D01GDF – NAG Fortran Library Routine Document

Note. Before using this routine, please read the Users' Note for your implementation to check the interpretation of bold italicised terms and other implementation-dependent details.

1 Purpose

D01GDF calculates an approximation to a definite integral in up to 20 dimensions, using the Korobov–Conroy number theoretic method. This routine is designed to be particularly efficient on vector processors.

2 Specification

```
SUBROUTINE D01GDF(NDIM, VECFUN, VECREG, NPTS, VK, NRAND, ITRANS,
1                 RES, ERR, IFAIL)
INTEGER           NDIM, NPTS, NRAND, ITRANS, IFAIL
real              VK(NDIM), RES, ERR
EXTERNAL          VECFUN, VECREG
```

3 Description

This routine calculates an approximation to the integral,

$$I = \int_{c_1}^{d_1} \dots \int_{c_n}^{d_n} f(x_1, \dots, x_n) dx_n \dots dx_1 \tag{1}$$

using the Korobov–Conroy number theoretic method ([1], [2], [3]). The region of integration defined in (1) is such that generally c_i and d_i may be functions of $x_1, x_2, \dots, x_{i-1}$, for $i = 2, 3, \dots, n$, with c_1 and d_1 constants. The integral is first of all transformed to an integral over the n-cube $[0, 1]^n$ by the change of variables

$$x_i = c_i + (d_i - c_i)y_i, \qquad i = 1, 2, \dots, n.$$

The method then uses as its basis the number theoretic formula for the n-cube, $[0, 1]^n$:

$$\int_0^1 \dots \int_0^1 g(x_1, \dots, x_n) dx_n \ \dots dx_1 = \frac{1}{p} \sum_{k=1}^{p} g\left(\left\{k\frac{a_1}{p}\right\}, \dots, \left\{k\frac{a_n}{p}\right\}\right) - E \tag{2}$$

where $\{x\}$ denotes the fractional part of x, $a_1, \dots, a_n$ are the so-called optimal coefficients, E is the error and p is a prime integer. (It is strictly only necessary that p be relatively prime to all $a_1, \dots, a_n$ and is in fact chosen to be even for some cases in Conroy, [3].) The method makes use of properties of the Fourier expansion of $g(x_1, \dots, x_n)$ which is assumed to have some degree of periodicity. Depending on the choice of $a_1, \dots, a_n$ the contributions from certain groups of Fourier coefficients are eliminated from the error, E. Korobov shows that $a_1, \dots, a_n$ can be chosen so that the error satisfies

$$E \leq CKp^{-\alpha} \ln^{\alpha\beta} p \tag{3}$$

where α and C are real numbers depending on the convergence rate of the Fourier series, β is a constant depending on n and K is a constant depending on α and n. There are a number of procedures for calculating these optimal coefficients. Korobov imposes the constraint that

$$a_1 = 1$$
$$a_i = a^{i-1} \pmod{p}$$

and gives a procedure for calculating the parameter, a, to satisfy the optimal conditions.

In this routine the periodisation is achieved by the simple transformation

$$x_i = y_i^2(3 - 2y_i), \qquad i = 1, 2, \dots, n.$$

More sophisticated periodisation procedures are available but in practice the degree of periodisation does not appear to be a critical requirement of the method.

An easily calculable error estimate is not available apart from repetition with an increasing sequence of values of p which can yield erratic results. The difficulties have been studied by Cranley and Patterson [4] who have proposed a Monte Carlo error estimate arising from converting (2) into a stochastic integration rule by the inclusion of a random origin shift which leaves the form of the error (3) unchanged; i.e., in the formula (2), $\left\{k\frac{a_i}{p}\right\}$ is replaced by $\left\{\alpha_i + k\frac{a_i}{p}\right\}$, for $i = 1, 2, \ldots, n$, where each α_i, is uniformly distributed over $[0, 1]$. Computing the integral for each of a sequence of random vectors α allows a 'standard error' to be estimated.

This routine provides built-in sets of optimal coefficients, corresponding to six different values of p. Alternatively, the optimal coefficients may be supplied by the user. D01GYF and D01GZF compute the optimal coefficients for the cases where p is a prime number or p is a product of two primes, respectively.

This routine is designed to be particularly efficient on vector processors, although it is very important that the user also codes the subroutines VECFUN and VECREG efficiently.

4 References

[1] Korobov N M (1957) The approximate calculation of multiple integrals using number theoretic methods *Dokl. Acad. Nauk SSSR* **115** 1062–1065

[2] Korobov N M (1963) *Number Theoretic Methods in Approximate Analysis* Fizmatgiz, Moscow

[3] Conroy H (1967) Molecular Shroedinger equation VIII. A new method for evaluting multi-dimensional integrals *J. Chem. Phys.* **47** 5307–5318

[4] Cranley R and Patterson T N L (1976) Randomisation of number theoretic methods for mulitple integration *SIAM J. Numer. Anal.* **13** 904–914

5 Parameters

1: NDIM — INTEGER *Input*

On entry: the number of dimensions of the integral, n.

