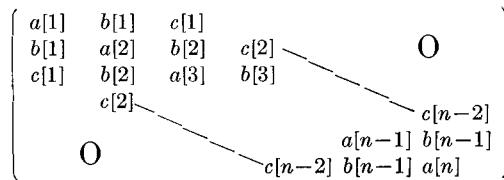ALGORITHM 104
REDUCTION TO JACOBI
H. RUTISHAUSER
Eidg. Technische Hochschule, Zurich, Switzerland

procedure *m21* (n, a, b, c, *inform*); **value** n;
  integer n; **array** a, b, c; **procedure** *inform*;
  comment: *m21* transforms symmetric bandmatrix

$$\begin{pmatrix} a[1] & b[1] & c[1] & & & & \\ b[1] & a[2] & b[2] & c[2] & & O & \\ c[1] & b[2] & a[3] & b[3] & & & \\ & c[2] & & & & & c[n-2] \\ & & & & & a[n-1] & b[n-1] \\ & O & & & c[n-2] & b[n-1] & a[n] \end{pmatrix}$$

represented by the arrays a, b, c by orthogonal transformation
to Jacobi form which is represented by the arrays a, b. The
method is described in H. RUTISHAUSER, "On Jacobi rotation
patterns," to appear in Proc. Symposium in Experimental
Arithmetic, Chicago, Apr. 12-14, 1962, Sect. 5. Note that decla-
rations must be given for the arrays a, b, c with subscripts
ranging from 1 to n. Also procedure *inform* must be declared.
It may serve to use the Jacobi rotations occurring inside *m21*
also for other purposes;
begin
  real p, g, d, s;
  integer k, j;
  b[n] := c[n] := c[n−1]: = 0;
  for k := 2 **step** 1 **until** n−1 **do**
  begin
    for j := k **step** 2 **until** n−1 **do**
    begin
      if k=j **then**
      begin
        p := sqrt(b[k−1]↑2 + c[k−1]↑2);
        if p=0 **then go to** *ex*;
        d := b[k−1]/p;
        s := −c[k−1]/p;
        b[k−1] := p;
        c[k−1] := 0
      end k=j
      else
      begin
        p := sqrt(c[j−2]↑2+g↑2);
        if p = 0 **then go to** *ex*;
        d := c[j−2]/p;
        s := −g/p;
        c[j−2] := p;
        p := d×b[j−1]−s×c[j−1];
        c[j−1] := s×b[j−1]+d×c[j−1];
        b[j−1] := p
      end j≠k;
  common: g := 2×b[j]×d×s;
    p := a[j]×d×d−g+a[j+1]×s×s;
    b[j] := (a[j]−a[j+1])×d×s+b[j]×(d×d−s×s);
    a[j+1] := a[j]×s×s+g+a[j+1]×d×d;
    a[j] := p;
    p := d×c[j]−s×b[j+1];
    b[j+1] := s×c[j]+d×b[j+1];
    c[j] := p;
    g := −s×c[j+1];
    c[j+1] := d×c[j+1];
    inform (n, j, d, s);
    comment: The Jacobi rotation which has been performed

in this turn of the j-loop is $A := U^T A U$ with

$$U = \begin{pmatrix} 1 & & & & & \\ & 1 & & & O & \\ & & d & s & & \\ & & −s & d & & \\ & O & & & 1 & \\ & & & & & 1 \end{pmatrix},$$

where the d's and s's are located at the crosspoints of
rows and columns j and j +1;
  end j;
ex: **end** k
end *m21*

ALGORITHM 105
NEWTON MAEHLY
F. L. BAUER AND J. STOER
Johannes Gutenberg-Universität, Mainz, Germany

procedure Newton Maehly (a, n, z, eps);
  value n, eps;
  array a, z;
  integer n;
  real eps;
  comment The procedure determines all zeros z[1:n] of the
    polynomial $p(x) := a[0] \times x{\uparrow}n + \cdots + a[n]$ of order n, if $p(x)$
    has only real zeros which have to be all different. The zeros
    $z[i]$ are ordered according to their magnitude: $z[1]>z[2]>$
    $\cdots>z[n]$. The approximations for each zero will be improved
    by iteration as long as abs(x1−x0)>eps × abs(x1) holds
    for two successive approximations x0 and x1;
begin **real** aa, pp, qq, x0, x1;
  integer i, m, s;
  array b, p, q[0:n−1];
  procedure Horner(p, q, n, x, pp, qq);
    value n, x;
    array p, q;
    real pp, x, qq;
    integer n;
  begin real s, s1;
    integer i;
    s := s1 := 0;
    for i := 0 **step** 1 **until** n−1 **do**
    begin s := s×x+p[i]; s1 := s1×x+q[i]; **end** ;
    pp := s×x+p[n]; qq := s1;
    end;
  p[0] := aa := a[0]; x0 := pp := 0; s := sign(a[0]);
  for i := 1 **step** 1 **until** n **do**
    if s × a[i]<0 **then**
      begin if pp=0 **then** pp := i;
            if x0<abs(a[i]) **then** x0 := abs(a[i]);
      end;
  x0 := **if** pp=0 **then** 0 **else** 1+exp(ln(abs(xo/aa))/pp);
  comment x0 is a first approximation for the largest zero which
    may be printed out at this point of the program;
  for i := 0 **step** 1 **until** n−1 **do** b[i] := (n−1)×a[i];

```
for m := 1 step 1 uhtil n do
  begin
    iteration:
    Horner (a, b, n, x0, pp, qq);  x1 := x0−pp/qq;
    if abs(x1−x0)>eps×abs(x1) then
      begin x0 := x1;
        comment  x0 is the last approximation for the zero
          being improved, which may be printed out at this
          point;
        go to iteration;
      end;
    z[m] := x1;
    comment  z[m] := x1 is the mth zero of the polynomial;
    pp := b[0] := b[0] − aa;  q[0] := pp;
    if m<n then
      begin for i := 1 step 1 until n−1 do
            begin pp := p[i] := x1×p[i−1]+a(i);
                  pp := b[i] := b(i)−pp;
                  q[i] := x1×q[i−1]+pp;
            end;
            Horner (p, q, n−1, x1, pp, qq);
            x0 := x1−pp/qq;
            comment  x0 is a first approximation for the
              next zero;
      end
  end
end Newton Maehly;
```

