - \lambda I$ is first triangularly decomposed using Gaussian elimination with partial pivoting. (Since $A - \lambda I$ is not positive definite we cannot safely use the LL^T decomposition.) Now for a general matrix we know that with partial pivoting, elements of the reduced matrices *can* greatly exceed those of the original matrix. However, for a tri-diagonal matrix this is not true as we shall now show.

We shall write

$$a_{ii} = \alpha_i \qquad a_{i,i+1} = a_{i+1,i} = \beta_i \tag{51.2}$$

and assume

$$|a_{ij}| \leqslant 1. \tag{51.3}$$

From this last assumption it follows that

$$|\lambda| \leqslant 3. \tag{51.4}$$

Elimination with partial pivoting is particularly simple because at each stage there are only two rows involving the current variable, of which one is an unmodified row of $(A - \lambda I)x$. Consider the elimination of x_i. The two relevant left-hand sides may be written in the form

$$p_i x_i + q_i x_{i+1} \tag{51.5}$$

$$\beta_i x_i + (\alpha_{i+1} - \lambda) x_{i+1} + \beta_{i+1} x_{i+2} \tag{51.6}$$

as will be seen from the analysis of the ith stage. We shall prove by induction that

$$|p_i| \leqslant 5 \text{ and } |q_i| < 1. \tag{51.7}$$

We assume (51.7) to be true up to the current step, which then proceeds as follows.

Either (i) $|p_i| \geqslant |\beta_i|$.

In this case there is no interchange and

$$m_i = \beta_i/p_i \quad |m_i| \leqslant 1 \tag{51.8}$$

$$p_{i+1} = (\alpha_{i+1} - \lambda) - m_i q_i \qquad |p_{i+1}| \leqslant (1+3) + 1 \times 1 = 5 \tag{51.9}$$

$$q_{i+1} = \beta_{i+1} \qquad |q_{i+1}| \leqslant 1 \tag{51.10}$$

and $p_i x_i + q_i x_{i+1}$ is the ith pivotal row.

Or (ii) $|p_i| < |\beta_i|$.

In this case there is an interchange and

$$m_i = p_i/\beta_i \quad |m_i| < 1 \tag{51.11}$$

$$p_{i+1} = q_i - m_i(\alpha_{i+1} - \lambda) \qquad |p_{i+1}| < 1 + 1(3+1) = 5 \tag{51.12}$$

$$q_{i+1} = -m_i \beta_{i+1} \qquad |q_{i+1}| < 1 \times 1 = 1 \tag{51.13}$$

and $\beta_i x_i + (\alpha_{i+1} - \lambda) x_{i+1} + \beta_{i+1} x_{i+2}$ is the ith pivotal row.

This proof shows that the matrices L and U are of the form

$$\begin{bmatrix} 1 & & & \\ \cdot & 1 & & \\ \cdot & \cdot & & \\ m_1 & \cdot & & \\ \cdot & m_2 & 1 & \\ \cdot & \cdot & m_{n-1} & 1 \end{bmatrix} \text{ and } \begin{bmatrix} u_1 & v_1 & w_1 & & & & \\ & u_2 & v_2 & w_2 & & & \\ & & - & - & - & - & - \\ & & & & u_{n-2} & v_{n-2} & w_{n-2} \\ & & & & & u_{n-1} & v_{n-1} \\ & & & & & & u_n \end{bmatrix} \tag{51.14}$$

respectively where there is one non-zero sub-diagonal element in each column of L, and w_i is non-zero only if there is an interchange at the ith stage. Clearly

$$\|L\|_2 \leqslant (2n)^{1/2}; \qquad \|U\|_2 \leqslant 6. \tag{51.15}$$

52. This argument shows that in the case of a tri-diagonal matrix with elements bounded by unity we have an *a priori* guarantee that no element arising in the elimination process will have a modulus exceeding 5. Notice that the proof did not require the symmetry of A and the result is true even when λ is complex.

The computed L and U satisfy

$$LU = (A - \lambda I)^* + E_1 \tag{52.1}$$

where $(A - \lambda I)^*$ denotes $A - \lambda I$ with its rows suitably permuted. Because of the interchanges, E_1 will not usually be tri-diagonal. (For example, if

interchanges take place at every stage, E_1 is null everywhere except in the last row.) However, there are at most three non-zero elements in any column of E. Hence we certainly have

$$\|E_1\|_2 < k_1 n^{1/2} 2^{-t} \tag{52.2}$$

where k_1 is a constant of order unity which depends on the type of arithmetic used.

Similarly the errors made in the back-substitution will mean that the computed solution corresponds to $(L+\delta L)(U+\delta U)$ where δL and δU are non-zero only in the positions occupied by the non-zero elements of L and U respectively. If we write $(L+\delta L)(U+\delta U) = LU + E_2$, then from the form of L and U we must have

$$\|E_2\|_2 < k_2 n^{1/2} 2^{-t} \tag{52.3}$$

where k_2 is a constant of order unity. The computed solution therefore satisfies

$$(A-\lambda I+F)x = b \tag{52.4}$$

where

$$\|F\|_2 < 2^{-t}(k_1 + k_2)n^{1/2}2^{-t} < K_1 n^{1/2}2^{-t} \text{ (say)}, \tag{52.5}$$

since the 2-norm is not affected by row permutations.

