An adaptive algorithm for numerical integration over an n-dimensional cube

ALGORITHM 006

Paul van Dooren and Luc de Ridder

ABSTRACT

A program is presented for automatic numerical integration over the n-dimensional cube. Interesting advantages of this program are the strong adaptivity of the algorithm combined with the use of a good basic integration rule. This is shown by some comparative tests with other programs.

1. INTRODUCTION

It is a wellknown fact that multiple integration by means of repeated integration or by computing the integral by a product formula is time consuming because the number of function evaluations grows exponentially with n.
In recent years more attention has been given to other methods.
Specifically for automatic numerical integration studies have been done not only on efficient integration formulas but also on algorithms.
In this article an automatic integrator for the n-cube is presented. It uses an adaptive and global [1] strategy based on a non-product formula. A comparison was done with four other programs, using 167 test-integrals of dimension 2, 3, 4, 5 and 6. The major conclusions are that product formulas are not suited to the problem and that an adaptive algorithm is almost a must.

2. THE BASIC RULE

The algorithm needs the computation of the integral

$$I = \int_{a_n}^{b_n} \dots \int_{a_1}^{b_1} f(x_1, x_2, \dots x_n)\, dx_1\, dx_2 \dots dx_n \qquad (1)$$

as well as an error estimate E. Therefore we approximate I by two integration rules I_7 and I_5, respectively of degree of exactness seven and five.

$$I_7 = A_7 f(0, 0 \dots 0) + B_7 \sum_{FS} f(\lambda_1, 0, \dots 0) + C_7 \sum_{FS} f(\lambda_2, 0, 0 \dots 0) + D_7 \sum_{FS} f(\mu, \mu, 0 \dots 0) + E_7 \sum_{FS} f(\nu, \nu, \nu, 0 \dots 0) \qquad (2)$$

$$I_5 = A_5 f(0, 0 \dots 0) + B_5 \sum_{FS} f(r, 0 \dots 0) + D_5 \sum_{FS} f(r, r, 0 \dots 0) \qquad (3)$$

$\sum_{FS}$ means 'fully symmetric' : summation over all permutations of the coordinates, sign changes included.
The formulas I_7 and I_5 are derived for the n-cube C_n with vertices $(\pm 1, \pm 1, \dots\dots, \pm 1)$. By a linear transformation they can also be applied for any region affine to C_n, e.g. the region of integral (1). The result I_7 is used to estimate $I : I \cong I_7$. $E = |I_7 - I_5|$ is considered as an estimate of the error $|I - I_7|$.
Phillips [2] and Stroud [3] give a solution for the parameters $A_7, B_7, \dots, D_5, \lambda_1, \lambda_2, \mu, \nu, r$, so that formula (2) has degree seven and formula (3) degree five. Since the solution for I_7 is not unique, we tried to force r to be equal to λ_1 or μ so that both cubature formulas have some common points. The best attempt was $r = \lambda_1$ for $n = 3$ and $n = 4$ and $r = \mu$ for all other values of n. This reduces the total number of function evaluations for both formulas I_7 and I_5 to $(4n^3 - 6n^2 + 20n + 3)/3$ for all n except for $n = 3$ (45) and $n = 4$ (97 evaluations). A product formula derived from a one dimensional k-point formula would need k^n integration points. For $n = 6$ and for a three-point Gauss formula (which is of degree five) we would need 729 points against 257 for I_5 and I_7 together.

3. ALGORITHM

We use an adaptive and global strategy which already proved to give very good results in one dimension [4], [5]. At each step the total integration area is partitioned

P. van Dooren and L. de Ridder, Applied Mathematics and Programming Division, Katholieke Universiteit Leuven, Celestijnenlaan 200B, B-3030 Heverlee, Belgium. (The present address of L. de Ridder is : Siemens N.V., FTE, B8020 Oostkamp, Belgium

into subregions B_i, all n-cubes.
To each of them the basic rule has been applied to obtain an integral estimate Int_i and an error estimate E_i. Obviously $HALF = \sum_i Int_i$ and $Eps = \sum_i E_i$ are approximations respectively for the integral and the error over the total integration area. The following step will consist in finding the subregion B_j with the largest estimate E_j of the whole set $\{E_i\}$, and subdividing it in smaller n-cubes to improve the global result. The iteration ends when Eps is smaller than the requested accuracy. To improve adaptivity the region B_j is not subdivided into 2^n regions (by dividing each dimension by two), but only into two regions. This seems to give the best results on condition that a good choice is made for the dimension that is halved. Therefore the fourth divided differences Dif_k in each direction are compared with each other and the dimension k^* with the largest value Dif_{k^*} - which is a measure for the difficulty of the integration in that dimension - will be selected. The subdivision policy of the algorithm is illustrated in the following figures.