ALGORITHM 106
COMPLEX NUMBER TO A REAL POWER
MARGARET L. JOHNSON AND WARD SANGREN

Computer Applications, Inc., San Diego, California

```
procedure  POWC (x, y, w, A, B);  value x, y, w;
  real x, y, w, A, B;
comment  This procedure takes a complex number (x+iy) to
  a real power w. The result is A+iB=(x+iy)^w. This procedure
  must be used with caution because although it is formally cor-
  rect, it may not give the desired results. For example, if w is a
  reciprocal integer it does not follow that the desired power
  (a root) will be calculated;
begin real  THETA, PHI, R;
  if x>0 then begin THETA := 0.0;  go to SOL 1 end;
  if x<0∧y≧0 then begin THETA := 3.1415927;
    go to SOL 1 end;
  if x<0∧y<0 then begin THETA := 3.1415927;
    go to SOL 1 end;
  if x=0∧y=0 then begin A := B := 0.0;  go to RETURN end:
  if x=0∧y<0 then begin PHI := 1.5707963;  go to SOL 2 end;
  if x=0∧y>0 then begin PHI := −1.5707963;
    go to SOL 2 end;
  SOL 1:  PHI := arctan (y/x)+THETA;
  SOL 2:  R := sqrt (x×x+y×y);
          R := exp (w×ln(R));
          A := R×cos (w×PHI);
          B := R×sin (w×PHI);
  RETURN:  end POWC
```

ALGORITHM 107
GAUSS'S METHOD
JAY W. COUNTS

University of Missouri, Columbia, Mo.

```
    procedure gauss (u, a, y);
    real array a, y;  real temp;  integer u;
```

```
comment  This procedure is for solving a system of
  linear equations by successive elimination of the un-
  knowns. The augmented matrix is a and u the number of
  unknowns. The solution vector is y. If the system hasn't
  any solution or many solutions, this is indicated by go
  to stop;
begin
  integer i, j, k, m, n;
  n := 0;
ck0:  n := n+1;
  for k := n step 1 until u do if a[k, n]≠0 then go to ck1;
  go to stop;
ck1:  if k=n then go to ck2;
  for m := n step 1 until u+1 do
  begin
    temp := a[n, m];  a[n, m] := a[k, m];  a[k, m] := temp
  end;
ck2:  for j := u+1 step−1 until n do a[n, j] := a[n, j]/a[n, n];
  for i := k+1 step 1 until u do
  for j := n+1 step 1 until u+1 do
  a[i, j] := a[i, j]−a[i, n]×a[n, j];
  if n≠u then go to ck0;
  for i := u step−1 until 1 do
  begin
    y[i] := a[i, u+1]/a[i, i];
    for k := i−1 step−1 until 1 do
    a[k, u+1] := a[k, u+1]−a[k, i]×y[i]
  end end;
```

ALGORITHM 108
DEFINITE EXPONENTIAL INTEGRALS A
YURI A. KRUGLYAK

Kharkov State University, Kharkov, U.S.S.R., AND

DONALD R. WHITMAN

Case Institute of Technology, Cleveland, Ohio

```
real procedure As (n, b);  value n, b;  integer n;  real b;
comment:  This procedure computes a value of integral
  A_{n−1}(1, b)=∫_1^∞ x^{n−1}exp(−bx) dx for any given positive integer, n,
  and any positive real parameter, b, by the recursion formula
  A_n(1, b)=A_0(1, b)+(n/b)A_{n−1}(1, b) with A_0(1, b)=exp(−b)/b;
begin  integer m;  real db;  real array a[1:n];
        a[1] := exp (−b)/b;
        if n=1 then go to exit;
comment  integral a[1]=A_0(1, b) was evaluated;
        db := 1/b;  for m := 2 step 1 until n do a[m] :=
        a[1]+db×(m−1)×a[m−1];
comment  integral a[n]=A_{n−1}(1, b) was evaluated;
        As := a[n] end As;
```

ALGORITHM 109
DEFINITE EXPONENTIAL INTEGRALS B
YURI A. KRUGLYAK

Kharkov State University, Kharkov, U.S.S.R., AND

DONALD R. WHITMAN

Case Institute of Technology, Cleveland, Ohio

```
real procedure  Bs(n, a);  value n, a;  integer n;  real a;
comment  This procedure computes a value of the integral
  B_{n−1}(a)=∫_{−1}^{+1} x^{n−1} exp(−ax)dx for any given positive integer, n,
  and any real parameter, a. If |a|<alim an expansion of
  exp(−ax) is used, otherwise the recursion formula B_n(a)=
  [(−1)^n e^a−e^{−a}+nB_{n−1}(a)]/a with b_0(a)=2 sinh(a)/a is used. The
  value of alim depends upon the highest n appearing in the
```