Now observe that if we can show that $\|x\|_2$ is very large then x is bound to be a 'satisfactory' eigenvector.* For if we write

$$y = x/\|x\|_2 \tag{52.6}$$

then

$$(A-\lambda I+F)y = \frac{b}{\|x\|_2} \tag{52.7}$$

$$(A-\lambda I)y = -Fy + \frac{b}{\|x\|_2} \tag{52.8}$$

$$\|(A-\lambda I)y\| \leqslant \|F\| + \frac{1}{\|x\|_2}$$

$$\leqslant K_1 n^{1/2}2^{-t} + \frac{1}{\|x\|_2}. \tag{52.9}$$

Hence if

$$\|x\|_2 > 2^t/Kn^{1/2} \text{ (say)}, \tag{52.10}$$

$$\|(A-\lambda I)y\| \leqslant 2Kn^{1/2}2^{-t} \tag{52.11}$$

and from equation (49.19) we see that this would be an exceptionally 'good' result.

53. Now we are assuming that λ is a very accurate eigenvalue in the sense that there is at least one eigenvalue of A such that

$$|\lambda_i - \lambda| = O(2^{-t}) = K_2 2^{-t}. \tag{53.1}$$

* Observe that we cannot expect to do better than to obtain an eigenvector of some matrix $A + G$ with a small $\|G\|_2$. Just how accurate an eigenvector of A this is, will inevitably depend on the condition of the appropriate vector.

Let us denote the λ_i satisfying this relation by $\lambda_1, \lambda_2, \ldots, \lambda_s$. The only danger then is that b should be so deficient in the eigenvectors corresponding to λ_1 to λ_s that relation (52.4) would not ensure that the proportion of such eigenvectors in x is higher than the proportion in b.

We show that $\|x\|_2 \gg \|b\|_2 = 1$ unless b is quite exceptionally deficient in these eigenvectors. We write

$$b = \sum_1^s c_i u_i + \sum_{s+1}^n c_i u_i \tag{53.2}$$

$$\left(\sum_1^n c_i^2\right)^{1/2} = 1 \tag{53.3}$$

$$x = \sum_1^s d_i u_i + \sum_{s+1}^n d_i u_i . \tag{53.4}$$

Hence from (52.4)

$$b = \sum_1^n (\lambda_i - \lambda) d_i u_i + Fx . \tag{53.5}$$

Equating components of the vectors in the sub-space spanned by $u_1, u_2, \ldots, u_s$, we have

$$\begin{aligned}\left(\sum_1^s c_i^2\right)^{1/2} &\leqslant \left[\sum_1^s (\lambda_i - \lambda)^2 d_i^2\right]^{1/2} + \|Fx\|_2 \\ &\leqslant K_2 2^{-t} \left(\sum_1^s d_i^2\right)^{1/2} + K_1 n^{1/2} 2^{-t} \|x\|_2 \\ &\leqslant (K_2 2^{-t} + K_1 n^{1/2} 2^{-t}) \|x\|_2 . \end{aligned} \tag{53.6}$$

Hence

$$\|x\|_2 \geqslant 2^t \left(\sum_1^s c_i^2\right)^{1/2} \Big/ (K_2 + K_1 n^{1/2}) . \tag{53.7}$$

Unless $\left(\sum_1^s c_i^2\right)^{1/2}$ is quite exceptionally small, $\|x\|_2$ will be much greater than unity. Hence, provided we can guard against a quite pathological deficiency of b, we can expect two iterations to produce *as good a vector as we will be able to obtain using t-digit arithmetic.* We are in any case limited to finding the eigenvector of LU rather than A, but if the tri-diagonal matrix has been obtained by a previous computation, the errors made in this are almost certain to be more important than those corresponding to E_1 in (52.1). Notice that compared with the exact solution of $(A - \lambda I)x_{r+1} = x_r$ the computed solution will usually be incorrect, even in the most significant figure. Our argument shows, nevertheless, that x_{r+1} will be an accurate eigenvector when $s = 1$ and λ is therefore close to an isolated eigenvalue. In fact, as in the example of section 30, the computed solution is almost exactly proportional to the true solution in the sense that (true solution) $- k$ (computed solution) is very small for some appropriate constant k. For high order matrices this almost constant ratio of corresponding elements

of the true solution and of the computed solution may not arise in such an obvious manner as in the example of section 30. The reader will find it instructive to carry out the computation with the matrix of (30.1). [See also Chapter 1, the end of section 36.]

54. In our experience, inverse iteration has proved to be the most successful of methods we have used for computing eigenvectors of a tri-diagonal matrix from accurate eigenvalues. However, it is important to pay a little attention to the choice of b. One might be tempted, for convenience, to take b equal to the first or last column, e_1 or e_n, of the unit matrix. *Such a vector, however, is often pathologically deficient in some of the eigenvectors of* A. If one of the off-diagonal elements of A is zero, this is obvious since A is then the direct sum of two tri-diagonal matrices A_1 and A_2, and the eigenvectors of A_1 may be converted into eigenvectors of A by supplying the appropriate number of zero components, and similarly for A_2. The eigenvectors are therefore all orthogonal either to e_1 or e_n. However, even when none of the off-diagonal elements is at all small it is very common for an eigenvector to be almost orthogonal to e_1 or e_n [**25**].