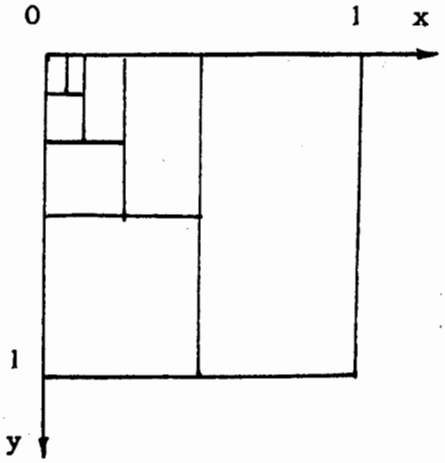

Fig. 1. $f_1 = (x+y)^{-0.5}$

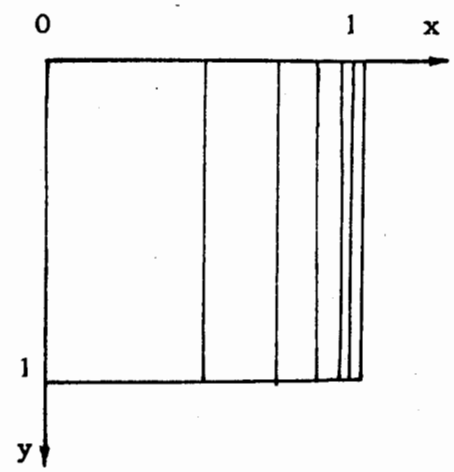

Fig. 2. $f_2 = (1-x)^{-0.5}$

Fig. 1 shows the subdivision pattern of the two dimensional integration area for a function with singularity in the upper left corner; fig. 2 for a function with a boundary singularity at $x = 1$.
For this subdivision strategy, the fourth differences have been chosen because all the information necessary to calculate them is at hand during the computation of I_7. The five points used in direction i are :

$$x_i = -\lambda_2,\ -\lambda_1,\ 0,\ \lambda_1,\ \lambda_2 \text{ and } x_j = 0 \text{ for } j \neq i$$

Since these points are not equidistant but symmetric, the fourth divided differences are up to a constant factor given by

$$Dif_i = [f(-\lambda_2) - 2f(0) + f(\lambda_2)] - \frac{\lambda_1^2}{\lambda_2^2}[f(-\lambda_1) - 2f(0) + f(\lambda_1)]$$

Note that no additional function evaluations are required.

4. THE PROGRAM

The program is written as a double precision basic FORTRAN FUNCTION subroutine.
It is possible to request an absolute accuracy as well as a relative one. As proposed by de Boor [6], both are combined in

$$Epsmax = \max\{Epsabs, Epsrel.\ |HALF|\}$$

The algorithm ends when Eps < Epsmax, when rounding errors occur or when the maximum number of function evaluations is reached.
The calling sequence is

I = HALF (AA, BB, N, F, EPSABS, EPSREL, MAX, EPS, NUMBER, IERR)

with input parameters :

AA(N), BB(N)	are two vectors of length N with the lower- and upperbounds of the n-cube
N	is the dimension of the space ($2 \leq N \leq 6$)
F	is the integrand (declared external in the calling program)
EPSABS	is the requested absolute accuracy
EPSREL	is the requested relative accuracy
MAX	is the maximum number of evaluations allowed

and output parameters :

EPS	is an estimation of the absolute error
NUMBER	is the number of evaluations required for the integration
IERR	is an error code = 0 when Eps < Epsmax = 1 when the requested accuracy cannot be obtained because MAX is too small = 2 when the requested accuracy cannot

be obtained even when MAX would be increased because of memory shortage
= 3 when rounding errors occur
= 4 if N < 2 or N > 6

HALF, F, AA, BB are declared double precision.
Subroutines called by HALF : CN7152.
No common zones are used.

