must be left as standard floating-point numbers in two designated registers.

EXERCISE AMB

The following function ln* $(1 + X)$ is an approximation for ln $(1 + X)$ in the range $0 \leq X \leq 1$:

$$\begin{aligned}\ln^*(1+X) &= 0.9974{,}442X \\ &\quad - 0.4712{,}839X^2 \\ &\quad + 0.2256{,}685X^3 \\ &\quad - 0.0587{,}527X^4\end{aligned}$$

Compute the error term, i.e. ln $(1 + X)$ − ln* $(1 + X)$, for $X = 0(0.02)1.0$. Set

x = the mean of their absolute values;

y = the absolute value of the greatest error.

Draw (by hand) a graph of the error against X.

EXERCISE AWE

Declare a real procedure

Simpint (f, a, b, n)

which integrates the function f over the range (a, b) using Simpson's rule with n intervals. This rule is given by the approximation

$$(h/3) \times (f(a) + 4f(a + h) + 2f(a + 2h) + 4f(a + 3h) + \cdots + 2f(b - 2h) + 4f(b - h) + f(b))$$

where $h = (b - a)/n$ and n is even.

Declare the real procedure

trap (x) = 0.92 × cosh (x) − cos (x)

and integrate it over $(-1, 1)$ using 2, 4, 8, 16, ⋯ intervals until two successive results differ by less than 10^{-9}. Set

x = the final result;

y = the final number of intervals.

Print out the results of the successive approximations: what would the result have been if the accuracy required was 10^{-6}? Any comments?

EXERCISE AT4

Read an integer n from data, followed by an $n \times n$ matrix A listed by rows (floating-point). Denote by $R_i(C_i)$ the sum of the absolute values of the elements in row i (column i). Set

x = trace (A), i.e. $A_{11} + A_{22} + \cdots + A_{nn}$;

y = maximum $(R_1, R_2, \cdots R_n, C_1, C_2, \cdots C_n)$.

(x = sum of eigenvalues, y = upper bound on magnitude of eigenvalues.)

RECEIVED AUGUST, 1968; REVISED NOVEMBER, 1968

REFERENCES

1. HOLLINGSWORTH, J. Automatic graders for programming classes. *Comm. ACM 3*, 10 (Oct. 1960), 528–529.
2. NAUR, P. Automatic grading of students' ALGOL programming. *BIT 4* (1964), 177–188.
3. FORSYTHE, G. E. AND WIRTH, N. Automatic grading programs. *Comm. ACM 8*, 5 (May 1965), 275–278.
4. TEMPERLY, J. F. AND SMITH, B. W. A grading procedure for PL/1 student exercises. *Comput. J. 10* (Feb. 1968), 368–370.

Algorithms

J. G. HERRIOT, Editor

The following algorithm by Bartels and Golub relates to the paper by the same authors in the Numerical Analysis department of this issue, on pages 266–268.

This concurrent publication in Communications follows a policy announced by the Editors of the two departments, J. G. Herriot and J. F. Traub, in the March 1967 issue.

ALGORITHM 350
SIMPLEX METHOD PROCEDURE EMPLOYING LU DECOMPOSITION* [H]

RICHARD H. BARTELS AND GENE H. GOLUB (Recd. 2 Aug. 1967 and 5 June 1968)

Computer Science Department, Stanford University, Stanford, CA 94305

* This project was supported in part by contracts NSF GP948 and ONR NR 044 211.

KEY WORDS AND PHRASES: simplex method, linear programming, LU decomposition, round-off errors, computational stability

CR CATEGORIES: 5.41

```
procedure linprog (m, n, kappa, G, b, d, x, z, ind, infeasible, unbounded, singular);
  value m, n; integer m, n, kappa; real z;
  array G, b, d, x; integer array ind; label infeasible, unbounded, singular;
comment linprog attacks the linear programming problem:
                maximize d^T x
                subject to Gx = b and x ≥ 0
```

Details about the methods used are given in a paper by Bartels and Golub [*Comm. ACM 12* (May 1969), 266–268].

The array $G[0{:}m-1, 0{:}n-1]$ contains the constraint coefficients. Array $b[0{:}m-1]$ contains the constraint vector, and $d[0{:}n-1]$ contains the objective function coefficients (cost vector). The computed solution will be stored in $x[0{:}n-1]$, and z will have the maximum value of the objective function if *linprog* terminates successfully. Error exit *singular* will be taken if a singular basis matrix is encountered. Error exit *infeasible* will be taken if the given problem has no basic feasible solution, and exit *unbounded* will be taken if the objective function is unbounded. If *kappa* = 0, problem (2) of the referenced paper will be set up and phase 1 entered. If $1 \leq$ *kappa* $\leq m - 1$, problem (4) of the paper will be set up and phase 1 entered. The last *kappa* columns of G will be preceded by the first $m -$ *kappa* columns of the identity matrix to form the initial basis matrix. If *kappa* = m, phase 2 computation will begin on problem (1) with variables numbered $ind[0], \cdots, ind[m-1]$ as the initial basic variables and variables numbered $ind[m], \cdots, ind[n-1]$ as the initial nonbasic variables. Hence each component of *ind* must hold an integer between 0 and $n - 1$ specified by the user. Finally, if *kappa* > m, problem (3) will be set up, and phase 2 computation will begin with variables numbered $ind[0], \cdots, ind[m]$ as the initial basic variables and variables numbered $ind[m+1], \cdots, ind[n+kappa-m-1]$ as the initial nonbasic variables. This option is of interest only because *linprog*, upon successful termination, leaves all variable numbers recorded in