calculations and upon the maximum errors in the last significant digits in the library procedures. For example, we have used alim=8 for $n_{max}=16$ with gamma=1×10^{-8}. The intrinsic function mod(E_1, E_2) which requires two integer arguments, is the conventional modulus;

```
begin      integer m;  real alim, delta, gamma, r, epsilon,
               s, k, a2, omega, da, jp, jm, q1, q2;  real array
               b[1:n];  if a=0 then
L1:        begin if mod(n−1, 2)=0 then
L2:        begin b[n] := 2/n;  go to exit end L2;
comment    integral b[n] = Bₙ₋₁(0) for odd n was evaluated;
           b[n] := 0;  go to exit end L1;
comment    integral b[n] = Bₙ₋₁(0) for even n was evaluated;
           if abs(a) alim then
L3:        begin delta := gamma;  if mod(n−1, 2)=0 then
L4:        begin r := 2/n;  epsilon := r×delta;  s := r;
               k := 0;  a2 := a↑2;
Even:      k := k+2;
           r := r×a2×(n+k−2)/(k×(k−1)×(n+k));
           s := s+r;  if r>epsilon then go to Even;
           b[n] := s+r;
           go to exit end L4;
comment    integral b[n]=Bₙ₋₁(a) for odd n and | a |<alim was
           evaluated;
           r := 2×a/(n+1);  omega := abs(r×delta);
           s := r;  k := 1;
           a2 := a↑2;
Odd:       k := k+2;
           r := r×a2×(n+k−2)/(k×(k−1)×(n+k));
           s := s+r;  if abs(r)>omega then go to Odd;
           b[n] := −(s+r);  go to exit end L3;
comment    integral b[n]=Bₙ₋₁(a) for even n and | a |<alim was
           evaluated;
           da := 1/a;  jp := da×exp(a);  jm := (da↑2)/jp;
           b[1] := jp−jm;
           if n=1 then go to exit;
comment    integral b[1] = B₀(a) for | a |≧alim was evaluated;
           q1 := −1;  q2 := 1;  for m := 2 step 1 until n do
L5:        begin b[m] := q1×jp−jm+q2×da×b[m−1];
           q1 := −q1;  q2 := q2+1 end L5;
comment    integral b[n]=Bₙ₋₁(a) for integer n≧2 and | a |≧alim
           was evaluated;
exit:      Bs := b[n] end Bs;
```

ALGORITHM 110
QUANTUM MECHANICAL INTEGRALS OF
   SLATER-TYPE ORBITALS
YURI A. KRUGLYAK
Kharkov State University, Kharkov, U.S.S.R., AND
DONALD R. WHITMAN
Case Institute of Technology, Cleveland, Ohio

real procedure INTSOLI (n, r, za, ab, As, Bs) Result:
   (s, i1, i2, i3); value n, r, za, zb; integer n; real r, za, zb;
   real array a[1:8], b[1:8], G[1:2×n]; integer array bc[1:2×n,
   1:2×n]; real procedure As, Bs;
comment   Procedure INTSOLI computes the quantum mechan-
   ical integrals $s=\langle\psi^{ai}_{n00} \mid \psi^{bi}_{210}\rangle$ (overlap integral),
           $i1=\langle\psi^{ai}_{n00} \mid Z_a*/r_{ai} \mid \psi^{bi}_{210}\rangle$ (exchange integral),
           $i2=\langle\psi^{ai}_{n00} \mid Z_b*/r_{bi} \mid \psi^{ai}_{n00}\rangle$ (coulomb integral),
   and $i3=\langle\psi^{bi}_{210} \mid Z_a*/r_{ai} \mid \psi^{bi}_{210}\rangle$ (coulomb integral).
Here $\mid\psi^{ai}_{nlm}\rangle$ is a Slater-type orbital of electron i centered on
atomic nucleus a. The integer n is the effective principal quan-
tum number with values 1, 2, 3 and 4. $Z_a*=za$ and $Z_b*=zb$
are effective nuclear charges. r_{bi} is the distance of electron

i from nucleus b. The input parameter r is the distance be-
tween the two centers a and b. All physical quantities are
given in atomic units;

```
begin      integer q, t, c, m;
           real g, zsa, zsb, ks, p, pt, lilya, s, k1, exc, i1, pppt,
           k2, sue, i2, pmpt, ptmp, k3, i3;
           bc[1, 1] := bc[2, 1] := bc[2, 2] := 1;
           for q := 3 step 1 until 2×n do
L6:        begin bc[q, 1] := 1;  for t := 2 step 1 until q−1 do
           bc[q, t] := bc[q−1, t−1]+bc[q−1, t];
           bc[q, q] := 1 end L6;
```

comment   binomial coefficients $bc[q, t]=\binom{q-1}{t-1}$ were computed
           using the recursion formula $\binom{q}{t}=\binom{q-1}{t-1}+\binom{q-1}{t}$;

procedure As(n, b) Result: (a[n]); value n, b; integer n;
           real b;
comment   procedure As computes a value of integral $A_{n-1}(1, b)$
           [see Algorithm 108, "Definite Exponential Inte-
           grals A," by Yuri A. Kruglyak and D. R. Whitman,
           Comm. ACM (July 1962)]. Any identifier occurring
           within the As is specified to be local to the As;
begin   integer m; real db; a[1] := exp(−b)/b;
           if n=1 then go to exitAs; db := 1/b;
           for m := 2 step 1 until n do a[m] := a[1] +
           db×(m−1)×a[m−1]
exitAs: end As;
procedure Bs(n, a) Result: (b[n]); value n, a; integer n;
           real a;
comment   procedure Bs computes a value of integral $B_{n-1}(a)$
           [see Algorithm 109, "Definite Exponential Inte-
           grals B" by Yuri A. Kruglyak and D. R. Whitman,
           Comm. ACM (July 1962)]. Any identifier occurring
           within the Bs is specified to be local to the Bs;
begin   integer m; real alim, delta, gamma, r, epsilon,
           s, k, a2, omega, da, up, jm, q1, q2;
           if a=0 then begin if mod(n−1, 2)=0
           then begin b[n] := 2/n; go to exitBs end;
           b[n] := 0; go to exitBs end;
           if abs(a)<alim then begin delta := gamma;
comment   we have used alim=8 and gamma=1×10^{-8};
           if mod(n−1, 2)=0 then begin r := 2/n;
           epsilon := r×delta;
           s := r; k := 0; a2 := a↑2;
Even: k := k+2;
           r := r×a2×(n+k−2)/((k×(k−1)×(n+k));
           s := s+r; if r>epsilon then go to Even;
           b[n] := s+r;
           go to exitBs end; r := 2×a/(n+1);
           omega := abs(r×delta);
           s := r; k := 1; a2 := a↑2;
Odd: k := k+2;
           r := r×a2×(n+k−2)/(k×(k−1)×(n+k));
           s := s+r; if abs(r)>omega then go to Odd;
           b[n] := −(s+r); go to exitBs end; da := 1/a;
           jp := da×exp(a);
           jm := (da↑2)/jp; b[1] := jp−jm;
           if n=1 then go to exitBs;
           q1 := −1; q2 := 1; for m := 2 step 1 until n
           do begin b[m] := q1×jp−jm+q2×da×b[m−1];
           q1 := −q1; q2 := q2+1 end
exitBs: end Bs;
           g := 1; for m := 1 step 1 until 2×n do g := g/m;
           G[2×n] := g;
```

comment    $1/(2n)!=G[2\times n]$ was evaluated;
           zsa := za/n;  zsb := zb/2;  ks := (r/2)↑(n+3)×
           (2×zsa)↑(n+1/2)×zsb↑(5/2)×G[2×n]↑(1/2);
           p := r×(zsa+zsb)/2;

