ALGORITHM 32
MULTINT

R. Don Freeman Jr.
Michigan State University, East Lansing, Michigan

real procedure MULTINT (n, Low, Upp, Funev, s, P, u, w);
value n;
real procedure Low, Upp, Funev; **array** s, u, w; **integer** n;

comment MULTINT will perform a single, double, triple,..., T-order integration depending on whether n=1, 2,..., T. The result is:

$$\text{MULTINT} = \int_{\text{Low}(1)}^{\text{Upp}(1)} \text{Funev}(1, x_1)\, dx_1 \int_{\text{Low}(2, x_1)}^{\text{Upp}(2, x_1)} \text{Funev}(2,x_1,x_2)\, dx_2 \cdots \int_{\text{Low}(n,x_1,\ldots,x_{n-1})}^{\text{Upp}(n,x_1,\ldots,x_{n-1})} \text{Funev}(x_1,\ldots,x_n)\, dx_n$$

The variable of integration is x[j]. j=1 refers to the outermost integral, j=n, the innermost integral. The code divides each interval equally into s[j] subintervals and performs a P-point Gaussian integration on each subinterval with weight functions w[k[j]] and abscissas u[k[j]]. P is the size of the arrays of weight functions and abscissas and must be provided by the main code along with these arrays.

Since the values x[1], x[2],..., x[n], are stored in an array, as are a, b, c, d, r, it is necessary to substitute an integer for the upper bound T of these arrays before the program is executed. This means, for example, if 3 is substituted for T, then the procedure will not do a 4th order integral unless it is retranslated with T ≥ 4.

The values of the lower and upper bounds and functions must of course be specified at the time of use. If each of these constituted a separate procedure, it would require writing and translating 3n different procedures. This is eliminated by grouping them into Low, Upp, and Funev which compute the lower and upper bounds and value of the functions respectively in each of the jth integrals. Since these are each essentially a collection of "subprocedures," the first statement of each should be a switch directing the code to the "subprocedure" which is used in the jth integral. Note that, for example, Low(3,x) is formally a function of x[1], x[2],..., x[T]; this is done merely because it is more convenient to make Low(j,x) formally a function of the whole array x for all j. Actually of course Low(3, x) would be a function of x[1] and x[2] only;

```
begin   real array a, b, c, d, r, x[1:T];
        integer array k, h[1:T]; real f; integer j, m;
        for j :=1 step 1 until T do
            x[j] := 0.0;
        m := 1;
        r[n+1] := d[n+1] := 1.0;
setup:  for j := m step 1 until n do
        begin
          a[j] := Low(j,x);
          b[j] := Upp(j,x);
          d[j] := (b[j]- a[j])/s[j];
          c[j] := a[j] + 0.5 × d[j];
          x[j] := c[j] + 0.5 × d[j] × u[1];
          r[j] := 0.0;
          h[j] := k[j] := 1; end;
          j := n;
sum:    f := Funev(j,x);
        r[j] := r[j] + r[j+1] × d[j+1] × f × w[k[j]];
        if (k[j] < P) then go to labk;
        if (h[j] < s[j]) then go to labh;
        j := j-1;
        if (j = 0) then go to exit;
        go to sum;
labh:   h[j] := h[j] + 1;
        c[j] := a[j] + (h[j] - 0.5) × d[j];
        k[j] := 1;
        go to initalx;
labk:   k[j] := k[j] + 1;
initalx: x[j] := c[j] + 0.5 × d[j] × u[k[j]];
        if (j=n) then go to sum;
        m := j+1;
        go to setup;
exit:  MULTINT := r[1] × d[1] × 0.5↑ n;  end
```

CERTIFICATION OF ALGORITHM 32
MULTINT [R. Don Freeman, *Comm. ACM*, Feb. 1961]

Henry C. Thacher, Jr.*
Reactor Engineering Div., Argonne National Laboratory, Argonne, Ill.

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

The procedure was transcribed into the ACT-III language for the LGP-30 computer, and was tested on the integrals:

$$(1)\quad \int_0^1\int_0^1\int_0^1\int_0^1 k[\cos u - 7u \sin u - 6u^2 \cos u + u^3 \sin u]\, dw\, dx\, dy\, dz = \sin k$$

where $u = kwxyz$, and

$$(2)\quad \int_0^1 \int_0^{\sqrt{1-x^2}} \int_0^{\sqrt{1-x^2-y^2}} \frac{dz\, dy\, dx}{x^2 + y^2 + (z-k)^2} = \pi \left(2 + \frac{1}{2}\left(\frac{1}{k} - k\right) \log \left|\frac{1+k}{1-k}\right|\right).$$

