Algorithm 464

Eigenvalues of a Real, Symmetric, Tridiagonal Matrix [F2]

Christian H. Reinsch [Recd. 11 Mar. 1971]
Mathematisches Institut der Technischen Universität, 8000 München 2, Arcisstra 21, Germany

Key Words and Phrases: eigenvalues, QR Algorithm
CR Categories: 5.14
Language: Algol

Description

This algorithm uses a rational variant of the QR transformation with explicit shift for the computation of all of the eigenvalues of a real, symmetric, and tridiagonal matrix. Details are described in [1]. Procedures *tred*1 or *tred*3 published in [2] may be used to reduce any real, symmetric matrix to tridiagonal form. Turn the matrix end-for-end if necessary to bring very large entries to the bottom right-hand corner.

References

1. Reinsch, C.H. A stable, rational QR algorithm for the computation of the eigenvalues of an Hermitian, tridiagonal matrix. *Math. Comp. 25* (1971), 591–597.
2. Martin, R.S., Reinsch, C.H., Wilkinson, J. H. Householder's tridiagonalization of a symmetric matrix. *Numer. Math. 11* (1968), 181–195.

Algorithm

```
procedure tqlrat (n,macheps) trans: (d,e2);
  value n, macheps;
  integer n; real macheps; array d, e2;
comment
               Input:
  n            order of the matrix,
  macheps      the machine precision, i.e. minimum of all x such that
               1 + x > 1 on the computer,
  d[1:n]       represents the diagonal of the matrix,
  e2[1:n]      represents the squares of the sub-diagonal entries,
               (e2[1] is arbitrary).
               Output:
  d[1:n]       the computed eigenvalues are stored in this array in
               ascending sequence,
  e2[1:n]      is used as working storage and the original informa-
               tion stored in this array is lost;
begin
  integer i, k, m; real b, b2, f, g, h, p2, r2, s2;
  for i := 2 step 1 until n do e2[i-1] := e2[i];
  e2[n] := b := b2 := f := 0.0;
  for k := 1 step 1 until n do
  begin
    h := macheps × macheps × (d[k]↑2 + e2[k]);
    if b2 < h then
    begin b := sqrt(h); b2 := h end;
    comment Test for splitting;
    for m := k step 1 until n do
      if e2[m] ≤ b2 then go to cont1;
cont1:
    if m = k then go to root;
    comment Form the shift from leading 2 × 2 block;
nextit:
    g := d[k]; p2 := sqrt(e2[k]);
    h := (d[k+1]-g)/(2.0×p2); r2 := sqrt(h×h+1.0);
    d[k] := h := p2/(if h<0.0 then h-r2 else h+r2);
    h := g - h; f := f + h;
    for i := k + 1 step 1 until n do d[i] := d[i] - h;
    comment Rational QL transformation, rows k through m;
    g := d[m]; if g = 0.0 then g := b;
    h := g; s2 := 0.0;
    for i := m - 1 step -1 until k do
    begin
      p2 := g × h; r2 := p2 + e2[i];
      e2[i+1] := s2 × r2; s2 := e2[i]/r2;
      d[i+1] := h + s2 × (h+d[i]);
      g := d[i] - e2[i]/g; if g = 0.0 then g := b;
      h := g × p2/r2
    end i;
    e2[k] := s2 × g × h; d[k] := h;
    if e2[k] > b2 then go to nextit;
root:
    h := d[k] + f;
    comment One eigenvalue found, sort eigenvalues;
    for i := k step -1 until 2 do
      if h < d[i-1] then d[i] := d[i-1] else go to cont2;
    i := 1;
cont2:
    d[i] := h
  end k
end tqlrat;
```

Algorithm 465

Student's t Frequency [S14]

G.W. Hill [Recd. 24 Aug. 1971, 23 Feb. 1972, 10 July 1972]
C.S.I.R.O., Division of Mathematical Statistics, Glen Osmond, South Australia

Key Words and Phrases: Student's t statistic, density function, series approximation
CR Categories: 5.12, 5.5
Language: Algol

Description

The frequency function for Student's t distribution,

$$f(t \mid n) = \frac{\Gamma(\frac{1}{2}n + \frac{1}{2})}{(\pi n)^{\frac{1}{2}}\Gamma(\frac{1}{2}n)} (1 + t^2/n)^{-(\frac{1}{2}n+\frac{1}{2})},$$

is evaluated for real t and real $n > 0$ to a precision near that of the processor, even for large values of n.

The factor involving t is evaluated as $exp(-\frac{1}{2}b)$ where b is computed as $(n + 1)ln(1 + t^2/n)$ if $t^2/n = c$ is large ($>cmax$, say) or, to avoid loss of precision for smaller c, by summing the series for $b = (t^2 + c)(1 - c/2 + c^2/3 - c^3/4 + \cdots)$ until negligible terms occur, i.e. $c^r/(r + 1) < \epsilon$, where ϵ is the relative magnitude of processor round-off. The relative error up to $\epsilon/cmax$ in evaluating $ln(1 + c)$ and the accumulated round-off error of order $\epsilon\sqrt{R}$ in summing a maximum of R terms of the series can be limited to about the same low level by choosing $cmax = R^{-\frac{1}{2}}$ where $R^{-\frac{1}{2}R}/R \approx \epsilon$. Thus for R = 12, 16, 23, or 32, values of $cmax \approx$ 0.2887, 0.25, 0.2085, or 0.1762, respectively, correspond to processor precision where $\epsilon = 2^{-24}$, 2^{-36}, 2^{-56}, or 2^{-84}, respectively.

Evaluation of the ratio of gamma functions by exponentiating the difference of almost equal values of their logarithms would involve considerable loss of precision for large n. This is avoided by use of the asymptotic series obtained by differencing the Stirling approximations, changing the variable to $a = n - \frac{1}{2}$, and exponentiating the result (see also [1]):

$$\frac{\Gamma(\frac{1}{2}n + \frac{1}{2})}{\Gamma(\frac{1}{2}n)} = (\tfrac{1}{2}a)^{\frac{1}{2}} \sum_{r=0} C_r (4a)^{-2r},$$

where $C_0 = C_1 = 1$, $C_2 = -19/2$, $C_3 = 631/2$, $C_4 = -174317/8$, $C_5 = 204\ 91783/8$, $C_6 = -73348\ 01895/16$, $C_7 = 185\ 85901\ 54455/16$, $C_8 = -5\ 06774\ 10817\ 68765/128$, $C_9 = 2236\ 25929\ 81667\ 88235/128$, $C_{10} = -24\ 80926\ 53157\ 85763\ 70237/256$.

The relative error of the sum of the first s terms is negligible for $n > nmin$ where $| C_s | \times [4\ (nmin - \frac{1}{2})]^{-2s} \approx \epsilon$, e.g. for $s = 5$ and $\epsilon = 2^{-24}$ or 2^{-36}, $nmin \approx 6.271$ or 13.76, respectively, and for $s = 10$ and $\epsilon = 2^{-56}$ or 2^{-84}, $nmin \approx 15.5$ or 40.89, respectively. For smaller n the ratio of gamma functions is obtained from the ratio for some $N \geq nmin$ by the relation:

$$\frac{\Gamma(\frac{1}{2}n + \frac{1}{2})}{\Gamma(\frac{1}{2}n)} = \frac{n}{(n+1)} \frac{(n+2)}{(n+3)} \cdots \frac{(N-2)}{(N-1)} \frac{\Gamma(\frac{1}{2}N + \frac{1}{2})}{\Gamma(\frac{1}{2}N)}.$$

For large n, processor underflow at line 21 is avoided by use of the normal approximation, which is adequate for values of $n > 1/\epsilon$, whose representation is unaffected by subtraction of 0.5. Protection against negative or zero n is provided by returning the distinctive value, -1.0, which may be supplemented by an error diagnostic process, if required.

For double precision calculations speed is improved by evaluating higher order terms of the gamma ratio series using single precision operations. Comparison of double precision ($\epsilon = 2^{-84}$) results with single precision results ($\epsilon = 2^{-36}$, $nmin = 13.76$, $cmax = 0.25$) for a Control Data 3200 indicated achievement generally of about ten significant decimal digits, dropping to about eight significant decimals for arguments beyond the 10^{-20} probability level.

Valuable comments from the referee are gratefully acknowledged.

Reference

1. Fields, J.L. A note on the asymptotic expansion of a ratio of Gamma functions. *Proc. Edinburgh Math. Soc. Ser. 2 15* (1966), 43–45.

Algorithm

```
real procedure t frequency (t, n);
  value t, n; real t, n;
if n ≤ 0.0 then t frequency := −1.0
else
begin
  real a, b, c, d, e, nmin, cmax;
  comment for 36-bit precision processor;
  nmin := 13.76; cmax := 0.25;
  b := t × t; c := b/n; a := d := b + c;
  if c > cmax then b := (n+1.0) × ln(1.0+c)
  else
  for e := 2.0, e + 1.0 while b ≠ d do
  begin a := −a × c; b := d; d := a/e + d end;
  a := n; c := 0.3989422804;
  comment 1/sqrt(2π) = 0.398942280401432677939946 1 . . . ;
  for e := a while e < nmin do
  begin c := c × a/(a+1.0); a := a + 2.0 end;
  a := a − 0.5;
  if a ≠ n then
  begin
    c := sqrt(a/n) × c; a := 0.25/a; a := a × a;
    c := ((((−21789.625×a+315.5)×a−9.5)×a+1.0)×a+1.0)
       × c
  end;
  t frequency := exp(−0.5×b) × c
end Student's t-frequency
```

Algorithm 466

Four Combinatorial Algorithms [G6]

Gideon Ehrlich [Recd. 25 Aug. 1971, 4 Jan. 1972, and 12 Dec. 1972]
Department of Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel

Key Words and Phrases: permutations and combinations
CR Categories: 5.39
Language: PL/I

Description

Each of the following algorithms produce, by successive calls, a sequence of all combinatorial configurations, belonging to the appropriate type.

PERMU Permutations of $N \geq 3$ objects: $X(1), X(2), \ldots, X(N)$.

COMBI Combinations of M natural numbers out of the first N.

COMPOMIN Compositions of an integer P to $M + 1$ ordered terms, $INDEX(k)$, each of which is not less than a given minimum $MIN(k)$.

COMPOMAX The same as *COMPOMIN* but each term has its own maximum *MAX* (k).

The four algorithms have in common the important property that they use neither loops nor recursion; thus the time needed for producing a new configuration is unaffected by the "size" (N, N and M, P and M respectively) of that configuration.

Each algorithm uses a single simple operation for producing a new configuration from the old one, that is:

PERMU A single transposition of two adjacent elements.

COMBI Replacing a single element x by a y having the property that there is no element between x and y belonging to the combination.

COMPOMIN(MAX) Changing the values of two adjacent terms (usually only by 1).

The algorithms are written in *PL*1(*F*).

Special instructions for the user and notes.

PERMU (1) The mean work-time is actually a decreasing function of N since, on $(N - 1)/N$ of the calls, it returns by the first *RETURN*. (2) The procedure operates directly on any object vector $x[1{:}N]$. (3) For the first permutation one must call *FIRSTPER*; for other permutations *PERMU* must be used. (4) Together with the last permutation, which is the original one, we will get *DONE* = '1'*B*. If we continue to call *PERMU*, the entire sequence will repeat indefinitely. If at any stage we set *DONE* = '0'*B*, then at the end of the appropriate sequence it will become '1'*B*. (5) The entire resulting sequence is the same as that of Johnson [1] and Trotter [2].