```
                pt := r×(zsa−zsb)/2;
                for c := 1 step 1 until n+3 do
ABSI:           begin As(c, p) Result: (a[c]);
                Bs(c, pt) Result: (b[c])
                end ABSI;
S:              lilya := 0;  for m := 0 step 1 until n do lilya :=
                lilya+bc[n+1, m+1]×(a[n−m+2]×(b[m+1]+
                b[m+3])−b[m+2]×(a[n−m+1]+a[n−m+3]));
                s := ks×lilya;
I1:             k1 := ks×2×za/r;  exc := 0;
                for m := 0 step 1 until n−1 do
                exc := exc+bc[n, m+1]×(a[n−m+1]×(b[m+1]+
                b[m+3])−b[m+2]×(a[n−m]+a[n−m+2]));
                i1 := k1×exc;
I2:             pppt := p+pt;  k2 := (r/2)↑(2×n)×(2×zsa)↑
                (2×n+1)×zsb×G[2×n];
                for c := 1 step 1 until 2×n do
BA2:            begin As(c, pppt) Result: (a[c]);
                Bs(c, pppt) Result: (b[c])
                end BA2;  sue := 0;
                for m := 0 step 1 until 2×n−1 do
                sue := sue+bc[2×n, m+1]×a[2×n−m]×b[m+1];
                i2 := k2×sue;
I3:             pmpt := p−pt;  ptmp := −pmpt;
                k3 := (r/2)↑4×2×za×zsb↑5;
                for c := 1 step 1 until 4 do
AB3:            begin As(c, pmpt) Result: (a[c]);
                Bs(c, ptmp) Result: (b[c])
                end AB3;  i3 := k3×(a[2]×(b[1]+2×b[3])−b[2]×
                (a[1]+2×a[3])+a[4]×b[3]−a[3]×b[4]) end
                INTSOLI;
```

ALGORITHM 111
MOLECULAR-ORBITAL CALCULATION OF MOLECULAR INTERACTIONS

YURI A. KRUGLYAK

Kharkov State University, Kharkov, U.S.S.R., AND

DONALD R. WHITMAN

Case Institute of Technology, Cleveland, Ohio

real procedure SOLI(n, za, zb, rem, coef2, r1, E1, dr1, drk1, acc, acc1, rk2, rk1, dr2, asy, rk3, dr3, As, Bs, RESULT) Result: (ra, Ep, Em, ca, ca2, cb, cb2, DEa, DEb, s, i1, i2, i3, haa, hab, hbb); **value** n, za, zb, rem, coef2, r1, E1, dr1, rk1, drk1, acc, acc1, rk2, dr2, asy, rk3, dr3; **integer** n, rem; **real** otherwise; **real array** a[1:8], b[1:8], G[1:8], zsap [1:9], rp[1:8], ans[1:rem]; **real procedure** As, Bs, RESULT;

comment This procedure calculates a one-electron approximation to the energy of interaction of molecular species by the use of the molecular-orbital (MO) method with a linear combination of Slater-type orbitals (LCSTO). The wave function used is $\mid \phi\rangle = c_a \mid \psi^a_{n00}\rangle + c_b \mid \psi^b_{210}\rangle$, where $\mid \psi^a_{n1m}\rangle$ is a STO centered on nucleus a. The effective principal quantum number n takes the integral value 1, 2, 3 or 4. The Hamiltonian used is: $\mathcal{K}_{ab} = -\Delta/2 - Z_a^*/r_a - Z_b^*/r_b + Z_a^* Z_b^*/R_{ab}$. Here Z_a^* and Z_b^* are effective nuclear charges, r_a and r_b the distances of the electron from nucleus a and b, and R_{ab} is the distance between nuclei a and b. The calculations are in atomic units, while the output ra is in Angstroms and DEa and DEb are in kcal/gm-ion. Abbreviations of the following type are used: $Za = Z_a^*$, $ra = R_{ab}(\text{Å})$, $haa = \langle \psi^a_{n00} \mid \mathcal{K}_{ab} \mid \psi^a_{n00}\rangle$, $DEa = D(a, b+\text{electron})$, $e1 = \langle \psi^a_{n00} \mid -\Delta/2 - Z_a^*/ra \mid \psi^a_{n00}\rangle$. The values of coef1 and coef2 are 627.71 (kcal/gm-ion) and 0.5291Å, respectively. $r1$, $E1$, $dr1$, $dr2$, $dr3$, acc, $acc1$, and asy are control parameters. The accuracy of the calculations (acc, $acc1$) is $1×10^{-5}$. The

initial values of R_{ab} and E_- are conveniently: $r1 = 0.4(\text{Å})$, and $E1 = 100(\text{a.u.})$. The steps are: $dr1 = 0.1$, $dr2 = 0.4$, and $dr3 = 0.01$, all in Angstroms. asy is $-1×10^{-3}$ (a.u.);

begin   integer q, t, c, m, f; **real** otherwise;
procedure As(n, b) Result: (a[n]); **value** n, b; **integer** n; **real** b;
comment any identifier occurring within the *As* is specified to be local to the *As*;
begin   integer m; **real** db; a[1] := exp(−b)/b;
            if n=1 **then go to** exitAs; db := 1/b;
            for m := 2 **step** 1 **until** n **do** a[m] := a[1]+db×
            (m−1)×a[m−1]
exitAs: **end** As;
procedure Bs(n, a) Result: (b[n]); **value** n, a; **integer** n; **real** a;
comment any identifier occurring within the *Bs* is specified to be local to the *Bs*;
begin   integer m; **real** otherwise; **if** a=0 **then begin if**
            mod(n−1, 2)=0 **then begin** b[n] := 2/n;
            go to exitBs **end**; b[n] := 0; **go to** exitBs **end**;
            if abs(a)<alim **then begin** delta := gamma;
            if mod(n−1, 2)=0 **then begin** r := 2/n;
            epsilon := r×delta; s := r; k := 0; a2 := a↑2;
Even: k := k+2; r := r×a2×(n+k−2)/(k(k−1)(n+k));
            s := s+r; **if** r>epsilon **then go to** Even;
            b[n] := s+r; **go to** exitBs **end**;
            r := 2×a/(n+1); omega := abs(r×delta);
            s := r; k:=1; a2 := a↑2;
Odd: k := k+2; r := r×a2×(n+k−2)/(k(k−1)(n+k));
            s := s+r; **if** abs(r)>omega **then go to** Odd;
            b[n] := −(s+r); **go to** exitBs
            end; da := 1/a; jp := da×exp(a);
            jm := (da↑2)/jp; b[1] := jp−jm;
            if n=1 **then go to** exitBs; q1 := −1; q2 := 1;
            for m := 2 **step** 1
            until n **do begin** b[m] := q1×jp−jm+
            q2×da×b[m−1];
            q1 := −q1; q2 := q2+1 **end**
exitBs: **end** Bs;
procedure Result(coef1); **real** coef1;
comment RESULT computes *Ep, Em, ca, ca2, cb, cb2, DEa, DEb, s, i1, i2, i3, nn, haa, hab, hbb*. Important: RESULT and any identifier occurring within the RESULT enter SOLI as nonlocal entities;
begin   r := ra×br; rp[1] := r; **for** c := 2 **step** 1 **until**
            n+4 **do** rp[c] := rp[c−1]×rp[1]; p := r×sum;
            pt := r×dif; ks := rp[n+3]×zss;
            for c:= 1 **step** 1 **until** n+3 **do begin** As(c, p)
            Result: (a[c]); Bs(c, pt) Result: (b[c]) **end**;
            lilya := 0; **for** m := 0 **step** 1 **until** n **do** lilya :=
            lilya+bc[n+1, m+1]×(a[n−m+2]×
            (b[m+1]+b[m+3])−b[m+2]×(a[n−m+1]
            +a[n−m+3])); s := ks×lilya: 1i12 := 2×s;
            k1 := ks×za/r; exc := 0; **for** m := 0 **step** 1
            until n−1 **do** exc := exc+bc[n, m+1]×(a[n−
            m+1]×(b[m+1]+b[m+3])−b[m+2]×(a[n−m]+
            a[n−m+2])); i1 := k1×exc; pppt := p+pt;
            k2 := rp[2×n]×zsbd; **for** c := 1 **step** 1 **until**
            2×n **do begin** As(c, pppt) Result: (a[c]);
            Bs(c, pppt) Result: (b[c]) **end**; sue := 0;
            for m := 0 **step** 1 **until** 2×n−1 **do** sue :=
            sue+bc[2×n, m+1]×a[2×n−m]×b[m+1];
            i2 := k2×sue; pmpt := p−pt;
            ptmp := −pmpt; k3 := rp[4]×z5;
            for c := 1 **step** 1 **until** 4 **do begin** As(c, pmpt)
            Result: (a[c]); Bs(c, ptmp) Result: (b[c]) **end**;
            i3 := k3×(a[2]×(b[1]+2×b[3])−b[2]×(a[1]+2×
            a[3])+a[4]×b[3]−a[3]×b[4]);
```