5. RESULTS AND CONCLUSIONS

We have compared HALF with four other available programs :
QB01A [7], Y1MITG [8], ROMBND [9] and FNTGRL [10], [11], all written in FORTRAN. The first two use repeated integration; ROMBND is based on Romberg integration; FNTGRL is an adaptive program that uses extrapolation. Only two of them, HALF and FNTGRL, give satisfying results for large dimensions ($n \geqslant 4$). The other three, which are based on product formulas, are only competitive for small dimensions ($n = 2, 3$). The principal feature of HALF and FNTGRL that favours them in regard to the other three programs is adaptivity. Both programs subdivide the integration area locally where some difficulty is met approaching the integral. By looking not only for the localization of the difficulty but also for the dimension in which it occurs, HALF is even more adaptive than FNTGRL.
The comparison of these five programs is done with 167 test functions. We give some results to compare the two best programs.

Simple integrands

1. $$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}\int_{0}^{\frac{\pi}{2}}\int_{0}^{1}\int_{0}^{2} x_1 x_2^2 \sin(x_3)\,(4+x_4+x_5+x_6)^{-1}\,dx_1\,dx_2\,dx_3\,dx_4\,dx_5\,dx_6$$
$$= 0.1434761888397263D\ 01$$

2. $$\int_{0}^{2}\int_{0}^{1}\int_{0}^{1}\int_{0}^{1} x_3^2 x_4 \exp(x_3 x_4)\,(x_1+x_2+1)^{-2}\,dx_1\,dx_2\,dx_3\,dx_4$$
$$= 0.5753641449035616D\ 00$$

3. $$\int_{0}^{1}\int_{0}^{1}\int_{0}^{1} 8\,(1+2\,(x+y+z))^{-1}\,dx\,dy\,dz$$
$$= 0.2152142832595894D\ 01$$

Oscillating integrands

4. $$\int_{0}^{\frac{\pi}{2}}\int_{0}^{\pi}\int_{0}^{\pi}\int_{0}^{\pi}\int_{0}^{\pi} \cos(x_1+x_2+x_3+x_4+x_5)\,dx_1\,dx_2\,dx_3\,dx_4\,dx_5 = 0.1600000000000000D\ 02$$

5. $$\int_{0}^{1}\int_{0}^{1}\int_{0}^{1}\int_{0}^{1} \sin(10\,x_1)\,dx_1\,dx_2\,dx_3\,dx_4$$
$$= 0.1839071529076452D\ 00$$

6. $$\int_{0}^{3\pi}\int_{0}^{3\pi} \cos(x+y)\,dx\,dy$$
$$= -0.4000000000000000D\ 01$$

Integrands with a peak

7. $$\int_{0}^{1}\int_{0}^{1}\int_{0}^{1} (x+y+z)^{-2}\,dx\,dy\,dz$$
$$= 0.8630462173553432D\ 00$$

8. $$\int_{0}^{1}\int_{0}^{1} 605\,y\,[(1+120\,(1-y))\,((1+120(1-y))^2+25x^2y^2)]^{-1}\,dx\,dy$$
$$= 0.1047591113142868D\ 01$$

9. $$\int_{0}^{1}\int_{0}^{1} [(x^2+.0001)((y+.25)^2+.0001)]^{-1}dxdy$$
$$= 0.4991249442241215D\ 03$$

Integrand having discontinuous derivative

10. $$\int_{0}^{1}\int_{0}^{1} \exp(|x+y-1|)\,dx\,dy =$$
$$= 0.1436563656918090D\ 01$$

MAX is the maximum number of function evaluations allowed

n	2	3	4	5	6
MAX	10 000	10 000	30 000	30 000	40 000

NUM is the number of function evaluations performed by the programs
ERROR is the actual relative error of the result.