COMBI Every combination is represented in two forms: (1) As a bit array of M '1's and $N - M$ '0's which is identical to $A(1), A(2), \ldots, A(N)$. (2) As an array C of M different integers not greater than N. The M elements are ordered according to their magnitude. If the second representation is not needed one can omit Z, H and C together with the last line of the procedure. For the first combination we can use the following initialization (for other initializations see [3]):

DECLARE A(0:*N*) *BIT* (1), (*X*, *Y*, *T*(*N*), *F*(0:*N*), *I*, *L*, *Z*, *H*(*N*), *C*(*M*)) *FIXED*;
DO K = 0 *TO N* − *M*; *A*(*K*) = '0'*B*; *END*;
DO K = *N* − *M* + 1 *TO N*; *A*(*K*) = '1'*B*; *END*;
DO K = 1 *TO M*; *C*(*K*) = *N* − *M* + *K*; *H*(*N* − *M* + *K*) = *K*; *END*;
$T(N - M) = -1$; $T(1) = 0$; $F(N) = N - M + 1$; $I = N - M$; $L = N$;

(The initialization was not done in the body of the procedure *COMBI* only in order to simplify the procedures *COMPOMIN-MAX*:.)

Instead of using such a large number of parameters it is possible to retain only A, I, L as parameters of the procedure and declare and initialize the other present parameters in the body of the procedure (as is done in *PERMU*). In such a case N, T, F, L, H must be declared as *STATIC* or *CONTROLLED* ('own' in *ALGOL*).

COMPOMIN Each of the $M + 1$ *MIN*(k), as well as P, can be any integer (positive, negative, or zero), but the sum S of all those minima cannot be greater than P.

For the first composition set *INDEX*(1) = $P - S +$ *MIN*(1) *INDEX*(k) = *MIN*(k), for $k > 1$.

Set $N = P - S + M$, and declare and initialize all variables that also appear in *COMBI* in the same way as was done for *COMBI*.

Together with the last composition, we will get $I = 0$ as a signal to halt.

COMPOMAX The instructions for *COMPOMIN* are valid for *COMPOMAX* provided: (1) *MIN* is replaced by *MAX* ($S \geq P$); and (2) N is initialized to $N = S - P + M$.

The vector C (but not H!) has no use in *COMPOMIN(MAX)*, so one can omit all statements in which it appears. A justification for the four algorithms and for some others can be found in [3].

Acknowledgment. I would like to thank Professor Shimon Even for guidance and encouragement.

References

1. Johnson, S.N. Generation of permutations by adjacent transformations. *Math. Comp. 17* (1963), 282–285.
2. Trotter, H.F. Algorithm 115, Perm. *Comm ACM 5* (Aug. 1962), pp. 434–435.
3. Ehrlich, G., Loopless algorithms for generation permutations combinations and other combinatorial configurations. *J. ACM 20* (July 1973), 500–513.

Algorithm

```
FIRSTPER:  PROCEDURE  (X,DONE);
DECLARE  (X(*),  (XN,XX) STATIC) DECIMAL, DONE BIT(1)
(N,S,V,M,L,I,DI,IPI) BINARY STATIC,
(P(0:N),IP(N-1),D(N-1),T(N)) BINARY CONTROLLED;
N=DIM(X,1);
IF ALLOCATION (P) THEN FREE P,IP,D,T;  ALLOCATE P,IP,D,T;
DO M=1  TO  N-1;  P(M),IP(M)=M; D(M)=-1;  END;
XN=X(N); V=-1; S,P(0),P(N)=N; M,L=1;
T(N)=N-1; T(N-1)=-2; T(2)=2;
DONE='0'B;
PERMU:  ENTRY  (X,DONE);
IF S¬=M THEN DO; X(S)=X(S+V); S=S+V; X(S)=XN; RETURN; END;
I=T(N);                     DI=D(I);
IP(I),IPI=IP(I)+DI;         M=P(IPI);                 IP(M)=IPI-DI;
P(IPI-DI)=M;                P(IPI)=I;                 M=IPI+L;
XX=X(M);                    X(M)=X(M-DI);             X(M-DI)=XX;
L=1-L;                      V=-V;                     M=N+1-S;
IF P(IPI+DI) < I   THEN
DO;   IF I=N-1 THEN RETURN;
      T(N)=N-1; T(N-1) = -I; RETURN;
END;
D(I)=-DI;
IF T(I) < 0   THEN
DO; IF T(I)¬=1-I  THEN T(I-1)=T(I); T(I)=I-1; END;
IF I ¬=N-1 THEN DO; T(N)=N-1; T(N-1)=-I-1; END;
T(I+1)=T(I);
IF I=2 & P(2)=2 THEN DONE='1'B;
END;
COMBI  PROCEDURE  (A,N,X,Y,T,F,I,L,Z,H,C);
DECLARE  A(*)BIT(1),  (N,X,Y,T(*),F(*),I,L,Z,H(*),C(*))  FIXED;
IF  T(I) < 0  THEN
DO;  IF -T(I)¬=I-1  THEN  T(I-1)=T(I);  T(I)=I-1; END;
IF ¬ A(I)  THEN
DO;  X=I;  Y=F(L);
     IF  A(I-1)  THEN  F(I)=F(I-1);  ELSE F(I)=I; IF F(L)=L THEN
     DO; L=I; I=T(I); GOTO CHANGE; END;
     IF L=N THEN
     DO; T(F(N))=-I-1; T(I+1)=T(I); I=F(N);
         F(N)=F(N)+1; GOTO CHANGE;
     END;
     T(L)=-I-1; T(I+1)=T(I);
     F(L)=F(L)+1; I=L; GOTO CHANGE
END;
Y=I;
IF I¬=L THEN
DO;
     F(L),X=F(L)-1; F(I-1)=F(I);
     IF L=N THEN
     DO; IF I=F(N) -1 THEN DO; I=T(I); GOTO CHANGE; END;
         T(F(N)-1)=-I-1; T(I+1)=T(I);
         I=F(N)-1; GOTO CHANGE;
     END;
     T(L)=-I-1; T(I+1)=T(I); I=L; GOTO CHANGE;
END;
X=N; F(L-1)=F(L); F(N)=N; L=N;
IF I=N-1 THEN DO; I=T(N-1); GOTO CHANGE; END;
T(N-1)=-I-1; T(I+1)=T(I); I=N-1;
CHANGE;
A(X)='1'B; A(Y)='0'B;
H(X),Z=H(Y); C(Z)=X;
END COMBI;
COMPOMIN:  PROCEDURE (INDEX,A,N,X,Y,T,F,I,L,Z,H,C);
DECLARE A(*)  BIT(1),
      (INDEX(*),N,X,Y,T (*),F(*),I,L,Z,H(*),C(*))   FIXED;
CALL COMBI   (A,N,X,Y,T,F,I,L,Z,H,C);
INDEX(Z)=INDEX(Z)+X-Y;     INDEX(Z+1)=INDEX(Z+1)+Y-X:
END COMPOMIN;
COMPOMAX: PROCEDURE (INDEX,A,N,X,Y,T,F,I,L,Z,H,C);
DECLARE A(*)  BIT(1),
      (INDEX(*),N,X,Y,T(*),F(*),I,L,Z,H(*),C(*))  FIXED;
CALL COMBI (A,N,X,Y,T,F,I,L,Z,H,C);
INDEX(Z)=INDEX(Z)-X+Y;      INDEX(Z+1)=INDEX(Z+1)-Y+X;
END COMPOMAX;
```

Algorithm 467

Matrix Transposition in Place [F1]

Norman Brenner [Recd. 14 Feb. 1972, 2 Aug. 1972]
M.I.T., Department of Earth and Planetary Sciences, Cambridge, MA 02139

Key Words and Phrases: transposition, matrix operations, permutations, primitive roots, number theory
CR Categories: 3.15, 5.14, 5.39
Language: Fortran

Description

Introduction. Since the problem of transposing a rectangular matrix in place was first proposed by Windley in 1959 [1], several algorithms have been used for its solution [2, 3, 7]. A significantly faster algorithm, based on a number theoretical analysis, is described and compared experimentally with existing algorithms.

Theory. A matrix a, of n_1 rows and n_2 columns, may be stored in a vector v in one of two ways. Element a_{ij} (0-origin subscripts) may be placed rowwise at v_k, $k = in_2 + j$, or columnwise at $v_{k'}$, $k' = i + jn_1$. Clearly, letting $n = n_1$ and $m = n_1n_2 - 1$,

$$k' \equiv nk \pmod{m}. \tag{1}$$

Transposition of the matrix is its conversion from one mode of storage to the other, by performing the permutation (1). This permutation may be done with a minimum of working storage in a minimum number of exchanges by breaking it into its subcycles. For example, for a 4×9 matrix, one subcycle representation is

(0) (1 4 16 29 11 9) (34 31 19 6 24 26)
(22 18 2 8 32 23) (13 17 33 27 3 12)
(5 20 10) (30 15 25) (7 28) (14 21) (35).

The notation for the sixth subcycle, for example, means that $v_5 \leftarrow v_{20} \leftarrow v_{10} \leftarrow v_5$.

For a subcycle starting with element s, the elements of the subcycle are $sn^r \pmod{m}$, for $r = 0, 1, \ldots$. The following theorems are easily established.

THEOREM 1. *All the elements of the subcycle beginning with s are divisible by $d = (s, m)$, the largest common factor of both s and m. They are divisible by no larger divisor of m.*

PROOF. Both m and s are divisible by d, and therefore so is any subcycle element $sn^r \pmod{m}$. But n and m have no common factors (since $m = nn_2 - 1$), so no divisor of m larger than d can divide sn^r. □

THEOREM 2. *For every subcycle beginning with s, there is another (possibly the same) subcycle beginning with $m - s$.*

PROOF. The elements of the second subcycle are just $-sn^r \pmod{m}$. It is the same subcycle if for some r, $n^r \equiv -1 \pmod{m'}$, for $m' = m/(s, m)$. □

The next theorem gives the group representation of the integers modulo m.

THEOREM 3. *Factor m into powers of primes, $m = p_1^{\alpha_1} \cdots p_l^{\alpha_l}$. Let r_i be a primitive root of p_i; that is, the powers $r_i^k \pmod{p_i}$ for $k = 0, 1, \ldots, p - 2$, comprise every positive integer less than p_i. Define the generator $g_i = 1 + Rm/p_i^{\alpha_i}$, where $R \equiv (r_i - 1)(m/p_i^{\alpha_i})^{-1} \pmod{p_i^{\alpha_i}}$. Define the Euler totient function $\phi(1) = 1$; otherwise $\phi(k) =$ the number of integers less than k having no common factor with it. Then, for any integer x less than m, there exist unique indices j_i for which $0 \leq j_i < \phi(p_i^{\alpha_i}/(x, p_i^{\alpha_i}))$ and $x \equiv (x, m)g_1^{j_1} \cdots g_l^{j_l} \pmod{m}$.*

PROOF. In [4]; if any $p_i = 2$, replace $g_i^{j_i}$ by $\pm 5^{j_i}$, where $0 \leq j_i < \phi(2^{\alpha_i - 2}/(x, 2^{\alpha_1 - 2}))$. □

For example, for $m = 35$, as in our example above, $x \equiv 22^{j_1}31^{j_2} \pmod{35}$ for $(x,35) = 1$ and for $0 \leq j_1 < 4$ and $0 \leq j_2 < 6$.

Index notation is analogous to logarithmic notation in that multiplication modulo m becomes merely addition of indices.

The following theorem solves the problem of the subcycle starting points. It is similar to the algorithm in [6].

THEOREM 4. *Let n and m be defined as for (1). Then, for any integer x less than m, upper bounds J_i may be found so that unique indices j_i exist in the range $0 \leq j_i < J_i$ and $x \equiv \pm(x, m) n^{j_0}g_1^{j_1} \cdots g_l^{j_l} \pmod{m}$.*

PROOF. Express n and -1 in index notation. Then, compute from the indices of n the smallest e such that $n^e \equiv 1 \pmod{m}$. Initially, set each $J_i = \phi(p_i^{\alpha_i}/(x, p_i^{\alpha_i}))$. Next, doing only index arithmetic, examine each power $\pm n^j$ for nontrivial relations of the form $g_i^{j_i} \equiv \pm n^j g_1^{j_1} \cdots g_l^{j_l} \pmod{m/(x, m)}$ where $0 \leq j_k < J_k$ for each k. Then set $J_i = j_i$. Stop when the product of the J_i and e equals $\phi(m/(x, m))$, which is the number of integers in subcycles divisible only by (x, m). □

Notice that the choice of J_i by this method is not unique. For example, continuing from above, for $(x, m) = 7$, $n = 4$, $x \equiv 7 \cdot 4^{j_0}22^{j_1} \pmod{35}$, for $0 \leq j_0 < 2$ and $0 \leq j_1 < 2$. The relations found were $(-1)^1 \equiv 4^1 \pmod{5}$, $22^2 \equiv 4^1 \pmod{5}$ and $31^1 \equiv 4^0 \pmod{5}$.