comment Two-center integrals s, $i1$, $i2$, and $i3$ were computed [see Algorithm 110, "Quantum Mechanical Integrals of Slater-Type Orbitals," by Yuri A. Kruglyak and D. R. Whitman, *Comm. ACM* (July 1962)]; nn := zz/(2×r); e2pnn := e2+nn; haa := e1−e2+nn; hbb := e2pnn−i3; hab := e2pnn×s−i1; den := 2−s×1i12; bsr := haa+hbb−hab×1i12; root := sqrt(bsr↑2−2×den×(haa×hbb−hab↑2)); Ep := (bsr+root)/den; Em := (bsr−root)/den; ans[f] := Em; DEa := coef1×(e2−Em); DEb := coef1×(e1−Em); Emhaa := Em−haa; Emhbb := Em−hbb; ES := Em×s; habmES := hab−ES; caDcb1 := habmES/Emhaa; cbDca2 := habmES/Emhbb; if abs(Emhaa)>abs(Emhbb) then begin col := caDcb1↑2; cb2 := 1/(1+1i12×caDcb1+col); ca2 := cb2×col; ca := sqrt(ca2); cb := ca/caDcb1 go to NATA end; co2 := cbDca2↑2; ca2 := 1/(1+1i12×cbDca2+co2); cb2 := ca2×co2; ca := sqrt(ca2); cb := ca×cbDca2

NATA: end RESULT;

Begin of program: bc[1, 1] := bc[2, 1] := bc[2, 2] := 1; for q := 3 step 1 until 8 do begin bc[q, 1] := 1; for t := 2 step 1 until q−1 do bc[q, t] := bc[q−1, t−1]+bc[q−1, t]; bc[q, q] := 1 end;

IZM: g := 1; for m := 1 step 1 until 2×n do g := g/m; G [2×n]:=g; zsa := za/n; zsap[1] := zsa×2; for c := 2 step 1 until 2×n+1 do xsap[c] := zsap[c−1]×zsap[1]; D := zsap[2×n+1]×G[2×n]; DS := sqrt(D); e1 := −zsap[2]×0.125×(4×n−3)/(2×n−1); zsb := zb×0.5; sum := zsa+zsb; dif := zsa−zsb; zsb5 := zsb↑5; zss := DS×sqrt(zsb5); zsbd := zsb×D; z5 := 2×za×zsb5; zz := za×zb; e2 := −(zsb↑2)/2; br := 0.5/coef2; f := 1; ans[1] := E1; ra := r1;

