Numerische Mathematik 1, 150—166 (1959)

The evaluation of the zeros of ill-conditioned polynomials. Part I

By

J. H. WILKINSON

1. Introduction

The design of a general purpose subroutine for the calculation of zeros of polynomials presents considerable difficulty. This is mainly because many polynomials which arise in practice are such that small changes in the coefficients produce much larger changes in some of the zeros. If the subroutine were intended merely for the calculation of the zeros of polynomials of which the coefficients were the primary data and were subject to end figure errors, then it would be reasonable to limit its scope to the accurate determination of those figures in the zeros which remained unaltered when the last figures of the data were changed. Since primary data is seldom accurate to more than ten figures, a subroutine based on the use of single precision floating arithmetic would normally be adequate and subroutines of this type have been written for a number of digital computers.

However if a subroutine for finding zeros of polynomials is to be used as a step in the solution of some larger problem it must meet much more exacting requirements. It may well happen that although the numbers which we seek to determine by calculating the zeros of a polynomial are well determined in the original problem in the sense that they undergo small changes when small changes are made in the data, the same is not true of their dependence on the coefficients of the polynomial. As is shown below this will often be the case when we attempt to determine the eigenvalues of a matrix by calculating its characteristic polynomial and finding its zeros. Because of the importance of the algebraic eigenvalue problem, the design of a general purpose routine for finding zeros of polynomials is discussed with special reference to its use for this purpose.

In this paper we are mainly concerned with an analysis of the problem of ill-conditioning in polynomials and we show that it cannot be overcome without, at some stage of the computation, resorting to high precision arithmetic. It is claimed that the iterative programmes which are described here reduce the volume of high precision computation to a minimum. The reader may well feel that since well conditioned problems frequently lead to ill-conditioned polynomials, transformation to explicit polynomial form should be avoided and this view was at one time shared by the author. However, experience with the routines described here for finding the zeros of polynomials has shown that the formal simplicity of the explicit representation is such an advantage, that it frequently

happens that high precision computation performed on the explicit form is faster than single precision computation performed on alternative forms*.

2. The Eigenvalues of a matrix as functions of its elements

Since a subroutine for finding zeros of polynomials is likely to be used to find the eigenvalues of a matrix, we consider first how the eigenvalues of a matrix are affected by small changes in its elements. A very simple analysis will suffice for our purposes. Let A be a square matrix with distinct eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_n$ and eigenvectors $u_1, u_2, \ldots u_n$ and let $v_1, v_2, \ldots v_n$ be the eigenvectors of A^T. If δA is a small perturbation of A then we assume that a typical eigenvector of $A+\delta A$ is $u_i + \sum_{j \neq i} \varepsilon_{ij} u_j$, where the ε_{ij} are small, and that it corresponds to an eigenvalue, $\lambda_i + \delta\lambda_i$. Then

$$(A + \delta A)\left(u_i + \sum_{j \neq i} \varepsilon_{ij} u_j\right) = (\lambda_i + \delta\lambda_i)\left(u_i + \sum_{j \neq i} \varepsilon_{ij} u_j\right) \tag{1}$$

$$\delta A\, u_i + \sum_{j \neq i} \varepsilon_{ij} A\, u_j = \sum_{j \neq i} \lambda_i \varepsilon_{ij} u_j + \delta\lambda_i u_i \tag{2}$$

Multiplying this equation by v_i^T we have

$$\delta\lambda_i = \frac{v_i^T\, \delta A\, u_i}{v_i^T\, u_i} \tag{3}$$

since $v_i^T u_j = 0,\ j \neq i$.

If the elements of δA are all bounded by ε, then for normalised v_i and u_i we have

$$|v_i^T\, \delta A\, u_i| \leqq n\varepsilon. \tag{4}$$

If A is symmetric then the v_i and the u_i are identical and $v_i^T u_i = 1$ for normalised vectors, so that

$$|\delta\lambda_i| \leqq n\varepsilon. \tag{5}$$

We can construct unsymmetric matrices for which $(v_i^T u_i)$ is arbitrarily small but it has been the author's experience in practice that values of $(v_i^T u_i)$ have not been unduly small and the eigenvalues have usually been well determined by the data. This is important because we are more likely to resort to the use of the characteristic equation for unsymmetric than for symmetric matrices. If we are to find eigenvalues via the characteristic equation we must know how the zeros of a polynomial depend on its coefficients.

3. Condition of a polynomial

The considerations of the previous section lead us to the problem of determining the perturbations in the zeros of a polynomial due to perturbations of its coefficients. Thus if the general polynomial of order n is denoted by

$$f(z) = z^n + a_{n-1} z^{n-1} + a_{n-2} z^{n-2} + \cdots + a_0 \tag{6}$$

* The paper is presented in two separate parts. In this part we develope the considerations which have led to the design of the subroutines on the computer DEUCE. In a second part we shall give a practical assessment of their performance on polynomials arising from a number of different sources.

we wish to determine the zeros of

$$f(z) + \delta f(z) \equiv f(z) + \delta a_r z^r$$

for each value of r.