Theorem 4 is more important in theory than in practice. The

Timing Tests

n_1	n_2	m	(all times in msec) Alg.302 T_1	Alg.380 $IWRK$=0 T_2	Alg.380 $IWRK$= $(n_1+n_2)/2$ T_3	$XPOS$ $NWORK$=0 T_4	$XPOS$ $NWORK$= $(n_1+n_2)/2$ T_5	T_1/T_4	T_2/T_4	T_3/T_5
45	50	$13 \cdot 173$	350	317	167	133	67	2,62	2,38	2,50
45	60	2699	558	123	117	90	100	6,20	1,37	1,17
46	50	$11^2 \cdot 19$	367	339	217	106	83	3,46	3,21	2,60
46	60	$31 \cdot 89$	425	350	250	133	83	3,19	2,63	3,00
47	50	$3^4 \cdot 29$	383	378	267	72	67	5,18	5,23	4,00
47	60	2819	483	127	133	90	100	5,36	1,41	1,33
45	180	$7 \cdot 13 \cdot 89$	1200	1050	816	517	300	2,25	2,03	2,72
45	200	8999	1767	408	416	283	300	6,25	1,44	1,39
46	180	$17 \cdot 487$	1816	1233	583	267	267	6,41	4,63	2,19
46	200	9199	1700	508	417	383	317	4,44	1,33	1,32
47	180	$11 \cdot 769$	1450	1133	667	383	267	3,78	2,96	2,50
47	200	$3 \cdot 13 \cdot 241$	983	1150	1067	550	467	1,69	2,09	2,29

tremendous labor in finding primitive roots for large primes (since a table of roots is very bulky) and in finding the index representation of n is not compensated for by time savings afterward; see the timing tests below. The same practical objection holds against the algorithm in [6].

Algorithm. An efficient program breaks naturally into two parts. First determine starting points for the subcycles and then move the data. In each part, the program below is significantly faster than Algorithm 380 in [3].

For each divisor d of m, the subcycles beginning with d and with $m - d$ are done. If the number of data moved is still less than $\phi(m/d)$, further subcycle starting points of the form sd are tried, for $s = 2, 3, \ldots$. The most general test is that sd is acceptable if no element in its subcycle is less than sd or greater than $m - sd$. Since this test requires much time-consuming computation, it is much faster to look for sd in a table where marks are made to indicate that an element has been moved. In some applications, a bit within each datum may be used. For example, if the data are all biased positive, the sign bit may be used; or, for normalized, nonzero, binary floating point data, the high bit of the fraction is always one and so may be used. In general, a special table of length *NWORK* is used. As in [3], $NWORK = (n_1 + n_2)/2$ was found to be sufficient for most cases. However, when m has many divisors, Algorithm 380 must perform the time-consuming general test for many possible starting points when the new algorithm need not.

The inner loop of the algorithm computes (1), moves data, marks in the table, and checks for loop closure. Since the major part of the time of the inner loop is calculating (1), time is saved over Algorithm 380 by moving elements v_k and v_{m-k} simultaneously. In special cases, further savings may be made. For example, m is divisible by 2 only when both n_1 and n_2 are odd. Then the subcycles beginning at $m/2 - s$ and $m/2 + s$ may be done simultaneously with the subcycles from s and $m - s$, thus reducing the number of times (1) is computed.

Timing tests. A set of test matrices were transposed on the 360/65 with all programs written in Fortran H, OPT = 2. The new algorithm was always faster than both Algorithm 380 [3] and Algorithm 302 [2] when $NWORK = (n_1 + n_2)/2$. When $NWORK = 0$, it was slower than Algorithm 380 (for $IWRK = 0$) and Algorithm 302 only for a few cases when $n_1 n_2 < 100$. It was especially faster than Algorithm 380 when $m = n_1 n_2 - 1$ had many factors and there were hence many subcycles.

An experiment was made for cases when m was prime. A known primitive root of m was then taken from a table [5] and was used to generate subcycle starting points. Since no time was wasted in finding the primitive root or in finding subcycle starting points, this test showed the maximum time savable by implementing Theorem 4. For $NWORK = (n_1 + n_2)/2$ and $m > 200$, no improvement was found over the normal algorithm. For $NWORK = 0$, the gain in speed was never more than 25 percent.

References

1. Windley, P.F. Transposing matrices in a digital computer. *Comp. J. 2* (Apr. 1959), 47–48.
2. Boothroyd, J. Algorithm 302, Transpose vector stored array. *Comm. ACM 10* (May 1967), 292–293.
3. Laflin, S., and Brebner, M.A. Algorithm 380: In-situ transposition of a rectangular matrix. *Comm. ACM 13* (May 1970), 324–326.
4. Bolker, E. *An Introduction to Number Theory: An Algebraic Approach.* Benjamin, New York, 1970.
5. Abramowitz, M., and Stegun, I. *Handbook of Mathematical Functions*, Table 24.8. Nat. Bur. of Standards, Washington, D.C., 1964.
6. Pall, G., and Seiden, E. A problem in Abelian Groups, with application to the transposition of a matrix on an electronic computer. *Math. Comp. 14* (1960), 189–192.
7. Knuth, D., *The Art of Computer Programming, Vol. I.* Addison-Wesley, Reading, Mass., 1967, p. 180, prob. 12, and p. 517, solution to prob. 12.

Algorithm

```
      SUBROUTINE XPOSE(A, N1, N2, N12, MOVED, NWORK)
C TRANSPOSITION OF A RECTANGULAR MATRIX IN SITU.
C BY NORMAN BRENNER, MIT, 1/72.  CF. ALG. 380, CACM, 5/70.
C TRANSPOSITION OF THE N1 BY N2 MATRIX A AMOUNTS TO
C REPLACING THE ELEMENT AT VECTOR POSITION I (O-ORIGIN)
C WITH THE ELEMENT AT POSITION N1*I (MOD N1*N2-1).
C EACH SUBCYCLE OF THIS PERMUTATION IS COMPLETED IN ORDER.
C MOVED IS A LOGICAL WORK ARRAY OF LENGTH NWORK.
      LOGICAL MOVED
      DIMENSION A(N12), MOVED(NWORK)
C REALLY A(N1,N2), BUT N12 = N1*N2
      DIMENSION IFACT(8), IPOWER(8), NEXP(8), IEXP(8)
      IF (N1.LT.2 .OR. N2.LT.2) RETURN
      N = N1
      M = N1*N2 - 1
      IF (N1.NE.N2) GO TO 30
C SQUARE MATRICES ARE DONE SEPARATELY FOR SPEED
      I1MIN = 2
      DO 20 I1MAX=N,M,N
        I2 = I1MIN + N - 1
        DO 10 I1=I1MIN,I1MAX
          ATEMP = A(I1)
          A(I1) = A(I2)
          A(I2) = ATEMP
          I2 = I2 + N
   10   CONTINUE
        I1MIN = I1MIN + N + 1
   20 CONTINUE
      RETURN
C MODULUS M IS FACTORED INTO PRIME POWERS.  EIGHT FACTORS
C SUFFICE UP TO M = 2*3*5*7*11*13*17*19 = 9,767,520.
   30 CALL FACTOR(M, IFACT, IPOWER, NEXP, NPOWER)
      DO 40 IP=1,NPOWER
        IEXP(IP) = 0
   40 CONTINUE
C GENERATE EVERY DIVISOR OF M LESS THAN M/2
      IDIV = 1
   50 IF (IDIV.GE.M/2) GO TO 190
C THE NUMBER OF ELEMENTS WHOSE INDEX IS DIVISIBLE BY IDIV
C AND BY NO OTHER DIVISOR OF M IS THE EULER TOTIENT
C FUNCTION, PHI(M/IDIV).
      NCOUNT = M/IDIV
      DO 60 IP=1,NPOWER
        IF (IEXP(IP).EQ.NEXP(IP)) GO TO 60
        NCOUNT = (NCOUNT/IFACT(IP))*(IFACT(IP)-1)
   60 CONTINUE
      DO 70 I=1,NWORK
        MOVED(I) = .FALSE.
   70 CONTINUE
C THE STARTING POINT OF A SUBCYCLE IS DIVISIBLE ONLY BY IDIV
C AND MUST NOT APPEAR IN ANY OTHER SUBCYCLE.
      ISTART = IDIV
   80 MMIST = M - ISTART
      IF (ISTART.EQ.IDIV) GO TO 120
      IF (ISTART.GT.NWORK) GO TO 90
      IF (MOVED(ISTART)) GO TO 160
   90 ISOID = ISTART/IDIV
      DO 100 IP=1,NPOWER
        IF (IEXP(IP).EQ.NEXP(IP)) GO TO 100
        IF (MOD(ISOID,IFACT(IP)).EQ.0) GO TO 160
  100 CONTINUE
      IF (ISTART.LE.NWORK) GO TO 120
      ITEST = ISTART
  110 ITEST = MOD(N*ITEST,M)
      IF (ITEST.LT.ISTART .OR. ITEST.GT.MMIST) GO TO 160
      IF (ITEST.GT.ISTART .AND. ITEST.LT.MMIST) GO TO 110
  120 ATEMP = A(ISTART+1)
      BTEMP = A(MMIST+1)
      IA1 = ISTART
  130 IA2 = MOD(N*IA1,M)
      MMIA1 = M - IA1
      MMIA2 = M - IA2
      IF (IA1.LE.NWORK) MOVED(IA1) = .TRUE.
      IF (MMIA1.LE.NWORK) MOVED(MMIA1) = .TRUE.
      NCOUNT = NCOUNT - 2
C MOVE TWO ELEMENTS, THE SECOND FROM THE NEGATIVE
C SUBCYCLE.  CHECK FIRST FOR SUBCYCLE CLOSURE.
      IF (IA2.EQ.ISTART) GO TO 140
      IF (MMIA2.EQ.ISTART) GO TO 150
      A(IA1+1) = A(IA2+1)
      A(MMIA1+1) = A(MMIA2+1)
      IA1 = IA2
      GO TO 130
  140 A(IA1+1) = ATEMP
      A(MMIA1+1) = BTEMP
      GO TO 160
  150 A(IA1+1) = BTEMP
      A(MMIA1+1) = ATEMP
  160 ISTART = ISTART + IDIV
      IF (NCOUNT.GT.0) GO TO 80
      DO 180 IP=1,NPOWER
        IF (IEXP(IP).EQ.NEXP(IP)) GO TO 170
        IEXP(IP) = IEXP(IP) + 1
        IDIV = IDIV*IFACT(IP)
        GO TO 50
  170   IEXP(IP) = 0
        IDIV = IDIV/IPOWER(IP)
```

```
  180 CØNTINUE
  190 RETURN
      END

      SUBRØUTINE FACTØR(N, IFACT, IPØWER, NEXP, NPØWER)
C FACTØR N INTØ ITS PRIME PØWERS, NPØWER IN NUMBER.
C E.G., FØR N=1960=2**3 *5 *7**2, NPØWER=3, IFACT=3,5,7,
C IPØWER=8,5,49, AND NEXP=3,1,2.
      DIMENSIØN IFACT(8), IPØWER(8), NEXP(8)
      IP = 0
      IFCUR = 0
      NPART = N
      IDIV = 2
   10 IQUØT = NPART/IDIV
      IF (NPART-IDIV*IQUØT) 60, 20, 60
   20 IF (IDIV-IFCUR) 40, 40, 30
   30 IP = IP + 1
      IFACT(IP) = IDIV
      IPØWER(IP) = IDIV
      IFCUR = IDIV
      NEXP(IP) = 1
      GØ TØ 50
   40 IPØWER(IP) = IDIV*IPØWER(IP)
      NEXP(IP) = NEXP(IP) + 1
   50 NPART = IQUØT
      GØ TØ 10
   60 IF (IQUØT-IDIV) 100, 100, 70
   70 IF (IDIV-2) 80, 80, 90
   80 IDIV = 3
      GØ TØ 10
   90 IDIV = IDIV + 2
      GØ TØ 10
  100 IF (NPART-1) 140, 140, 110
  110 IF (NPART-IFCUR) 130, 130, 120
  120 IP = IP + 1
      IFACT(IP) = NPART
      IPØWER(IP) = NPART
      NEXP(IP) = 1
      GØ TØ 140
  130 IPØWER(IP) = NPART*IPØWER(IP)
      NEXP(IP) = NEXP(IP) + 1
  140 NPØWER = IP
      RETURN
      END
```

