ALGORITHM 24
SOLUTION OF TRI-DIAGONAL LINEAR EQUATIONS

B. Leavenworth
American Machine & Foundry Co., Greenwich, Conn.

```
procedure TRIDAG (n, A, B, C, D) ; integer n ;
array A, B, C, D ;
comment: This procedure[1] finds the solution of an n × n system
    of linear equations whose matrix is in tridiagonal form, that
    is, a_ij = 0 for |i − j| ≧ 2. Parameters are: the main diagonal
    B_p, the diagonal just below A_r, the diagonal just above C_r,
    the right-hand side D: (where p = 1, . . . , n and r = 1, . . . ,
    n − 1) and the matrix order n. The solution vector replaces
    the input vector D and the vector B is also destroyed in the
    process ;
begin
    real w ; integer j ;
    D[1] := D[1]/B[1] ; w := B[1] ;
    for j := 2 step 1 until n do
        begin B[j − 1] := C[j − 1]/w ; w := B[j] − A[j − 1]
          × B[j − 1] ;
            D[j] := (D[j] − A[j − 1] × D[j − 1])/w end ;
    for j := 1 step 1 until n − 1 do
        D[n − j] := D[n − j] − B[n − j] × D[n − j + 1]
end TRIDAG
```

[1] D. W. Peaceman and H. H. Rachford, Jr., The Numerical Solution of Parabolic and Elliptic Differential Equations, Journal of the Soc. for Ind. and Applied Math. Vol. 3 March 1955.

ALGORITHM 25
REAL ZEROS OF AN ARBITRARY FUNCTION

B. Leavenworth
American Machine and Foundary Co., Greenwich, Conn.

```
procedure ZEROS(n, C, FUNCTION, m, ep1, ep2, ep3, eta) ;
integer n, m ; real ep1, ep2, ep3, eta ; array C ;
procedure FUNCTION ;
comment: This procedure finds the real zeros of an arbitrary
    function using Muller's method[1, 2] and is adapted from a
    FORTRAN code by Frank.[3] Each iteration determines a zero
    of the quadratic passing through the last three function
    values. Parameters include the number of roots desired n.
    If C_i is zero, starting values are −1, 1, 0 respectively. If
    C_i = β then the starting values are .9β, 1.1β, β. The procedure
    FUNCTION(rt, frt) must be supplied to evaluate the func-
    tion value frt, given the argument rt. m is the maximum
    number of iterations permitted. ep1 is the relative conver-
    gence criterion on successive iterates. ep2 is the absolute
    convergence criterion on the function value. eta is the spread
    for multiple roots, that is, if |rt − C_i| < ep3 where C_i is a
    previously found root, then rt is replaced by rt + eta ;
begin integer L, jk, i, mm ; real p, p1, p2, x0, x1, x2, rt,
                frt, fprt, d, dd, di, h, bi, den, dn, dm, tem ;
    switch S : S1, S2, S3, S4 ;
    for L := 1 step 1 until m do
    begin jk := 0 ; if C[L] = 0 then go to initial else
    go to assign ;
initial: p := −1 ; p1 := 1 ; p2 := 0 ; go to start ;
assign: p := .9 × C[L] ; p1 := 1.1 × C[L] ; p2 := C[L] ;
start: rt := p ; go to fn ;
enter: go to S[if jk < 4 then jk else 4] ;
S1: rt := p1 ; x0 := fprt ; go to fn ;
S2: rt := p2 ; x1 := fprt ; go to fn ;
S3: x2 := fprt ; h := if C[L] = 0 then −1
    else −.1 × C[L] ; d := −.5 ;
loop: dd := 1 + d ; bi := x0 × d ↑ 2 − x1 × dd ↑ 2 × x2 ;
        (dd + d) ;
    den := bi ↑ 2 − 4 × x2 × d × dd × (x0 × d − (x1 × dd) + x2)
    if den ≦ 0 then den := 0 else den := sqrt(den) ;
    dn := bi + den ; dm := bi − den ;
    if abs(dn) ≦ abs(dm) then den := dm else den := dn ;
    if den = 0 then den := 1 ;
    di := −2 × x2 × dd/den ; h := di × h ; rt := rt + h
    go to if abs(h/rt) < ep1 then call else fn ;
S4: if abs(fprt) < abs(x2 × 10) then
    begin x0 := x1 ; x1 := x2 ; x2 := fprt ; d := di
    go to loop end else begin di := di × .5 ; h := h × .5
    rt := rt − h ; go to fn end ;
fn: jk := jk + 1 ; if jk < m then mm := 1 else mm := 0
call: FUNCTION(rt, frt) ; if mm = 1 then go to compute
    else go to root ;
compute: fprt := frt ;
    for i := 2 step 1 until L do
    begin tem := rt − C[i − 1] ; if abs(tem) < ep3 then go
    change else fprt := fprt/tem end
test: if abs(frt) < ep2 ∧ abs(fprt) < ep2 then go to root
    else go to enter ;
change: rt := rt + eta ; jk := jk − 1 ; go to fn ;
root: C[L] := rt end L
end ZEROS
```

[1] D. E. Muller, A Method for Solving Algebraic Equatic Using an Automatic Computer, *MTAC 10* (1956).

[2] W. L. Frank, Finding Zeros of Arbitrary Functions, *J. AC 5* (1958).

[3] W. L. Frank, RWGRT, General Root Finder 704 Fortr Source Language Subroutine Share Distribution # 635. Para eters used by Frank are: ep1 = 10^{-6}, ep2 = 10^{-20}, ep3 = 10^{-} eta = 10^{-3}.

REMARK ON ALGORITHM 15
ROOTFINDER II (Henry C. Thacher, Jr., *Com ACM*, August 1960)

George E. Forsythe and John G. Herriot, Stanf University, Stanford, California

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