Requested relative accuracy 1. E - 2

	FNTGRL		HALF	
	ERROR	NUM	ERROR	NUM
1	0.88E-3	411	0.50E-4	257
2	0.16E-2	735	0.32E-3	873
3	0.30E-3	81	0.89E-3	45
4	0.19E-2	3071	0.33E-3	3171
5	0.95E-5	3087	0.13E-6	873
6	0.11E-2	247	0.12E-2	289
7	0.22E-1	513	0.49E-2	3015
8	0.12E-2	1183	0.14E-3	357
9	0.61E-4	1209	0.20 E-3	391
10	0.30E-2	39	0.40E-2	153

Requested relative accuracy 1.E-3

	FNTGRL		HALF	
	ERROR	NUM	ERROR	NUM
1	0.31E-3	4521	0.69E-4	2313
2	0.97E-4	6419	0.84E-4	1843
3	0.14E-3	135	0.26E-4	405
4	0.84E-4	15687	0.49E-5	25217
5	0.95E-5	3087	0.13E-6	873
6	0.11E-2	247	0.85E-4	493
7	0.20E-1	8397	0.31E-3	6885
8	0.69E-3	2925	0.56E-4	391
9	0.68E-5	3757	0.12E-3	527
10	0.78E-2	91	0.75E-3	391

Requested relative accuracy 1.E-4

	FNTGRL		HALF	
	ERROR	NUM	ERROR	NUM
1	0.73E-4	24249	0.26E-5	7967
2	0.32E-4	18081	0.33E-5	9215
3	0.86E-5	891	0.13E-4	495
4	0.29E-5	28967	0.24E-5	29747
5	0.30E-4	25039	0.17E-6	1261
6	0.15E-5	871	0.25E-5	1003
7	0.20E-1	8397	0.89E-4	9945
8	0.68E-3	4615	0.22E-4	459
9	0.18E-4	6227	0.79E-5	901
10	0.21E-2	247	0.80E-4	1343

When a program cannot reach the requested accuracy, the returned results are underlined.

Alle computations were done on a IBM 370/158. As expected, HALF detects very well the difficulties of the integrands with peak or singularity (7, 8 and 9) and especially the integrands that vary strongly in only one dimension (5).

ACKNOWLEDGEMENTS

The authors are greatly indebted to R. Piessens for drawing their attention to the problem and for his help and suggestions.
This research was awarded the 'IBM Belgium Reward for Informatics 1974'. It was supported by the 'Fonds voor Kollektief Fundamenteel Onderzoek' (Belgium) under grant no 10.174.

REFERENCES

1. Malcolm M. A. and Simpson R. B., "Local versus global strategies for adaptive quadrature", ACM Trans. on Math. Softw., Vol. 1, No 2, pp. 129-146, (1975).

2. Philips G. M., "Numerical integration over an N-dimensional rectangular region", Comp. J., 10, pp. 297-299, (1967).

3. Stroud A. H., "Approximate calculation of multiple integrals", Prentice Hall, Englewood Cliffs, New Jersey, 1971.

4. Piessens R., "An algorithm for automatic integration", Angewandte Informatik, pp. 399-401, (1973).

5. Piessens R., "A quadrature routine with round-off error guard", Rep. TW 17, Appl. Math. and Progr. Div., Kath. Univ. of Leuven, Belgium, (1974).

6. de Boor C., "On writing an automatic integration algorithm", Mathematical Software, J. Rice, Ed., Academic Press, New York, pp. 201-209, 1971.

7. Hopgood F. R., "SUBROUTINE QBO1A", Harwell Subroutine Library, Compiled by M. J. Hopper, Theoretical Physics Division, Atomic Energy Research Establishment, Harwell Berkshire.

8. Schotsmans L., "SCK-CEN SUBROUTINE Y1MITG note FO 4/4" (621-75/68-22-18.01.68), Nuclear Center, Mol, Belgium, (1968).

9. Fowler R. H., "SUBROUTINE ROMBND", Computing Technology Center, Numerical Analysis Library, Union Carbide Corporation, Oak Ridge, Tennessee, pp. 87-92, (1970).