If λ_i is an isolated root of $f(z)$, then $\lambda_i + \delta\lambda_i$ is a root of $f(z) + \delta f(z)$ if

$$f(\lambda_i + \delta\lambda_i) + \delta a_r(\lambda_i + \delta\lambda_i)^r = 0 \tag{7}$$

or, since $f(\lambda_i) = 0$

$$\delta\lambda_i = -\delta a_r \lambda_i^r / f'(\lambda_i) \quad \text{to the first order of small quantities.} \tag{8}$$

For a double root $f'(\lambda_i) = 0$, and equation (7) gives

$$\begin{aligned} \tfrac{1}{2}(\delta\lambda_i)^2 f''(\lambda_i) &= -\delta a_r \lambda_i^r \\ (\delta\lambda_i)^2 &= -2\delta a_r \lambda_i^r / f''(\lambda_i). \end{aligned} \tag{9}$$

There are similar results for roots of higher multiplicity. For a double root the perturbations involve the factor $(\delta a_r)^{\frac{1}{2}}$, so that we expect changes of order 10^{-10} in a coefficient to make changes of order 10^{-5} in a root. The sensitivity of multiple roots and of very close roots has been widely discussed in the literature. The fact that a succession of "moderately close" roots also results in a very poor determination has received much less attention though OLVER [3] mentions this in passing in connexion with one of his examples. The author's experience suggests that it is the overriding practical problem.

By way of illustration, we consider a polynomial of moderate degree with roots which can certainly not be regarded as pathologically close. The polynomial is of degree 20 and is defined by

$$f(x) = (x+1)(x+2)\ldots(x+20). \tag{10}$$

Since for any polynomial with roots $\lambda_1, \lambda_2, \ldots, \lambda_{20}$ we have

$$f'(\lambda_i) = \prod_{j \neq i} (\lambda_i - \lambda_j) \tag{11}$$

equation (8) gives for the 20th root

$$\delta\lambda_{20} = \frac{20^r \delta a_r}{19!}. \tag{12}$$

This takes its greatest value for $r = 19$, namely

$$\delta\lambda_{20} = \frac{20^{19}}{19!}\delta a_{19} \doteq 0.43 \times 10^8 \delta a_{19}. \tag{13}$$

The 20th root is not the most sensitive to variations in a_{19} however. We have in fact

$$\delta\lambda_{15} = \frac{15^{19}}{5!\,14!}\delta a_{19} \doteq 0.21 \times 10^{10} \delta a_{19} \tag{14}$$

and

$$\delta\lambda_{16} = \frac{16^{19}}{4!\,15!}\delta a_{19} \doteq 0.24 \times 10^{10} \delta a_{19}. \tag{15}$$

The multiplying factors are so large that for a change of order 10^{-7} in δa_{19}, the linear approximation given by (8) is completely invalid. The roots of the polynomial

$$(x+1)(x+2)\ldots(x+20)+2^{-23}x^{19} \tag{16}$$

for instance, are given below in Example 1. It will be seen that the small roots are little affected by the variation but that the larger ones are completely changed, 10 of them becoming non-trivially complex. Example 2 on the other hand shows a perturbation of a_{19} for which the linear theory is sufficiently accurate.

Example 1. Accurate roots of polynomial $(x+1)(x+2)\ldots(x+19)(x+20)+2^{-23}x^{19}$

− 1.000000000	− 10.095266145 ± 0.643500904 i
− 2.000000000	− 11.793633881 ± 1.652329728 i
− 3.000000000	− 13.992358137 ± 2.518830070 i
− 4.000000000	− 16.730737466 ± 2.812624894 i
− 4.999999928	− 19.502439400 ± 1.940330347 i
− 6.000006944	
− 6.999697234	Note that 5 pairs of roots have become complex. Changes are so great that the linearised perturbation theory is inapplicable.
− 8.007267603	
− 8.917250249	
− 20.846908101	

Example 2. Accurate roots of polynomial $(x+1)(x+2)\ldots(x+19)(x+20)+2^{-55}x^{19}$

− 1.000000000	− 6.000000000	− 10.999999999	− 16.000000067
− 2.000000000	− 7.000000000	− 12.000000006	− 16.999999947
− 3.000000000	− 8.000000000	− 12.999999983	− 18.000000028
− 4.000000000	− 9.000000000	− 14.000000037	− 18.999999991
− 5.000000000	− 10.000000000	− 14.999999941	− 20.000000001

Changes are now all small enough for linear perturbation to be applied.

We call those roots of a polynomial which are sensitive to small changes in the coefficients, "ill-conditioned" roots. The above example makes it clear that a polynomial may have some ill-conditioned and some well-conditioned roots. The condition of a polynomial is not changed by multiplying all its roots by a constant scale factor. It will readily be verified that the polynomial with roots 1.00, 0.95, 0.90, ..., 0.05 is as ill-conditioned as the one we have just considered.

4. The use of the characteristic equation

We now analyse the significance of the above results when the eigenvalues of a matrix are determined by finding the roots of its characteristic equation. We consider a matrix of order 20 with well determined eigenvalues which have approximately the values -1 to -20. (We prefer not to have roots exactly equal to these values because comments about rounding errors are inapplicable for examples in which small integers occur.) Suppose we have a method of calculating the characteristic equation which determines the coefficients correctly as ten-digit floating decimal numbers, so that the only errors are those that are inherent in the representation of the coefficients by numbers of a finite length. The coefficient of x^{19} is approximately 210 so that the rounding error in this

coefficient has the maximum value $\frac{1}{2}\,10^{-7}$. The results quoted above show that such an error will be fatal to the accuracy of the larger eigenvalues. It is worthwhile considering the effect of errors in the other coefficients since further aspects of the difficulty are thereby illustrated. The coefficients of the polynomial are of the order of magnitude shown below.

$$x^{20}+10^{3}x^{19}+10^{5}x^{18}+10^{7}x^{17}+10^{8}x^{16}+10^{10}x^{15}+10^{11}x^{14}+10^{12}x^{13}+$$
$$+10^{13}x^{12}+10^{14}x^{11}+10^{16}x^{10}+10^{17}x^{9}+10^{17}x^{8}+10^{18}x^{7}+10^{19}x^{6}+$$
$$+10^{19}x^{5}+10^{19}x^{4}+10^{20}x^{3}+10^{20}x^{2}+10^{19}x+10^{19}.$$

When considering the perturbations due to the error in the coefficient of x^r we must take account not only of the factor $\lambda_i^r/f(\lambda_i)$ but also of the variation in size of δa_r itself. The maximum rounding error in a_{18} for ten-digit representation is $\frac{1}{2}\,10^{-5}$ and because this is larger than that in a_{19}, the rounding of this coefficient can have the more severe effect. The worst case is the perturbation in λ_{16} due to the rounding error in a_{15} which can have a maximum value of

$$\frac{1}{2}\,10^{10-10}\,\frac{16^{15}}{4!\,15!} \doteqdot (0.18)\,10^{5}.$$

This is well beyond the region where a linear approximation is valid.