KOM: ra := ra+dr1; if ra>rk1 then ra := ra+drk1; f := f+1; RESULT (coef1); if ans [f]−ans[f−1]≦ acc then begin if ra>rk2 then go to IZM; go to KOM end; ansf := ans[f]; d1 := ra;

CLEV: ra := ra+dr2; RESULT (coef1); if e1<e2 then begin if Em−e1≦asy∧ra<rk3 then go to CLEV; go to KHAR end; if e1≧e2 then begin if Em−e2≦ asy∧ra<rk3 then go to CLEV; go to KHAR end;

KHAR: ra := d1; ans[f−1] := ansf;

CASE: ra := ra−dr3; f := f+1; RESULT (coef1); if ans[f]−ans[f−1]≦acc1 then go to CASE; go to IZM end SOLI;

REMARK ON ALGORITHM 34
GAMMA FUNCTION [M. F. Lipp, *Comm. ACM* 4 (Feb. 1961)]

MARGARET L. JOHNSON AND WARD SANGREN
Computer Applications, Inc., San Diego, Calif.

The coefficients used in the calculation of the Hasting's polynomial are used in reverse order. The algorithm should have
a[1]=−.19352782; a[2]=.48219939; a[3]=−.75670408; a[4]=.91820686; a[5]=−.89705694; a[6]=.98820589; a[7]=−.57719165; a[8]=1.0; y=.03586834;
for i := 1 step 1 until 8 do y := y×x+a[i];.

Further, since Gamma $(x)=\Gamma(1+x)$, the divisor x in the statement labeled minus should be $x+1$.

REMARK ON ALGORITHM 48
LOGARITHM OF A COMPLEX NUMBER [John R. Herndon, *Comm. ACM* 4 (Apr. 1961)]

MARGARET L. JOHNSON AND WARD SANGREN
Computer Applications, Inc., San Diego, Calif.

Considerable care must be taken in using the arctan function. In Algorithm 48 two such difficulties are ignored. First, it is necessary, because of a resulting division by zero, to deal separately with the case where the real part of the complex number is zero. Second, if the real part of the complex number is negative and the argument of the logarithm is to have a value between $−\pi$ and π then the action depends upon the sign of the imaginary part of the complex number. For clarity the following procedure exhibits in sequence the alternatives:

procedure LOGC (a, b, c, d); value a, b; real a, b, c, d;
comment This procedure computes the number $c+di$ which is equal to $\log_e (a+bi)$. It is assumed that the arctan has a value between $−\pi/2$ and $\pi/2$.
begin if a>0 then begin THETA := 0; go to SOL end; if a<0∧b≧0 then begin THETA := 3.1415927; go to SOL end; if a<0∧b<0 then begin THETA := −3.1415927; go to SOL end; if a=0∧b=0 then begin c := d := 0; go to RETURN end; if a=0∧b>0 then begin c := ln(b); d := 1.570963; go to RETURN end; if a=0∧b<0 then begin c := ln(abs(b)); d := 1.570963; go to RETURN end;
SOL: d := arctan (b/a) + THETA; c := sqrt(a×a+b×b); c := ln(c);
RETURN: end LOGC

CERTIFICATION OF ALGORITHM 51
ADJUST INVERSE OF A MATRIX WHEN AN ELEMENT IS PERTURBED [John R. Herndon, *Comm. ACM* 4 (Apr. 1961)]

RICHARD GEORGE*
Argonne National Laboratory, Argonne, Ill.

This procedure was programmed in FORTRAN and reduced to machine code mechanically. It was run on the Argonne-built computing machine, GEORGE. A floating-point routine was used which allows maximum accuracy to 31 bits.

The procedure was tested for matrices with n ranging from 2 to 10. For each value of n, there were 20 successive trials; each trial consisted of a random perturbation of a randomly selected element of the matrix M, followed by a use of ADJUST, followed by the matrix multiplication N := B·M. For each trial, the adjustment was evaluated by computing

$$\text{sum} := \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} N[i,j] \right\} - n.$$

For random perturbations between $−1.0$ and $+1.0$, the value of sum never exceeded $2.0_{10}−8$.

There are two typographical errors present:

$$B[r,s] = A[r,s] - t \times A[r,i] \times Aj,s] \text{ end}$$

should be

$$B[r,s] := A[r,s] - t \times A[r,i] \times A[j,s] \text{ end}$$

CERTIFICATION OF ALGORITHM 57
BER OR BEI FUNCTION [J. R. Herndon, *Comm. ACM*
4 (Apr. 1961)]
A. P. RELPH
The English Electric Co. Whetstone, England

Algorithm 57 was translated using the DEUCE ALGOL compiler.
No corrections were required, and the results were satisfactory.

CERTIFICATION OF ALGORITHM 70
INTERPOLATION BY AITKEN [C. J. Mifsud, *Comm.*
ACM 4 (Nov. 1961)]
A. P. RELPH
The English Electric Co., Whetstone, England

Algorithm 70 was translated using the DEUCE ALGOL compiler
and gave satisfactory results after semicolons had been added to

$$t := j+1 \quad to\ make\ it \quad t := j+1;$$

and $\quad (x[i]-x[j]) \ \text{end} \quad to\ make\ it \quad (x[i]-x[j]) \ \text{end};$

The identifier t can be eliminated and the algorithm shortened
by the following changes:

Replace **begin integer** i, j, t;   *by*   **begin integer** i, j;
Replace   t := j+1;   *by*   **for** i := j+1 **step** 1 **until**
    for i := t **step** 1 **until**   n **do**
     n **do**

CERTIFICATION OF ALGORITHM 75
FACTORS [J. E. L. Peck, *Comm. ACM* 5 (Jan. 1962)]
A. P. RELPH
The English Electric Co., Whetstone, England

