4. Bisection Routine

S. Gorn
University of Pennsylvania Computer Center
Philadelphia, Pa.

```
comment     This procedure evaluates a function at the end-points
            of a real interval, switching to an error exit (fools
            exit) FLSXT if there is no change of sign. Otherwise
            it finds a root by iterated bisection and evaluation
            at the midpoint, halting if either the value of the
            function is less than the free variable ε or two suc-
            cessive approximations of the root differ by less
            than ε1. ε should be chosen of the order of error in
            evaluating the function (otherwise time would be
            wasted), and ε1 of the order of desired accuracy. ε1
            must not be less than two units in the last place
            carried by the machine or else indefinite cycling will
            occur due to round-off on bisection. Although this
            method is of 0 order, and therefore among the slow-
            est, it is applicable to any continuous function. The
            fact that no differentiability conditions have to be
            checked makes it, therefore, an 'old work-horse'
            among routines for finding real roots which have
            already been isolated. The free variables y1 and y2
            are (presumably) the end-points of an interval within
            which there is an odd number of roots of the real
            function F. α is the temporary exit for the evalua-
            tion of F.;
procedure   Bisec(y1, y2, ε, ε1, F( ), FLSXT) =: (x)
begin
Bisec:      i := 1 ; j := 1 ; k := 1 ; x := y2
α:          f := F(x) ; if (abs(f) ≦ ε) ; return
            go to γi
First val:  i := 2 ; f1 := f ; x := y1 ; go to α
Succ val:   if (sign(f) = sign(f1)) ; go to δj ; go to ηk
Sec val:    j := 2 ; k := 2
Midpoint:   x := y1/2 + y2/2 ; go to α
Reg δ:      y2 := x
Precision:  if (abs(y1 − y2) ≧ ε1) ; go to Midpoint
            return
Reg η:      y1 := x ; go to Precision
            integer (i, j, k)
            switch γ := (First val, Succ val)
            switch δ := (FLSXT, Reg δ)
            switch η := (Sec val, Reg η)
end Bisec
```

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.