10. Genz A., "An adaptive multidimensional quadrature procedure", Comp. Phys. Comm., 4, pp. 11-15, (1972).

11. Genz A., "Some extrapolation methods for the numerical calculations of multidimensional integrals", Software for Numerical Mathematics, Academic Press, pp. 159-172, 1974.

```
      DOUBLE PRECISION FUNCTION HALF(AA,BB,N,F,EPSABS,EPSREL
     *                              ,MAX,EPS,NUMBER,IERR)
C
C    THIS SUBROUTINE CALCULATES THE INTEGRAL OF THE FUNCTION
C    F(X). X IS A VECTOR OF LENGTH N THAT REPRESENTS THE
C    VARIABLES OF THE FUNCTION F. THE INTEGRAL IS EVALUATED
C    OVER THE N-CUBE WITH LOWER-BOUNDS AA(N) AND UPPERBOUNDS
C    BB(N). IT TRIES TO MAKE EPS LESS THAN MAX(EPSABS,
C    EPSREL*ABS(RES))
C
      INTEGER C,I,ICOUN1,IEERST,IERR,IFOUT,IG,IKEY,ILAAT,
     *IVOLG,IVOOR,I2,I3,J,K,KEUS,KEY,KEY2,KIES,KM1,K1,K2,L,
     *LB,LK,LK2,L1,M,MAX,N,NU,NUM,NUMBER,NUM2,O,V
      REAL ACCUR,E,EIKEY,EPS,EPSABS,EPSAB2,EPSMAX,EPSN,
     *EPSO,EPSREL,EPSRE2,RESN,RESO,SOM1
      DOUBLE PRECISION AA,AS,AX,BB,DABS,INT,OB,OR,RES,SOM2,
     *VOL,VOLUME
      DIMENSION AA(6),AS(6),AX(6),BB(6),C(450),E(100),
     *IG(100),INT(100),IVOLG(100),IVOOR(100),KIES(100),
     *O(450),OB(6),OR(6),V(100)
      DATA ACCUR/1.E-14/
      EXTERNAL F
C
C*** ORGANISATION OF THE AVAILABLE STORAGE ***
C
      IF(N.LE.6.AND.N.GE.2) GO TO 10
      IERR=4
      RETURN
   10 IF(MAX.GT.10000*N) MAX=10000*N
      NUM=(((4*N-6)*N+20)*N+3)/3
      IF(N.EQ.3) NUM=45
      IF(N.EQ.4) NUM=97
      I2=(MAX-NUM)/(NUM*2)+6
      M=I2/2
      IF(M.GT.100) M=100
      IF((M-1)*N.GT.450) M=450/N+1
C
C*** CALCULATION OF THE TOTAL VOLUME ***
C    IF VOLUME IS ZERO THEN RETURN HALF=0.D0
C
      VOL=1.D0
      DO 30 I=1,N
        IF(AA(I).NE.BB(I)) GO TO 20
        HALF=0.D0
        EPS=0.
        NUMBER=0
        RETURN
   20   OB(I)=(AA(I)+BB(I))*0.5D0
        AS(I)=BB(I)-OB(I)
        VOL=VOL*AS(I)*2.D0
   30 CONTINUE
      VOL=DABS(VOL)
C
C*** FIRST STEP ***
C    THE BASIC RULE CN7152 IS APPLIED TO THE TOTAL REGION
C
```

```
      IERR=0
      KEUS=100
      NUMBER=NUM
      EPSMAX=AMAX1(EPSABS,ACCUR)
      CALL CN7152 (OB,AS,N,RES,EPS,F,VCL,KEUS,IFOUT,EPSMAX)
      HALF=RES
      EPSAB2=ABS(EPSABS)
      EPSRE2=ABS(EPSREL)
      IF(EPSRE2.GT.0.05) EPSRE2=0.05
      IF(EPSRE2.LT.ACCUR) EPSRE2=ACCUR
      RESN=DABS(HALF)
      IF(IFOUT.EQ.1) GO TO 40
      EPSMAX=AMIN1(AMAX1(EPSRE2*RESN,EPSAB2),AMAX1(0.05*RESN
     *,ACCUR))
      IF(EPS.LE.EPSMAX)RETURN
C
C*** INITIALISATION OF THE ALGORITHM ***
C
   40 VOLUME=VOL
      NUM2=NUM+NUM
      EPSO=EPS
      SOM2=0.D0
      SOM1=0.
      EPS=0.
      HALF=0.D0
      ICOUN1=0
      KEY2=0
      ILAAT=2
      IEERST=2
      I=3
      K=KEUS
      E(1)=-1.
      IVOLG(2)=1