Algorithm 468

Algorithm for Automatic Numerical Integration Over a Finite Interval [D1]

T.N.L. Patterson [Recd. 20 Jan. 1971, 27 Nov. 1972, 12 Dec. 1972, 26 Mar. 1973]
Department of Applied Mathematics and Theoretical Physics, The Queen's University of Belfast, Belfast BT7 1NN Northern Ireland

Key Words and Phrases: automatic integration, numerical integration, automatic quadrature, numerical quadrature
CR Categories: 5.16
Language: Fortran

Editor's note: *Algorithm 468 described here is available on magnetic tape from the Department of Computer Science, University of Colorado, Boulder, CO 80302. The cost for the tape is $16.00 (U.S. and Canada) or $18.00 (elsewhere). If the user sends a small tape (wt. less than 1 lb.) the algorithm will be copied on it and returned to him at a charge of $10.00 (U.S. only). All orders are to be prepaid with checks payable to ACM Algorithms. The algorithm is recorded as one file of BCD 80 character card images at 556 B.P.I., even parity, on seven track tape. We will supply algorithm at a density of 800 B.P.I. if requested. Cards for algorithms are sequenced starting at 10 and incremented by 10. The sequence number is right justified in column 80. Although we will make every attempt to insure that the algorithm conforms to the description printed here, we cannot guarantee it, nor can we guarantee that the algorithm is correct.—L.D.F. and A.K.C.*

Description

Purpose. The algorithm attempts to calculate automatically the integral of $F(x)$ over the finite interval $[A, B]$ with relative error not exceeding a specified value ϵ.

Method. The method uses a basic integration algorithm applied under the control of algorithms which invoke, if necessary, adaptive or nonadaptive subdivision of the range of integration. The basic algorithm is sufficiently powerful that the subdivision processes will normally only be required on very difficult integrals and might be regarded as a rescue operation.

The Basic Algorithm. The basic algorithm, *QUAD*, uses a family of interlacing whole-interval, common-point, quadrature formulas. The construction of the family is described in detail in [1]. Beginning with the 3-point Gauss rule, a new 7-point rule is derived, with three of the abscissae coinciding with the original Gauss abscissae; the remaining four are chosen so as to give the greatest possible increase in polynomial integrating degree; the resulting 7-point rule has degree 11. The procedure is repeated, adding eight new abscissae to the 7-point rule to produce a 15-point rule of degree 23. Continuing, rules using 31, 63, 127, and 255 points of respective degree 47, 95, 191, and 383 are derived. The 255-point rule has not previously been published. In addition, a 1-point rule (abscissa at the mid-point of the interval of integration) is included in the family to make eight members in all. The 3-point Gauss rule is in fact formally the extension of this 1-point rule. The successive application of these rules, until the two most recent results differ relatively by ϵ or better, is the basis of the method. Due to their interlacing form, no integral evaluations need to be wasted in passing from one rule to the next.

The algorithm has been used for some time on practical problems and has been found to generally perform reliably and efficiently. Its domain of applicability generally coincides with that of the Gauss formula, which is much wider than commonly supposed [2]. It will perform best on "smooth" functions, but the degree of deterioration of performance when applied to functions with various types of eccentricities depends more on the harshness of these eccentricities than on their presence as such. Integrands with large peaks or even singularities at the ends of the interval of integration are handled reasonably well. It may be noted that none of the rules actually uses the end points of the interval as abscissae. Peaks in the integrand at the center of the interval and discontinuities in the integrand are less easily dealt with. Although it is recommended that the algorithm be applied using the control algorithms described later, if desired it can be used directly as follows.

The algorithm is entered by the statement:

CALL QUAD (A, B, RESULT, K, EPSIL, NPTS, ICHECK, F)

The user supplies:

A lower limit of integration.
B upper limit of integration.
EPSIL required relative error.
F $F(X)$ is a user written function to calculate the integrand.

The algorithm returns:

RESULT an array whose successive elements *RESULT*(1), *RESULT*(2), etc., contain the results of applying the successive members of the family of rules. The number of rules actually applied depends on *EPSIL*. The array should be declared by the calling program to have at least eight elements.

K element, *RESULT*(*K*), of array *RESULT* contains the value of the integral to the required relative accuracy. *K* is determined from the convergence criterion:

$$|\,RESULT\,(K) - RESULT\,(K-1)\,| \le EPSIL^{*}\,|\,RESULT\,(K)\,|$$

NPTS number of integrand evaluations.

ICHECK this flag will normally be 0 on exiting from the subroutine. However, if the convergence criterion above is not satisfied after exhausting all members of the family of rules, then the flag is set to 1.

Table I. Test Integrals and Their Values

1.	$\int_0^1 \sqrt{x}\, dx = \frac{2}{3}$
2.	$\int_{-1}^1 [0.92 \cosh(x) - \cos(x)]\, dx \doteq 0.4794282267$
3.	$\int_{-1}^1 dx/(x^4 + x^2 + 0.9) \doteq 1.582232964$
4.	$\int_0^1 x^{\frac{3}{2}}\, dx = \frac{2}{5}$
5.	$\int_0^1 dx/(1 + x^4) \doteq 0.8669729873$
6.	$\int_0^1 dx/(1 + 0.5 \sin(31.4159x)) \doteq 1.154700669$
7.	$\int_0^1 x\, dx/(e^x - 1) \doteq 0.7775046341$
8.	$\int_{0.1}^1 \sin(314.159x)/(3.14159x)\, dx \doteq 0.009098645256$
9.	$\int_0^{10} 50\, dx/(2500x^2 + 1)/3.14159 \doteq 0.4993638029$
10.	$\int_0^{3.1415927} \cos(\cos(x) + 3\sin(x) + 2\cos(2x) + 3\cos(3x) + 3\sin(2x))\, dx \doteq 0.8386763234$
11.	$\int_0^1 \ln(x)\, dx = -1.0$
12.	$\int_0^1 4\pi^2 x \sin(20\pi x)\cos(2\pi x)\, dx \doteq -0.6346651825$
13.	$\int_0^1 dx/(1 + (230x - 30)^2) \doteq 0.0013492485650$

The control algorithms. Two control algorithms are provided, *QSUBA* and *QSUB*, which if necessary invoke subdivision respectively in either an adaptive or a nonadaptive manner. *QSUBA* is generally more efficient than *QSUB*, but since there are reasons for believing [2] that adaptive subdivision is intrinsically less reliable than the nonadaptive form, an alternative is provided.

The adaptive algorithm QSUBA. QUAD is first applied to the whole interval. If a converged result is not obtained (that is, the convergence criterion is not satisfied), the following adaptive subdivision strategy is invoked. At each stage of the process an interval is presented for subdivision (initially the whole interval (A, B)). The interval is halved, and *QUAD* applied to each subinterval. If *QUAD* fails to converge on the first subinterval, the subinterval is stacked for future subdivision and the second subinterval immediately examined. If *QUAD* fails to converge on the second subinterval, it is immediately subdivided and the whole process repeated. Each time a converged result is obtained it is accumulated as the partial value of the integral. When *QUAD* converges on both subintervals the interval last stacked is chosen next for subdivision and the process repeated. A subinterval is not examined again once a converged result is obtained for it, so that a spurious convergence is more likely to slip through than for the nonadaptive algorithm *QSUB*.

The convergence criterion is slightly relaxed in that a panel is deemed to have been successfully integrated if either *QUAD* converges or the estimated absolute error committed on this panel does not exceed ϵ times the estimated absolute value of the integral over (A, B). This relaxation is to try to take account of a common situation where one particular panel causes special difficulty, perhaps due to a singularity of some type. In this case, *QUAD* could obtain nearly exact answers on all other panels, and so the relative error for the total integration would be almost entirely due to the delinquent panel. Without this condition the computation might continue despite the requested relative error being achieved. The risk of underestimating the relative error is increased by this procedure and a warning is provided when it is used.

The algorithm is written as a function with value that of the integral. The call takes the form:

QSUBA(A, B, EPSIL, NPTS, ICHECK, RELERR, F)

and causes $F(x)$ to be integrated over (A, B) with relative error hopefully not exceeding *EPSIL*. *RELERR* gives a crude estimate of the actual relative error obtained by summing the absolute values of the errors produced by *QUAD* on each panel (estimated as the differences of the last two iterates of *QUAD*) and dividing by the calculated value of the integral. The reliability of the algorithm will decrease for large *EPSIL*. It is recommended that *EPSIL* should generally be less than about 0.001. *F* should be declared *EXTERNAL* in the calling program. *NPTS* is the number of integrand evaluations used. The outcome of the integration is indicated by *ICHECK*:

ICHECK = 0. Convergence obtained without invoking subdivision. This corresponds to the direct use of *QUAD*.

ICHECK = 1. Subdivision invoked and a converged result obtained.

ICHECK = 2. Subdivision invoked and a converged result obtained but at some point the relaxed convergence criterion was used. If confidence in the result needs bolstering, *EPSIL* and *RELERR* may be checked for a serious discrepancy.

ICHECK negative. If during the subdivision process the stack of delinquent intervals becomes full a result is obtained, which may be unreliable, by continuing the integration and ignoring convergence failures of *QUAD* which cannot be accommodated on the stack. This occurrence is noted by returning *ICHECK* with negative sign.

The nonadaptive algorithm QSUB. QUAD is first applied to the whole interval. If a converged result is not obtained the following nonadaptive subdivision strategy is invoked.

Let the interval (A, B) be divided into 2^N panels at step N of the subdivision process. *QUAD* is first applied to the subdivided interval on which it last failed to converge, and if convergence is now achieved, the remaining panels are integrated. Should a convergence failure occur on any panel, the integration at that point is terminated and the procedure repeated with N increased by one. The strategy insures that possibly delinquent intervals are examined before work, which later might have to be discarded, is invested on well behaved panels. The process is complete when no convergence failure occurs on any panel, and the sum of the results obtained by *QUAD* on each panel is taken as the value of the integral.

The process is very cautious in that the subdivision of the interval (A, B) is uniform the fineness of which is controlled by the success of *QUAD*. In this way it is much more difficult for a spurious convergence to slip through than for *QSUBA*. The convergence criterion is relaxed as described for *QSUBA*.

The algorithm is used in the same way as *QSUBA* and is called with the same arguments as *QSUBA*. One of the possible values of *ICHECK* has a different interpretation:

ICHECK negative. If during the subdivision process the upper limit on the number of panels which may be generated is reached, a result is obtained, which may be unreliable, by continuing the integration ignoring convergence failures of *QUAD*. This occurrence is noted by returning *ICHECK* with negative sign.

Tests. The algorithms have been found to perform reliably on a large number of practical problems. To give a feeling for the performance, results for a number of contrived examples are given using the adaptive control algorithm, *QSUBA*. It would be difficult to justify these examples as acid tests of any method, but they have the advantage of having being quoted at various times in the literature.

For comparison a number of automatic procedures were used, which include *SQUANK* [3] (adaptive Simpson), as well as the

Table II. Relative Error Requested, 10^{-3}

Integral	N_{CADRE}	N_{QSUBA}	T_{CADRE}/T_{QSUBA}
1	17	15	1.8
2	17	7	2.9
3	33	15	4.4
4	9	7	1.9
5	9	7	2.2
6	175	127	3.2
7	9	7	1.8
8	1137	255	8.5
9	97	127	2.4
10	107	63	2.2
11	137	31	9.9
12	252	63	6.3
13	129	787	.52

N and T with appropriate subscripts give respectively the number of integrand evaluations and the time taken for the computation.

