ALGORITHM 15

ROOTFINDER II (Modification of Algorithm 2. ROOTFINDER, J. Wegstein, *Communications ACM*, February, 1960)

HENRY C. THACHER, JR.,* Argonne National Laboratory, Argonne, Illinois

```
procedure    ROOT II (f, a, eps, n, g, c, m); integer n, m;
               real procedure f; real a, eps, g, c;
comment      ROOT II computes a value of g = y satisfying
               the equation y = f(y). The iteration will con-
               verge to Y providing that at some time in the
               iteration a g is reached such that abs(g − Y) ×
               abs(d(df/dy)/dy) < 2 × abs((df/dy) − 1),
               where the derivatives are evaluated at Y.
               Input includes (1) f, a procedure for computing
               f(y), (2) a, an initial approximation to the
               root, (3) eps, a tolerance for the relative error
               in g, and (4) n, a maximum number of itera-
               tions to be performed. Output includes: (1) g,
               the required root, (2) c = f(g) − g, (3) m, a
               parameter indicating the success of the pro-
               cedure. If the tolerance was not met, m < 0.
               | m − 1 | gives the number of times that the
               correction to g exceeded the preceding one, an
               indication of instability. ;
begin integer  j; real b, d, h;
               m := 1; if f(0) = 0 then begin g := 0;
                 go to return end
               else g := f(a); b := d := c := a − g;
               if c = 0 then go to return else
               for j := 1 step 1 until n do begin c := f(g) − g;
               if (abs(c/g) < eps then go to return else h :=
                 b/c;
               if h < 0 ∨ h > 2 then m := m + 1 else
               d := d/(h − 1); b := c; g := g + d end
                 iteration
comment      if the system is known to be stable, the if clause
               of the last statement can be omitted;
               m := − m return end
```

* Work supported by the U. S. Atomic Energy Commission.

CERTIFICATION OF ALGORITHM 15

ROOTFINDER II (Revision by Henry C. Thacher, Jr., *Communications ACM*, August, 1960)

HENRY C. THACHER, JR.,* Argonne National Laboratory, Argonne, Illinois

The revision of ROOTFINDER suggested in the preceding remark was programmed by hand for the Royal Precision LGP-30 computer, using a 28-bit mantissa floating point interpretive system (24.2).

The program was tested for the following equations:

(1.k) $f(y) = \arctan y + k\pi$ $(k = 0, 1, 2, 3, 4, 6, 8)$

(2) $f(y) = (y^3 + 1)^{1/3}$

(3) $f(y) = y^3 - 1$

These both have the root 1.3247180428

(4.k) $f(y) = \sinh \alpha_k y$

(5.k) $f(y) = \cosh \alpha_k y$

$(\alpha_1 = -1.2,\ \alpha_2 = -0.5,\ \alpha_3 = 0.5,\ \alpha_4 = 1.2)$

Typical results of these tests were as follows.

f(y)	ϵ	a	g	$[f(g_{-1}) - g_{-1}] \times 10^7$	Remarks
1.0	10^{-8}	1.0000	0.0000000	0.00	
1.1	10^{-8}	3.1415	4.4934094	0.15	
1.2	10^{-8}	6.2832	7.7252518	0.60	
1.3	10^{-8}	9.4248	10.904122	0.00	
1.4	10^{-8}	12.5664	14.066194	0.00	
1.6	10^{-8}	18.8496	20.371303	0.60	
1.8	10^{-8}	25.1327	26.666054	0.60	
1.2	10^{-8}	1.3	1.3247179	0.00	
1.2	10^{-8}	0.5	1.3247179	0.00	
3	10^{-9}	9.0	4.4804900	197.74×10^7	Diverged 2 times, not converged after 20 iter.
3	10^{-9}	5.0	1.3482797	$.51 \times 10^7$	
3	10^{-9}	3.0	1.3247180	0.0	Converged in less than 20 iter.
3	10^{-9}	2.0			Diverged 2 times. Term. with h = 1.
3	10^{-9}	1.1	1.3247180	1384.24	Diverged 9 times. Converged after 20 iter.
3	10^{-9}	1.0			Terminated when g became 0.
3	10^{-9}	0.8	1.3247180	0.00	Diverged 4 times. Conv. in less than 20.
3	10^{-9}	0.6	1.6161598	4.39×10^7	Diverged 2 times. No conv. after 20.
4.k	10^{-9}	1.0	0.00000000	0.0000000	For all k.
5.1, 5.4	10^{-8}	1.0	0.09179585	0.793×10^7	Diverged 7 times. No conv. after 20 iter.
5.2, 5.3	10^{-8}	1.0	1.11787755	0.037	

Function (3) is of particular interest, since it does not converge for most algorithms. With the Wegstein iteration, convergence was obtained, or would have been obtained with a few more iterations for a wide range of initial guesses.

* Work supported by the U. S. Atomic Energy Commission.

REMARK ON ALGORITHM 15

ROOTFINDER II (Henry C. Thacher, Jr., *Comm. ACM*, August 1960)

GEORGE E. FORSYTHE AND JOHN G. HERRIOT, Stanford University, Stanford, California

As pointed out by Lieberstein (*Comm. ACM*, January 1959, p. 5), this algorithm is precisely the Newton method of chords or the scant method applied to g(x) = f(x) — x = 0. Thus convergence is not of second order but rather (for simple roots) of order $\frac{1}{2}(\sqrt{5} - 1) = 1.618$, as shown by Jeeves (*Comm. ACM*, August 1958, pp. 9–10). In the first portion of the algorithm b, c, d, should be set equal to g— a instead of a— g in order to be consistent with the iteration portion. Doing this will usually cut down the number of iterations. Not only is a preliminary test for a zero root desirable but the possibility that g may be zero at any stage of the iteration should be considered in writing the return criterion. The possibility that h = 1 should also be checked and appropriate action taken. Algorithm 26 takes care of these matters and also corrects some minor errors in Algorithm 15. This method is certainly not the best rootfinder that could be written.

