2. ROOTFINDER

J. Wegstein
National Bureau of Standards, Washington 25, D. C.

```
comment     This procedure computes a value of g=x satis-
            fying the equation x=f(x). The procedure calling
            statement gives the function, an initial approxi-
            mation a≠0 to the root, and a tolerance
            paramater ϵ for determining the number of sig-
            nificant figures in the solution. This accelerated
            iteration or secant method is described by the
            author in Communications, June, 1958.;
procedure   Root(f( ), a, ϵ) =: (g)
begin
Root        b := a  ;  c := f(b)  ;  g := c
            if (c=a)  ;  return
            d := a  ;  b := c  ;  e := c
Hob:        c := f(b)
            g := (d×c−b×e)/(c−e−b+d)
            if (abs((g−b)/g)≦ϵ)  ;  return
            e := c  ;  d := b  ;  b := g  ;  go to Hob
end
```

CERTIFICATION

2. ROOTFINDER, J. Wegstein, *Communications ACM*, February, 1960

Henry C. Thacher, Jr.,* Argonne National Laboratory, Argonne, Illinois

Rootfinder was coded for the Royal-Precision LGP-30 Computer, using an interpretive floating point system with 28 bits of significance. The translation from ALGOL was made by hand. Provision was made to terminate the iteration after ten cycles if convergence had not been secured.

The program was tested against the following functions:

(1)	$f(x) = (x + 1)^{1/3}$	(Root = 1.3247180)
(2)	$f(x) = \tan x$	
(3.α)	$f(x) = 2\pi\alpha + \tan^{-1} x$	(α = 1, 2, 3, 4)
(4.α)	$f(x) = \sinh \alpha x$	(α = −1.2, −0.5, 0.5, 1.2)

Selected results were as follows:

f(x)	α	ϵ	x_{k-1}	x_k	
1	1.3	10^{-7}, 10^{-6}	$1.3247\underline{233}$	$1.32\underline{58637}$	(1)
1	1.3	10^{-5}		$1.324716\underline{5}$	(1)
2	5	10^{-3}	−.4674691	$-.\underline{36021288}$	(1, 2)
2	4	10^{-3}	+.84880381	$+.\underline{69496143}$	(1, 2)
3.1	1	10^{-5}		$7.72525\underline{31}$	
3.2	2	10^{-5}		$14.0661\underline{55}$	
3.3	3	10^{-5}		$20.371\underline{026}$	
3.4	4	10^{-5}		$26.665\underline{767}$	

(1) No convergence after 10 iterations. Underlined figures are incorrect.

(2) For this function, $f'(0) = 1$; so convergence is not to be expected at this root. However, the algorithm did not find any other root.

It should be noted that the convergence criterion used fails for a zero root. The provision to terminate after a given number of cycles is therefore essential. Also, double precision is desirable.

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

REMARK ON ALGORITHM 2
ROOTFINDER (J. Wegstein, *Communications ACM*, February, 1960)

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

$$\frac{y_k - Y}{y_{k-1} - Y} = \frac{(y_{k-2} - Y)f''}{2(f' - 1) + (y_{k-1} - y_{k-2})f''} + O(y_{k-1} - Y)^3$$

where Y is the desired root, and the derivatives f′ and f″ are evaluated there. Convergence is thus second order, provided that $|f''|\,|y_{k-1} - Y| < 2\,|f' - 1|$.

The algorithm is, however, somewhat unstable numerically because of the factor $f(y_{k-1}) - f(y_{k-2}) - y_{k-1} + y_{k-2}$ in the denominator.

Experience has shown that the minimum for ϵ is about one half the precision being used. Provision to indicate when round-off errors are causing random oscillations of g would be a desirable addition.

The criterion used for terminating the iteration renders the algorithm unsuitable for a zero root. A preliminary test for a zero root would be desirable. In addition, the algorithm should include provision for exit after a stated number of iterations.

Algorithm 15 appears to offer advantages along these lines.

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

This algorithm has the convergence factor

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

$$x = n\pi + y$$

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

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.