published the two groups in English Electric can go ahead with their respective translators, each with its separate field of application.

Somebody mentioned recursive procedures, and people playing with them. Most published examples show you how to do this—for instance, the famous factorial example—and it would be difficult to think of a less efficient procedure for factorial than this. However, if you have a procedure, written by somebody else, which uses, say, an integration routine, it would be possible to use this procedure, even within the integration routine, without having to delve into the procedure. That is the real benefit of recursive procedures.

Mr. F. S. Ellis (*Elliott Brothers*): Dr. Dijkstra has quoted certain ratios in connection with ALGOL programs and handwritten programs. Can he give the corresponding ratios in time involved in writing the handwritten programs and the ALGOL programs? Mr. Glennie quoted a figure of 3 to 1 in cost, but what about time? Do you more than make up for the difference in cost by the saving in time?

Dr. Dijkstra: The corresponding ratios in writing time might be 100 or 1,000. It goes up on an exponential scale in practice. That is why so few comparisons with run-time systems are available.

Mr. Glennie: The difficulty of answering questions as to the effect of ALGOL and FORTRAN on programming times is that there is no such thing as a standardized programmer. But I would say that you might expect to get a gain of 2 or 3 to 1 with FORTRAN with the average programmer, assuming the problems were defined.

The Chairman: I am sorry to have to draw this discussion to a close, but we have to meet again at 2 o'clock. I know that you would wish me to add your thanks to mine, to our four speakers. We are indebted to Dr. Dijkstra, Mr. Hoare, Mr. Duncan and Mr. Glennie for making this morning's session a great success. Thank you very much. (*The conference adjourned.*)

Computation of the latent roots of a Hessenberg matrix by Bairstow's method

By D. C. Handscomb

This paper gives the details of an algorithm, equivalent to Bairstow's process, for finding the real quadratic factors of the characteristic polynomial of a Hessenberg matrix, working from the matrix itself. This includes a method of removing known factors without introducing serious errors into the remaining roots.

Introduction

1. Bairstow's method of finding the real quadratic factors of a polynomial is well known (Wilkinson, 1959b), and can obviously be applied to the problem of determining the latent roots of a matrix from its characteristic polynomial. However, the roots of a polynomial are often very sensitive to errors in its coefficients, and it is therefore necessary to carry the calculation to much higher precision than is ultimately required.

We shall here describe an alternative direct method of finding the roots of a lower Hessenberg matrix $\{a_{ij}\}$, in which $a_{ij} = 0$ when $j > i + 1$. Wilkinson (1959a) has shown that any real matrix may be turned into this form, by similarity transformations, without great loss of accuracy.

We do not subject the matrix to any further transformation, so avoiding the inconvenience of multiple-precision arithmetic. On the other hand, unless the matrix is small or very sparse, this direct method calls for many more multiplications than are needed to solve the expanded polynomial.

The fact that we never construct a characteristic polynomial compels us to find a substitute for the usual method of dividing out each factor before attempting to find the next. The device which we use is in fact more accurate than the usual method, and could profitably be used on explicit polynomials also.

Bairstow's Process for Hessenberg Matrices

2. In order to clarify the subsequent removal of factors, we shall present Bairstow's process in terms of an algebraic congruence. If $F(\lambda)$ is the polynomial whose quadratic factors are sought, and $(\lambda^2 + p\lambda + q)$ is an approximation to such a factor, then we first find the coefficients of the congruence

$$F(\lambda) \equiv \{(\lambda a + b) + (\lambda^2 + p\lambda + q)(\lambda c + d)\} \quad \text{modulo } (\lambda^2 + p\lambda + q)^2. \quad (1)$$

In finding these coefficients, we may multiply $F(\lambda)$ by any constant scale factor, since all the expressions which we shall encounter are homogeneous in a, b, c, and d.