Algorithm 75 was translated using the DEUCE ALGOL compiler
and gave satisfactory results after the following corrections had
been made:

    begin if q=0∨(an÷q)×q=an **then**
     begin if q>1∧p=1 **then**
was changed to
    begin if q≦1 **then go to** NO CONSTANT;
     if (an÷q)×q=an **then**
     begin if p=q **then**

---

    begin c := c×a0;   a0 := 1
    end
was changed to
    begin c := c×a[0];   a[0] := 1;
    end

There are now r (r>0) rational linear factors (u$_i$ x−v$_i$),
    1<i<r,
was changed to
    If r>0 there are now r rational linear factors (u$_i$x−v$_i$), 1≦i≦ r,

To return to the state ($p=1$, $q=0$) after every factor or constant
is found is inefficient. This can be avoided by substituting a[0]
and a[n] for the identifiers a0 and an respectively. The procedure
then becomes:

```
procedure factors (n, a, u, v, r, c);  value n, a;
   integer array a, u, v;
integer r, n, c;
begin integer p, q;
   r := 0;  c := 1;
ZERO:  if a[n]=0 then
        begin r := r+1;  u[r] := 1;  v[r] := 0;  n := n−1;
          go to ZERO
        end;
        for p := 1 step 1 until abs (a[0]) do
        begin if (a[0]÷p)×p=a[0] then
          begin for q := 1 step 1 until abs (a[n]) do
            begin if q=1 then go to NO CONSTANT;
        TRY AGAIN:  if (a[n]÷q)×q=a[n] then
                  begin integer j;
                    for j := 0 step 1 until n−1 do
                      if (a[j]÷q)×q≠a[j] then go to
                      NO CONSTANT;
                    for j := 0 step 1 until n do
                      a[j] := a[j]/q;
                    c := c×q;  go to TRY AGAIN
                  end;
        NO CONSTANT:  begin integer f, g, i;  f := a[0];
                  g := 1;
                  for i := 1 step 1 until n do
                  begin g := g×p;
                    f := f×q+a[i]×g
                  end;
                  if f=0 then
                  begin r := r+1;  u[r] := p;
                    v[r] := q;
                    begin integer i, t;  t := 0;
                      for i := 0 step 1 until n do
                      begin a[i] := t := (a[i]+t)/p;
                        t := t×q
                      end;
                      n := n−1
                    end
                    go to if n=0 then REDUCED
                      else NO CONSTANT
                  end;
                  q := −q;  if q<0 then go to NO
                      CONSTANT
                  end
            end
          end
        end;
        REDUCED:  if n=0 then
            begin c := c×a[0];  a[0] := 1
            end
end
```


CERTIFICATION OF ALGORITHM 84
SIMPSON'S INTEGRATION [P. E. Hennion, *Comm.*
ACM 5 (Apr. 1962)]
A. P. RELPH
The English Electric Co., Whetstone, England

Simpson's Integration was translated using the DEUCE ALGOL
compiler and, with no corrections, gave satisfactory results.
It is not stated in the comment that integer *n* needs to be even.

CERTIFICATION OF ALGORITHM 108
DEFINITE EXPONENTIAL INTEGRALS A [Yuri A. Kruglyak and Donald R. Whitman, *Comm. ACM 5* (July 1962)]

YURI A. KRUGLYAK
Kharkov State University, Kharkov, U.S.S.R. and
DONALD R. WHITMAN
Case Institute of Technology, Cleveland, Ohio

Integrals $A_n(1,b) = \int_1^\infty x^n \exp(-bx)dx$ occur in physical problems involving spheroidal coordinates, particularly in quantum chemistry calculations. This algorithm was programmed for the Burrough's 220 computer using Burrough's Algebraic Compiler. The program was used to compute tables of $A_n(1,b)$ in the ranges $n=0(1)15$, and $b=0.01(0.01)30.14$. For example, for $n=0(1)15$, and $b=0.25$ and $b=24.0$, the results below were obtained. These are compared with the results (columns 3 and 5) obtained by James Miller, John M. Gerhauser, and F. A. Matsen [*Quantum Chemistry Integrals and Tables*, University of Texas Press, 1959].

n	$b=0.25$	$b=0.25$ (Miller et al.)	$b=24.0$	$b=24.0$ (Miller et al.)
0	.31152031, 01	.31152031322856, 01	.15729727,−11	.15729727267830,−11
1	.15576015, 02	.15576015661428, 02	.16385132,−11	.16385132570656,−11
2	.12772332, 03	.12772332842371, 03	.17095154,−11	.17095154982051,−11
3	.15357950, 04	.15357951442168, 04	.17866621,−11	.17866621640586,−11
4	.24575835, 05	.24575837510601, 05	.18707497,−11	.18707497541261,−11
5	.49151976, 06	.49151986541516, 06	.19627122,−11	.19627122588926,−11
6	.11796476, 08	.11796479885167, 08	.20636507,−11	.20636507915061,−11
7	.33030132, 09	.33030143989988, 09	.21748707,−11	.21748708743056,−11
8	.10569642, 11	.10569646079911, 11	.22979295,−11	.22979296848848,−11
9	.38050711, 12	.38050725887992, 12	.24346962,−11	.24346963586148,−11
10	.15220284, 14	.15220290355200, 14	.25874294,−11	.25874295428724,−11
11	.66969248, 15	.66969277562880, 15	.27588778,−11	.27588779339328,−11
12	.32145238, 17	.32145253230182, 17	.29524115,−11	.29524116937494,−11
13	.16715523, 19	.16715531679695, 19	.31721955,−11	.31721957275639,−11
14	.93606928, 20	.93606977406291, 20	.34234200,−11	.34234202345285,−11
15	.56164156, 22	.56164186443775, 22	.37126102,−11	.37126103733633,−11

The accuracy is at least six significant figures over the entire range. This accuracy is completely satisfactory for all quantum chemical calculations.