ind in their final order and provides *kappa* with an appropriate value. This permits *linprog* to be reentered at the phase 2 point after modifications have been made to *G*, *b*, or *d*. An understanding of the simplex method and the accompanying paper by Bartels and Golub will make clear what modifications can be permitted. If phase 1 is to be executed, *ind* must have array bounds [0:*m*+*n*−*kappa*] to allow for artificial variables. Otherwise, *ind* must have bounds [0:*n*+*kappa*−*m*−1]. The values in array *b* must be nonnegative if phase 1 is to be executed. The contents of *m*, *n*, *G*, *b*, and *d* are left unchanged by *linprog*;

```
begin
  real procedure ip2(ii, ll, uu, aa, bb, cc);
    value uu; integer ii, ll, uu; real aa, bb, cc;
  begin
  comment ip2 must produce a double-precision, accumulated
    inner product. Jensen's device is used. The main statement in
    ip2 is
     for ii := ll step 1 until uu do sum := sum + aa × bb
    where the local variable sum has been initialized by cc. How-
    ever, the multiplication aa × bb must produce a double-pre-
    cision result, so sum represents a double-precision accumu-
    lated sum. After all products have been summed together, sum
    is to be rounded to single-precision and used as the value of
    ip2;
  end ip2;
  procedure trisolv(fis, fid, fie, sis, sie, fi, si, sol, rhs, mat, piv);
    value fid, fie; integer fis, fid, fie, sis, sie, fi, si; real sol, rhs,
    mat, piv;
  comment trisolv solves a triangular system of linear equa-
    tions. The off-diagonal part of the system's coefficient matrix
    is given by mat, the diagonal part by piv, and the right-hand
    side of the system by rhs. The solution is developed in sol.
    By appropriately setting the first five parameters, either an
    upper or a lower triangular system can be treated. Column by
    column LU decomposition of a matrix can be compactly ex-
    pressed using trisolv;
  begin real tt, pv;
    for fi := fis step fid until fie do
    begin tt := −ip2(si, sis, sie, sol, mat, −rhs);
      si := fi; pv := piv;
      sol := if pv = 1.0 then tt else tt/pv
    end
  end trisolv;
  array q, h, w, y, v[0:m], P[0:m, 0:m];
  integer array ix[0:m+n], ro[0:m];
  integer mu, nu, alpha, beta, gamma, gm1, im1, i, j, k, l;
  real t1, t2, infinity, prevz, eta;
  real procedure Gmat(ri, ci);
    value ri, ci; integer ri, ci;
      Gmat := if ri = m then (if ci < n then 0 else 1.0)
              else if ci < n then G[ri, ci]
              else if ci − n = ri then 1.0 else 0;
  real procedure dvec(ii); value ii; integer ii;
    dvec := if ii < n then d[ii] else 0;
  procedure decompose (mat, bottom, top);
    value bottom, top; integer bottom, top; real mat;
  comment This procedure performs a column-by-column re-
    duction of the matrix given by mat, forming an upper and a
    lower triangular matrix into the array P. (Each diagonal ele-
    ment of the lower triangular matrix is 1.) Interchanges of rows
    take place so that the largest pivot in each column is em-
    ployed. If P already contains the LU decomposition of a
    matrix differing from mat in only the (beta)-th column, ad-
    vantage is taken of this. The parameters bottom and top enable
    decompose to concentrate on a lower right-hand submatrix of
    mat. This feature saves computation during phase 1. If mat
    is singular, exit singular is taken;
  begin
    for i := beta step 1 until mu do
    begin
      im1 := i − 1; l := ix[i];
      trisolv(if i=beta then bottom else top, 1, im1, bottom, j − 1,
        j, k, P[ro[k], i], mat, P[ro[j], k], 1.0);
      trisolv(i, 1, mu, bottom, im1, j, k, P[ro[k], i], mat,
        P[ro[j], k], 1.0);
      t1 := 0;
      for j := i step 1 until mu do
      begin
        t2 := P[ro[j], i];
        if abs(t1) < abs(t2) then begin t1 := t2; k := j end
      end;
      if t1 = 0 then go to singular;
      if i = mu then go to decompover;
      j := ro[i]; ro[i] := ro[k]; ro[k] := j;
      for j := i + 1 step 1 until mu do P[ro[j], i] :=
        P[ro[j], i]/t1
    end;
decompover:
  end decompose;
  procedure findbeta;
  comment This procedure determines which of the basic
    variables is to become nonbasic;
  begin
    t1 := infinity;
    for i := 0 step 1 until mu do
    begin
      if y[i] > 0 then
      begin
        t2 := h[i]/y[i];
        if t2 < t1 then begin t1 := t2; beta := i end
      end
    end
  end findbeta;
  procedure findalpha(mat, vec); real mat, vec;
  comment This procedure determines which of the nonbasic
    variables is to be made basic;
  begin
    t1 := infinity;
    for i := mu + 1 step 1 until nu do
    begin
      k := ix[i];
      t2 := ip2(j, 0, mu, mat, w[j], vec);
      if t2 < t1 then begin alpha := i; t1 := t2 end
    end
  end findalpha;
  procedure refine(mat, rhs, od, lp, up, vec, fi, si, ord, ill); value
    ord; integer ord, fi, si; real mat, rhs, od, lp, up, vec; label
    ill;
  comment This procedure makes an iterative refinement of
    vec, which is the solution of the matrix equation mat × vec =
    rhs. The matrix mat has order ord. The LU decomposition of
    mat is specified by od, lp, and up. Exit ill is taken if mat is too
    ill-conditioned for the refinement process to be successful.
    Note the global identifier eta, whose value and purpose are
    given in the next comment;
  begin
    array cor[0:ord]; real cnorm, snorm, eps, tt; integer cnt;
    cnt := 0; eps := 5 × eta;
loop:
    cnorm := snorm := 0; cnt := cnt + 1;
    for fi := 0 step 1 until ord do
    begin
      cor[fi] := −ip2(si, 0, ord, mat, vec, −rhs);
      si := fi; tt := abs(vec);
      if tt > snorm then snorm := tt
    end;
  trisolv(0, 1, ord, 0, fi−1, fi, si, cor[si], cor[fi], od, lp);
```

```
    trisolv(ord, −1, 0, fi+1, ord, fi, si, cor[si], cor[fi], od, up);
    for si := 0 step 1 until ord do
    begin
      tt := cor[si];
      vec := vec + tt;
      if abs(tt) > cnorm then cnorm := abs(tt)
    end;
    if cnt > 15 then go to ill;
    if snorm ≠ 0 then
      begin if cnorm/snorm > eps then go to loop end
  end refine;
  comment At this point, infinity and eta are set to special
    values. Set infinity to the largest positive single-precision
    floating-point number. Set eta to the largest positive floating-