The second step is to replace p and q, respectively, by $(p + \delta p)$ and $(q + \delta q)$, where

$$\left.\begin{aligned} \delta p &= \frac{ad - bc}{d^2 - cdp + c^2 q} \\ \text{and} \qquad \delta q &= \frac{acq + bd - bcp}{d^2 - cdp + c^2 q} \end{aligned}\right\}. \quad (2)$$

Under appropriate conditions, these new values are then better approximations to the coefficients of the quadratic factor.

It may easily be shown that this is Bairstow's process. If we differentiate the congruence (1) with respect to p

and q, and evaluate it at the roots of $(\lambda^2 + p\lambda + q)$, we find that

$$\partial a/\partial p = cp - d,\ \partial b/\partial p = cq,$$
$$\partial a/\partial q = -c,\ \partial b/\partial q = -d. \quad (3)$$

The condition that a and b should vanish when p and q become $(p + \delta p)$ and $(q + \delta q)$ is then, to the first order,

$$a + (cp - d)\delta p - c\delta q = b + cq\delta p - d\delta q = 0. \quad (4)$$

These equations are satisfied by (2).

3. In the case where $F(\lambda) = \det(A - \lambda I)$, A being a lower Hessenberg matrix of order n, the values of a, b, c, and d, may be found directly from the matrix by the following process. Starting with $w_1 = y_1 = z_1 = 0$, and $x_1 = 1$, we generate sets of numbers a_i, b_i, c_i, d_i (for $i = 1, 2, \ldots, n$), and w_i, x_i, y_i, z_i (for $i = 2, \ldots, n$), by means of the equations

$$\left.\begin{aligned} a_i &= pw_i - x_i + \sum_{j=1}^{i} a_{ij}w_j \\ b_i &= qw_i + \sum_{j=1}^{i} a_{ij}x_j \\ c_i &= py_i - z_i + \sum_{j=1}^{i} a_{ij}y_j \\ d_i &= qy_i - w_i + \sum_{j=1}^{i} a_{ij}z_j \end{aligned}\right\}, \quad (5)$$

and

$$\left.\begin{aligned} w_i &= -a_{i-1}/a_{i-1,i} \\ x_i &= -b_{i-1}/a_{i-1,i} \\ y_i &= -c_{i-1}/a_{i-1,i} \\ z_i &= -d_{i-1}/a_{i-1,i} \end{aligned}\right\}. \quad (6)$$

Then $$a = a_n,\ b = b_n,\ c = c_n,\ d = d_n. \quad (7)$$

It can be shown, by induction on the value of n, that the quantities (7) satisfy the congruence (1), apart from a constant factor.

If any superdiagonal element $a_{i-1,i}$ of the matrix vanishes, it must be approximated by a small number to permit the divisions (6) to be performed. It is in such a case that scaling may be required, in order to keep all numbers within range. We scale them down by reducing in the same ratio all the quantities a_i, b_i, c_i, d_i, w_i, x_i, y_i, z_i, so far generated.

It will be seen that we need to keep only the current values of a_i, b_i, c_i, and d_i, but that we need the whole sequence of values of w_i, x_i, y_i, and z_i. The number of working locations required, apart from those occupied by the matrix, is therefore of the order of $4n$, and the number of multiplications required to generate these numbers is of the order of $2n^2$, provided that we do not scale them down too often.

The process represented by equations (5) and (6) is similar to the forward-substitution process which Hyman uses (Wilkinson, 1960), together with Muller's interpolation method, to find the complex roots of the matrix, one at a time. Either process can locate well-conditioned roots to almost the full precision of the computation.

Removal of Quadratic Factors

4. If Bairstow's method is to give us automatically a complete set of quadratic factors, we must have a means of removing factors of $F(\lambda)$ as we find them. If $F(\lambda)$ is given explicitly as a polynomial, they may be removed explicitly by long division; however, Wilkinson (1959b) has shown that this process introduces errors into any roots which are smaller than the roots of the dividing factors, so that it is essential that the roots should be found in increasing order of magnitude. In the following method, on the other hand, the order in which the factors are found is irrelevant.