Table III. Relative Error Requested, 10^{-6}

1	33	63	.75
2	33	15	2.6
3	49	31	3.0
4	129	31	5.0
5	17	15	2.0
6	401	255	2.9
7	9	7	1.8
8	2633	255	18.
9	281	255	2.4
10	193	63	3.8
11	233	795	.74
12	532	127	6.4
13	305	1001	.90

Table IV. Relative Error Requested, 10^{-8}

1	65	255	.36
2	33	15	2.7
3	97	31	4.9
4	545	31	20.
5	65	31	3.6
6	569	255	3.8
7	17	15	1.6
8	4001	255	24.
9	337	255	2.8
10	305	127	2.8
11	297	2415	.28
12	932	127	10.
13	481	1017	1.1

modified Havie integrator [4] and *CADRE* [5] (both based on the Romberg scheme). The latter algorithm, which attempts to detect certain types of singularities using the Romberg table, was found, on the examples tried, to be the best overall competitor to *QSUBA*, and only this comparison is quoted. The Havie algorithm was particularly poor and had the disturbing feature of converging spuriously on periodic integrands. Thacher [6] has described the shortcomings of Romberg integration, and Algorithm 400 appears to exhibit them. *SQUANK* was found to be quite good when used at low accuracy, but the performance deteriorated as the demand for accuracy increased. It also gave trouble on some of the more awkward integrals such as 8 and 11. *SQUANK* also computes the integral in the context of absolute error, and since this is meaningless unless an estimate of the order of magnitude of the integral is known, the algorithm can hardly be described as automatic. *CADRE* allows a choice of absolute or relative error. A criticism sometimes levied at relative error is that should the integral turn out to be zero a difficulty will arise. The only advice that can be offered in this respect is that, should a user suspect that this is likely to happen, a constant should be added to the integrand reflecting some appropriate quantity such as the maximum of the integrand. The constant which will be integrated exactly can be removed after the algorithm has done its work.

The test integrals are listed in Table I, and the results obtained for various required relative accuracies in Tables II, III, and IV. Generally *QSUBA* is superior by a substantial margin. The methods are compared in terms of the number of integrand evaluations needed to obtain the required accuracy and also in terms of the times required. For simple integrands the bookkeeping time of some methods can be significant, and *QUAD* can obtain a considerable advantage by its relative simplicity. Integrals 11 and 13 are interesting examples of this. The number of integrand evaluations exceeding 255 indicates that *QSUBA* invoked subdivision to obtain the result. In Tables III and IV *QSUBA* returned *ICHECK* = 2 on integral 11, but the requested tolerance was achieved.

Integral 8 caused special difficulty to *CADRE*, and for Tables III and IV a converged result could be obtained only after a relatively large investment of computer time. The feature of *CADRE* to detect certain singularities should show up in integrals 1 and 11, but the gain does not emerge until high accuracy is requested as in Table IV. For harsher singularities the gain would likely become apparent earlier.

References

1. Patterson, T.N.L. The optimum addition of points to quadrature formulae. *Math. Comp. 22* (1968), 847–856.
2. Cranley, R., and Patterson, T.N.L. On the automatic numerical evaluation of definite integrals. *Comp. J., 14* (1971), 189–198.
3. Lyness, J.N. Algorithm 379, SQUANK. *Comm. ACM 13* (Apr. 1970), 260–263.
4. Wallick, G.C. Algorithm 400, Modified Havie integration. *Comm. ACM 13* (Oct. 1970), 622–624.
5. de Boor, Carl. CADRE: An algorithm for numerical quadrature. *Mathematical Software*. J.R. Rice (Ed.) Academic Press, New York, 1971, pp. 417–449.
6. Thacher, H.C. Jr. Remark on Algorithm 60, *Comm. ACM* (July, 1964), 420–421.