    point number such that 1.0 + eta = 1.0 − eta = 1.0 in single-
    precision arithmetic. The convergence of the iterative re-
    finement process which is applied in refine is determined using
    eta;
  prevz := −infinity;
  for i := 0 step 1 until m do ro[i] := i;
  comment Determine from kappa whether phase 1 is to be
    skipped;
  if kappa ≥ m then
  begin
    nu := n + kappa − m−1;  l := 0;
    for i := 0 step 1 until nu do
    begin
      j := ind[i];  if j ≥ n then l := 1;  ix[i] := j
    end;
    mu := if l = 0 then m − 1 else m;
    go to phase 2
  end;
  mu := m − 1;  gamma := m − kappa;  gm1 := gamma − 1;
  nu := n + gm1;  l := n − m;
  comment Set up the appropriate phase 1 problem;
  for i := 0 step 1 until gm1 do
  begin
    ix[i] := n + i;
    P[i, i] := 1.0;
    for j := i + 1 step 1 until gm1 do P[i, j] := P[j, i] := 0;
    for j := gamma + 1 step 1 until mu do P[i, j] := G[i, l+j]
  end;
  for i := gamma step 1 until mu do
  begin
    ix[i] := l + i;
    for j := 0 step 1 until gm1 do P[i,j] := 0
  end;
  for i := m step 1 until nu do ix[i] := i − m;
  beta := gamma;
  go to no removal;
new phase 1 cycle:;
  comment Begin a new simplex step on the phase 1 problem.
    Check the phase 1 problem objective function;
  if ip2(i, 0, mu, w[i], b[i], 0) = 0 then go to phase 2;
  comment Determine which nonbasic variable is to become
    basic;
  findalpha(G[j,k],0);
  if t1 ≥ 0 then go to infeasible;
  j := ix[alpha];
  comment Solve a linear system for a vector y;
  trisolv(gamma, 1, mu, gamma, l − 1, l, k, v[k], G[ro[l],j],
    P[ro[l],k], 1.0);
  trisolv(mu, −1, gamma, l + 1, mu, l, k, y[k], v[l],
    P[ro[l],k], P[ro[l],l]);
  for i := 0 step 1 until gm1 do
  begin
    l := ro[i];
    y[i] := −ip2(k, gamma, mu, y[k], P[l,k], −G[l,j])
  end;
  comment Use the vector y to determine which basic variable
    becomes nonbasic. If the variable which has become non-
    basic is an artificial variable, remove it entirely from the
    problem and make an appropriate row and column inter-
    change upon the basis matrix P;
  findbeta;
  if beta ≥ gamma then
  begin
    k := ix[alpha]; ix[alpha] := ix[beta]; ix[beta] := k;
    go to no removal
  end;
  k := ro[gm1]; i := ro[gm1] := ro[beta]; ro[beta] := k;
  P[k, beta] := 1.0; P[i, beta] := 0;
  ix[beta] := ix[gm1]; ix[gm1] := ix[alpha]; beta := gm1;
  for i := alpha + 1 step 1 until nu do ix[i−1] := ix[i];
  gamma := gm1; gm1 := gm1 − 1; nu := nu − 1;
no removal:;
  comment Produce the LU decomposition of the new basis
    matrix;
  k := ix[beta];
  for i := 0 step 1 until gm1 do P[ro[i],beta] := G[ro[i],k];
  decompose(G[ro[j],l], gamma, gamma);
  comment Find the basic solution h;
  trisolv(gamma, 1, mu, gamma, j − 1, j, k, v[k],
    b[ro[j]], P[ro[j],k], 1.0);
  trisolv(mu, −1, gamma, j + 1, mu, j, k, h[k], v[j],
    P[ro[j],k], P[ro[j],j]);
  for i := 0 step 1 until gm1 do
  begin
    k := ro[i];
    h[i] := −ip2(j, gamma, mu, h[j], P[k,j], −b[k]);
    w[k] := −1.0
  end;
  comment Solve a linear system for the vector, w, of simplex
    multipliers;
  for i := gamma step 1 until mu do
  begin
    t1 := 0;
    for j := 0 step 1 until gm1 do t1 := t1 + P[ro[j],i];
    v[i] := t1
  end;
  trisolv(gamma, 1, mu, gamma, i − 1, i, j, v[j], v[i],
    P[ro[j], i] P[ro[i],i]);
  trisolv (mu, − 1, gamma, i + 1, mu, i, j, w[ro[j]], v[i], P[ro[j], i],
    1.0);
  go to new phase 1 cycle;
phase 2:;
  comment Set up the appropriate phase 2 problem and make
    an initial LU decomposition if necessary;
  beta := 0;
  if kappa < m then
  begin
    if gamma > 0 then
    begin
      kappa := m; nu := nu + 1; mu := m;
      ix[nu] := ix[mu]; ix[mu] := n + m
    end
  end;
  if kappa ≥ m then go to decomp
  else trisolv(0, 1, mu, 0, j − 1, j, k, q[k], if ro[j] = m then 0 else
    b[ro[j]], P[ro[j],k], 1.0);
new phase 2 cycle: ;
  comment Begin a new simplex step on the phase 1 problem.
    Solve a linear system for the vector, w, of simplex multipliers;
  trisolv(0, 1, mu, 0, i − 1, i, j, v[j], dvec(ix[i]), P[ro[j],i], P[ro[i],i]);
  trisolv(mu, −1, 0, i + 1, mu, i, j, w[ro[j]], v[i], P[ro[j],i], 1.0);
```

```
    comment Determine which nonbasic variable is to become
      basic;
    findalpha(Gmat(j,k), -dvec(k));
    comment Check whether the solution has been found;
    if t1 ≥ 0 then go to finished;
  not done yet:
    i := ix[alpha];
    comment Solve a linear system for a vector y;
    trisolv(0, 1, mu, 0, j - 1, j, k, v[k], Gmat(ro[j],i), P[ro[j],k], 1.0);
    trisolv(mu, -1, 0, j + 1, mu, j, k, y[k], v[j], P[ro[j],k], P[ro[j],j]);
    comment Use y to determine which basic variable is to be-
      come nonbasic;
    findbeta;
    if t1 = infinity then go to unbounded;
    k := ix[beta]; ix[beta] := ix[alpha]; ix[alpha] := k;
  decomp: ;
    comment Produce the LU decomposition of the new basis
      matrix;
    decompose(Gmat(ro[j],l), 0, beta);
    comment Compute the basic solution h;
    trisolv(beta, 1, mu, 0, j - 1, j, k, q[k], if ro[j] = m then 0 else
      b[ro[j]], P[ro[j],k], 1.0);
    trisolv(mu, -1, 0, j + 1, mu, j, k, h[k], q[j], P[ro[j],k], P[ro[j],j]);
    go to new phase 2 cycle;
  finished: ;
    comment Refine w and the basic solution h. Compute the
      objective function. Check the refined results to determine
      whether the optimum has been reached. If the check indicates
      nonoptimality but the objective function is less than any
      value previously computed for it, return the best basic solu-
      tion obtained so far and print a warning that the solution
      has doubtful validity;
    refine(Gmat(ro[j],ix[i]), dvec(ix[i]), P[ro[j],i], P[ro[i],i], 1.0,
      w[ro[j]], i, j, mu, singular);
    z := ip2(i, 0, m - 1, w[i], b[i], 0);
    if z < prevz then
      begin comment Print out "doubtful solution"; end
    else
    begin
      prevz := z;
      refine(Gmat(ro[j], ix[k]), if ro[j] = m then 0 else b[ro[j]],
        P[ro[j],k], 1.0, P[ro[j],j], h[k], j, k, mu, singular);
      l := n - 1; kappa := nu + 1;
      for i := 0 step 1 until l do x[i] := 0;
      for i := 0 step 1 until nu do ind[i] := ix[i];
      for i := 0 step 1 until mu do
      begin
        j := ix[i];
        if j < n then x[j] := h[i]
      end;
      findalpha(Gmat(j,k), -dvec(k));
      if t1 < 0 then go to not done yet
    end
  end linprog
```