We have found the following technique to be very satisfactory in practice. We choose b so that

$$Le = b \tag{54.1}$$

where e is the vector having a unit element in every position. We never actually compute b, but instead compute our first x as the solution of

$$Ux = e. \tag{54.2}$$

The first iteration is thus actually simpler than subsequent iterations. We have found that the first x is usually already a very good eigenvector so that the b which corresponds to (54.1) has a component of the relevant eigenvector that is not at all small. Note that this b is a function of λ, so that we are using a different initial vector for each eigenvalue. Two iterations have always sufficed to give a vector that is as good as could be obtained with t-digit arithmetic. If we normalize each x_r to give y_r before performing the next iteration, then we have a reliable indication of the effectiveness of the current step. (We need only normalize so that $\|y_r\|_\infty = 1$, or even so that $\frac{1}{2} \leqslant \|y_r\|_\infty < 1$; the latter involves only multiplication by powers of two.) If $\|x_{r+1}\|_\infty / \|y_r\|_\infty = O(n^{-1/2}2^t)$ then (52.11) shows that x_{r+1} is as good a vector as the process can give. We have therefore only to choose some appropriate K and stop when $\|x_{r+1}\|_\infty / \|y_r\|_\infty > Kn^{-1/2}2^t$.

CALCULATION OF THE EIGENVALUES OF A LOWER HESSENBERG MATRIX

55. Our second example has been chosen so as to illustrate the advantages of floating-point accumulation. We consider the calculation of the eigenvalues of a lower Hessenberg matrix H by the method of Hyman [**14**]. The essential feature of the technique is the derivation of an auxiliary function whose zeros are also the zeros of $\det(H - \lambda I)$.

Now if we add any multiples of columns 2 to n to column 1 of $H-\lambda I$ its determinant is unaltered. Accordingly we define $x_2, x_3, \ldots, x_n$ to be such that, if we replace the first column of $H-\lambda I$ by column $1+\sum_{i=2}^{n}(x_i \times \text{column } i)$ the first $n-1$ elements of that column vanish. The relevant algorithm is:

$$x_1 = 1 \tag{55.1}$$

$$-x_{i+1} = h_{i1}x_1 + h_{i2}x_2 + \ldots + (h_{ii}-\lambda)x_i/h_{i,i+1} \quad (i = 1, \ldots, n-1). \tag{55.2}$$

It follows that

$$\det(H-\lambda I) = (-1)^{n+1}h_{12}h_{23}\ldots h_{n-1,n}[h_{n1}x_1 + h_{n2}x_2 + \ldots + (h_{nn}-\lambda)x_n] \tag{55.3}$$

and the zeros of $\det(H-\lambda I)$ therefore coincide with those of

$$h_{n1}x_1 + h_{n2}x_2 + \ldots + (h_{nn}-\lambda)x_n = f(\lambda) \text{ (say)}. \tag{55.4}$$

Consider now the practical realisation of this technique, using floating-point arithmetic, for a matrix with elements satisfying

$$|h_{ij}| \leqslant 1. \tag{55.5}$$

In standard floating-point we have

$$-x_{i+1} \equiv fl[h_{i1}x_1 + h_{i2}x_2 + \ldots + (h_{ii}-\lambda)x_i/h_{i,i+1}] \tag{55.6}$$

where $x_1, x_2, \ldots, x_i$ are standard floating-point numbers obtained in the previous steps. Now we need only consider values of λ satisfying

$$|\lambda| \leqslant \|H\|_E = [\tfrac{1}{2}(n^2+3n-2)]^{1/2} < \frac{1}{\sqrt{2}}(n+2). \tag{55.7}$$

Equation (55.6) gives

$$-x_{i+1} = \frac{h_{i1}x_1(1+E_{i1}) + h_{i2}x_2(1+E_{i2}) + \ldots + h_{i,i-1}x_{i-1}(1+E_{i,i-1}) + (h_{ii}-\lambda)x_i(1+\epsilon_{i1})(1+E_{i1})}{h_{i,i+1}(1+\epsilon_{i2})} \tag{55.8}$$

where from section 26 of Chapter 1,

$$|E_{is}| \leqslant (i+2-s)2^{-t_1} \tag{55.9}$$

$$|\epsilon_{ij}| \leqslant 2^{-t_1}. \tag{55.10}$$

The computed value of $f(\lambda)$ is

$$h_{n1}x_1(1+E_{n1}) + h_{n2}x_2(1+E_{n2}) + \ldots + (h_{nn}-\lambda)x_n(1+\epsilon_{n1})(1+E_{nn}). \tag{55.11}$$

It is therefore the exact value for the matrix $H+\delta H$, where δH is a function of λ but is uniformly bounded. For example, when $n = 5$ we have

$$|\delta H| < 2^{-t_1}\begin{bmatrix} 3 & 1 & & & \\ 3 & 3 & 1 & & \\ 4 & 3 & 3 & 1 & \\ 5 & 4 & 3 & 3 & 1 \\ 6 & 5 & 4 & 3 & 3 \end{bmatrix} + |\lambda|2^{-t_1}\begin{bmatrix} 3 & & & & \\ & 3 & & & \\ & & 3 & & \\ & & & 3 & \\ & & & & 3 \end{bmatrix} \tag{55.12}$$

provided $n2^{-t} < 0{\cdot}1$.

The right-hand side, F, of (55.12) is such that

$$\|F\|_E \sim \left(\frac{n^2}{\sqrt{12}} + 3\lambda\right)(1{\cdot}06)2^{-t_1}$$

$$\sim \frac{1}{\sqrt{12}}\, n^2 2^{-t_1} \tag{55.13}$$

from (55.7).

