Example 2. Acoustic Waveguide Function

$$f(x) = P\frac{\sin(2\pi S)}{S} + \rho\cos(2\pi S), \quad \rho = 2.50$$

$$P = \sqrt{(k^2 - s^2)}, \quad Re(P) < 0; \qquad S = \sqrt{((x/e)^2 - k^2)}, \quad Re(s) < 0,$$
$$k = 0.0, \quad e = 0.285; \qquad P1 = 0.0, \quad P2 = 0.02, \quad P3 = 0.04$$

Iteration	$x = u + iv$		$f(x) = u + iv$	
	u	v	u	v
1	0.67672247−01	0.44008261−02	0.24001462−00	0.49014406−01
2	0.71677143−01	0.51274250−02	0.23259824−01	0.98558815−02
3	0.72102452−01	0.53056952−02	0.20055826−03	0.19331276−07
4	0.72100262− 01	0.53092607−02	0.39539112−07	0.40978193−07

Example 3. The 20 Roots of Unity

Starting values at approximately 19, 27, and 35 degrees. Conjugates accepted as roots. $eps1 = eps2 = .5_{10} - 7$. Roots are at $e^{r\cdot\pi/10}$, $r = 0, \cdots, 19$.

Root	r	Number of iterations
0	3	5
1	17	*
2	4	15
3	16	*
4	7	22
5	13	*
6	9	14
7	11	*
8	5	14
9	15	*
10	1	10
11	19	*
12	2	7
13	18	*
14	6	10
15	14	*
16	0	5
17	8	8
18	12	*
19	10	2

* Asterisk in column 3 indicates conjugate taken; i.e., $z_1 = \bar{z}_0$, $z_3 = \bar{z}_2$, *etc.*

different machines have been made. In every case differences can be satisfactorily explained by (1) different BCD-binary conversion on different machines, (2) different word lengths, or (3) differences in library subroutines on the various machines.

Miscellaneous Comments. Both versions of Muller's method (Algorithm 196 and Traub's iteration function) have the advantage over other one-point-with-memory methods in that it is possible to locate complex zeros using real starting values. There is, of course, the possibility of spending unnecessary time doing complex arithmetic. As a general purpose library routine, unless one is looking for complex zeros, there are other iterative functions that require less space and have the same order; there are also others requiring less space and enjoying higher order, although they usually involve computing derivatives. For our purposes, the Traub version of Muller's method has proved quite satisfactory.

Example 2 is included to indicate the behavior of the Algorithm using a non-algebraic function. With this function both Algorithm 196 and the Traub modification agreed exactly except for a difference in the last digit in the first iterate.

Example 3 is included to show the order in which the 20 roots of unity are found and the number of iterations required to attain the specified accuracy. Values were checked against the NBS *Tables of Sines and Cosines to 15 Decimal Places*. All but two roots were correct to 9 significant figures; the remaining two were correct to 8.

Acknowledgment. The support of this work by the Atomic Energy Commission, Contract AT(29-2)-1103, is gratefully acknowledged.

References:

1. Frank, Werner L. Finding zeros of arbitrary functions. *J. ACM 5* (1958), 154–160.
2. Muller, David E. A method for solving algebraic equations using an automatic computer. *MTAC 10* (1956), 208–215.
3. Traub, J. F. *Iterative Methods for the Solution of Equations.* Prentice-Hall, Englewood Cliffs, N. J., 1964.
4. IBM Systems Reference Library. File No. 7090-27. Form C28-6389-2. IBM 7090/7094 IBSYS Operating System. Version 12. IBJOB Processor.

REMARK ON ALGORITHM 291 [S14] LOGARITHM OF GAMMA FUNCTION [M.C. Pike and I. D. Hill, *Comm. ACM 9* (Sept. 1966), 684]

Miss M. R. Hoare (Recd. 24 Aug. 1967)
c/o C. Hoare and Co., 37 Fleet St., London, E.C.4.

(1) **if** $x < 7.0$ **then**
begin $f := 1.0$; $x := x - 1.0$;
for $x := x + 1.0$ **while** $x < 7.0$ **do**
would be better written as:
if $x < 7.0$ **then**
begin $f := 1.0$;
for $x := x, x + 1.0$ **while** $x < 7.0$ **do**
This avoids unnecessary operations.

(2) In the final statement, **loggamma** should read *loggamma*

REMARK ON ALGORITHM 307 [A1] SYMMETRIC GROUP CHARACTERS [J. K. S. McKay, *Comm. ACM 10* (July 1967), 451]

J. K. S. McKay (Recd. 13 Sept. 1967)
Dept. of Computer Science, University of Edinburgh, Edinburgh, Scotland

Three corrections are noted.

(1) Line 39:

own integer array *p*[0:*n*,0:*n*];

should be moved to the line after the **begin** in line 32.

(2) At *E* the line should read

E: **if** *rep*[*j*2] ≥ (**if** *j*2=*k* **then** 0 **else** *rep*[*j*2+1])
then go to *E*;

(3) Three lines, later

coeff := −1 *coeff*;

should read

coeff := −*coeff*;