CERTIFICATION OF ALGORITHM 292 [S22] REGULAR COULOMB WAVE FUNCTIONS [Walter Gautschi, *Comm. ACM 9* (Nov. 1966), 793] AND OF REMARK ON ALGORITHM 292 [S22] REGULAR COULOMB WAVE FUNCTIONS [Walter Gautschi, *Comm. ACM 12* (May 1969), 280]

K. S. KÖLBIG (Recd. 10 Oct. 1967)
Applied Mathematics Group, Data Handling Division, European Organization for Nuclear Research (CERN), 1211 Geneva 23, Switzerland

KEY WORDS AND PHRASES: Coulomb wave functions, wave functions, regular Coulomb wave functions
CR CATEGORIES: 5.12

Both the original and the revised version of the procedure *Coulomb* have been translated into FORTRAN and tested on a Control Data 6600 computer. It became apparent that the following changes in the original version are necessary:

1. The second sentence in the **comment** following the statement labeled *L1* in procedure *Coulomb* should be replaced by:

   Similarly for the letter n in the next statement, which is a place holder for the number of digits carried in the main program.

2. The second statement after this **comment** (beginning "*outstring* . . .") should be changed to

```
   if abs(d1 × epsilon) < 10-m-1 then
   outstring (1, 'The requested accuracy cannot be guaranteed.
   Use of the procedure minimal in a higher precision mode ap-
   pears indicated.');
```

Since the original version of *Coulomb* is to be superseded by the revised one (see Remark), detailed test results are given here only for the latter. Most of the tests have already been described in the Algorithm itself or in the Remark. Those presented here are obtained on a different machine, and the results differ slightly in some cases from the previous ones. The tests included the following:

(i) Generation of $\Phi_L(\eta,\rho) = [C_L(\eta)\rho^{L+1}]^{-1} F_L(\eta,\rho)$, $L = 0(1)21$, to 8 significant digits ($d = 8$) for $\eta = -5(1)5$, $\rho = .2(.2)5$. The results were in complete agreement with the values tabulated in [4] of Algorithm 292. In the cases where more than 8 significant digits are tabulated, the highest discrepancy was one unit in the last digit; e.g. for $L = 0$, $\eta = 5$, $\rho > 3.4$, 10 to 11 correct significant digits have been found.