If the eigenvalues of H are $\lambda_1, \lambda_2, \ldots, \lambda_n$ and those of $H + \delta H$ are $\lambda_1', \lambda_2', \ldots, \lambda_n'$, then denoting the computed value of $f(\lambda)$ by $\bar{f}(\lambda)$, we have

$$\begin{aligned} \bar{f}(\lambda) &= \frac{\Pi(\lambda_i'-\lambda)(-1)^{n+1}}{h_{12}h_{23}\ldots h_{n-1,n}(1+\epsilon_{12})(1+\epsilon_{22})\ldots(1+\epsilon_{n-1,2})} \\ &= P(1+E)\,\Pi(\lambda_i'-\lambda) \end{aligned} \tag{55.14}$$

where

$$P = (-1)^{n+1}/h_{12}h_{23}\ldots h_{n-1,n} \tag{55.15}$$

$$(1-2^{-t})^{n-1} \leqslant 1+E \leqslant (1+2^{-t})^{n-1}. \tag{55.16}$$

CALCULATION OF $f(\lambda)$ USING FLOATING-POINT ACCUMULATION

56. Although the result of the previous section is by no means unsatisfactory, we can achieve a substantially better result if we use floating-point accumulation. We then have

$$-x_{i+1} = fl_2\left[\frac{h_{i1}x_1 + h_{i2}x_2 + \ldots + h_{ii}x_i - \lambda x_i}{h_{i,i+1}}\right] \tag{56.1}$$

where the $x_1, \ldots, x_i$ are standard single-precision numbers obtained in the previous stages. Notice that we have used $h_{ii}x_i - \lambda x_i$ and not $(h_{ii}-\lambda)x_i$ so as to avoid a rounding error when forming $h_{ii} - \lambda$. Now the numerator may be accumulated in floating-point and the mantissa of $h_{i,i+1}$ then divided into the double-precision mantissa of the numerator. We have therefore

$$-x_{i+1} = \frac{\begin{aligned} h_{i1}x_1(1+E_{i1}) + h_{i2}x_2(1+E_{i2}) + \ldots + h_{ii}x_i(1+E_{ii}) - \\ - \lambda x_i(1+E_{i,i+1}) \end{aligned}}{h_{i,i+1}(1+\epsilon_i)} \tag{56.2}$$

where from section 32 of Chapter 1

$$|E_{is}| \leqslant \tfrac{3}{2}(i+3-s)2^{-2t_2} \tag{56.3}$$

$$|\epsilon_i| \leqslant 2^{-t} \tag{56.4}$$

The computed $f(\lambda)$ is now the exact value for the matrix $H + \delta H$ where, typically for $n = 5$, we have

$$|\delta H| < (1\cdot5)\,2^{-2t_2}\begin{bmatrix}3&0&0&0&0\\4&3&0&0&0\\5&4&3&0&0\\6&5&4&3&0\\7&6&5&4&3\end{bmatrix} + (1\cdot5)\,2^{-2t_2}\begin{bmatrix}2\lambda&&&&\\&2\lambda&&&\\&&2\lambda&&\\&&&2\lambda&\\&&&&2\lambda\end{bmatrix}$$

$$+\,2^{-t}\begin{bmatrix}0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\\0&0&0&0&0\end{bmatrix}. \tag{56.5}$$

The three matrices on the right have 2-norms which are bounded by $(1\cdot5)2^{-2t_2}\dfrac{(n+2)^2}{\sqrt{12}}$, $(3\cdot0)2^{-2t_2}\dfrac{(n+2)}{\sqrt{2}}$, and 2^{-t} respectively. Hence provided $n^2 2^{-t}$ is appreciably less than unity the third matrix is the most important.

PERTURBATION OF THE EIGENVALUES

57. If λ_i is an isolated eigenvalue of H then from (47.11), for sufficiently large t, the corresponding eigenvalue λ_i' of $H + \delta H$ will satisfy

$$|\lambda_i' - \lambda_i| \leqslant \frac{\|\delta H\|_2}{|s_i|}(1 + A_i 2^{-t}) \tag{57.1}$$

where A_i is independent of t.

If $|s_i|$ is small, λ_i is bound to be a sensitive eigenvalue and we are almost certain to suffer from this with any technique. The method we have just described reduces $\|\delta H\|_2$ to an exceptionally low level when we use floating-point accumulation.

Starting from this method of calculating $\det(H - \lambda I)$, we can use any of a number of techniques for locating the zeros of a computable function. Newton's method, Laguerre's method, successive linear interpolation and Muller's successive quadratic interpolation have all been used. As with the calculation of the zeros of polynomials discussed in Chapter 2, the fundamental limitation on the attainable accuracy is our ability to evaluate $\det(H - \lambda I)$ in the neighbourhood of a zero. For a given precision of

computation, all of these methods, based on the calculation of $\det(H-\lambda I)$ using floating-point accumulation, give results which, in our experience, are more accurate than those obtainable by any other method.