      IVOOR(1)=2
      V(2)=0
      DO 50 L=1,N
        O(L)=0
        C(L)=1
   50 CONTINUE
      DO 160 I3=3,I2
C
C*** ITERATION ALGORITHM ***
C     THE INTERVAL WITH THE LARGEST ABSOLUTE ERROR E(IEERST)
C     IS DIVIDED IN TWO SUBREGIONS AND THE BASIC RULE IS
C     APPLIED TO BOTH OF THEM
C     A VECTOR IS MEMORIZED WITH ALL THE INTERVALS RANGED IN
C     A DECREASING SEQUENCE WITH A POINTER IEERST TO THE TOP
C
        V(IEERST)=V(IEERST)+1
        V(I)=V(IEERST)
        L1=(IEERST-2)*N
        LK=L1+K
        O(LK)=2*O(LK)+1
        C(LK)=2*C(LK)
        LB=(I-2)*N
        LK2=LB+K
```

```
      DO 60 K2=1,N
        LB=LB+1
        L1=L1+1
        O(LB)=O(L1)
        C(LB)=C(L1)
        AX(K2)=AS(K2)/FLOAT(C(LB))
        OR(K2)=OB(K2)+AX(K2)*FLOAT(O(LB))
   60 CONTINUE
      VOL=VOLUME/2.D0**V(I)
      O(LK2)=O(LK)-2
      CALL CN7152(OR,AX,N,INT(IEERST),E(IEERST),F,VOL,
     *            KEUS,IG(IEERST),EPSMAX)
      KIES(IEERST)=KEUS
      OR(K)=OR(K)-2.D0*AX(K)
      CALL CN7152(OR,AX,N,INT(I),E(I),F,VOL,KEUS,IG(I),
     *            EPSMAX)
      KIES(I)=KEUS
      IFOUT=IFOUT+IG(I)+IG(IEERST)
      NUMBER=NUMBER+NUM2
      EPSN=E(IEERST)+E(I)
      EPS=EPS+EPSN
      HALF=HALF+INT(IEERST)+INT(I)
C
C*** TEST FOR STOPPING ITERATION ***
C    THE RESULT IS RETURNED IF THE ACCURACY EPSMAX IS
C    OBTAINED, IF THE MAXIMUM NUMBER OF EVALUATIONS WILL BE
C    OVERPASSED OR IF ROUNDING ERRORS OCCUR.
C
      RESO=RESN
      RESN=DABS(HALF)
      IF(EPSN.LE.EPSO.OR.(IG(I)+IG(IEERST)).NE.0) GO TO 70
      IF(RESO.LT.RESN*1.0001.OR.RESO.GT.RESN*0.9999)
     * ICOUN1=ICOUN1+1
   70 IF(ICOUN1.GT.21) IERR=3
      EPSMAX=AMIN1(AMAX1(EPSRE2*RESN,EPSAB2),AMAX1(0.05*
     * RESN,ACCUR))
      IF(IERR.EQ.3) GO TO 90
      IF(NUMBER.LE.(MAX-NUM2).AND.SOM1.LE.EPSMAX) GO TO 80
      IERR=1
      IF(M.EQ.100) IERR=2
      GO TO 90
   80 IF(100*IFOUT.GT.15*MAX0((I3-6),0)) GO TO 110
      IF(EPS.GT.EPSMAX) GO TO 110
   90 EPS=SOM1
      K2=I
      IF(KEY2.GE.1) K2=M
      HALF=SOM2
      DO 100 J=2,K2
        EPS=EPS+E(J)
        HALF=HALF+INT(J)
  100 CONTINUE
      IF(IERR.GE.1) RETURN
      IF(100*IFOUT.GT.15*MAX0((I3-6),0)) GO TO 110
      IF(EPS.LE.EPSMAX) RETURN
```

```
C
C*** THE TWO NEW SUBREGIONS ARE RANGED IN THE SEQUENCE E ***
C
  110    IKEY=I
         IF(E(IEERST).GT.E(I)) IKEY=IEERST
         KEY=I+IEERST
         IEERST=IVOLG(IEERST)
         K1=IEERST
         KM1=1
         NU=0
  120    EIKEY=E(IKEY)
         DO 130 K2=1,100
           IF(EIKEY.GE.E(K1)) GO TO 140
           KM1=K1
           K1=IVOLG(K1)
  130    CONTINUE
  140    IVOLG(KM1)=IKEY
         IVOLG(IKEY)=K1
         IVOOR(K1)=IKEY
         IVOOR(IKEY)=KM1
         IF(NU.EQ.1) GO TO 150
         IF(KM1.EQ.1) IEERST=IKEY