The inadequacy of single precision floating arithmetic for such polynomials is further illustrated by the following simple observation. If we take the equation

$$z^{n}+a_{n-1}z^{n-1}+\cdots+a_{0}=0$$

and multiply it by a ten-figure constant k, rounding the new coefficients ka_r to 10 significant decimals then any roots which are very sensitive to end-figure changes in the coefficients of the original polynomials, will be substantially modified by the transformation. Thus we cannot perform the simplest operation involving rounding errors on such a polynomial without modifying these roots. If the coefficients of the polynomial are primary data and are subject to independent errors of any kind, we can claim that only those figures are meaningful which remain unaltered when the last figures of the data are modified and no purpose is served in trying to calculate the roots to higher accuracy by regarding the coefficients as exact.

If, on the other hand, we are using the characteristic equation to determine the eigenvalues of a matrix, then provided these eigenvalues are well determined by the data, we are justified in forming the characteristic equation to any accuracy which may be necessary to determine its roots correctly. It may be argued that if the elements of the original matrix have errors in their kth significant figure then this is true also of the coefficients of the characteristic equation and no purpose can be served by calculating them more accurately. The fallacy in this argument is that errors in the coefficients of the characteristic equation arising from those in the elements of the matrix are not independent and, since the *exact* characteristic equation of the given matrix has for its roots the exact

eigenvalues of the matrix, we may calculate those eigenvalues to any accuracy by forming a sufficiently accurate characteristic equation. The above example shows that if a symmetric matrix of order 20 has eigenvalues approximately equal to $1, 2, \ldots, 20$ we must determine the characteristic equation correct to 18 or 19 decimal places to obtain all the eigenvalues correct to 5.

It is tempting to conclude from this that, since matrices commonly arise which have unfavourable distributions of roots, the determination of the characteristic equation is always inadvisable. The author's experience of the last few years has convinced him that this is not completely true and what follows may be regarded as a qualified attempt to reinstate the use of the characteristic equation.

5. Analysis of some standard root distributions

Before investigating methods of calculating zeros of polynomials we shall analyse the condition of a number of polynomials whose zeros have typical distributions. The above example has shown that a linear distribution of positive roots gives rise to ill-conditioned polynomials. It is interesting to note that the polynomial of order 20 given above is in some respects worse conditioned than may be a polynomial with a multiple root. For example, the polynomial $f(z)$ defined by

$$f(z) = (z - \tfrac{1}{2})^3 (z^{17} + 1) \tag{17}$$

has a treble root at $z = \frac{1}{2}$. If the coefficient of z^r is changed to $(a_r + \delta a_r)$ the perturbed roots become $(\frac{1}{2} + \varepsilon)$ where ε satisfies

$$\varepsilon^3 (2^{-17} + 1) = -\delta a_r 2^{-r} \quad \text{or} \quad \varepsilon^3 = -\delta a_r \left(\frac{2^{-r}}{1 + 2^{-17}}\right). \tag{18}$$

The expression on the right is less than δa_r in modulus for all values of r. Errors in the 10th figures of the coefficients therefore produce errors of order 10^{-3} at most in the treble root. This is a far more favourable result than we obtained for our polynomial above for which 10-figure coefficients gave no correct figures in some roots. The treble root is worse only in the sense that we need to increase the precision of the coefficients by three figures to improve this root by one figure, whereas in the earlier polynomial if the precision of the coefficients is sufficient to give at least one figure correct in the roots then every added figure of precision in the coefficients gives one extra figure in the roots.

Not all linear distributions are as ill-conditioned as that from 1 to 20. For roots $(k+1), (k+2), \ldots, (k+20)$ we have

$$\frac{\partial \lambda_i}{\partial a_r} = \frac{(k+i)^r}{(i-1)!\,(20-i)!}. \tag{19}$$

For positive k the polynomial becomes increasingly ill-conditioned for increasing k. For $k = 20$ we have, for example

$$\frac{\partial \lambda_{15}}{\partial a_{19}} = \frac{(35)^{19}}{5!\,14!} = (0.21)\,10^{17}. \tag{20}$$

For $k=-10$ however the polynomial is far better conditioned, the worst root being the 18th for which we have

$$\frac{\partial \lambda_{18}}{\partial a_{19}} = \frac{(8)^{19}}{2!\,17!} = (0.20)\,10^3. \tag{21}$$

If we knew in advance that a matrix A had a linear distribution of eigenvalues, then a simple transformation $(A-pI)$ with $p=\frac{1}{n}\sum a_{ii}$ would lead to a much better conditioned characteristic equation. Unfortunately we do not usually have such information in advance and for some distributions the transformation makes the characteristic equation far more ill-conditioned.