(ii) Computation of $F_0(\eta,\rho)$, $F_0'(\eta,\rho) = (d/d\rho)\, F_0(\eta,\rho)$ to 5 significant digits for $\eta = 0(2)12$, $\rho = 0(5)40$, using $F'_0 = (\rho^{-1} + \eta)F_0 - (1+\eta^2)^{\frac{1}{2}}\, F_1$. Comparison with [5] of Algorithm 292 revealed frequent discrepancies of one unit in the fifth digit. For $\eta = 2$, $\rho = 40$ the discrepancy in F_0 is 80 units of the fifth digit. This is probably an error in the table.

(iii) Computation to 8 significant digits of $F_0(\eta,\rho)$, $F_0'(\eta,\rho)$ for $\rho = 2\eta$, $\rho = .5(.5)20(2)50$. The results agreed completely with those published in [1] of Algorithm 292.

(iv) Computation (with $d = 10$) of the miscellaneous values of $F_0(\eta,\rho)$ and $\Phi_0(\eta,\rho)$ given in the Remark on Algorithm 292. The results obtained differ slightly from those given in the Remark. In the worst case, $\eta = 50$, $\rho = 120$, the discrepancy is 16 units in the tenth digit.

(v) After changing the dimensions of the arrays *lambda*, *lmin* into [0:600] and adjusting the upper limit for *nu* to 600 (see Remark on Algorithm 292), $F_L(\eta,\rho)$ has been calculated with $d = 6$ for $\eta = -200(20)\ 200$, $\rho = 20(20)\ 200$, $Lmax = 0(50)100$ merely to test whether overflow occurs or not. The following table indi-

cates where overflow, indefinite results, or convergence difficulties in the generation of λ_L (see Algorithm 292) have been observed.

η	$\rho \geq$
20	200
40	200
60	180
80	100
100	80
120	60
140	60
160	60
180	40
200	40

(vi) Calculation of $F_L(\eta,\rho)$ for $L = 0(50)100$ with $d = 7$ for $\eta = 1$, $\rho = 10^{-n}$, $n = -20(1)-1$. Underflow occurred for $L = 50$, $n \leq 5$; $L = 100$, $n \leq 2$. The valid results have been compared with those obtained by summation of the power series for $\Phi_L(\eta,\rho)$ (see [4, (1.3) and (4.4)] of Algorithm 292). Agreement has been found to 7 significant digits.

(vii) Calculation of $\Phi_L(\eta,\rho)$ to 13 significant digits ($d=13$) for $\rho = 5$, $\eta = 0(1)5$, $L = 0(10)100$. The results have been compared with those obtained by summation in double-precision mode (27 digits) of the power series mentioned in (vi). Agreement was found to at least 12 significant digits. The constant 2π in the statement *t1* := . . . on page 795 of Algorithm 292 was supplied here with 14 significant digits, as required by the **comment.**

Acknowledgment. I wish to thank Professor Gautschi for useful remarks and comments.

CERTIFICATION OF ALGORITHM 300 [S22] COULOMB WAVE FUNCTIONS [J. H. Gunn, *Comm. ACM 10* (Apr. 1967), 244]

K. S. Kölbig (Recd. 8 Feb. 1968)
Applied Mathematics Group, Data Handling Division, European Organization for Nuclear Research (CERN), 1211 Geneva 23, Switzerland

KEY WORDS AND PHRASES: Coulomb wave functions, wave functions
CR CATEGORIES: 5.12

The procedure *Coulomb* was checked for a few parameter values using the Algol compiler of the CDC 3800 computer at CERN. It was found that for $\rho = \eta$ better results were obtained if the first line of the second **if** statement was altered to read:

if *rho* $\leq$ (5 $\times$ *eta* $-$ 15)/3 $\vee$ *rho* $<$ *eta* **then**

It was also necessary to correct a misprint in the first constant following the **comment** "*G*[0] and *Gd*[0] are calculated on the transition line for *rhom* = 2 $\times$ *eta*, ref. formulas 10.3–10.4, Fröberg." The line following this **comment** should read:

G[0] := 1.223404016 $\times$ *eta* $\uparrow$ (1/6) $\times$ (1 + 0.0495957017/*eta* $\uparrow$ (4/3)

The procedure was then translated into Fortran and tested in more detail on a CDC 6600 computer. The tests included the following:

(i) Generation of $\Phi_L(\eta,\rho) = [C_L(\eta)\rho^{L+1}]^{-1} F_L(\eta,\rho)$, $L = 0(1)21$ for $\eta = 1(1)5$, $\rho = 5$. The results were compared with values tabulated in [1]. In most cases, 6 to 7 significant digits agreed, except for $\eta = 1$, where agreement was found to 3 to 4 significant digits. It is interesting to compare some results for $\rho = \eta = 5$ obtained with and without the first of the above corrections:

$L \backslash \Phi_L$	Without correction	With correction	Table [1] and Gautschi [2]
0	$6.554097_{10}3$	$6.552297_{10}3$	$6.552292_{10}3$
5	$1.865738_{10}1$	$1.865226_{10}1$	$1.865225_{10}1$
10	$5.354953_{10}0$	$5.353482_{10}0$	$5.353478_{10}0$
20	$2.440859_{10}0$	$2.440188_{10}0$	$2.440187_{10}0$

(ii) Computation of $F_0(\eta,\rho)$, $F_0'(\eta,\rho) = (d/d\rho)F_0(\eta,\rho)$ for $\eta = 2(2)12$, $\rho = 5(5)30$. Comparison with the table of Tubis [3] revealed frequent discrepancies of 1 (occasionally 2) units of the fifth significant digit. However, disagreement was observed in many fewer cases when comparing the calculated results with those obtained by Gautschi's algorithm [2].