For the method of bisection the analysis of the process is almost identical with that given in Chapter 2. If λ_i is an isolated zero, then in the neighbourhood of λ_i, $\bar{f}(\lambda)$ has the same sign as $f(\lambda)$ unless λ lies in the interval

$$|\lambda_i' - \lambda_i| \leqslant \frac{\|\delta H\|_2}{|s_i|}(1+A_i 2^{-t}) \tag{57.2}$$

surrounding λ_i. The λ_i' of limiting accuracy therefore has an error which is effectively bounded by $2^{-t}/|s_i|$ and it seems inherently improbable that we can do better than this with t-digit computation. Even without floating-point accumulation, the method gives high accuracy, and when the statistical effect is taken into account, we see from (55.13) that we can expect λ_i' to satisfy some such relation as

$$|\lambda_i' - \lambda_i| \leqslant Kn2^{-t}/|s_i|\,. \tag{57.3}$$

The calculation of x_{i+1} in equation (55.2) involves a division by the element $h_{i,i+1}$, and it might well be thought that if this element is small the process would be unstable. Such fears are completely unfounded. The perturbation corresponding to this division is $h_{i,i+1}\epsilon_{i2}$ or $h_{i,i+1}\epsilon_i$ in the notations of (55.8) and (56.2) respectively. *Hence the smaller $h_{i,i+1}$, the smaller is the corresponding perturbation.* This is not absolutely conclusive since there remains the possibility that the eigensystems of matrices having small $h_{i,i+1}$ are particularly sensitive to perturbations in these elements. This is not so however; there would appear to be no correlation between small $h_{i,i+1}$ and ill-conditioned eigensystems. On the whole then, small $h_{i,i+1}$ are conducive to higher rather than lower accuracy and if an $h_{i,i+1}$ is zero we may safely replace it by a small value and continue with the process if this is desirable from the point of view of simplicity of programme.

NUMERICAL EXAMPLE

58. A remarkable example of the effectiveness of floating-point accumulation is provided by the results obtained for the matrix H_{12} defined by

$$H_{12} = \begin{bmatrix} 12 & 11 & & & & & \\ 11 & 11 & 10 & & & & \\ 10 & 10 & 10 & 9 & & & \\ - & - & - & - & - & - & - \\ 2 & 2 & 2 & 2 & \ldots & 2 & 1 \\ 1 & 1 & 1 & 1 & \ldots & 1 & 1 \end{bmatrix}. \tag{58.1}$$

The eigenvalues are given in Table 4 together with the values of s_i. It will be seen that the smaller eigenvalues are very ill-conditioned with values of s_i of order 10^{-7} to 10^{-8}, while the larger eigenvalues are well-conditioned.

That some of the eigenvalues must be ill-conditioned follows immediately from consideration of $\det(H_{12})$ and $\det(H_{12}+\delta H)$ for suitable δH. In fact if δH is $\epsilon E_{12,1}$ (that is, is null apart from the (12, 1) element) then we have, by subtracting each column from the column on its left,

$$\det(H_{12}) = \prod_{i=1}^{n} \lambda_i = 1 \tag{58.2}$$

$$\det(H_{12}+\epsilon E_{12,1}) = \prod_{i=1}^{n} \lambda_1' = 1 - 11!\epsilon$$

$$\doteqdot 1 - (0\cdot 399)10^8\epsilon. \tag{58.3}$$

On the other hand

$$\det(H_{12}+\epsilon E_{p,p+1}) = 1 - \epsilon. \tag{58.4}$$

Some of the eigenvalues *must* therefore be sensitive to changes in the (12, 1) element, but it seems likely that they are all insensitive to change in any $(p, p+1)$ element.

We give the left-hand and right-hand eigenvectors corresponding to λ_{10}, λ_{11} and λ_{12}, the three most sensitive eigenvalues. Notice that each u_i is almost orthogonal to the corresponding v_i. The perturbation in λ_i corresponding to a perturbation ϵE_{pq} in H is given, for sufficiently small ϵ, by

$$\delta\lambda_i \doteqdot \frac{\epsilon v_i^T E_{pq} u_i}{v_i^T u_i} = \frac{\epsilon v_{ip} u_{iq}}{v_i^T u_i} \tag{58.5}$$

where v_{ip} and u_{iq} are the pth component of v_i and the qth component of u_i respectively. It will readily be verified that $|v_{ip}u_{i,p+1}|$ is considerably smaller than $|s_i|$ for all p from 1 to 11 and for $i = 10, 11, 12$. Hence even the sensitive eigenvalues are not much affected by perturbations in the elements of the super-diagonal.

TABLE 4

Eigenvalues of H_{12}

$\lambda_1 = 32\cdot 22889\ 15$	$\lambda_5 = 3\cdot 51185\ 595$	$\lambda_9 = 0\cdot 14364\ 6520$
$\lambda_2 = 20\cdot 19898\ 86$	$\lambda_6 = 1\cdot 55398\ 871$	$\lambda_{10} = 0\cdot 08122\ 76592\ 40405$
$\lambda_3 = 12\cdot 31107\ 74$	$\lambda_7 = 0\cdot 64350\ 5319$	$\lambda_{11} = 0\cdot 04950\ 74291\ 85278$
$\lambda_4 = 6\cdot 96153\ 309$	$\lambda_8 = 0\cdot 28474\ 9721$	$\lambda_{12} = 0\cdot 03102\ 80606\ 44010$

Last three eigenvalues computed using standard floating-point

$\lambda_{10} = 0\cdot 08122\ 76460$ $\lambda_{11} = 0\cdot 04950\ 74419$ $\lambda_{12} = 0\cdot 03102\ 79318$

Last three eigenvalues computed using floating-point accumulation

$$\lambda_{10} = 0{\cdot}08122\ 76592\ 40403 \quad \text{Error } 2 \times 10^{-15}$$
$$\lambda_{11} = 0{\cdot}04950\ 74291\ 85279 \quad \text{Error } 1 \times 10^{-15}$$
$$\lambda_{12} = 0{\cdot}03102\ 80606\ 44008 \quad \text{Error } 2 \times 10^{-15}$$