As a second example we consider a polynomial with roots in geometric progression $2^{-1}, 2^{-2}, 2^{-3}, \ldots, 2^{-20}$. The coefficients of this polynomial vary enormously in size; their order of magnitude is indicated below.

$$\begin{aligned} &x^{20} + x^{19} + 2^{-1}x^{18} + 2^{-4}x^{17} + 2^{-8}x^{16} + 2^{-13}x^{15} + 2^{-19}x^{14} + 2^{-26}x^{13} + \\ &+ 2^{-34}x^{12} + 2^{-43}x^{11} + 2^{-53}x^{10} + 2^{-64}x^{9} + 2^{-76}x^{8} + 2^{-89}x^{7} + \\ &+ 2^{-103}x^{6} + 2^{-118}x^{5} + 2^{-134}x^{4} + 2^{-151}x^{3} + 2^{-169}x^{2} + 2^{-189}x + 2^{-209}. \end{aligned} \tag{22}$$

If a matrix has eigenvalues with a distribution of this type then it is essential that the coefficients of the characteristic equation be determined to a fixed number of significant figures rather than to a fixed number of decimal places. Determination to 10 decimal places, for example, would, at best, give the coefficients from a_{11} to a_0 as zeros and therefore give 12 zero roots. At worst, it might produce for the later coefficients, random numbers of the order of magnitude of 10^{-10}. This would give a number of spurious roots in place of the smaller roots. Equation (8) yields

$$\delta\lambda_k = -\delta a_r\, 2^{-kr}/PQ \quad \text{where} \tag{23}$$

$$P = (2^{-k}-2^{-1})(2^{-k}-2^{-2})\ldots(2^{-k}-2^{-k+1}) \tag{24}$$

and

$$Q = (2^{-k}-2^{-k-1})(2^{-k}-2^{-k-2})\ldots(2^{-k}-2^{-20}). \tag{25}$$

We have

$$P = \pm\frac{1}{2^{\frac{1}{2}k(k-1)}}\left[(1-2^{-1})(1-2^{-2})\ldots(1-2^{-k+1})\right] \tag{26}$$

and

$$Q = \frac{1}{2^{k(20-k)}}\left[(1-2^{-1})(1-2^{-2})\ldots(1-2^{-20+k})\right]. \tag{27}$$

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

$$|PQ| > (\tfrac{1}{16})\,2^{-\frac{1}{2}k(39-k)} \tag{28}$$

and

$$|\delta\lambda_k| < 16\,|\delta a_r|\,2^{\frac{1}{2}k(39-k-2r)}. \tag{29}$$

Because of the great disparity in the size of the roots it is more reasonable to consider the change in a root relative to the root itself, so that equation (29) gives

$$\left|\frac{\delta\lambda_k}{\lambda_k}\right| < 16\,|\delta a_r|\,2^{\frac{1}{2}k(37-k-2r)}. \tag{30}$$

For a fixed value of r this takes its maximum value when $k=(19-r)$ for which

$$\left|\frac{\delta\lambda_k}{\lambda_k}\right| < 16\,|\delta a_r|\,2^{\frac{1}{2}(19-r)(18-r)} \text{ for all } k. \tag{31}$$

From the order of magnitude of the a_r given above we can verify that

$$|a_r| < 2^{-\frac{1}{2}(19-r)(18-r)};$$

indeed for higher values of r the coefficients are much smaller than this. We can write equation (31) in the form

$$\begin{aligned}\left|\frac{\delta\lambda_k}{\lambda_k}\right| &< 16\left|\frac{\delta a_r}{a_r}\right||a_r|\,2^{\frac{1}{2}(19-r)(18-r)}\\ &< 16\left|\frac{\delta a_r}{a_r}\right| \text{ for all } k\end{aligned} \tag{32}$$

with a much stricter inequality holding for most values of r. Equation (32) implies that the determination of the coefficients of the polynomial to 10 significant decimals gives all roots correct to at least 8 significant decimal places, so that all roots are well conditioned. Example 3 below shows the perturbation of the roots due to a small change in a_{19}.

Example 3. Accurate roots of $(x+2^{-1})(x+2^{-2})\ldots(x+2^{-20})+2^{-31}x^{19}=0$

$10^{-6}\times -0.953674316$	$10^{-3}\times -0.976562500$
$10^{-5}\times -0.190734863$	$10^{-2}\times -0.195312500$
$10^{-5}\times -0.381469727$	$10^{-2}\times -0.390625000$
$10^{-5}\times -0.762939453$	$10^{-2}\times -0.781250000$
$10^{-4}\times -0.152587891$	$10^{-1}\times -0.156250000$
$10^{-4}\times -0.305175781$	$10^{-1}\times -0.312500000$
$10^{-4}\times -0.610351563$	$10^{-1}\times -0.624999999$
$10^{-3}\times -0.122070313$	-0.125000000
$10^{-3}\times -0.244140625$	-0.249999998
$10^{-3}\times -0.488281250$	-0.500000001

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

We have considered zeros in geometric progression with a ratio of $\frac{1}{2}$. It is easy to see from the above that for a ratio less than $\frac{1}{2}$ the condition of the polynomial is even better, while for ratios greater than $\frac{1}{2}$ it becomes steadily worse as the ratio increases until for a ratio of 1 we have all the zeros coincident.

Finally we consider the distribution $\lambda_r=e^{2\pi ir/20}$. The corresponding polynomial is $x^{20}+1$ and $f'(x)=20x^{19}$. The variations are given by

$$\delta\lambda_k = \delta a_r\,[\lambda_k^r/f'(\lambda_k)] \tag{33}$$

or

$$|\delta\lambda_k| = \tfrac{1}{20}|\delta a_r| \text{ since } |\lambda_k| = 1 \text{ for all roots.}$$

The polynomial is therefore extremely well-conditioned.