(iii) Computation of $F_0(\eta,\rho)$, $F_0'(\eta,\rho)$, $G_0(\eta,\rho)$, and $G_0'(\eta,\rho)$ for $\rho = 2\eta$, $\rho = 5(.5)20(2)30$. Comparing the results with the table of Abramowitz and Rabinowitz [4] or with the values obtained with Gautschi's algorithm, the following discrepancies were found in units of the seventh decimal place:

F_0 —frequently 1, occasionally 2, units for $\rho \leq 10$;
F_0'—frequently 1 unit for $\rho \leq 8.5$;
G_0 —for $\rho \leq 8$ up to 40 units, for $8 < \rho \leq 14.5$ up to 2 or 3 units;
G_0'—for $\rho \leq 7.5$ up to 13 units.

(iv) Calculation of $G_0(\eta,\rho)$, $G_0'(\eta,\rho)$ for $\eta = .5(.5)20$, $\rho = 5(1)20$. The results have been compared with the tables given by Abramowitz [5]. Agreement was found in most cases to 5 significant digits. Discrepancies of 1, occasionally more, units of the fifth significant digit were found, mainly for arguments near a line separating two methods used in the algorithm. In some cases (in the immediate neighborhood of a zero of G_0 or G_0') there was agreement to only 2 or 3 significant digits.

(v) Generation of $F_L(\eta,\rho)$, $F_L'(\eta,\rho)$, $G_L(\eta,\rho)$, $G_L'(\eta,\rho)$, $\sigma_L(\eta)$ for $L = 0(1)10$, $\rho = 5,10$, $\eta = 1(1)5,10,25$. As a first step, the results were compared with values given in a table by Lutz and Karvelis [6]. Since important discrepancies were noted for $\eta = 1$, $\rho = 5$ and $\eta \geq 4$, the values for F_L and F_L' were also calculated by Gautschi's algorithm, known to be correct by checking it against the table [1]. Lutz and Karvelis give 6 significant digits, but without commenting on a possible error tolerance. They state, "we test [the generated functions] to see how closely the Wronskian relation $F_L'G_L - F_LG_L' = 1$ is obeyed." Comparison of their values with those obtained from Gautschi's algorithm shows, for $\eta < 4$, occasional discrepancies of 1 unit in the sixth significant digit. For $\eta \geq 4$ [disregarding some obvious misprints, e.g. for $G_1(2,10)$ and $G_{10}'(10,10)$] there are discrepancies which in a few cases exceed a 100 units in the sixth significant digit. Because of this, the table of Lutz and Karvelis was used for checking the procedure *Coulomb* only for $\eta < 4$. For $\eta \geq 4$ check values were obtained from Gautschi's algorithm (F_L and F_L' only). The following discrepancies were found in units of the sixth significant digit:

$\eta = 1$, $\rho = 5$: F_L—up to 119 units ($L = 8$).
F_L'—up to 87 units ($L = 0$).
G_L—up to 350 units ($L = 2$).
G_L'—up to 247 units ($L = 0$).

$\eta = 1$, $\rho = 10$;
$\eta = 2,3$: 1 or 2 units in several cases, exceptionally more; one isolated case $G_3(3,10)$ with 23 units. Comparison with Gautschi's values (where possible) gives better agreement.

$\eta \geq 4$: Occasionally 1 unit for F_L and F_L'.

$\sigma_L(\eta)$ nearly always agreed to 6 significant digits for all tested η. To complete the check, values of the functions at $\eta = 1$, $\rho = 5$,

and $\eta = \rho = 5$ were calculated using the ALGOL procedure. The results agreed with those calculated by the FORTRAN program to the 6 significant digits which were compared.

REFERENCES:

1. NATIONAL BUREAU OF STANDARDS. *Tables of Coulomb Wave Functions, Vol. I.* Appl. Math. Ser. 17, U.S. Govt. Printing Office, Washington, D.C., 1952.
2. GAUTSCHI, W. Algorithm 292. Regular Coulomb wave functions. *Comm. ACM 9* (Nov. 1966), 793–795.
3. TUBIS, A. *Tables of Nonrelativistic Coulomb Wave Functions.* LA-2150, Los Alamos Sci. Lab., Los Alamos, New Mexico, 1958.
4. ABRAMOWITZ, M., AND RABINOWITZ, P. Evaluation of Coulomb wave functions along the transition line. *Phys. Rev. 96* (1954), 77–79.
5. ——, AND STEGUN, I. A. (Eds.) *Handbook of Mathematical Functions.* NBS Appl. Math. Ser. 55, U.S. Govt. Printing Office, Washington, D.C., 1965.
6. LUTZ, H. F., AND KARVELIS, M.D. Numerical calculation of Coulomb wave functions for repulsive Coulomb fields. *Nucl. Phys. 43* (1963), 31–44.

REMARK ON ALGORITHM 292 [S22] REGULAR COULOMB WAVE FUNCTIONS [Walter Gautschi, *Comm. ACM 9* (Nov. 1966), 793]

WALTER GAUTSCHI (Recd. 5 July 1967)

Computer Sciences Department, Purdue University, Lafayette, Indiana, and Argonne National Laboratory, Argonne, Illinois

* This work was performed under the auspices of the United States Atomic Energy Commission.