Algorithm

```
      SUBRØUTINE QUAD(A, B, RESULT, K, EPSIL, NPTS, ICHECK, F)
      DIMENSIØN FUNCT(127), P(381), RESULT(8)
C THIS SUBRØUTINE ATTEMPTS TØ CALCULATE THE INTEGRAL ØF F(X)
C ØVER THE INTERVAL *A* TØ *B* WITH RELATIVE ERRØR NØT
C EXCEEDING *EPSIL*.
C THE RESULT IS ØBTAINED USING A SEQUENCE ØF 1,3,7,15,31,63,
C 127, AND 255 PØINT INTERLACING FØRMULAE(NØ INTEGRAND
C EVALUATIØNS ARE WASTED) ØF RESPECTIVE DEGREE 1,5,11,23,
C 47,95,191 AND 383. THE FØRMULAE ARE BASED ØN THE ØPTIMAL
C EXTENSIØN ØF THE 3-PØINT GAUSS FØRMULA.  DETAILS ØF
C THE FØRMULAE ARE GIVEN IN 'THE ØPTIMUM ADDITIØN ØF PØINTS
C TØ QUADRATURE FØRMULAE' BY T.N.L. PATTERSØN,MATHS.CØMP.
C VØL 22,847-856,1968.
C                        *** INPUT ***
C A        LØWER LIMIT ØF INTEGRATIØN.
C B        UPPER LIMIT ØF INTEGRATIØN.
C EPSIL    RELATIVE ACCURACY REQUIRED. WHEN THE RELATIVE
C          DIFFERENCE ØF TWØ SUCCESSIVE FØRMULAE DØES NØT
C          EXCEED *EPSIL* THE LAST FØRMULA CØMPUTED IS TAKEN
C          AS THE RESULT.
C F        F(X) IS THE INTEGRAND.
C                        *** ØUTPUT ***
C RESULT   THIS ARRAY,WHICH SHØULD BE DECLARED TØ HAVE AT
C          LEAST 8 ELEMENTS, HØLDS THE RESULTS ØBTAINED BY
C          THE 1,3,7, ETC., PØINT FØRMULAE. THE NUMBER ØF
C          FØRMULAE CØMPUTED DEPENDS ØN *EPSIL*.
C K        RESULT(K) HØLDS THE VALUE ØF THE INTEGRAL TØ THE
C          SPECIFIED RELATIVE ACCURACY.
C NPTS     NUMBER INTEGRAND EVALUATIØNS.
C ICHECK   ØN EXIT NØRMALLY ICHECK=0. HØWEVER IF CØNVERGENCE
C          TØ THE ACCURACY REQUESTED IS NØT ACHIEVED ICHECK=1
C          ØN EXIT.
C ABSCISSAE AND WEIGHTS ØF QUADRATURE RULES ARE STACKED IN
C ARRAY *P* IN THE ØRDER IN WHICH THEY ARE NEEDED.
      DATA
     * P( 1),P( 2),P( 3),P( 4),P( 5),P( 6),P( 7),
     * P( 8),P( 9),P(10),P(11),P(12),P(13),P(14),
     * P(15),P(16),P(17),P(18),P(19),P(20),P(21),
     * P(22),P(23),P(24),P(25),P(26),P(27),P(28)/
     *  0.77459666924148337704E 00,0.55555555555555555556E 00,
     *  0.88888888888888888889E 00,0.26848808986833344073E 00,
     *  0.96049126870802028342E 00,0.10465622602646726519E 00,
     *  0.43424374934680255800E 00,0.40139741477596222291E 00,
     *  0.45091653865847414235E 00,0.13441525524378422036E 00,
     *  0.51603282997079739697E-01,0.20062852937698902103E 00,
     *  0.99383196321275502221E 00,0.17001719629940260339E-01,
```

```
*  0.88845923287225699889E 00,0.92927195315124537686E-01,
*  0.62110294673722640294E 00,0.17151190913639138079E 00,
*  0.22338668642896688163E 00,0.21915685840158749640E 00,
*  0.22551049979820668739E 00,0.67207754295990703540E-01,
*  0.25807598096176653565E-01,0.10031427861179557877E 00,
*  0.84345657393211062463E-02,0.46462893261757986541E-01,
*  0.85755920049990351154E-01,0.10957842105592463824E 00/
 DATA
* P(29),P(30),P(31),P(32),P(33),P(34),P(35),
* P(36),P(37),P(38),P(39),P(40),P(41),P(42),
* P(43),P(44),P(45),P(46),P(47),P(48),P(49)/
* P(50),P(51),P(52),P(53),P(54),P(55),P(56)/
*  0.99909812496766759766E 00,0.25447807915618744154E-02,
*  0.98153114955374010687E 00,0.16446049854387810934E-01,
*  0.92965485742974005667E 00,0.35957103307129322097E-01,
*  0.83672593816886873550E 00,0.56979509494123357412E-01,
*  0.70249620649152707861E 00,0.76879620499003531043E-01,
*  0.53131974364437562397E 00,0.93627109981264473617E-01,
*  0.33113539325797683309E 00,0.10566989358023480974E 00,
*  0.11248894313318662575E 00,0.11195687302095345688E 00,
*  0.11275525672076869161E 00,0.33603877148207730542E-01,
*  0.12903800100351265626E-01,0.50157139305899537414E-01,
*  0.42176304415585548391E-02,0.23231446639910269443E-01,
*  0.42877960025007734493E-01,0.54789210527962865032E-01,
*  0.12651565562300680114E-02,0.82230079572359296693E-02,
*  0.17978551568128270333E-01,0.28489754745833548613E-01/
 DATA
* P(57),P(58),P(59),P(60),P(61),P(62),P(63),
* P(64),P(65),P(66),P(67),P(68),P(69),P(70),
* P(71),P(72),P(73),P(74),P(75),P(76),P(77)/
* P(78),P(79),P(80),P(81),P(82),P(83),P(84)/
*  0.38439810249455532039E-01,0.46813554990628012403E-01,
*  0.52834946790116519862E-01,0.55978436510476319408E-01,
*  0.99987288812035761194E 00,0.36322148184553065969E-03,
*  0.99720625937222195908E 00,0.25790497946856882724E-02,
*  0.98868475754742947994E 00,0.61155068221172463397E-02,
*  0.97218287474858179658E 00,0.10498246909621321898E-01,
*  0.94634285837340290515E 00,0.15406750466559497802E-01,
*  0.91037115695700429250E 00,0.20594233915912711149E-01,
*  0.86390793819369047715E 00,0.25869679327214746911E-01,
*  0.80694053195021761186E 00,0.31073551111687964880E-01,
*  0.73975604435269475868E 00,0.36064432780782572640E-01,
*  0.66290966002478059546E 00,0.40715510116944318934E-01,
*  0.57719571005204581484E 00,0.44914531653632197414E-01,
*  0.48361802694584102756E 00,0.48564330406673198716E-01/
 DATA
* P( 85),P( 86),P( 87),P( 88),P( 89),P( 90),P( 91),
* P( 92),P( 93),P( 94),P( 95),P( 96),P( 97),P( 98),
* P( 99),P(100),P(101),P(102),P(103),P(104),P(105),
* P(106),P(107),P(108),P(109),P(110),P(111),P(112)/
*  0.38335932419873034692E 00,0.51583253952048458777E-01,
*  0.27774982202182431507E 00,0.53905499335266063927E-01,
*  0.16823525155220746498E 00,0.55481404356559363988E-01,
*  0.56344313046592789972E-01,0.56277699831254301273E-01,
*  0.56377628360384717388E-01,0.16801938574103865271E-01,
*  0.64519000501757369228E-02,0.25078569652949768707E-01,
*  0.21088152457266328793E-02,0.11615723319955134727E-01,
*  0.21438980012503867246E-01,0.27394605263981432516E-01,
*  0.63260731936263354422E-03,0.41115039786546930472E-02,
*  0.89892757840641357233E-02,0.14244877372916774306E-01,
*  0.19219905124727766019E-01,0.23406777495314006201E-01,
*  0.26417473395058259931E-01,0.27989218255238159704E-01,
*  0.18073956444538835782E-03,0.12895240826104173912E-02,
*  0.30577534101755311361E-02,0.52491234548088591251E-02/
 DATA
* P(113),P(114),P(115),P(116),P(117),P(118),P(119),
* P(120),P(121),P(122),P(123),P(124),P(125),P(126),
* P(127),P(128),P(129),P(130),P(131),P(132),P(133),
* P(134),P(135),P(136),P(137),P(138),P(139),P(140)/
*  0.77033752332797418482E-02,0.10297116957956355524E-01,
*  0.12934839663607373455E-01,0.15536775555843982440E-01,
*  0.18032216390391286320E-01,0.20357755058472159467E-01,
*  0.22457265826816098707E-01,0.24282165203336599358E-01,
*  0.25791626976024229388E-01,0.26952749667633031963E-01,
*  0.27740702178279681994E-01,0.28138849915627150636E-01,
*  0.99998243035489159858E 00,0.50536095207862517625E-04,
*  0.99959879967191068325E 00,0.37774664632698466027E-03,
*  0.99831663531840739253E 00,0.93836984854238150079E-03,
*  0.99572410469840718851E 00,0.16811428654214699063E-02,
*  0.99149572117810613240E 00,0.25687649437940203731E-02,
*  0.98537149959852037111E 00,0.35728927835172996494E-02,
*  0.97714151463970571416E 00,0.46710503721143217474E-02,
*  0.96663785155841656709E 00,0.58434498758356395076E-02/
 DATA
* P(141),P(142),P(143),P(144),P(145),P(146),P(147),
* P(148),P(149),P(150),P(151),P(152),P(153),P(154),
* P(155),P(156),P(157),P(158),P(159),P(160),P(161),
* P(162),P(163),P(164),P(165),P(166),P(167),P(168)/
*  0.95373000642576113641E 00,0.70724899954335554680E-02,
*  0.93832039777959288365E 00,0.83428387539681577056E-02,
*  0.92034002547001242073E 00,0.96411777297025366953E-02,
*  0.89974489977694003664E 00,0.10955733387837901648E-01,
*  0.87651341448470526974E 00,0.12275830560082770087E-01,
*  0.85064449476835027976E 00,0.13591571009765546790E-01,
*  0.82215625436498040737E 00,0.14893641664815182035E-01,
*  0.79108493379984836143E 00,0.16173218729577719942E-01,
*  0.75748396638051363793E 00,0.17421930159464173747E-01,
*  0.72142308537009891548E 00,0.18631848256138790186E-01,
*  0.68298743109107922809E 00,0.19795495048097499488E-01,
*  0.64227664250975951377E 00,0.20905851445812023852E-01,
*  0.59940393024224289297E 00,0.21956366305317824939E-01,
*  0.55449513263193254887E 00,0.22940964229387748761E-01/
 DATA
* P(169),P(170),P(171),P(172),P(173),P(174),P(175),
* P(176),P(177),P(178),P(179),P(180),P(181),P(182),
* P(183),P(184),P(185),P(186),P(187),P(188),P(189),
* P(190),P(191),P(192),P(193),P(194),P(195),P(196)/
*  0.50768775753371660215E 00,0.23854052106038540080E-01,
*  0.45913001198983233287E 00,0.24690524744487676909E-01,
*  0.40897982122988867241E 00,0.25445769965464765813E-01,
*  0.35740383783153215238E 00,0.26115673376706097680E-01,
*  0.30457644155671404334E 00,0.26696622927450359906E-01,
*  0.25067873030348317661E 00,0.27185513229624791819E-01,
*  0.19589750271110015392E 00,0.27579749566481873035E-01,
*  0.14042423315256017459E 00,0.27877251476613701609E-01,
*  0.84454040083710883710E-01,0.28076455793817246607E-01,
*  0.28184648949745694339E-01,0.28176319033016602131E-01,
*  0.28188814180192358694E-01,0.84009692870519326354E-02,
*  0.32259500250878684614E-02,0.12539284826474884353E-01,
*  0.10544076228633167722E-02,0.58078616599775673635E-02,
*  0.10719490006251933623E-01,0.13697302631990716258E-01/
 DATA
* P(197),P(198),P(199),P(200),P(201),P(202),P(203),
* P(204),P(205),P(206),P(207),P(208),P(209),P(210),
* P(211),P(212),P(213),P(214),P(215),P(216),P(217),
* P(218),P(219),P(220),P(221),P(222),P(223),P(224)/
*  0.31630366082226447689E-03,0.20557519893273465236E-02,
*  0.44946378920320678616E-02,0.71224386864583871532E-02,
*  0.96099525623638830097E-02,0.11703388747657003101E-01,
*  0.13208736697529129966E-01,0.13994609127619079852E-01,
*  0.90372734658751149261E-04,0.64476204130572477933E-03,
*  0.15288767050877655684E-02,0.26245617274044295626E-02,
*  0.38516876166398709241E-02,0.51485584789781777618E-02,
*  0.64674198318036867274E-02,0.77683877779219912200E-02,
*  0.90161081951956431600E-02,0.10178877529236079733E-01,
*  0.11228632913408049354E-01,0.12141082601668299679E-01,
*  0.12895813488012114694E-01,0.13476374833816515982E-01,
*  0.13870351089139840997E-01,0.14069424957813575318E-01,
*  0.25157870384280661489E-04,0.18887326450650491366E-03,
*  0.46918492424785040975E-03,0.84057143271072246365E-03/
 DATA
* P(225),P(226),P(227),P(228),P(229),P(230),P(231),
* P(232),P(233),P(234),P(235),P(236),P(237),P(238),
* P(239),P(240),P(241),P(242),P(243),P(244),P(245),
* P(246),P(247),P(248),P(249),P(250),P(251),P(252)/
*  0.12843824718970101768E-02,0.17864463917586498247E-02,
*  0.23355251860571608737E-02,0.29217249379178197538E-02,
*  0.35362449977167777340E-02,0.41714193769840788528E-02,
*  0.48205888648512683476E-02,0.54778666939189508240E-02,
*  0.61379152800413850435E-02,0.67957855048827733948E-02,
*  0.74468208324075910174E-02,0.80866093647888599710E-02,
*  0.87109650797320868736E-02,0.93159241280693950932E-02,
*  0.98977475240487497440E-02,0.10452925722906011926E-01,
*  0.10978183152658912470E-01,0.11470482114693874380E-01,
*  0.11927026053019270040E-01,0.12345262372243838455E-01,
*  0.12722884982732382906E-01,0.13057836688353048840E-01,
*  0.13348311463725179953E-01,0.13592756614812395910E-01,
*  0.13789874783240936517E-01,0.13938625738306850804E-01,
*  0.14038227896908623303E-01,0.14088159516508301065E-01/
 DATA
* P(253),P(254),P(255),P(256),P(257),P(258),P(259),
* P(260),P(261),P(262),P(263),P(264),P(265),P(266),
* P(267),P(268),P(269),P(270),P(271),P(272),P(273),
* P(274),P(275),P(276),P(277),P(278),P(279),P(280)/
*  0.99999759637974846462E 00,0.69379364324108267170E-05,
*  0.99994399620705437576E 00,0.53275293669780613125E-04,
*  0.99976049092443204733E 00,0.13575491094922871973E-03,
*  0.99938033802502358193E 00,0.24921240048299729402E-03,
*  0.99874561446809511470E 00,0.38974528447328229322E-03,
*  0.99780535449595727456E 00,0.55429531493037471492E-03,
*  0.99651414591489027385E 00,0.74028280424450333046E-03,
*  0.99483150280062100052E 00,0.94536151685852538246E-03,
*  0.99272134428278861533E 00,0.11674841174299594077E-02,
*  0.99015137040077015918E 00,0.14049079956551446427E-02,
*  0.98709252795403406719E 00,0.16561127281544526052E-02,
*  0.98351865757863272876E 00,0.19197129710138724125E-02,
*  0.97940628167086268381E 00,0.21944069253638388388E-02,
*  0.97473445975240266776E 00,0.24789582266575679307E-02/
 DATA
* P(281),P(282),P(283),P(284),P(285),P(286),P(287),
* P(288),P(289),P(290),P(291),P(292),P(293),P(294),
* P(295),P(296),P(297),P(298),P(299),P(300),P(301),
* P(302),P(303),P(304),P(305),P(306),P(307),P(308)/
*  0.96948465950245923177E 00,0.27721957645934509940E-02,
*  0.96364062156981213252E 00,0.30730184347025783234E-02,
*  0.95718821610986096274E 00,0.33803979910869203823E-02,
*  0.95011529752129487656E 00,0.36933779170256508183E-02,
*  0.94241156519108305981E 00,0.40110687240750233989E-02,
*  0.93406843615772578800E 00,0.43326409680929828545E-02,
*  0.92507893290707565236E 00,0.46573172997568547773E-02,
*  0.91543758715576504064E 00,0.49843645647655386012E-02,
*  0.90514035881326159519E 00,0.53130866051870565663E-02,
*  0.89418456833555902286E 00,0.56428181013844441585E-02,
*  0.88256884024734190684E 00,0.59729195655081658049E-02,
*  0.87029305554811390585E 00,0.63027734490857587172E-02,
*  0.85735831088623215653E 00,0.66317812429018878941E-02,
*  0.84376688267270860104E 00,0.69593614093904229394E-02/
 DATA
* P(309),P(310),P(311),P(312),P(313),P(314),P(315),
* P(316),P(317),P(318),P(319),P(320),P(321),P(322),
* P(323),P(324),P(325),P(326),P(327),P(328),P(329),
* P(330),P(331),P(332),P(333),P(334),P(335),P(336)/
*  0.82952219463740140018E 00,0.72849479805538070639E-02,
*  0.81462878765513741344E 00,0.76079896657190565832E-02,
*  0.79909229096084140180E 00,0.79279493342948491103E-02,
*  0.78291939411828301639E 00,0.82443037630328680306E-02,
*  0.76611781930376009072E 00,0.85565435613076896192E-02,
*  0.74869629361693660282E 00,0.88641732094824942641E-02,
*  0.73066452124218126133E 00,0.91667111635607884067E-02,
*  0.71203315536225203459E 00,0.94636899938300652943E-02,
*  0.69281376977911470289E 00,0.97546565363174114611E-02,
*  0.67301883023041847920E 00,0.10039172044056840798E-01,
*  0.65266166541001749610E 00,0.10316812330947621682E-01,
*  0.63175643771119423041E 00,0.10587167904885197931E-01,
*  0.61031811371518640016E 00,0.10849844089337314099E-01,
*  0.58836243444766254143E 00,0.11104461134006926537E-01/