CERTIFICATION OF ALGORITHM 109
DEFINITE EXPONENTIAL INTEGRALS B [Yuri A. Kruglyak, D. R. Whitman, *Comm. ACM 5* (July 1962)]

YURI A. KRUGLYAK
Kharkov State University, Kharkov, U.S.S.R., and
DONALD R. WHITMAN
Case Institute of Technology, Cleveland, Ohio

Integrals $B_n(a) = \int_{-1}^{+1} x^n \exp(-ax)dx$ occur in physical problems involving spheroidal coordinates, particularly in quantum chemistry calculations. This algorithm was programmed for the Burroughs-220 computer using a Burroughs Algebraic Compiler. The program was used to compute tables of $B_n(a)$ in the ranges $n=0(1)15$, and $a=0.00(0.01)32.54$. For example, for $n=0(1)15$ and $a=0.25$, and $a=24.0$ the results below were obtained. These are compared with the results (columns 3 and 5) obtained by James Miller, John M. Gerhauser, and F. A. Matsen [*Quantum Chemistry Integrals and Tables*, University of Texas Press, Austin, 1959].

n	$a=0.25$	$a=0.25$ (Miller et al.)	$a=24.0$	$a=24.0$ (Miller et al.)
0	.20208984, 01	.20208985344653, 01	.11037134, 10	.110371342208, 10
1	−.16771064, 00	−.16771066117520, 00	−.10577253, 10	−.105772536282, 10
2	.67921322, 00	.67921324506375, 00	.10155696, 10	.101556964184, 10
3	−.10074584, 00	−.10074585827159, 00	−.97676725, 09	−.976767216847, 09
4	.40896479, 00	.40896480211998, 00	.94091887, 09	.940918885936, 09
5	−.72008754, −01	−.72008756636929, −01	−.90768866, 09	−.907688654174, 09
6	.29268836, 00	.29268837517905, 00	.87679129, 09	.876921258533, 09
7	−.56030292, −01	−.56030294023170, −01	−.84798262, 09	−.847982638338, 09
8	.22792911, 00	.22792912573392, 00	.82105258, 09	.821052542631, 09
9	−.45856272, −01	−.45856272975462, −01	−.79581870, 09	−.795818718590, 09
10	.18664760, 00	.18664761544688, 00	.77212229, 09	.772122289331, 09
11	−.38809718, −01	−.38809719373731, −01	−.74982404, 09	−.749824039467, 09
12	.15803198, 00	.15803200452627, 00	.72880141, 09	.728801402343, 09
13	−.33640562, −01	−.33640563670387, −01	−.70894600, 09	−.708945995807, 09
14	.13702696, 00	.13702696892367, 00	.69016158, 09	.690161591189, 09
15	−.29686662, −01	−.29686663616401, −01	−.67236245, 09	−.672362427583, 09

The accuracy is at least six significant figures in the ranges mentioned above. This accuracy is enough for the majority of quantum chemistry calculations.

CERTIFICATION OF ALGORITHM 110
QUANTUM MECHANICAL INTEGRALS OF SLATER-TYPE ORBITALS [Yuri A. Kruglyak and Donald R. Whitman, *Comm. ACM 5* (July 1962)]

YURI A. KRUGLYAK
Kharkov State University, Kharkov, U.S.S.R. and
DONALD R. WHITMAN
Case Institute of Technology, Cleveland, Ohio

This procedure was written and tested in the Burroughs 220 version of the ALGOL language in the spring of 1961 at Case Institute of Technology. The program was used to compute tables of quatnum mechanical integrals s, $i1$, $i2$, and $i3$ in the ranges: $r(\text{Å})=0.64(0.02)1.40(0.10)3.10$; $Z_b{}^*=0.25(0.50)3.75$, 3.90, 4.25, 4.55, 4.75, 5.20, 5.25; $Z_a{}^*=0.7$, 1.0 for $n=1$; 1.3(1.0)3.3 for $n=2$; 0.2, 2.2(1.0)4.2 for $n=3$; and 0.2, 2.2, 3.2 for $n=4$. The table at the right shows typical results compared with values from *Integraltafeln zur Quantenchemie* by H. Preuss (Springer-Verlag, 1957), Zweiter Band. Accuracy is at least six significant figures in the ranges mentioned above. This is ample for the overwhelming majority of quantum chemistry calculations.

Certification of INTSOLI

Input n	r	za	zb		K. and W. Result	Preuss' Result — Notation		Table No.
1 5	0.5	0.2		s	0.14841691	0.5 0.1 [1a 3b]	0.148417	23
1 1	4.5	8.0		s	0.35203437	4.5 4.0 [1a 3b]	0.352034	30
2 1	20	20		s	$0.25032133 \times 10^{-1}$	10 10 [2a 3b]	0.250321×10^{-1}	40
1 5	0.5	0.2		$i1$	0.22058816	0.5 0.1 5[a⁻¹ | 1a 3b]	0.220588	59
1 1	4.5	8.0		$i1$	0.96587055	4.5 4.0 1[a⁻¹ | 1a 3b]	0.965871	66
2 1	20	20		$i1$	$0.58102500 \times 10^{-1}$	10 10 1[a⁻¹ | 2a 3b]	0.581025×10^{-1}	76
1 1	0.5	0.2		$i2$	0.44818080	0.5 0.5 1[b⁻¹ | 1a 1a]	0.448181	41
1 5	0.5	0.2		$i2$	0.97641725	0.5 0.5 5[b⁻¹ | 1a 1a]	0.976417	45
2 1	20	20		$i2$	0.99999530	10 10 1[b⁻¹ | 2a 2a]	0.100000×10^{1}	58
1 1	10	1		$i3$	0.26217432	0.5 0.5 1[b⁻¹ | 3a 3a]	0.262174	41
1 1	10	10		$i3$	0.11093011×10^{1}	5 5 1[b⁻¹ | 3a 3a]	0.110929×10^{1}	50
1 1	10	20		$i3$	0.10300137×10^{1}	10 10 1[b⁻¹ | 3a 3a]	0.103001×10^{1}	58