KEY WORDS AND PHRASES: Coulomb wave functions, wave functions, regular Coulomb wave functions

CR CATEGORIES: 5.12

The following changes are suggested to eliminate the need for multiple-precision arithmetic. The underlying theory will be published in *Aequationes Math.*

1. Remove the procedure *minimal.*
2. Change the statement (near the bottom of page 794)

   nu := **if** $s \geq -.36788$ **then** *entier* $(r \times t(s))$ **else** 1

   to read:

   nu := **if** $s \geq -.36788$ **then** *entier* $(r \times t(s))$ **else** $r/2.7183$
3. Change the statement labeled *L*1 to read

   *L*1: *d*1 := 2 × *eta*/(*exp*(2×*eta*×*arctan*(1/*omega*))−1)

   and rephrase the comment following this statement to read:

   comment The letter *n* in the following statement is a place holder for a machine-dependent integer, namely, the number of (equivalent) decimal digits carried in the mantissa of floating-point numbers. This integer must be properly substituted by the user;
4. Omit the output statement

   outstring (1, 'The requested accuracy cannot be guaranteed. Use of the procedure *minimal* in a higher precision mode appears indicated');
5. Insert the statement

   *r*1 := *lmin*[1];

   between the two lines

   end;

   and

   lam[0] := −*r*1; *lam*[1] := 1; *t*1 := *d*1/(1+*r*1↑2);
6. Change the line (near the middle of page 795)

   s := *sqrt*(*t*1/(*exp*(*t*1)−1));

   to read

   s := *exp*(−*t*1/4)/*sqrt*((*exp*(*t*1/2) − *exp*(−*t*1/2))/*t*1);

   (These statements are mathematically equivalent, but the latter delays overflow as the value of *t*1 becomes large.)
7. If large values of $|\eta|$ and/or ρ, say exceeding 100, are contemplated, it may be necessary to increase the dimension of the arrays *lambda* and *lmin* (if they are declared at the beginning of the procedure *Coulomb*) and to correspondingly increase the upper limit for *nu* in the conditional clause

   if *nu* < 300

   near the top of page 795. The user, in this case, should also be prepared to encounter overflow difficulties, especially in the later entries of the array *lam.*

With these revisions the algorithm produced correct results on the CDC 3600 for the three tests described at the end of Algorithm 292. It was also used (with input parameter $d = 10$) to compute miscellaneous values of $F_0(\eta, \rho)$ and $\Phi_0(\eta, \rho)$ published in a paper by C. E. Fröberg (Numerical treatment of Coulomb wave functions. *Rev. Mod. Phys. 27* (1955), 399–411). The results are summarized in the table below.

η	ρ	Algorithm 292 (revised)	Fröberg
9	50	$F_0 = 9.357085680_{10} - 1$	$9.3570855_{10} - 1$
50	80	$F_0 = 1.203662491_{10} - 3$	$1.203665_{10} - 3$
50	120	$F_0 = 2.002599349_{10} - 1$	$2.00255_{10} - 1$
100	4	$\Phi_0 = 5.722985154_{10}21$	$5.722985155_{10}21$
200	1	$\Phi_0 = 7.236604732_{10}14$	$7.236604731_{10}14$

In addition, the algorithm was run (with $d = 6$, and *lambda*, *lmin* being declared as arrays of dimension [0 : 600]) for $\eta = -200(20)200$, $\rho = 20(20)200$, $Lmax = 0(50)100$. Apparently valid results were obtained as long as $\eta \leq 100$, though no tables seem to exist to check these results against. Overflow was observed in some of the entries of the array *lam*, for $\eta = 120$, $\rho \geq 120$; $\eta = 140$, $\rho \geq 60$; $\eta = 160$, $\rho \geq 40$; and $\eta = 200$, $\rho \geq 20$. (For the purpose of this test, a number is considered to overflow if its modulus exceeds $_{10}300$.)

REMARK ON ALGORITHM 331 GAUSSIAN QUADRATURE FORMULAS [D1] [Walter Gautschi, *Comm. ACM 11* (June 1968), 432]

I. D. HILL (Recd. 12 Sept. 1968)

Medical Research Council, Computer Unit (London), London, N.1, England

KEY WORDS AND PHRASES: quadrature, Gaussian quadrature, numerical integration, weight function, orthogonal polynomials

CR CATEGORIES: 5.16

1. On pages 434 and 435 there are five strings, all of which have identical opening and closing string quotes.' and' should be replaced by 'and' in each case.

2. No space symbols appear in these strings. ␣ should be inserted in each space. Otherwise, no spaces will appear in the printed messages.

3. In the second string, the hyphen in the word "violated" should be deleted.

4. In the first column of page 433 there appear:

kmax := *entier*(*capn*/2);

and

if *capn*/2 ≠ *kmax* **then**

Both these are critically dependent upon rounding error in the real division. Presumably,

kmax := *capn* ÷ 2;

and

if *capn* ≠ 2 × *kmax* **then**

are intended.

5. A semicolon is necessary before the final **end** (on page 436). As things stand, this **end** is part of the comment, and the algorithm never finishes.

Alternatively, the semicolon after **end** *Gauss*, two columns earlier, could be deleted (in which case the symbol **comment** could also be deleted if desired, but need not be). If this were done, the final **end** would terminate the comment without the need for a preceding semicolon.

REMARK ON ALGORITHM 334 [G5]
NORMAL RANDOM DEVIATES [James R. Bell, *Comm. ACM 11* (July 1968), 498]

R. KNOP* (Recd. 5 Aug. 1968 and 8 Nov. 1968)
Physics Dept., University of Maryland, College Park, MD 20742

This work was supported in part by an Atomic Energy Commission contract.

* Present address: Physics Dept., Rutgers University, New Brunswick, NJ 08903

KEY WORDS AND PHRASES: normal deviates, normal distribution, random number, random number generator, simulation, probability distribution, frequency distribution, random
CR CATEGORIES: 5.13, 5.5

Algorithm 334 produces pairs of normally distributed random deviates with zero mean and unit variance by the method of Box and Muller [1]. The sine and cosine required by the Box-Muller method are calculated by the von Neumann rejection technique [2]. This technique allows the calculation of the sine and cosine of an angle uniformly distributed over the interval $(0, 2\pi)$ without referencing the sine, cosine, or square root functions. We note however, that Algorithm 334 require as square root calculation in inverting the distribution function of the radius (equal to $L \times S$ in the notation of the algorithm).