         KM1=IKEY
         NU=1
         IKEY=KEY-IKEY
         GO TO 120
  150    EPSO=E(IEERST)
         EPS=EPS-EPSO
         HALF=HALF-INT(IEERST)
         K=KIES(IEERST)
         IFOUT=IFOUT-IG(IEERST)
         I=I+1
         IF(K1.EQ.1) ILAAT=IKEY
         IF(I3.EQ.M) KEY2=1
         IF(I3.EQ.I2-M) KEY2=2
         IF(KEY2.LT.1) GO TO 160
         I=ILAAT
         ILAAT=IVOOR(ILAAT)
         IVOOR(1)=ILAAT
         IVOLG(ILAAT)=1
         SOM1=SOM1+E(I)
         SOM2=SOM2+INT(I)
         IF(KEY2.NE.2) GO TO 160
         ILAAT=IVOOR(ILAAT)
         IVOOR(1)=ILAAT
         IVOLG(ILAAT)=1
  160 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CN7152(OR,AX,N,INT,E,F,VOL,KEUS,IG,EPSMAX)
C
C   THIS PROGRAM EVALUATES THE INTEGRAL OF THE FUNCTION F
C   OVER THE N-CUBE WITH ORIGIN OR(N) AND HALF INTERVAL
C   LENGTH AS(N). IT IS USED BY HALF AS BASIC RULE FOR
C   INTEGRATION AND CALCULATES ALSO THE DIMENSION IN WHICH
C   THE FUNCTION VARIATES THE MOST
C
      INTEGER I,IG,IM1,I1,I2,K,KEUS,KEY34,KEY8,KM2,L,N,NM1
      REAL EPSMAX,E,VOLUME
      DOUBLE PRECISION ACCUR,AX,C0,C1,C2,DABS,DIFI,DIFMAX,
     *DIFO,DMIN1,D0,D1,D2,D3,D4,EPS,EVAL3,EVAL5,E0,F,FUN1,
     *FUN2,INT,OR,R,RU,R1,R2,R3,R4,S,SOM1,SOM2,SOM3,SOM4,
     *SQ35,S1,S2,S21,T,U,V,VOL,Y,Z
      DIMENSION AX(6),D0(5),D1(5),D2(5),D3(5),D4(5),OR(6),
     *R2(6),R4(5),S(5),S2(5),Y(6)
      DATA R4/.960768922830527D0,4*.95D0/,S/0.D0,.640512615
     *2203485D0,.426401432711220 9D0,2*0.D0/,S2/.774596669241
     *4834D0,.975900729485332D0,.698430295769578 2D0,2*.7745
     *966692414834D0/,ACCUR/1.D-14/,        SQ35/.7745966692
     *414834D0/,R2/.349927106111882 6D0,.835167185090805D0,.
     *108617550621409 9D0,.489872430601854 2D0,.653284336022 94
     *89D0,.758D0/,D0/-.476543209876542 8D0,.343481416576218 0
     *D0,-.125718605664488 0D3,-.128551874558714 2D1,-.1726490
     *001750501D1/,D1/.478649237472766 7D-1,-.497639054241362
     *8D0,.124801229134794 5D0,.631854503879701 3D-1,.87997094
     *74426657D-1/,D2/.244110384894698 6D0,.385362203561993 2D
     *0,.164797906826299 1D2,.199659976982815 5D0,.32496714980
     *03278D0/,D3/.771604938271604 8D-1,.107275390625D0,-.308
     *1018518518519D0,-.514403292181069 9D-1,-.943070270233196
     *15D-1/,D4/0.D0,.5359375D-2,.398848379626629 6D-1,2*.214
     *3347050754458D-1/
      IF(KEUS.NE.100) GO TO 10
      KEUS=1
      KEY34=0
      IF(N.EQ.3.OR.N.EQ.4) KEY34=1
      NM1=N-1
      Z=N
      E0=1.D0-Z/1.8D0
      C0=((25.D0*Z-115.D0)*Z+162.D0)/162.D0
      C1=(70.D0-25.D0*Z)/162.D0
      C2=25.D0/324.D0
      R1=R4(NM1)
      R=R2(NM1)
      S1=S(NM1)
      S21=S2(NM1)
      RU=R*R/SQ35/SQ35
      VOLUME=VOL
   10 DO 20 I=1,N
         Y(I)=OR(I)
   20    CONTINUE