6. Iterative methods for zeros of polynomials

For reasons which will be explained below, we have used iterative methods for finding the zeros of polynomials. Probably the simplest method is that due to Newton in which a sequence of values x_r is determined from the relation

$$x_{r+1} = x_r - f(x_r)/f'(x_r). \tag{34}$$

If x_r is a close approximation to a root, a, and

$$x_r = a + h$$

then

$$\begin{aligned} x_{r+1} &= (a+h) - \frac{f(a+h)}{f'(a+h)}, \\ &= (a+h) - \frac{f(a) + hf'(a) + \frac{h^2}{2}f''(a) + \frac{h^3}{6}f'''(a) + \cdots}{f'(a) + hf''(a) + \frac{h^2}{2}f'''(a) + \cdots}, \\ &= a + \frac{\frac{h^2}{2}f''(a) + \frac{h^3}{3}f'''(a) + \cdots}{f'(a) + hf''(a) + \frac{h^2}{2}f'''(a)}. \end{aligned} \tag{35}$$

If $f'(a) \neq 0$ we have

$$x_{r+1} = a + h^2\frac{f''(a)}{2f'(a)} + O(h^3) \tag{36}$$

and for this reason the process is usually described as quadratically convergent. It is sometimes said that in the later stages the number of correct decimal places doubles itself with each iteration. This clain ignores the effect of the factor $f''/2f'(a)$ and for ill-conditioned roots this factor may be quite important. If it is equal to 1000 for example, then if k_r denotes the number of correct decimal places in the rth iteration we have ultimately

$$k_{r+1} = 2k_r - 3. \tag{37}$$

If we are not proceeding beyond an accuracy of 8 or 10 decimals it is unreasonable to refer to an eventual doubling of figures.

The above remarks refer to the mathematical process in which it is assumed that $f(x_r)$ and $f'(x_r)$ are calculated exactly. In practice, for an ill-conditioned root, the accuracy to which $f(x_r)$ can be calculated in the neighbourhood of that root may severely limit the accuracy attainable by iteration. Let us consider the evaluation of the polynomial whose zeros are 1, 2, ..., 20 for a value of $x = 20.00012345$. A rough graph of the function shows us that an exact application of NEWTON'S method gives monotonic convergence to the root 20, and the given approximation is already in the region of rapid convergence. If, however, we

attempt to evaluate the function using 10-decimal floating arithmetic we immediately run into difficulties. Let us suppose that we evaluate the polynomial

$$x^n + a_{n-1}x^{n-1} + \cdots + a_0$$

by "nested multiplication", that is by calculating the sequence

$$1,\ x + a_{n-1},\ x(x + a_{n-1}) + a_{n-2}, \ldots .$$

Now the polynomial in question may be written

$$x^{20} - 210x^{19} + \cdots + 20! \tag{38}$$

and at the second stage we calculate $(x - 210)$ which is

$$20.00012345 - 210 = -189.99987655 .$$

Since we are using ten-decimal floating arithmetic we must replace this by -189.9998766 with an error of $\frac{1}{2}(10^{-7})$. Ignoring for the moment the later rounding errors in the calculation of $f(x)$, this single error leads to an error in the final value of

$$(\tfrac{1}{2}10^{-7})\, x^{19} \doteqdot \tfrac{1}{2}(10^{-7})\, 20^{19}. \tag{39}$$

Now the correct value of $f(x)$ is $(x-1)(x-2)\ldots(x-20) \doteqdot 19!\,(0.00012345)$ and it will be seen that this is much smaller than the error. The evaluation of the function using ten-figure floating arithmetic is completely inaccurate and we cannot expect the use of NEWTON'S method with the *calculated* value of $f(x)$ to produce an improved value for the root. The inaccuracy does not spring primarily from the use of nested multiplication. If we evaluate the polynomial by calculating each term separately the result is just as bad.

We may relate the emergence of an incorrect value of the function to the condition of the root as follows. At the second stage of the evaluation above we replaced the number -189.99987655 by -189.9998766. Another way of describing this would be to say that our evaluation is exact up to that stage for the polynomial beginning with

$$x^{20} - 210.0000005\, x^{19} + \cdots .$$

In a similar way the error made at the next stage may be interpreted as the use of a modified coefficient of x^{18} and so on. However, we know from our earlier analysis that the change in the coefficient of x^{19} above completely alters the higher roots of the polynomial. More generally we may say that for a value of x equal to $(20 + \delta)$ the error in $f(x)$ due to error in the first stage, may be as great as

$$(\tfrac{1}{2}10^{-7})\, 20^{19} \quad \text{(approximately)}.$$

The true value of $f(x)$ is approximately $19!\,\delta$ so that the first error alone completely invalidates the calculation when

$$(\tfrac{1}{2}10^{-7})\, 20^{19} > 19!\,\delta \quad \text{or} \quad \delta < \frac{(\tfrac{1}{2}10^{-7})\, 20^{19}}{19!}. \tag{40}$$

Comparison with equation (13) shows this to be exactly the estimate, on the basis of a linear theory, of the maximum variation in the root 20 due to a rounding error in a_{19}.