Values of $s_i = \cos\theta_i = v_i^T u_i$

$s_1 = +0{\cdot}30424\ 083$	$s_5 = +0{\cdot}14446\ 703$	$s_9 = +0{\cdot}00000\ 01498$
$s_2 = -0{\cdot}20079\ 033$	$s_6 = -0{\cdot}00462\ 656$	$s_{10} = -0{\cdot}00000\ 00375$
$s_3 = +0{\cdot}31822\ 599$	$s_7 = +0{\cdot}00006\ 913$	$s_{11} = +0{\cdot}00000\ 00258$
$s_4 = -0{\cdot}58447\ 355$	$s_8 = -0{\cdot}00000\ 178$	$s_{12} = -0{\cdot}00000\ 00547$

Note that the first five eigenvalues are quite well-conditioned; λ_4 in particular is almost as well-conditioned as an eigenvalue of a symmetric matrix. Because the last four s_i are so small they have been given to extra decimal places.

Right-hand eigenvectors corresponding to λ_{10}, λ_{11} *and* λ_{12}

u_{10}	u_{11}	u_{12}
$-0{\cdot}67714\ 11$	$-0{\cdot}67599\ 46$	$-0{\cdot}67531\ 89$
$+0{\cdot}73369\ 91$	$+0{\cdot}73440\ 62$	$+0{\cdot}73480\ 66$
$-0{\cdot}05625\ 42$	$-0{\cdot}06061\ 69$	$-0{\cdot}06315\ 65$
$-0{\cdot}00084\ 53$	$+0{\cdot}00211\ 69$	$+0{\cdot}00385\ 87$
$+0{\cdot}00060\ 06$	$+0{\cdot}00011\ 26$	$-0{\cdot}00019\ 87$
$-0{\cdot}00006\ 06$	$-0{\cdot}00002\ 68$	$+0{\cdot}00000\ 91$
$+0{\cdot}00000\ 09$	$+0{\cdot}00000\ 29$	$-0{\cdot}00000\ 04$
$+0{\cdot}00000\ 07$	$-0{\cdot}00000\ 02$	$+0{\cdot}00000\ 00$
$-0{\cdot}00000\ 01$	$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$
$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$
$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$
$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$

Left-hand eigenvectors corresponding to λ_{10}, λ_{11} *and* λ_{12}

v_{10}	v_{11}	v_{12}
$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$
$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$
$+0{\cdot}00000\ 00$	$+0{\cdot}00000\ 00$	$-0{\cdot}00000\ 03$
$-0{\cdot}00000\ 10$	$+0{\cdot}00000\ 03$	$+0{\cdot}00000\ 38$
$+0{\cdot}00001\ 10$	$-0{\cdot}00001\ 41$	$-0{\cdot}00004\ 51$
$-0{\cdot}00002\ 37$	$+0{\cdot}00021\ 95$	$+0{\cdot}00043\ 20$
$-0{\cdot}00068\ 17$	$-0{\cdot}00221\ 72$	$-0{\cdot}00331\ 87$
$+0{\cdot}00946\ 06$	$+0{\cdot}01596\ 10$	$+0{\cdot}02009\ 76$
$-0{\cdot}06504\ 47$	$-0{\cdot}08251\ 35$	$-0{\cdot}09284\ 49$
$+0{\cdot}26984\ 68$	$+0{\cdot}29459\ 26$	$+0{\cdot}30854\ 36$
$-0{\cdot}64994\ 86$	$-0{\cdot}65581\ 13$	$-0{\cdot}65861\ 25$
$+0{\cdot}70740\ 99$	$+0{\cdot}68997\ 00$	$+0{\cdot}67970\ 23$

The eigenvalue problem was solved on ACE using Hyman's method of evaluating $\det(H_{12}-\lambda I)$ and several different techniques for locating the zeros. When standard floating-point computation was used ($t = 46$) the last three eigenvalues obtained by successive linear interpolation were as shown. The underlined are the first incorrect figures; the errors are of the order 10^{-7} to 10^{-8}. This was to be expected since we were using effectively 14-decimal computation and the relevant s_i are of order 10^{-7} to 10^{-8}. The rounding errors made in the evaluation are equivalent to perturbations of order 2^{-t} in the elements of H. (Results of the same accuracy were obtained by bisection, Newton's method and Muller's method.)

Using floating-point accumulation again with $t = 46$, *we obtained the sensitive eigenvalues with errors of order* 10^{-15}. *These are equivalent to* 1 *or* 2 *in the last figure retained.* Again much the same accuracy was obtained using the various different methods of locating the zeros as our analysis has led us to expect. The rounding errors made in the evaluation are equivalent to perturbations of order 2^{-2t} in elements on and below the diagonal. The factors associated with these perturbations are not greater than $12 \times 15 \times 1{\cdot}5$ so that, even ignoring the statistical effect, these perturbations are negligible. The equivalent perturbations in the super-diagonal are of order 2^{-t}, but the eigenvalues are least sensitive to changes here. Although this example presents floating-point accumulation in a particularly favourable light, it is perhaps worth mentioning that it is quite common for eigenvalues to be far less sensitive to changes in near-diagonal elements than in those which are more remote.