We suppose that a, b, c, and d, have been derived by (5), (6), and (7), and that $(\lambda^2 + P\lambda + Q)$ is a known factor of $F(\lambda)$. It may then be shown that

$$(\lambda^2 + P\lambda + Q)\{(\lambda a'' + b'') + (\lambda^2 + p\lambda + q)(\lambda c'' + d'')\}$$
$$\equiv e^2\{(\lambda a + b) + (\lambda^2 + p\lambda + q)(\lambda c + d)\}$$
$$\text{modulo } (\lambda^2 + p\lambda + q)^2, \quad (8)$$

where

$$\left.\begin{aligned} p' &= P - p & q' &= Q - q \\ f &= pp' - q' & e &= fq' - qp'^2 \\ a' &= bp' - aq' & b' &= bf - aqp' \\ c' &= ce - a' & d' &= de - b' - a'p' \\ a'' &= a'e & b'' &= b'e \\ c'' &= d'p' - c'q' & d'' &= d'f - c'qp' \end{aligned}\right\}. \quad (9)$$

Therefore we may effectively divide out the factor $(\lambda^2 + P\lambda + Q)$, by replacing a, b, c, and d, by a'', b'', c'', and d'', repeating the process for every known factor. (Again we are free to introduce a scale factor, should the numbers become too large or small.)

If $(\lambda^2 + P\lambda + Q)$ is not an exact factor of $F(\lambda)$, the result of this replacement is simply that we find ourselves using Bairstow's process to locate the roots of the rational function $F(\lambda)/(\lambda^2 + P\lambda + Q)$. This process will lead correctly to another factor of $F(\lambda)$, provided that neither root of $(\lambda^2 + p\lambda + q)$ comes too close to a root of $(\lambda^2 + P\lambda + Q)$. We may notice that the condition for two such roots actually to coincide is that the quantity e, which appears in equations (9), should vanish.

Example

5. To illustrate the virtues of this method of factor-removal, when the factor is subject to error, we apply it to the quartic polynomial

$$F(\lambda) = \lambda^4 - 111\lambda^2 + 110\lambda = (\lambda^2 - \lambda)(\lambda^2 + \lambda - 110). \quad (10)$$

Suppose first that we have been given $(\lambda^2 + \lambda - 100)$ as an approximation to the second factor, and that we are now using Bairstow's process to find the other factor, starting from $(\lambda^2 + 2\lambda + 1)$. The first step (1) gives us $a = 328$, $b = 108$, $c = -4$, and $d = -109$. The algorithm (9) for factor-removal then gives $a'' = -330200000$, $b'' = -111200000$, $c'' = -406920$, $d'' = 109373120$, from which we deduce the improved

quadratic $(\lambda^2 - 1{\cdot}0004\lambda - 0{\cdot}005)$. The next iteration gives us $(\lambda^2 - 0{\cdot}9999986\lambda + 0{\cdot}000018)$, and convergence is quadratic. If we had simply divided out the second factor, we should have been left with the quadratic $(\lambda^2 - \lambda - 10)$, with no hope of improvement.

Conversely, if we had been given $(\lambda^2 + 2\lambda + 1)$ for the first factor, using Bairstow's method to improve on $(\lambda^2 + \lambda - 100)$ as an approximation to the second, the method of this paper would have given us $(\lambda^2 + 0{\cdot}938\lambda - 111{\cdot}18)$ and $(\lambda^2 + 0{\cdot}9998\lambda - 110{\cdot}003)$ as successive approximations, while simple long division would have left us with $(\lambda^2 - 2\lambda - 108)$.

The point to notice here is that, with our method, Bairstow's process will eventually converge to the correct factor, whereas, with the conventional method, convergence is limited by the accuracy of previously obtained factors.