In general we may say that if we take an approximation to a zero of a polynomial and then, working to a given precision, iterate using NEWTON'S method, we derive a succession of values which reach a limiting accuracy, after which further iteration yields no improvement. For well-conditioned roots this final accuracy will be the full precision of the computation, but for ill-conditioned roots it will fall short of this. The extreme point is reached when the errors made in the evaluation of $f(x)$ are greater than its true value so that the calculated value of $f(x)$ has no correct figures. As an example of this the DEUCE double precision programme (18 decimals floating) will calculate the root 20 of the above polynomial to 8 or 9 significant figures. It will not advance beyond this accuracy, however long iteration is continued. Further if a value of x of higher accuracy is used, the next iteration will produce a value with only 8 or 9 figures correct, that is, iteration will "*spoil*" a very accurate root. (For the polynomial with roots exactly equal to $1, 2, \ldots, 20$ a double precision programme will in fact take the value 20 exactly and produce 20 as its next approximation. This is due to the fact that no rounding errors are involved in this particular calculation. Later remarks in this paper may be invalidated by such special cases and we will not refer to this again.)

These remarks are of particular importance when we consider iteration for a root which is so ill-conditioned that our calculations to a given number of decimals fail to determine even one figure of the polynomial correctly for values in its neighbourhood. An example of this is encountered in the use of single precision arithmetic to calculate the roots 15 or 16 of the above polynomial. For values of x between 15 and 16 single precision arithmetic gives no correct figures in the calculated values of $f(x)$. If we started with an approximation in this region the process would show no signs of convergence, even for values for which the exact use of NEWTON'S method would be successful.

The above comments should not be construed as a criticism of NEWTON'S method or of iterative methods in general, since these will often involve the calculation of $f(x)$. The limitation is a fundamental one and will reveal itself in almost any technique applied to the solution of such a polynomial equation. Indeed, in general iterative techniques deal with ill-conditioning at least as satisfactorily as any other in which computation is performed to the same precision. In the root squaring method, for example, the polynomial

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$$

is first replaced by the polynomial $f(x)\, f(-x)$, which is a polynomial of degree n in $(-x^2)$. The new polynomial is

$$a_n^2 (-x^2)^n + (a_{n-1}^2 - 2 a_n a_{n-2}) (-x^2)^{n-1} + \cdots .$$

The presence of rounding errors introduced in the calculation of these new coefficients will mean that it corresponds exactly to the polynomial

$$(a_n + \delta a_n) x^n + (a_{n-1} + \delta a_{n-1}) x^{n-1} + \cdots + (a_0 + \delta a_0)$$

instead of the true original polynomial and the δa_r are comparable with the values which arise indirectly in iterative methods in the evaluation of $f(x)$. A further comparison of root squaring and iteration is made later.

The inescapable fact is that for any root of any polynomial, a minimum precision in the computation is essential in order to calculate that root to a required precision. High precision computation is unavoidable for the resolution of very ill-conditioned roots. On the other hand multiple precision floating point arithmetic is comparatively slow, so that we do not want to work for any extended period to an unnecessarily large number of figures. We therefore recommend that iterative techniques be used in the following way. We iterate using single precision floating arithmetic until either (a) the root has converged to a pre-assigned number of figures or (b) we have completed a certain fixed number of iterations. We then continue with double precision arithmetic again continuing until either (a) the root has converged to a pre-assigned number of figures (greater than in the first stage) or (b) we have completed a fixed number of iterations. We can continue in this way using higher precision arithmetic. The most accurate that has been used on DEUCE is treble precision. This has 3 words for the fraction, and one word for the index and gives about 27 decimal digits. The condition (b) prevents us from iterating indefinitely for a root which needs higher precision for its resolution. Although a very large number of polynomial equations, arising from different sources, has been solved, treble precision has always proved adequate so far, though one could easily invent examples with any degree of ill-conditioning.

7. Existing DEUCE programmes

Several iterative techniques for finding zeros of polynomials have been programmed for DEUCE. The first of these is based on NEWTON'S method and works in the complex plane. Starting from an arbitrary value of z_0 (we comment on the initial choice later) a sequence of values is found by NEWTON'S rule, and when convergence has reached a point at which the relation

$$\left|\frac{z_{r+1}-z_r}{z_r}\right| < 2^{-k_1}, \tag{41}$$

is satisfied, where k_1 is a pre-assigned integer, this value is accepted as a root.

The calculation of $f(z)$ and $f'(z)$ is carried out as follows. The two sequences s_r and s'_r defined by

$$s_{n+1}=0 \quad s_r = z\,s_{r+1}+a_r \quad (r=0 \text{ to } n), \tag{42}$$

$$s'_{n+1}=0 \quad s'_r = z\,s'_{r+1}+s_{r+1} \quad (r=0 \text{ to } n) \tag{43}$$

are calculated simultaneously to obtain $f(z)=s_0$ and $f'(z)=s'_0$. When a zero, x, has been accepted, it is divided out from the polynomial to obtain $f(z)/(z-x)$, a polynomial of degree $(n-1)$. The coefficients of the reduced polynomial are precisely the values s_r of the sequence above calculated for $z=x$, so that the programme which performs iteration will also perform the division. Most of the polynomials which arise in practice have real coefficients so that when a complex

zero is found, it is to be expected that its conjugate will be a zero of the reduced polynomial. This is therefore used as the first approximation to a zero of the reduced polynomial. It is not automatically assumed that the complex conjugate is a zero of that polynomial however, and iteration is performed. Because this first guess is an accurate approximation to a zero, the second of a complex conjugate pair is always found with only one or two iterations. Note that in this way we do not obtain a pair which are exactly conjugates. The extent to which they are complex conjugates proves to be a very good measure of their accuracy.

A real first approximation to a root of a polynomial with real coefficients, produces a succession of real values and cannot converge to a complex root. To guard against this, each root that is accepted is tested to see if it is a real root. The criterion for a root $(x+iy)$ being real is taken to be that

$$\left|\frac{y}{x}\right| < 2^{-k_2} \tag{44}$$

where k_2 is a pre-assigned integer. When a real root, according to this criterion, has been accepted, the first guess for the next root is taken to be $(1+i)(x-iy)$ in order to ensure departure from the real axis. As a final step, when all the roots have been found, the original polynomial is used to produce improved roots by iteration. We comment on this below.