For matrices of the same form as H_{12} but of higher order, the effect is even more marked. For $n = 17$ some of the s_i are of order 10^{-13} and standard floating-point gives no accuracy in the smaller zeros. With floating-point accumulation, almost full accuracy is still attained.

Additional comments

We have given only an elementary treatment of vector and matrix norms but it should be adequate to cover the requirements in error analysis of matrix processes. The notation $\|x\|_p$ and $\|A\|_p$ for the Hölder norms has been preferred to those of Householder [11] and Faddeeva [4] since it was felt that many readers will already be acquainted with the concepts in this notation in connexion with classical analysis. We would like to stress the value of the Euclidean norm $\|A\|_E$ for floating-point error analysis. For a more extensive treatment of matrix norms and their application the reader is referred to numerous papers by Householder and Bauer [2], [3], [11], [12] and [13].

The method of Lanczos [15] for the computation of the eigenvalues of a symmetric matrix has been discussed in numerous papers [15], [26]. The need for reorthogonalization of each computed vector with respect to earlier vectors is often attributed to the *accumulation* of rounding errors. In fact *accumulation* of errors plays an insignificant role and it is cancellation which is the main problem. By an analysis similar to that of sections 7–10 it can be shown that, provided reorthogonalization is used, the normalized Lanczos vectors x_i form the columns of a matrix X such that

$$AX = XT + E \tag{1}$$

where the columns of X are orthogonal to working accuracy, T is tri-diagonal and the elements of E satisfy

$$|e_{ij}| \leqslant 2^{-t} f(n) \tag{2}$$

for some simple function $f(n)$ of n which depends on the type of arithmetic which is used. Without reorthogonalization the columns of X may fall almost arbitrarily short of orthogonality and no such relations as (1) and (2) may hold.

The first published error analysis of techniques based on Gaussian elimination were those of Turing [23] and of von Neumann and Goldstine [18]. The latter paper may be said to have laid the foundation for modern error analysis, in its rigorous treatment of the inversion of a positive definite matrix by elimination. It is, however, a fairly difficult paper to read, and in the author's experience comparatively few computers understand why it is that the cumulative effect of the rounding errors is so much less than was at one time feared. In the paper 'Rounding errors in algebraic processes' [28] we attempted to show in the simplest possible manner, without any pretensions to rigour, how this comes about. In a later paper [30] we gave a completely rigorous analysis, based on a similar approach, which covered not only positive definite matrices but also general non-symmetric matrices.

The role played by pivoting in elimination techniques has often been misunderstood. It was employed by von Neumann and Goldstine for a positive definite matrix so that fixed-point computation without scaling should be possible. This has sometimes given the impression that it was essential for stability, though in fact for a positive definite matrix it is quite irrelevant.

For matrices of a more general type, some form of pivoting is usually of fundamental importance and the use of floating-point arithmetic does not remove the need for it. The terms *partial* and *complete* pivoting were introduced by the author in [30], where their significance was discussed in some detail. An outstanding problem in this field is the determination of the maximum attainable value of the ratio of the nth pivotal element of a matrix of order n to the maximum element of the original matrix. A bound was given in [30] but it is evident that this was a severe overestimate.

The analysis in sections 37–46 of the convergence of the method of successive refinement of a solution of $Ax = b$, using either the approximate triangular decomposition of A or an approximate inverse, has not appeared elsewhere, though the technique has been quite widely used. Our main object here was to show for which parts of the computation inner-product accumulation is essential and to explain the behaviour of the residuals corresponding to the successive approximations.

The condition of a matrix with respect to the eigenvalue problem was discussed by Bauer and Fike [2] and section 46 follows their treatment very closely. Section 47 on the sensitivity of the individual eigenvalues was developed by the author in connexion with the determination of rigorous *a posteriori* error bounds [3]. Section 49 covers mainly classical theory. For an elegant exposition and generalization of these results based on matrix norms, the reader is referred to the paper by Bauer and Householder [3].

The calculation of the eigenvectors of a symmetric tri-diagonal matrix is of fundamental importance since the methods of Lanczos, Givens and Householder all reduce a general real symmetric matrix to this form by orthogonal similarity transformations. A satisfactory

numerical solution of this problem is rather more difficult than might be expected. The discussion given here was sketched in a paper by the author [25]. An outstanding requirement in this field is a simple automatic procedure which will give a set of vectors which are orthogonal to working accuracy and which accurately span the invariant subspace corresponding to pathologically close (possibly coincident) eigenvalues.

We have included an analysis of Hyman's method because it is commonly believed that it is unstable, whereas it is in fact one of the most stable which has yet been developed. The version developed by Parlett [22] in which Hyman's method is combined with that of Laguerre is both fast and accurate.

Perhaps the most satisfactory error analyses in the matrix field have been those connected with orthogonal transformations. We have excluded these on the grounds that they would have extended the size of the book unduly. The first such analysis was that of Givens' [8] for his own techniques of computing eigenvalues of real symmetric matrices. Although later analyses have been much more concise the broad outlines of Givens work have generally been followed. The reader is referred to papers by Householder [12], Goldstine *et al.* [9], Ortega [21] and Wilkinson [32].

Since this book went to press McKeeman **[16a]** has published a procedure in BALGOL for the accurate solution of linear equations. This is based on the techniques described in Section 37, but uses floating-point arithmetic with accumulation of all relevant inner-products. The general behaviour of the residuals and the successive approximations follow very closely the pattern we have described for block-floating computation.