 DATA
* P(337),P(338),P(339),P(340),P(341),P(342),P(343),
* P(344),P(345),P(346),P(347),P(348),P(349),P(350),
* P(351),P(352),P(353),P(354),P(355),P(356),P(357),
* P(358),P(359),P(360),P(361),P(362),P(363),P(364)/
*  0.56590588542365442262E 00,0.11350654315980596602E-01,
*  0.54296566649831149049E 00,0.11588074033043952568E-01,
*  0.51955966153745702199E 00,0.11816385890830235763E-01,
*  0.49570640791876146017E 00,0.12035270785279562630E-01,
*  0.47142506587165887693E 00,0.12244424981611985899E-01,
*  0.44673538766202847374E 00,0.12443560190714035263E-01,
*  0.42165768662616330006E 00,0.12632403643542078765E-01,
*  0.39621280605761593918E 00,0.12810698163877361967E-01,
*  0.37042208795007823014E 00,0.12978202239537399286E-01,
```

```
     * 0.344307341599438O2278E 00,0.131346900919601528 36E-01,
     * 0.317890812068476683 18E 00,0.132799517439305306 50E-01,
     * 0.291195148518246681 96E 00,0.134137930851100985 13E-01,
     * 0.264243372410926761 94E 00,0.135360359349562136 14E-01,
     * 0.237058845589829727 21E 00,0.136465181025712914 28E-01/
      DATA
     * P(365),P(366),P(367),P(368),P(369),P(370),P(371),
     * P(372),P(373),P(374),P(375),P(376),P(377),P(378),
     * P(379),P(380),P(381)/
     * 0.209665238243181194 77E 00,0.137450934430018966 32E-01,
     * 0.182086496759252198 25E 00,0.138316319095064286 76E-01,
     * 0.154346811481378108 69E 00,0.139060196013254612 64E-01,
     * 0.126470584372301966 85E 00,0.139681588065169385 16E-01,
     * 0.984823965981192020 90E-01,0.140179680394566088 10E-01,
     * 0.704069760428551790 63E-01,0.140553820726499642 77E-01,
     * 0.422691647653636032 12E-01,0.140803519625536613 25E-01,
     * 0.140938864107824626 14E-01,0.140928450691604083 55E-01,
     * 0.140944070900961793 47E-01/
      ICHECK = 0
C CHECK FØR TRIVIAL CASE.
      IF (A.EQ.B) GØ TØ 70
C SCALE FACTØRS.
      SUM = (B+A)/2.0
      DIFF = (B-A)/2.0
C 1-PØINT GAUSS
      FZERØ = F(SUM)
      RESULT(1) = 2.0*FZERØ*DIFF
      I = 0
      IØLD = 0
      INEW = 1
      K = 2
      ACUM = 0.0
      GØ TØ 30
   10 IF (K.EQ.8) GØ TØ 50
      K = K + 1
      ACUM = 0.0
C CØNTRIBUTIØN FRØM FUNCTIØN VALUES ALREADY CØMPUTED.
      DØ 20 J=1,IØLD
        I = I + 1
        ACUM = ACUM + P(I)*FUNCT(J)
   20 CØNTINUE
C CØNTRIBUTIØN FRØM NEW FUNCTIØN VALUES.
   30 IØLD = IØLD + INEW
      DØ 40 J=INEW,IØLD
        I = I + 1
        X = P(I)*DIFF
        FUNCT(J) = F(SUM+X) + F(SUM-X)
        I = I + 1
        ACUM = ACUM + P(I)*FUNCT(J)
   40 CØNTINUE
      INEW = IØLD + 1
      I = I + 1
      RESULT(K) = (ACUM+P(I)*FZERØ)*DIFF
C CHECK FØR CØNVERGENCE.
      IF (ABS(RESULT(K)-RESULT(K-1))-EPSIL*ABS(RESULT(K))) 60,
     * 60, 10
C CØNVERGENCE NØT ACHIEVED.
   50 ICHECK = 1
C NØRMAL TERMINATIØN.
   60 NPTS = INEW + IØLD
      RETURN
C TRIVIAL CASE
   70 K = 2
      RESULT(1) = 0.0
      RESULT(2) = 0.0
      NPTS = 0
      RETURN
      END

      FUNCTIØN QSUB(A, B, EPSIL, NPTS, ICHECK, RELERR, F)
C   THIS FUNCTIØN RØUTINE PERFØRMS AUTØMATIC INTEGRATIØN
C ØVER A FINITE INTERVAL USING THE BASIC INTEGRATIØN
C ALGØRITHM QUAD, TØGETHER WITH, IF NECESSARY, A NØN-
C ADAPTIVE SUBDIVISIØN PRØCESS.
C   THE CALL TAKES THE FØRM
C          QSUB(A,B,EPSIL,NPTS,ICHECK,RELERR,F)
C AND CAUSES F(X) TØ BE INTEGRATED ØVER (A,B) WITH RELATIVE
C ERRØR HØPEFULLY NØT EXCEEDING EPSIL.  SHØULD QUAD CØNVERGE
C (ICHECK=0) THEN QSUB WILL RETURN THE VALUE ØBTAINED BY IT
C ØTHERWISE SUBDIVISIØN WILL BE INVØKED AS A RESCUE
C ØPERATIØN IN A NØN-ADAPTIVE MANNER. THE ARGUMENT RELERR
C GIVES A CRUDE ESTIMATE ØF THE ACTUAL RELATIVE ERRØR
C ØBTAINED.
C   THE SUBDIVISIØN STRATEGY IS AS FØLLØWS
C LET THE INTERVAL (A,B) BE DIVIDED INTØ 2**N PANELS AT STEP
C N ØF THE SUBDIVISIØN PRØCESS.  QUAD IS APPLIED FIRST TØ
C THE SUBDIVIDED INTERVAL ØN WHICH QUAD LAST FAILED TØ
C CØNVERGE AND IF CØNVERGENCE IS NØW ACHIEVED THE REMAINING
C PANELS ARE INTEGRATED.  SHØULD A CØNVERGENCE FAILURE ØCCUR
C ØN ANY PANEL THE INTEGRATIØN AT THAT PØINT IS TERMINATED
C AND THE PRØCEDURE REPEATED WITH N INCREASED BY 1. THE
C STRATEGY INSURES THAT PØSSIBLY DELINQUENT INTERVALS ARE
C EXAMINED BEFØRE WØRK, WHICH LATER MIGHT HAVE TØ BE
C DISCARDED, IS INVESTED ØN WELL BEHAVED PANELS. THE
C PRØCESS IS CØMPLETE WHEN NØ CØNVERGENCE FAILURE ØCCURS ØN
C ANY PANEL AND THE SUM ØF THE RESULTS ØBTAINED BY QUAD ØN
C EACH PANEL IS TAKEN AS THE VALUE ØF THE INTEGRAL.
C   THE PRØCESS IS VERY CAUTIØUS IN THAT THE SUBDIVISIØN ØF
C THE INTERVAL (A,B) IS UNIFØRM, THE FINENESS ØF WHICH IS
C CØNTRØLLED BY THE SUCCESS ØF QUAD.  IN THIS WAY IT IS
C RATHER DIFFICULT FØR A SPURIØUS CØNVERGENCE TØ SLIP
C THRØUGH.
C   THE CØNVERGENCE CRITERIØN ØF QUAD IS SLIGHTLY RELAXED
C IN THAT A PANEL IS DEEMED TØ HAVE BEEN SUCCESSFULLY
C INTEGRATED IF EITHER QUAD CØNVERGES ØR THE ESTIMATED
C ABSØLUTE ERRØR CØMMITTED ØN THIS PANEL DØES NØT EXCEED
C EPSIL TIMES THE ESTIMATED ABSØLUTE VALUE ØF THE INTEGRAL
C ØVER (A,B).  THIS RELAXATIØN IS TØ TRY TØ TAKE ACCØUNT ØF
C A CØMMØN SITUATIØN WHERE ØNE PARTICULAR PANEL CAUSES
C SPECIAL DIFFICULTY, PERHAPS DUE TØ A SINGULARITY ØF SØME
C TYPE.  IN THIS CASE QUAD CØULD ØBTAIN NEARLY EXACT
C ANSWERS ØN ALL ØTHER PANELS AND SØ THE RELATIVE ERRØR FØR
C THE TØTAL INTEGRATIØN WØULD BE ALMØST ENTIRELY DUE TØ THE
C DELINQUENT PANEL.  WITHØUT THIS CØNDITIØN THE CØMPUTATIØN
C MIGHT CØNTINUE DESPITE THE REQUESTED RELATIVE ERRØR BEING
C ACHIEVED.
C   THE ØUTCØME ØF THE INTEGRATIØN IS INDICATED BY ICHECK.
C   ICHECK=0  -  CØNVERGENCE ØBTAINED WITHØUT INVØKING
C                SUBDIVISIØN.  THIS CØRRESPØNDS TØ THE
C                DIRECT USE ØF QUAD.
C   ICHECK=1  -  RESULT ØBTAINED AFTER INVØKING SUBDIVISIØN.
C   ICHECK=2  -  AS FØR ICHECK=1 BUT AT SØME PØINT THE
C                RELAXED CØNVERGENCE CRITERIØN WAS USED.
C                THE RISK ØF UNDERESTIMATING THE RELATIVE
C                ERRØR WILL BE INCREASED.  IF NECESSARY,
C                CØNFIDENCE MAY BE RESTØRED BY CHECKING
C                EPSIL AND RELERR FØR A SERIØUS DISCREPANCY.
C   ICHECK NEGATIVE
C                IF DURING THE SUBDIVISIØN PRØCESS THE
C                ALLØWED UPPER LIMIT ØN THE NUMBER ØF PANELS
C                THAT MAY BE GENERATED (PRESENTLY 4096) IS
C                REACHED A RESULT IS ØBTAINED WHICH MAY BE
C                UNRELIABLE BY CØNTINUING THE INTEGRATIØN
C                WITHØUT FURTHER SUBDIVISIØN IGNØRING
C                CØNVERGENCE FAILURES. THIS ØCCURRENCE IS
C                FLAGGED BY RETURNING ICHECK WITH NEGATIVE
C                SIGN.
C  THE RELIABILITY ØF THE ALGØRITHM WILL DECREASE FØR LARGE
C  VALUES ØF EPSIL.  IT IS RECØMMENDED THAT EPSIL SHØULD
C  GENERALLY BE LESS THAN ABØUT 0.001.
      DIMENSIØN RESULT(8)
      INTEGER BAD, ØUT
      LØGICAL RHS
      EXTERNAL F
      DATA NMAX/4096/
      CALL QUAD(A, B, RESULT, K, EPSIL, NPTS, ICHECK, F)
      QSUB = RESULT(K)
      RELERR = 0.0
      IF (QSUB.NE.0.0) RELERR =
     * ABS((RESULT(K)-RESULT(K-1))/QSUB)
C CHECK IF SUBDIVISIØN IS NEEDED.
      IF (ICHECK.EQ.0) RETURN
C SUBDIVIDE
      ESTIM = ABS(QSUB*EPSIL)
      IC = 1
      RHS = .FALSE.
      N = 1
      H = B - A
      BAD = 1
   10 QSUB = 0.0
      RELERR = 0.0
      H = H*0.5
      N = N + N
C INTERVAL (A,B) DIVIDED INTØ N EQUAL SUBINTERVALS.
C INTEGRATE ØVER SUBINTERVALS BAD TØ (BAD+1) WHERE TRØUBLE
C HAS ØCCURRED.
      M1 = BAD
      M2 = BAD + 1
      ØUT = 1
      GØ TØ 50
C INTEGRATE ØVER SUBINTERVALS 1 TØ (BAD-1)
   20 M1 = 1
      M2 = BAD - 1
      RHS = .FALSE.
      ØUT = 2
      GØ TØ 50
C INTEGRATE ØVER SUBINTERVALS (BAD+2) TØ N.
   30 M1 = BAD + 2
      M2 = N
      ØUT = 3
      GØ TØ 50
C SUBDIVISIØN RESULT
   40 ICHECK = IC
      RELERR = RELERR/ABS(QSUB)
      RETURN
C INTEGRATE ØVER SUBINTERVALS M1 TØ M2.
   50 IF (M1.GT.M2) GØ TØ 90
      DØ 80 JJ=M1,M2
        J = JJ
C EXAMINE FIRST THE LEFT ØR RIGHT HALF ØF THE SUBDIVIDED
C TRØUBLESØME INTERVAL DEPENDING ØN THE ØBSERVED TREND.
        IF (RHS) J = M2 + M1 - JJ
        ALPHA = A + H*(J-1)
        BETA = ALPHA + H
        CALL QUAD(ALPHA, BETA, RESULT, M, EPSIL, NF, ICHECK, F)
        CØMP = ABS(RESULT(M)-RESULT(M-1))
        NPTS = NPTS + NF
        IF (ICHECK.NE.1) GØ TØ 70
        IF (CØMP.LE.ESTIM) GØ TØ 100
C SUBINTERVAL J HAS CAUSED TRØUBLE.
C CHECK IF FURTHER SUBDIVISIØN SHØULD BE CARRIED ØUT.
        IF (N.EQ.NMAX) GØ TØ 60
        BAD = 2*J - 1
        RHS = .FALSE.
        IF ((J-2*(J/2)).EQ.0) RHS = .TRUE.
        GØ TØ 10
   60   IC = -IABS(IC)
   70   QSUB = QSUB + RESULT(M)
   80 CØNTINUE
      RELERR = RELERR + CØMP
   90 GØ TØ (20,30,40), ØUT
C RELAXED CØNVERGENCE
  100 IC = ISIGN(2,IC)
      GØ TØ 70
      END

      FUNCTIØN QSUBA(A, B, EPSIL, NPTS, ICHECK, RELERR, F)
C   THIS FUNCTIØN RØUTINE PERFØRMS AUTØMATIC INTEGRATIØN
C ØVER A FINITE INTERVAL USING THE BASIC INTEGRATIØN
C ALGØRITHM QUAD TØGETHER WITH, IF NECESSARY AN ADAPTIVE
C SUBDIVISIØN PRØCESS.  IT IS GENERALLY MØRE EFFICIENT THAN
C THE NØN-ADAPTIVE ALGØRITHM QSUB BUT IS LIKELY TØ BE LESS
C RELIABLE(SEE CØMP.J.,14,189,1971).
C   THE CALL TAKES THE FØRM
C          QSUBA(A,B,EPSIL,NPTS,ICHECK,RELERR,F)
C AND CAUSES F(X) TØ BE INTEGRATED ØVER (A,B) WITH RELATIVE
C ERRØR HØPEFULLY NØT EXCEEDING EPSIL.  SHØULD QUAD CØNVERGE
C (ICHECK=0) THEN QSUBA WILL RETURN THE VALUE ØBTAINED BY IT
C ØTHERWISE SUBDIVISIØN WILL BE INVØKED AS A RESCUE
C ØPERATIØN IN AN ADAPTIVE MANNER. THE ARGUMENT RELERR GIVES
C A CRUDE ESTIMATE ØF THE ACTUAL RELATIVE ERRØR ØBTAINED.
```

```
C   THE SUBDIVISIØN STRATEGY IS AS FØLLØWS
C AT EACH STAGE ØF THE PRØCESS AN INTERVAL IS PRESENTED FØR
C SUBDIVISIØN (INITIALLY THIS WILL BE THE WHØLE INTERVAL
C (A,B)).  THE INTERVAL IS HALVED AND QUAD APPLIED TØ EACH
C SUBINTERVAL.  SHØULD QUAD FAIL ØN THE FIRST SUBINTERVAL
C THE SUBINTERVAL IS STACKED FØR FUTURE SUBDIVISIØN AND THE
C SECØND SUBINTERVAL IMMEDIATELY EXAMINED.  SHØULD QUAD FAIL
C ØN THE SECØND SUBINTERVAL THE SUBINTERVAL IS
C IMMEDIATELY SUBDIVIDED AND THE WHØLE PRØCESS REPEATED.
C EACH TIME A CØNVERGED RESULT IS ØBTAINED IT IS
C ACCUMULATED AS THE PARTIAL VALUE ØF THE INTEGRAL. WHEN
C QUAD CØNVERGES ØN BØTH SUBINTERVALS THE INTERVAL LAST
C STACKED IS CHØSEN NEXT FØR SUBDIVISIØN AND THE PRØCESS
C REPEATED.  A SUBINTERVAL IS NØT EXAMINED AGAIN ØNCE A
C CØNVERGED RESULT IS ØBTAINED FØR IT SØ THAT A SPURIØUS
C CØNVERGENCE IS MØRE LIKELY TØ SLIP THRØUGH THAN FØR THE
C NØN-ADAPTIVE ALGØRITHM QSUB.
C   THE CØNVERGENCE CRITERIØN ØF QUAD IS SLIGHTLY RELAXED
C IN THAT A PANEL IS DEEMED TØ HAVE BEEN SUCCESSFULLY
C INTEGRATED IF EITHER QUAD CØNVERGES ØR THE ESTIMATED
C ABSØLUTE ERRØR CØMMITTED ØN THIS PANEL DØES NØT EXCEED
C EPSIL TIMES THE ESTIMATED ABSØLUTE VALUE ØF THE INTEGRAL
C ØVER (A,B).  THIS RELAXATIØN IS TØ TRY TØ TAKE ACCØUNT ØF
C A CØMMØN SITUATIØN WHERE ØNE PARTICULAR PANEL CAUSES
C SPECIAL DIFFICULTY, PERHAPS DUE TØ A SINGULARITY ØF SØME
C TYPE.  IN THIS CASE QUAD CØULD ØBTAIN NEARLY EXACT
C ANSWERS ØN ALL ØTHER PANELS AND SØ THE RELATIVE ERRØR FØR
C THE TØTAL INTEGRATIØN WØULD BE ALMØST ENTIRELY DUE TØ THE
C DELINQUENT PANEL.  WITHØUT THIS CØNDITIØN THE CØMPUTATIØN
C MIGHT CØNTINUE DESPITE THE REQUESTED RELATIVE ERRØR BEING
C ACHIEVED.
C   THE ØUTCØME ØF THE INTEGRATIØN IS INDICATED BY ICHECK.
C   ICHECK=0  -  CØNVERGENCE ØBTAINED WITHØUT INVØKING SUB-
C                DIVISIØN.  THIS WØULD CØRRESPØND TØ THE
C                DIRECT USE ØF QUAD.
C   ICHECK=1  -  RESULT ØBTAINED AFTER INVØKING SUBDIVISIØN.
C   ICHECK=2  -  AS FØR ICHECKB1 BUT AT SØME PØINT THE
C                RELAXED CØNVERGENCE CRITERIØN WAS USED.
C                THE RISK ØF UNDERESTIMATING THE RELATIVE
C                ERRØR WILL BE INCREASED.  IF NECESSARY,
C                CØNFIDENCE MAY BE RESTØRED BY CHECKING
C                EPSIL AND RELERR FØR A SERIØUS DISCREPANCY.
C   ICHECK NEGATIVE
C                IF DURING THE SUBDIVISIØN PRØCESS THE STACK
C                ØF DELINQUENT INTERVALS BECØMES FULL (IT IS
C                PRESENTLY SET TØ HØLD AT MØST 100 NUMBERS)
C                A RESULT IS ØBTAINED BY CØNTINUING THE
C                INTEGRATIØN IGNØRING CØNVERGENCE FAILURES
C                WHICH CANNØT BE ACCØMMØDATED ØN THE STACK.
C                THIS ØCCURRENCE IS FLAGGED BY RETURNING
C                ICHECK WITH NEGATIVE SIGN.
C  THE RELIABILITY ØF THE ALGØRITHM WILL DECREASE FØR LARGE
C  VALUES ØF EPSIL.  IT IS RECØMMENDED THAT EPSIL SHØULD
C  GENERALLY BE LESS THAN ABØUT 0.001.
      DIMENSIØN RESULT(8), STACK(100)
      EXTERNAL F
      DATA ISMAX/100/
      CALL QUAD(A, B, RESULT, K, EPSIL, NPTS, ICHECK, F)
      QSUBA = RESULT(K)
      RELERR = 0.0
      IF (QSUBA.NE.0.0)
     * RELERR = ABS((RESULT(K)-RESULT(K-1))/QSUBA)
C CHECK IF SUBDIVISIØN IS NEEDED
      IF (ICHECK.EQ.0) RETURN
C SUBDIVIDE
      ESTIM = ABS(QSUBA*EPSIL)
      RELERR = 0.0
      QSUBA = 0.0
      IS = 1
      IC = 1
      SUB1 = A
      SUB3 = B
   10 SUB2 = (SUB1+SUB3)*0.5
      CALL QUAD(SUB1, SUB2, RESULT, K, EPSIL, NF, ICHECK, F)
      NPTS = NPTS + NF
      CØMP = ABS(RESULT(K)-RESULT(K-1))
      IF (ICHECK.EQ.0) GØ TØ 30
      IF (CØMP.LE.ESTIM) GØ TØ 70
      IF (IS.GE.ISMAX) GØ TØ 20
C STACK SUBINTERVAL (SUB1,SUB2) FØR FUTURE EXAMINATIØN
      STACK(IS) = SUB1
      IS = IS + 1
      STACK(IS) = SUB2
      IS = IS + 1
      GØ TØ 40
   20 IC = -IABS(IC)
   30 QSUBA = QSUBA + RESULT(K)
      RELERR = RELERR + CØMP
   40 CALL QUAD(SUB2, SUB3, RESULT, K, EPSIL, NF, ICHECK, F)
      NPTS = NPTS + NF
      CØMP = ABS(RESULT(K)-RESULT(K-1))
      IF (ICHECK.EQ.0) GØ TØ 50
      IF (CØMP.LE.ESTIM) GØ TØ 80
C SUBDIVIDE INTERVAL (SUB2,SUB3)
      SUB1 = SUB2
      GØ TØ 10
   50 QSUBA = QSUBA + RESULT(K)
      RELERR = RELERR + CØMP
      IF (IS.EQ.1) GØ TØ 60
C SUBDIVIDE THE DELINQUENT INTERVAL LAST STACKED
      IS = IS - 1
      SUB3 = STACK(IS)
      IS = IS - 1
      SUB1 = STACK(IS)
      GØ TØ 10
C SUBDIVISIØN RESULT
   60 ICHECK = IC
      RELERR = RELERR/ABS(QSUBA)
      RETURN
C RELAXED CØNVERGENCE
   70 IC = ISIGN(2,IC)
      GØ TØ 30
   80 IC = ISIGN(2,IC)
      GØ TØ 50
      END
```