The above technique has been coded using single-precision floating arithmetic and only a little experience of its use was necessary to show its limitations. Polynomials were frequently derived for which the process failed to converge even when the tolerance k_1 was lowered almost to zero. An analysis of these examples showed that this was invariably due to the phenomenon discussed above, namely the poor evaluation of $f(z)$ and it was decided that higher precision arithmetic should be used. Since, however, polynomials with complex coefficients had proved to be very rare it was decided that it was worth programming a method which took advantage of this. The method due to BAIRSTOW [*1*] was therefore preferred, while a third, more general technique due to MÜLLER [*4*] was programmed for polynomials with complex coefficients.

BAIRSTOW's method is essentially a technique for finding the real quadratic factors of a polynomial with real coefficients. Starting from an approximate factor (x^2-px-l), an improved factor $x^2-(p+\delta p)\,x-(l+\delta l)$ is found as follows. We write

$$f(x) = (x^2-px-l)\,q(x) + r(x) \tag{45}$$

where $q(x)$ is the quotient (of degree $n-2$) and $r(x)$ is the remainder (degree unity). We may write

$$r(x) = a\,x + b.$$

Differentiating (45) with respect to l and p, we have

$$0 = -q(x) + (x^2-px-l)\frac{\partial}{\partial l}q(x) + \frac{\partial}{\partial l}r(x) \tag{46}$$

and

$$0 = -xq(x) + (x^2-px-l)\frac{\partial}{\partial p}q(x) + \frac{\partial}{\partial p}r(x).$$

Equations (46) show that $\frac{\partial}{\partial l} r(x)$ is the remainder when $q(x)$ is divided by $(x^2 - px - l)$ and $\frac{\partial}{\partial p} r(x)$ is the remainder when $xq(x)$ is divided by $(x^2 - px - l)$. If, therefore, we write

$$q(x) - (x^2 - px - l)\, T(x) + cx + d \tag{47}$$

then $\frac{\partial}{\partial l} r(x) = cx + d$ and

$$\begin{aligned} \frac{\partial}{\partial p} r(x) &\equiv x(cx + d) \text{ modulo } (x^2 - px - l) \\ &= x(cp + d) + cl. \end{aligned} \tag{48}$$

We now choose δp and δl so that

$$r + \frac{\partial r}{\partial p}\delta p + \frac{\partial r}{\partial l}\delta l = 0 \text{ giving}$$

$$a + (cp + d)\,\delta p + c\,\delta l = 0, \tag{49}$$

$$b + (cl)\,\delta p + d\,\delta l = 0. \tag{50}$$

The most convenient way of programming the process seems to be that described by OLVER [3]. The quantities q_s and T_s defined by the following recursive formulae are first derived

$$q_{n+2} = q_{n+1} = 0 \qquad q_s = pq_{s+1} + lq_{s+2} + a_s \qquad s = n, n-1, \ldots, 0, \tag{51}$$

$$T_n = T_{n-1} = 0 \qquad T_s = pT_{s+1} + lT_{s+2} + q_{s+2} \qquad s = n-2, n-1, \ldots, 0. \tag{52}$$

The next approximation is then derived from the formulae

$$D\,\delta p = T_1 q_0 - T_0 q_1 \qquad D\,\delta l = Mq_1 - T_0 q_0 \tag{53}$$

where

$$M = lT_1 + pT_0 \qquad D = T_0^2 - MT_1. \tag{54}$$

The process gives quadratic convergence of both p and l as is to be expected from equations (49) and (50) in which squares and products of δp and δl have been ignored. It does not require the factors of $(x^2 - px - l)$ to be complex and it differs from NEWTON'S formulae in the complex plane by second order quantities. Thus applied to a function $f(x)$ of degree 2 it produces in one step the obvious quadratic factor starting from any quadratic factor as a first approximation. NEWTON'S process in the complex plane does not do this.

After accepting a quadratic factor, that factor is divided into $f(x)$ to give a polynomial of degree two lower. The coefficients of this polynomial are the $q_n, q_{n-1}, \ldots, q_2$ calculated from equation (51) so that the same programme is used for dividing out as for iterating. After a quadratic factor has been divided out, it is used as the first approximation to the next factor to be calculated. This has considerable advantages as will be seen in the next section.

Single-, double-, and treble-precision Bairstow programmes exist on DEUCE and again there is the facility for using the original polynomias to improve all the calculated factors by iteration. Together they give the most powerful and fastest method that has been coded for the machine.

For polynomials with complex coefficients the following iterative scheme is used. A sequence of values z_r is derived, z_{r+1} being calculated from z_{r-2}, z_{r-1}, z_r and the corresponding values of $f(z)$, as the root nearer to z_r of the quadratic through the points $(z_{r-2}, f(z_{r-2}))$ $(z_{r-1}, f(z_{r-1}))$ $(z_r, f(z_r))$. MÜLLER [4] has described a particularly elegant method for finding this root. The auxiliary quantities h_i, λ_i and δ_i defined by the relations

$$z_i - z_{i-1} = h_i \qquad \frac{h_i}{h_{i-1}} = \lambda_i \qquad \delta_i = 1 + \lambda_i \tag{55}$$

are introduced. λ_{i+1} is then determined as the smaller root of the quadratic

$$A\lambda^2 + B\lambda + C = 0$$

where

$$A = \lambda_i[f_{i-2}\lambda_i - f_{i-1}\delta_i + f_i] \qquad B = f_{i-2}\lambda_i^2 - f_{i-1}\delta_i^2 + f_i(\lambda_i + \delta_i)$$
$$C = +f_i\delta_i. \tag{56}$$

The iteration is continued until

$$\left|\frac{h_{i+1}}{z_i}\right| < 10^{-k}$$

where k is a pre-assigned integer.