We suggest that since the square root calculation seems unavoidable, it can be used to obtain the required sine and cosine by more conventional means. Thus we propose sampling points from a density uniform over the unit disk in the X, Y-plane and calculating the sine and cosine from their definition in terms of the legs and hypotenuse of a right triangle. The following changes in Algorithm 334 are then necessary:

a. Replace $X := R$ by $X := 2 \times R - 1$

b. Replace $L := sqrt(-2 \times ln(R))/S$ by
$L := sqrt(-2 \times ln(R)/S)$

c. Replace $D1 := (XX - YY) \times L$ by $D1 := X \times L$

d. Replace $D2 := 2 \times X \times Y \times L$ by $D2 := Y \times L$

Acknowledgment. The author thanks B. Kehoe for comments concerning this algorithm.

REFERENCES:

1. BOX, G., AND MULLER, M. A note on the generation of normal deviates. *Ann. Math. Stat. 28* (1958), 610.

2. VON NEUMANN, J. Various techniques used in connection with random digits. In Nat. Bur. Standards Appl. Math. Ser. 12, US Govt. Printing Off., Washington, D. C., 1959, p. 36.

REMARK ON ALGORITHM 340 [C2]
ROOTS OF POLYNOMIALS BY A ROOT-SQUARING AND RESULTANT ROUTINE [Albert Noltemeier, *Comm. ACM 11* (Nov. 1968), 779]

ALBERT NOLTEMEIER (Recd. 6 Jan. 1969)
Technische Universität Hannover, Rechenzentrum, Hannover, Germany

KEY WORDS AND PHRASES: rootfinders, roots of polynomial equations, polynomial zeros, root-squaring operations, Graeffe method, resultant procedure, subresultant procedure, testing of roots, acceptance criteria
CR CATEGORIES: 5.15

The following misprints were found in the algorithm and should be corrected as indicated:

1. In the comment, in the first column on page 780, the last line before the paragraph beginning with the word "Parameters" ends with a semicolon; it should end with a period.

2. In the seventh line following the word "Parameters" the abbreviation CDC should appear in capital letters.

3. In the procedure body, in the second column on page 780, the line before the label *SQUARING OPERATION* is missing. It should read as follows:

for *j* := 0 **step** 1 **until** *d* **do** *a*[*j*, 0] := *c*[*j*];

Corrigenda

INDEX BY SUBJECT TO ALGORITHMS 1960–1968

In this index [*Comm. ACM, 11* (Dec. 1968), 827–830], 10 lines were omitted from the classification *G6*. The complete classification *G6* should appear as follows:

```
G6              PERMUTATIONS AND COMBINATIONS
G6    71 PERMUTATIONS                          11-61(497),4-62(209),
G6    71  8-62(439)
G6    85 PERMUTATIONS                          4-62(208),4-62(209),
G6    86  8-62(440)
G6    87 PERMUTATION GENERATOR                 4-62(209),8-62(440),
G6    87  10-62(514),7-67(452)
G6    94 COMBINATIONS                          6-62(344),11-62(557),
G6    94  12-62(606)
G6   102 PERMUTATIONS IN LEXIC. ORDEP          6-62(346),10-62(514),
G6   102  7-67(452)
G6   115 PERMUTATIONS                          8-62(434),10-62(514),
G6   115  12-62(606)
G6   130 PERMUTE                              11-62(551),7-67(452)
G6   152 COMBINATIONS                         2-63(68),7-63(385)
G6   154 COMBINATION IN LEXIC. ORDER          3-63(103),8-63(449)
G6   155 COMBINATION IN ANY ORDER             3-63(103),8-63(449)
G6   156 ALGEBRA OF SETS                      3-63(103),8-63(450)
G6   160 COMB. OF M THINGS N AT A TIME        4-63(161),8-63(450),
G6   160  10-63(618)
G6   161 COMBS. 1,2,UP TO N AT A TIME         4-63(161),8-63(450),
G6   161  10-63(619)
G6   202 PERMUTATIONS IN LEXIC. ORDER         9-63(517),9-65(556),
G6   202  7-67(452)
G6   235 RANDOM PERMUTATION                   7-64(420),7-65(445)
G6   242 PERMUTATIONS WITH REPETITIONS        10-64(585)
G6   250 INVERSE PERMUTATION                  2-65(104),11-65(670)
G6   306 PERMUTATIONS WITH REPETITIONS        7-67(450)
G6   308 PERMUT. IN PSEUDOLEXIC.ORDER        7-67(452)
G6   317 PERMUTATION                          11-67(729)
G6   323 PERMUTATIONS IN LEXIC ORDER          2-68(117)
G6   329 DISTR OF INDISTINGUISHABLE OBJ       6-68(430)
G6       ALL PERMUTATIONS OF N OBJECTS        COMP.BULL.V9(104)
G6       PERMUTNS OF VECTOR-LEXIC ORDER       COMP.J.V10(311)
G6       PERMUTN OF VECTOR                    COMP.J.V10(311)
G6       FAST PERMUTN OF VECTOR               COMP.J.V10(311)
```

In addition in Classification H, Algorithm 332 should be Algorithm 333 and the line should read

H 333 MINIT ALGORITHM FOR LIN PROG 6-68(437)

Thanks are due to Louis C. Semprebon, Dartmouth College, for calling attention to these errors.