Algorithm 469 Arithmetic Over a Finite Field [A1]

C. Lam* and J. McKay† [Recd. 8 Oct. 1971]
* Department of Mathematics, Caltech University, Pasadena, CA 91101 † School of Computer Science, McGill University, P.O. Box 6070, Montreal 101, P.Q. Canada

Key Words and Phrases: algebra; CR Categories: 5.19
Language: Algol

Description

The rational operations of arithmetic over the finite field F_q, of $q = p^n (n \geq 1)$ elements, may be performed with this algorithm.

On entry $a[i]$ contains $a_i \in F_p$ with $0 \leq a_i < p$, $i = 0, \ldots, n-1$, and $x \in F_q$ satisfies the primitive irreducible polynomial $P(x) = x^n + \sum_{k=0}^{n-1} a_k x^k$. fq produces e_i in $e[i]$, $i = -1, \ldots, q-2$, where $1 + x^i = x^{e_i}$ with the convention that -1 represents $*$ and $x* = 0$. During execution the range of the a_i is altered to $-p < a_i \leq 0$, $i = 0 \ldots n-1$. The storage used is $2q + n + 6$ locations including the final array e.

With appropriate conventions for $*$, multiplication and division are trivial, and addition and subtraction are given by $x^a + x^b = x^a(1 + x^{b-a})$ for $a \leq b$ and $x^a - x^b = x^a + x^{\frac{1}{2}(q-1)} x^b$ when $p \neq 2$. For small values of q, it is suggested that addition and multiplication tables be generated by this algorithm. A description of the method and its generalization to a multi-step process when n is composite is in [2]. A list of primitive irreducible polynomials is given in [1]. Further useful information (especially for $p = 2$) is to be found in [3].

References

1. Alanen, A.J., and Knuth, D.E. Tables of finite fields. *Sankhyā-(A) 26* (1964), 305–328.
2. Cannon, J.J. Ph.D. Th., 1967 U. of Sydney, Sydney, Australia.
3. Conway, J.H., and Guy, M.J.T. Information on finite fields. In *Computers in Mathematical Research.* North-Holland Pub. Co., Amsterdam, 1967.

Algorithm

```
procedure fq(p, n, a, e);
  integer p, n; integer array a, e
begin
  integer array c[0:n−1], f[0:p ↑ n−1]; integer q, i, j, d, s, w;
    q := p ↑ n;
  for i := 0 step 1 until n − 1 do if a[i] ≠ 0 then a[i] := a[i] − p;
  for i := 1 step 1 until n − 1 do c[i] := 0;
    c[0] := 1;  f[1] := 0;  f[0] := −1:
  for i := 1 step 1 until q − 2 do
    begin
    d := e[n − 1];  s := 0;
    for j := n − 1 step −1 until 1 do
      begin
      w := c[i−1] − d × a[j];  w := w − w ÷ p × p;
      c[j] := w;  s := p × s + w
      end;
    w := −d × a[0];  w := w − w ÷ p × p;  c[0] := w;
      f[p × s + w] := i
    end;
for i := q step −p until p do
    begin
    e[f[i−1]] := f[i−p];
for j := i − p step 1 until i − 2 do e[f[j]] := f[j+1]
    end
end
```