When a zero x has been accepted we can divide out the factor $(z-x)$ to form the polynomial of lower order, $f(z)/(z-x)$ as was done with the other two methods.

Alternatively we can continue with the function $f(z)/(z-x)$ itself and find *its* zeros. The second alternative is more general in that it can be used to find zeros of functions other than polynomials. When r zeros $x_1, x_2, \ldots, x_r$ have been found we continue with the function $f_r(z)$ defined by

$$f_r(z) = \frac{f(z)}{(z-x_1)(z-x_2)\ldots(z-x_r)}. \tag{57}$$

In the present situation this second alternative requires far more computation than the first since instead of working with polynomials of progressively diminishing order we work with a function of increasing complexity. Its use has nevertheless sometimes been recommended on the grounds that the explicit division by successive factors $(z-x_r)$ leads to such a serious accumulation of rounding errors that the later roots are inaccurate. While recognising the great value of the alternative technique and its great generality, in our experience it is unnecessary for polynomials. We comment further on this in section 8.

The Müller technique has been programmed using both single- and double-precision arithmetic. The programmes have been used in a more general manner

to find zeros of functions other than those expressed explicitly as polynomials, and for such work they have proved very effective, particularly for functions for which the calculation of the derivative is difficult. For finding the zeros of polynomials with real coefficients they have proved inferior to the Bairstow programmes. This is because the number of iterations required to locate a root starting from arbitrary values has proved to be appreciably greater in general for the Müller technique than for Bairstow. The amount of work in an iteration is much the same in both techniques since MÜLLER'S technique involves n complex multiplications and BAIRSTOW'S technique $4n$ real multiplications.

In the Newton programme after a complex root had been divided out its complex conjugate was used as the first trial root of the reduced polynomial. A corresponding technique is used on DEUCE in the Müller programme in the version in which the accepted factors are divided out. After accepting a root x and dividing by the factor $(z-x)$ the first 3 values for which the reduced function is evaluated are $\bar{x}(1-k)$, $\bar{x}(1+k)$ and $\bar{x}$, where $\bar{x}$ is the complex conjugate of x and k is a suitable constant. The value of k does not appear to be very critical and $k=\frac{1}{4}$ has been used on DEUCE. These three values correspond to $\lambda=-\frac{1}{2}$ and their use ensures that the second of a complex pair is located almost at once though as remarked before in connexion with NEWTON'S method, we will not in general find exactly complex conjugate values. After finding x and then $\bar{x}$ the first three values for which the next reduced function is evaluated are $\bar{\bar{x}}(1-k)$, $\bar{\bar{x}}(1+k)$ and $\bar{\bar{x}}$. Now $\bar{\bar{x}}=x$ so that the three values will be almost exactly $x(1-k)$, $x(1+k)$ and x. We may regard this merely as a device for ensuring that we begin to look for the next root in the neighbourhood of x.

For general functions, for which we work explicitly with $\frac{f(z)}{(z-x)(z-\bar{x})}$ rather than dividing out, we do not wish to work with the value $z=x$ since this makes the denominator exactly zero. We therefore start with the three values $y(1-k)$, $y(1+k)$ and y where $y=x(1+\delta)$. On DEUCE the value $\delta=2^{-8}$ has been used which ensures that we start looking for the next root in the neighbourhood of x.

It is interesting to consider the most economical way of determining the zeros of polynomials of arbitrarily bad condition defined explicitly by their coefficients. If it is assumed that the zeros are required correct to single precision only, that is to about 9—12 significant decimal digits, then the Müller technique has distinct advantages. A programme has been designed for DEUCE which employs floating arithmetic with up to 31 words in the fractional parts of the numbers but requires only multiplications of single word numbers by multiple word numbers. We begin iterating for each zero using single precision arithmetic and steadily increase the precision until the zero has converged to single precision. A well-conditioned root will therefore be determined without using more than double precision arithmetic and that only for one iteration, while iteration for ill-conditioned roots will employ only such precision as is essential. In the evaluation of the function we need to add together floating numbers both of which may be of high precision, but all multiplications will be of a single precision by a high precision number. In the equations (55) and (56) we need employ only single precision arithmetic. When a zero z_1 has been found correct to single precision we continue with the function $f(z)/(z-z_1)$ and again the numerator and denomi-

nator are required only to single-precision. The virtue of the Müller technique is that we do not have to determine well-conditioned zeros to high precision in order to obtain to single precision ill-conditioned zeros found subsequently. An assessment of this programme shows that even if all the zeros require five-word arithmetic, it will be as fast as the existing treble precision programme. In case it may be felt that the use of high precision arithmetic is somewhat extravagant it should be emphasized that if this programme uses r word arithmetic to calculate a zero of a given polynomial then no other method will evaluate that zero to single precision without, at some stage, using at least r word arithmetic. If the explicit polynomial has been derived by expanding some other expression, then we may well question the wisdom of this step. If on the other hand we are considering the strict problem of calculating zeros of explicit polynomials the above programme would appear to achieve results with the maximum efficiency.

Acknowledgment. The work described above has been carried out as part of the research programme of the National Physical Laboratory and is published by permission of the Director of the Laboratory.

List of references see Part II, p. 180.

Mathematics Division,
National Physical Laboratory,
Teddington, Middlesex

(Received May 6, 1959)