Mercury Program

6. This process has been programmed for the Mercury computer, the complete program including also a routine which reduces a general square matrix to Hessenberg form by elementary similarity transformations, and a routine which obtains the latent vectors by applying to the Hessenberg matrix an elimination process similar to Wilkinson's method for codiagonal matrices (Wilkinson, 1958). Single-precision arithmetic (29-bit floating binary) is employed throughout.

To start the Bairstow process, the program always takes a quadratic whose roots are near to the average of those roots of $F(\lambda)$ which it has yet to find. This average is easily deduced from the diagonal elements of the matrix.

Precautions, similar to those described by Wilkinson (1959b), are taken to speed up the approach to convergence. If we define the length of an iterative step to be $\{(\delta p/p)^2 + (\delta q/q)^2\}^{\frac{1}{2}}$, special action is taken whenever the length of one step is more than 3 times the length of the preceding step. The program distinguishes two cases here, according to whether or not one root of the quadratic, nevertheless, continues to converge. In case one root continues to converge, it is assumed that the other root of the quadratic corresponds in fact to a pair of complex roots of $F(\lambda)$, and the next trial factor is found by dividing the right-hand side of the congruence (1) by $(\lambda - \lambda_1)$, where λ_1 is the convergent root. This has the effect of diverting the search to the suspected complex roots. In the second case we can only proceed empirically, and the program in fact simply reduces δp and δq so that the length of the new step becomes exactly 3 times that of the previous step.

This program has been in use for some months, and has usually proved successful. On average it is slightly faster than a similar program which locates the roots by Muller's method, to the same accuracy. Failure to converge has generally been attributable to the presence of a cluster of nearly-equal roots. This condition is, however, neither necessary nor sufficient to cause failure. For instance, the program found all the roots of a certain Hessenberg matrix, of order 22, in $12\frac{1}{2}$ minutes. The actual and computed roots are shown in Table 1, and we observe that this matrix has a cluster of 7 roots, all very near the origin. It appears that only one root (the 19th) has been catastrophically miscalculated as a result of this. The error in the sixth root arises from it having been determined from the same quadratic factor as the 19th.

Table 1

	ACTUAL ROOT	COMPUTED ROOT
1	0·6120842	0·6120842
2, 3	0·2859394 ± *i*0·06599475	0·2859394 ± *i*0·06599477
4	0·1859416	0·1859416
5	0·1652502	0·1652502
6	0·1439745	0·1440724
7, 8	0·1222192 ± *i*0·01961939	0·1222191 ± *i*0·01961939
9	0·07952003	0·07952002
10, 11	0·04149856 ± *i*0·06709468	0·04149855 ± *i*0·06709469
12	0·07313159	0·07313159
13	0·06936648	0·06936650
14	0·06520933	0·06520932
15	− 0·02779673	− 0·02779672
16	0·0000015735	0·0000015728
17, 18	0·0000007035 ± *i*0·000001319	0·0000007033 ± *i*0·000001317
19	− 0·000000739	− 0·000161
20	0·000000984	0·000000988
21	0·0000003004	0·0000003007
22	− 0·00000007984	− 0·00000007966

Acknowledgements

The author is grateful to Dr. L. Fox for reading the manuscript of this paper, and to Mr. J. H. Wilkinson for supplying him with the correct figures for Table 1.

References

WILKINSON, J. H. (1958). "The Calculation of the Eigenvectors of Codiagonal Matrices," *The Computer Journal*, Vol. 1, p. 90.
WILKINSON, J. H. (1959a). "Stability of the Reduction of a Matrix to Almost-Triangular and Triangular Forms . . .," *J. Ass. Comp. Mach.*, Vol. 6, p. 336.
WILKINSON, J. H. (1959b). "Evaluation of Zeros of Ill-Conditioned Polynomials," *Num. Math.*, Vol. 1, p. 150.
WILKINSON, J. H. (1960). "Error Analysis of Floating-Point Computations," *Num. Math.*, Vol. 2, p. 319.