BIBLIOGRAPHY

1. BAREISS, E. H. 1960 Resultant procedure and the mechanization of the Graeffe process. *J. Assoc. comp. Mach.* **7,** 346–386.

2. BAUER, F. L. and FIKE, C. T. 1960 Norms and exclusion theorems. *Numer. Math.* **2,** 137–141.

3. BAUER, F. L. and HOUSEHOLDER, A. S. 1960 Moments and characteristic roots. *Numer. Math.* **2,** 42–53.

4. FADDEEVA, V. N. 1959 *Computational methods of linear algebra.* (Translated by C. D. Benster.) New York: Dover; London: Constable.

5. FORSYTHE, G. E. 1958 Singularity and near singularity in numerical analysis. *Amer. math. Mon.* **65,** 229–240.

6. GERSCHGORIN, S. 1931 Über die Abgrenzung der Eigenwerte einer Matrix. *Izv. Akad. Nauk. SSSR, Ser. fiz.-mat. Nauk.* **6,** 749–754.

7. GIVENS, W. 1953 A method of computing eigenvalues and eigenvectors suggested by classical results on symmetric matrices. *Appl. Math. Ser. nat. Bur. Stand.* **29,** 117–122.

8. GIVENS, W. 1954 Numerical computation of the characteristic values of a real symmetric matrix. Oak Ridge National Laboratory, ORNL–1574.

9. GOLDSTINE, H. H., MURRAY, F. J. and VON NEUMANN, J. 1959 The Jacobi method for real symmetric matrices. *J. Assoc. comp. Mach.* **6,** 59–96.

10. GOURSAT, E. fifth edition 1933, *Cours d'analyse mathématique, Tome II.* Paris: Gauthier Villars.

11. HOUSEHOLDER, A. S. 1958 The approximate solution of matrix problems. *J. Assoc. comp. Mach.* **5,** 205–243.

12. HOUSEHOLDER, A. S. 1958 Generated error in rotational tridiagonalization. *J. Assoc. comp. Mach.* **5,** 335–338.

13. HOUSEHOLDER, A. S. *The theory of matrices in numerical analysis.* In the press.

14. HYMAN, M. A. 1957 *Eigenvalues and eigenvectors of general matrices.* Presented at the 12th National meeting of the Association for Computing Machinery, Houston, Texas.

15. LANCZOS, C. 1950 An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. nat. Bur. Stand.* **45,** 255–282.

16. LEHMER, D. H. 1961 A machine method for solving polynomial equations. *J. Assoc. comp. Mach.* **8,** 151–162.

16a. MCKEEMAN, W. M. 1963 An accurate algorithm for the solution of simultaneous linear algebraic equations. Applied Mathematics and Statistics Laboratories, Stanford University, *Tech. Rep.* No. 26.

17. MULLER, D. E. 1956 A method for solving algebraic equations using an automatic computer. *Math. Tab., Wash.* **10,** 208–215.

18. VON NEUMANN, J. and GOLDSTINE, H. H. 1947 Numerical inverting of matrices of high order. *Bull. Amer. math. Soc.* **53,** 1021–1099.

19. OLVER, F. W. J. 1952 The evaluation of zeros of high-degree polynomials. *Phil. Trans. roy. Soc.* A, **244,** 385–415.

20. ORTEGA, J. M. 1960 On Sturm sequences for tridiagonal matrices. Applied Mathematics and Statistics Laboratories, Stanford University, *Tech. Rep.* No. 4.

21. ORTEGA, J. M. 1962 An error analysis of Householder's method for the symmetric eigenvalue problem. Applied Mathematics and Statistical Laboratories, Stanford University, *Tech. Rep.* No. 18.

22. PARLETT, B. 1962 Applications of Laguerre's method to the matrix eigenvalue problem. Applied Mathematics and Statistical Laboratories, Stanford University, *Tech. Rep.* No. 21.

23. TURING, A. M. 1948 Rounding-off errors in matrix processes. *Quart. J. Mech.* **1,** 287–308.

24. WIELANDT, H. 1944 Das iterationsverfahren bei nicht selbstadjungierten linearen Eigenwertanfgaben. *Bericht der aerodynamischen Versuchsanstalt Gottingen,* 44/J/37.

25. WILKINSON, J. H. 1958 The calculation of the eigenvectors of codiagonal matrices. *Computer J.* **1,** 90–96.

26. WILKINSON, J. H. 1958 The calculation of eigenvectors by the method of Lanczos. *Computer J.* **1,** 148–152.

27. WILKINSON, J. H. 1959 The evaluation of the zeros of ill-conditioned polynomials. *Numer. Math.* **1,** 150–180.

28. WILKINSON, J. H. 1960 Rounding errors in algebraic processes. *Information Processing,* 44–53. Paris: UNESCO; Munich: Oldenbourg; London: Butterworths.

29. WILKINSON, J. H. 1960 Error analysis of floating-point computation. *Numer. Math.* **2,** 319–340.

30. WILKINSON, J. H. 1961 Error analysis of direct methods of matrix inversion. *J. Assoc. comp. Mach.* **8,** 281–330.

31. WILKINSON, J. H. 1961 Rigorous error bounds for computed eigensystems. *Computer J.* **4,** 230–241.

32. WILKINSON, J. H. 1962 Error analysis of eigenvalue techniques based on orthogonal transformations. *J. Soc. indus. appl. Math.* **10,** 162–195.

INDEX

S.O. Code No. 48–120–32