REMARKS ON ALGORITHMS 2 AND 3 (*Comm. ACM*, February 1960), ALGORITHM 15 (*Comm. ACM*, August 1960) AND ALGORITHMS 25 AND 26 (*Comm. ACM*, November 1960)

J. H. WILKINSON
National Physical Laboratory, Teddington.

Algorithms 2, 15, 25 and 26 were all concerned with the calculation of zeros of arbitrary functions by successive linear or quadratic interpolation. The main limiting factor on the accuracy attainable with such procedures is the condition of the *method* of evaluating the function in the neighbourhood of the zeros. It is this condition which should determine the tolerance which is allowed for the relative error. With a well-conditioned method of evaluation quite a strict convergence criterion will be met, even when the function has multiple roots.

For example, a real quadratic root solver (of a type similar to Algorithm 25) has been used on ACE to find the zeros of triple-diagonal matrices T having $t_{ii} = a_i$, $t_{i+1,i} = b_{i+1}$, $t_{i,i+1} = c_{i+1}$. As an extreme case I took $a_1 = a_2 = \cdots = a_5 = 0$, $a_6 = a_7 = \cdots = a_{10} = 1$, $a_{11} = 2$, $b_i = 1$, $c_i = 0$ so that the function which was being evaluated was $x^5(x-1)^5(x-2)$. In spite of the multiplicity of the roots, the answers obtained using floating-point arithmetic with a 46-bit mantissa had errors no greater than 2^{-44}. Results of similar accuracy have been obtained for the same problem using linear interpolation in place of the quadratic. This is because the method of evaluation which was used, the two-term recurrence relation for the leading principal minors, *is a very well-conditioned method of evaluation.* Knowing this, I was able to set a tolerance of 2^{-42} with confidence. If the *same function* had been evaluated from its explicit polynomial expansion, then a tolerance of about 2^{-7} would have been necessary and the multiple roots would have obtained with very low accuracy.

To find the zero roots it is necessary to have an absolute tolerance for $|x_{r+1} - x_r|$ as well as the relative tolerance condition. It is undesirable that the preliminary detection of a zero root should be necessary. The great power of rootfinders of this type is that, since we are not saddled with the problem of calculating the derivative, we have great freedom of choice in evaluating the function itself. This freedom is encroached upon if we frame the rootfinder so that it finds the zeros of $x = f(x)$ since the true function $x - f(x)$ is arbitrarily separated into two parts. The formal advantage of using this formulation is very slight. Thus, in Certification 2 (June 1960), the calculation of the zeros of $x = \tan x$ was attempted. If the function $(-x + \tan x)$ were used with a general zero finder then, provided the method of evaluation was, for example

$$x = n\pi + y$$

$$\tan x - x = -n\pi + \frac{\frac{y^3}{3} - \frac{y^5}{30} - \cdots}{\cos y},$$

the multiple zeros at $x = 0$ could be found as accurately as any of the others. With a slight modification of common sine and cosine routines, this could be evaluated as

$$-n\pi + \frac{(\sin y - y) - y(\cos y - 1)}{1 + (\cos y - 1)}$$

and the evaluation is then well-conditioned in the neighbourhood of $x = 0$. As regards the number of iterations needed, the restriction to 10 (Certification 2) is rather unreasonably small. For example, the direct evaluation of $x^{60} - 1$ is well conditioned, but starting with the values $x = 2$ and $x = 1.5$ a considerable number of iterations are needed to find the root $x = 1$. Similarly a very large number are needed for Newton's method, starting with $x = 2$. If the time for evaluating the derivative is about the same as that for evaluating the function (often it is much longer), then linear interpolation is usually faster, and quadratic interpolation much faster, than Newton.

In all of the algorithms, including that for Bairstow, it is useful to have some criterion which limits the permissible change from one value of the independent variable to the next [1]. This condition is met to some extent in Algorithm 25 by the condition S4, that abs(fprt) < abs(x2 × 10), but here the limitation is placed on the permissible increase in the value of the function from one step to the next. Algorithms 3 and 25 have tolerances on the size of the function and on the size of the remainders r1 and r0 respectively. They are very difficult tolerances to assign since these quantities may take very small values without our wishing to accept the value of x as a root. In Algorithm 3 (Comm. ACM June 1960) it is useful to return to the original polynomial and to iterate with each of the computed factors. This eliminates the loss of accuracy which may occur if the factors are not found in increasing order. This presumably was the case in Certification 3 when the roots of $x^5 + 7x^4 + 5x^3 + 6x^2 + 3x + 2 = 0$ were attempted. On ACE, however, all roots of this polynomial were found very accurately and convergence was very fast using single-precision, but the roots emerged in increasing order. The reference to *slow* convergence is puzzling. On ACE, convergence was fast for all the initial approximations to p and q which were tried. When the initial approximations used were such that the real root $x = -6.35099\,36103$ and the spurious zero were found first, the remaining two quadratic factors were of lower accuracy, though this was, of course, rectified by iteration in the original polynomial. When either of the other two factors was found first, then all factors were fully accurate even without iteration in the original polynomial [1].

REFERENCE

[1] J. H. WILKINSON. The evaluation of the zeros of ill-conditioned polynomials Parts I and II. *Num. Math.* 1 (1959), 150–180.