```
begin
    ES[k] := ti[I[k]]  ;
    LS[k] := te[J[k]] − DIJ[k]  ;
    EF[k] := ti[I[k]] + DIJ[k]  ;
    LF[k] := te[J[k]]  ;
    TF[k] := te[J[k]] − ti[I[k]] − DIJ[k]  ;
    FF[k] := ti[J[k]] − ti[I[k]] − DIJ[k]
  end
end CRITICALPATH
```

REFERENCES

(1) JAMES E. KELLEY, JR. AND MORGAN R. WALKER, "Critical-Path Planning and Scheduling," 1959 Proceedings of the Eastern Joint Computer Conference.

(2) M. C. FRISHBERG, "Least Cost Estimating and Scheduling — Scheduling Phase Only," IBM 650 Program Library File No. 10.3.005.

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.

CERTIFICATION OF ALGORITHM 4 BISECTION ROUTINE (S. Gorn, *Comm. ACM*, March 1960)

PATTY JANE RADER,* Argonne National Laboratory, Argonne, Illinois

BISEC was coded for the Royal-Precision LGP-30 computer, using an interpretive floating point system (24.2) with 28 bits of significance.

The following minor correction was found necessary.

α: go to γ_1 should be **go to** γ_i

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

After this correction was made, the program ran smoothly for F(x) = cos x, using the following parameters:

y_1	y_2	ϵ	ϵ_1	Results
0	1	.001	.001	FLSXT
0	2	.001	.001	1.5703
1.5	2	.001	.001	1.5703
1.55	2	.1	.1	1.5500
1.5	2	.001	.1	1.5625

These combinations test all loops of the program.

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

REMARK ON ALGORITHM 16
CROUT WITH PIVOTING (G. E. Forsythe, *Comm. ACM*, *3* (Sept. 1960), 507–8.)

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

This procedure contains the following errors:

a. In SOLVE, the expression

```
c[k] := c[k] − INNERPRODUCT
   (B[k, p], c[p], p  1, k − 1)
```

should read:

```
c[k] := c[k] − INNERPRODUCT
   (B[k, p], c[p], p, 1, k − 1)
```

b. In CROUT, the specification part should read:

```
array A, b, y  ; integer n  ; integer array pivot  ;
```

c. In SOLVE, the specification part should read:

```
array B, c, z  ; integer n  ; integer array pivot  ;
```

The efficiency of the algorithm will be improved by the following changes:

a. In the elimination phase of CROUT, replace

```
for i := k + 1 step 1 until n do
begin quote := 1.0/A[k, k]  ;  A[i, k] := quot XA[i, k] end  ;
```

by

```
quot := 1.0/A[k, k]  ;  for i := k + 1 step 1 until n do
    A[i, k] := quot XA[i, k]  ;
```

b. Omit INNERPRODUCT from the formal parameter list in both CROUT and SOLVE, and declare INNERPRODUCT either locally, or globally. This avoids any reference to INNERPRODUCT in the calling sequence produced by a compiler.

It is also to be noted that a minor modification of CROUT allows it to be used to evaluate the determinant of A.

All of these suggestions are included in a later algorithm.

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

REMARK ON ALGORITHM 25
REAL ZEROS OF AN ARBITRARY FUNCTION (B. Leavenworth, *Comm. ACM*, November 1960)

ROBERT M. COLLINGE
Burroughs Corporation, Pasadena, California

On attempting to use this algorithm, I discovered the two following errors:

(1) The line following the SWITCH statement should read:

```
for L := 1 step 1 until n do
```

(2) The line starting with the label loop: should read:

```
loop: dd  :=  1 + d  ; bi = x0 × d↑2 − x1 × dd↑2
    + x2 × (dd + d)  ;
```

With these two modifications incorporated the algorithm was translated into the language of the Burroughs Algebraic Compiler and has been used successfully on the Burroughs 220 Computer.

"COMPUTERS— KEY TO TOTAL SYSTEMS CONTROL" IS THEME OF 1961 EASTERN JOINT COMPUTER CONFERENCE

Bruce G. Oldfield, Program Chairman, Calls For Papers To Be Presented December 12-14, Sheraton-Park Hotel, Washington, D.C.

The 1961 Eastern Joint Computer Conference Committee has announced that the theme for this year's conference, to be held December 12–14 at the Sheraton-Park Hotel in Washington, D. C., will be "Computers—Key to Total Systems Control".

Bruce G. Oldfield, Program Chairman, states that this theme reflects one of the most significant trends in modern computer technology. "Until quite recently, computers were considered to be data processing ends in themselves," Mr. Oldfield points out. "Now they are more and more being treated as merely one element—although the most vital one—in total systems for government, defense, industry and business management operations. Other important elements in the closed loop for the 'total system" are data acquisition, digital data communications, display, and actual control or guidance."

The 1961 EJCC will follow this total systems approach by presenting the latest advances in equipment and concepts leading toward computer control of present and future systems. Mr. Oldfield called for papers in such representative areas as:

Business Management Control	Network Control
Military and Space Command Control Systems	Man-Machine Systems
Industrial Process Control	Self Organizing Systems
Real Time Systems	High Speed Digital Data Communications

Each person wishing to contribute a paper to the program should submit two copies of both a 100-word abstract and a two-page summary to:

Bruce G. Oldfield
IBM Federal Systems Division
326 E. Montgomery Avenue
Rockville, Maryland

The deadline for submission of abstracts and summaries is June 20, 1961. Authors whose papers are chosen for presentation will be promptly notified.

Inasmuch as papers will be published prior to the Conference and made available to the attendees, the full text of papers chosen for presentation must be submitted to the Program Chairman by September 1, 1961.