The Algol procedures for the second integral are:

```
real procedure Low (j,x);
Low := 0;
real procedure Upp(j,x); comment z ≡ x[3], y ≡ x[2], x ≡
  x(1);
begin
integer i; real temp;
temp := 1.0;
for i := j-1 step - 1 until 1 do
temp := temp - x[j] × x[j];
Upp := sqrt(temp)
end;
real procedure Funev(j,x);
comment The real parameter k is global;
Funev := if j < 3 then 1.0 else 1/(x[1]×x[1]+x[2]×x[2]+(x[3]-k)
  ↑2);
```

The first integral was tested only with $s[j] = 1$, and with various Gaussian formulas for integrals over the interval (−1,+1). Results were as follows:

k	$\pi/2$	π	$3\pi/2$	2π
true	1.0000000	0.0000000	−1.0000000	0.0000000
p = 2	0.993704	−0.0333603	+0.020166	6.881490
p = 3	1.000032	0.0000848	−1.061651	−0.597419
p = 4	0.999999	0.0000001	−0.998407	+0.0027035
p = 5	1.000000	−0.0000002	−1.000028	−0.0007857

For the second integral, two values of $s = s[1] = s[2] = s[3]$ were used, and two values of p. Results were as follows:

k	1/2		2	
true	11.46027376		1.10609687	
s	1	2	1	2
p = 2	5.454460	11.838651	1.0368770	1.1184305
p = 3	9.361666	12.408984	1.1343551	1.1094278

The effect of the pole at $(0,0,k)$ is obvious.

For the algorithm to run in any compiler, the semicolon following $x[T]$; in the fourth line above the end of the comment must be deleted. The array bounds on the arrays r and d must be increased to $[1 : T+1]$.

For a system which permits variable array bounds, the introduction of the integer T appears superfluous. For such a system, T may be replaced by n throughout with a probable gain in efficiency. For most translators, the presence of undefined elements in an array will not cause difficulties, provided these elements do not appear in an expression before they are assigned a value. The statement "**for** $j := 1$ **step** 1 **until** T **do** $x[j] := 0.0$;" is thus superfluous. The semicolon before the **end** which precedes the label "*sum*" also appears unnecessary.

In spite of these minor corrections, the algorithm appears to be extremely convenient for multiple quadratures over arbitrary regions using the Cartesian product of any explicit one-dimensional formula (and not merely a Gaussian formula) for integrating over the range $[-1,1]$. If endpoints are used in the formula, it will, of course, repeat the calculation for each section of the range.

REMARKS ON ALGORITHM 32 [D1]
MULTINT [R. Don Freeman, Jr., *Comm. ACM 4* (Feb. 1961), 106]
AND
CERTIFICATION OF ALGORITHM 32 [Henry C. Thacher, Jr., *Comm. ACM 6* (Feb. 1963), 69]

K. S. Kölbig
Data Handling Division, European Organization for Nuclear Research (CERN), 1211 Geneva 23, Switzerland

KEY WORDS AND PHRASES: numerical integration, multidimensional integration, Gaussian integration
CR CATEGORIES: 5.16

The real procedure *MULTINT* was corrected according to the certification. It was then compiled on a CDC 3800 computer and tested on the second integral given in the certification. It became apparent that

(i) Equation (2) of the certification should read

$$\int_{-1}^{1}\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}}\int_{-\sqrt{1-x^2-y^2}}^{\sqrt{1-x^2-y^2}} \frac{dz\,dy\,dx}{x^2+y^2+(z-k)^2} = \pi\left(2+\left(\frac{1}{k}-k\right)\log\left|\frac{1+k}{1-k}\right|\right) \qquad (2)$$

It should be noted that the right-hand side of equation (2) as printed in the certification does not correspond either to the original limits or to those given above.

(ii) the statement

Low := 0;

in the real procedure *Low* should be replaced by

Low := $-Upp(j, x)$;

(iii) the second line of the **for** statement in the real procedure *Upp* should read

temp := *temp* − $x[i] \times x[i]$;

After making these corrections, it is possible to obtain results corresponding to a permuted version of the table given in the certification, which should be replaced by the following:

k	$\frac{1}{2}$		2	
true	11.46027375		1.10609686	
s	1	2	1	2
$P = 2$	5.454466	9.361670	1.0368787	1.1184317
$P = 3$	11.838664	12.408983	1.1343568	1.1094294

In addition, since several compilers require specifications, it would be desirable

(i) to change the last specification in the heading of *MULTINT* to read

integer n, P;

(ii) to insert the specifications

integer j; **array** x;

in the heading of the real procedures *Low*, *Upp*, and *Funev*.

Some of these additions were necessary in order to ensure correct results with the compiler used for the tests.