      SOM4=F(Y)
      SOM3=SOM4+SOM4
      SOM1=0.D0
      SOM2=0.D0
```

```
      DIFMAX=0.D0
      DIF0=0.D0
      DO 30 I=1,N
        T=AX(I)*SQ35
        Y(I)=OR(I)+T
        FUN1=F(Y)
        Y(I)=OR(I)-T
        FUN2=F(Y)
        SOM1=SOM1+FUN1+FUN2
        DIF0=DIF0+DABS(FUN1-FUN2)
        DIFI=SOM3-FUN1-FUN2
        T=AX(I)*R
        Y(I)=OR(I)+T
        FUN1=F(Y)
        Y(I)=OR(I)-T
        FUN2=F(Y)
        SOM2=SOM2+FUN1+FUN2
        Y(I)=OR(I)
        DIFI=DABS(SOM3-FUN1-FUN2-DIFI*RU)
        IF(DIFI.LE.DIFMAX) GO TO 30
        DIFMAX=DIFI
        KEUS=I
   30 CONTINUE
      IF(DIFMAX.LE.ACCUR) KEUS=KEUS+1
      IF(KEUS.GT.N) KEUS=1
      INT=D0(NM1)*SOM4+D2(NM1)*SOM2
      EVAL3=E0*SOM4+SOM1/3.6D0
      EVAL5=C0*SOM4+C1*SOM1
      IF(KEY34.EQ.1) GO TO 50
      SOM1=0.D0
      DO 40 I=1,N
        T=AX(I)*R1
        Y(I)=OR(I)+T
        SOM1=SOM1+F(Y)
        Y(I)=OR(I)-T
        SOM1=SOM1+F(Y)
        Y(I)=OR(I)
   40 CONTINUE
   50 INT=INT+D1(NM1)*SOM1
      R3=SQ35
      KEY8=0
      DO 90 I1=1,2
        SOM2=0.D0
        DO 80 I=2,N
          IM1=I-1
          U=AX(IM1)*R3
          DO 70 I2=1,2
            Y(IM1)=OR(IM1)+U
            DO 60 L=I,N
              T=AX(L)*R3
              Y(L)=OR(L)+T
              SOM2=SOM2+F(Y)
              Y(L)=OR(L)-T
              SOM2=SOM2+F(Y)
              Y(L)=OR(L)
```

```
 60          CONTINUE
             U=-U
 70        CONTINUE
           Y(IM1)=OR(IM1)
 80      CONTINUE
         IF(KEY8.EQ.1) GO TO 100
         EVAL5=EVAL5+C2*SOM2
         IF(KEY34.EQ.0) GO TO 100
         KEY8=1
         R3=S1
 90   CONTINUE
100   INT=INT+D3(NM1)*SOM2
      IF(N.EQ.2) GO TO 160
      SOM2=0.D0
      DO 150 K=3,N
        KM2=K-2
        V=AX(KM2)*S21
        DO 140 I1=1,2
          Y(KM2)=OR(KM2)+V
          DO 130 I=K,N
            IM1=I-1
            U=AX(IM1)*S21
            DO 120 I2=1,2
              Y(IM1)=OR(IM1)+U
              DO 110 L=I,N
                T=AX(L)*S21
                Y(L)=OR(L)+T
                SOM2=SOM2+F(Y)
                Y(L)=OR(L)-T
                SOM2=SOM2+F(Y)
                Y(L)=OR(L)
110           CONTINUE
              U=-U
120         CONTINUE
            Y(IM1)=OR(IM1)
130       CONTINUE
          V=-V
140     CONTINUE
        Y(KM2)=OR(KM2)
150   CONTINUE
      INT=INT+SOM2*D4(NM1)
160   EPS=DABS(EVAL5-INT)
      IG=0
      IF(DMIN1(DABS(INT-EVAL3),DABS(INT-SOM4)).LE.EPS*.95D0.
     *AND.EPS.GT.ACCUR*Z**3) IG=1
      IF(IG.EQ.1.OR.N.LT.4.OR.DIFO.GE.ACCUR*Z) GO TO 180
      DO 170 I=1,N
        Y(I)=OR(I)+AX(I)*R2(I)
170   CONTINUE
      IF(DABS(SOM4-F(Y)).GE.ACCUR) IG=1
180   IF(IG.EQ.1) EPS=(EPS+EPSMAX*VOL/VOLUME)*0.5D0
      E=EPS*VOL
      INT=INT*VOL
      RETURN
      END
```