Constraint: $1 \leq \text{NDIM} \leq 20$.

2: VECFUN — SUBROUTINE, supplied by the user. *External Procedure*

VECFUN must evaluate the integrand at a specified set of points.

Its specification is:

```
SUBROUTINE VECFUN(NDIM, X, FV, M)
INTEGER           NDIM, M
real              X(M,NDIM), FV(M)
```

1: NDIM — INTEGER *Input*

On entry: the number of dimensions of the integral, n.

2: X(M,NDIM) — ***real*** array *Input*

On entry: the co-ordinates of the m points at which the integrand must be evaluated. X(i, j) contains the jth co-ordinate of the ith point.

3: FV(M) — ***real*** array *Output*

On exit: FV(i) must contain the value of the integrand of the ith point. i.e., FV(i) = $f(\text{X}(i, 1), \text{X}(i, 2), \ldots, \text{X}(i, \text{NDIM}))$, for $i = 1, 2, \ldots$,M.

4: M — INTEGER *Input*

On entry: the number of points m at which the integrand is to be evaluated.

VECFUN must be declared as EXTERNAL in the (sub)program from which D01GDF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

3: VECREG — SUBROUTINE, supplied by the user. *External Procedure*

VECREG must evaluate the limits of integration in any dimension for a set of points.

Its specification is:

```
SUBROUTINE VECREG(NDIM, X, J, C, D, M)
INTEGER           NDIM, J, M
real              X(M,NDIM), C(M), D(M)
```

1: NDIM — INTEGER *Input*

On entry: the number of dimensions of the integral, n.

2: X(M,NDIM) — ***real*** array *Input*

On entry: for $i = 1, 2, \ldots, m$, X(i,1), X(i,2),...,X($i, j-1$) contain the current values of the first $j-1$ co-ordinates of the ith point, which may be used if necessary in calculating the m values of c_j and d_j.

3: J — INTEGER *Input*

On entry: the index, j, of the dimension for which the limits of the range of integration are required.

4: C(M) — ***real*** array *Output*

On exit: C(i) must be set to the lower limit of the range for X(i, j), for $i = 1, 2, \ldots, m$.

5: D(M) — ***real*** array *Output*

On exit: D(i) must be set to the upper limit of the range for X(i, j), for $i = 1, 2, \ldots, m$.

6: M — INTEGER *Input*

On entry: the number of points m at which the limits of integration must be specified.

VECREG must be declared as EXTERNAL in the (sub)program from which D01GDF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

4: NPTS — INTEGER *Input*

On entry: the Korobov rule to be used. There are two alternatives depending on the value of NPTS.

(a) $1 \leq \text{NPTS} \leq 6$.

In this case one of six preset rules is chosen using 2129, 5003, 10007, 20011, 40009 or 80021 points depending on the respective value of NPTS being 1, 2, 3, 4, 5 or 6.

(b) $\text{NPTS} > 6$.

NPTS is the number of actual points to be used with corresponding optimal coefficients supplied in the array VK.

Constraint: $\text{NPTS} \geq 1$

5: VK(NDIM) — ***real*** array *Input/Output*

On entry: If $\text{NPTS} > 6$, VK must contain the n optimal coefficients (which may be calculated using D01GYF or D01GZF); if $\text{NPTS} \leq 6$, VK need not be set.

On exit: if $\text{NPTS} > 6$, VK is unchanged; if $\text{NPTS} \leq 6$, VK contains the n optimal coefficients used by the preset rule.

6: NRAND — INTEGER *Input*

On entry: the number of random samples to be generated (generally a small value, say 3 to 5, is sufficient). The estimate, RES, of the value of the integral returned by the routine is then the average of NRAND calculations with different random origin shifts. If $\text{NPTS} > 6$, the total number of integrand evaluations will be NRAND $\times$ NPTS. If $1 \leq \text{NPTS} \leq 6$, then the number of integrand

evaluations will be NRAND $\times p$, where p is the number of points corresponding to the six preset rules. For reasons of efficiency, these values are calculated a number at a time in VECFUN.

Constraint: NRAND ≥ 1

7: ITRANS — INTEGER *Input*

On entry: indicates whether the periodising transformation is to be used:

if ITRANS $= 0$, the transformation is to be used.
if ITRANS $\neq 0$, the transformation is to be suppressed (to cover cases where the integrand may already be periodic or where the user desires to specify a particular transformation in the definition of VECFUN).

Suggested value: ITRANS $= 0$.

8: RES — ***real*** *Output*

On exit: an estimate of the value of the integral.

9: ERR — ***real*** *Output*

On exit: the standard error as computed from NRAND sample values. If NRAND $= 1$, then ERR contains zero.

10: IFAIL — INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.

On exit: IFAIL $= 0$ unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors detected by the routine:

IFAIL $= 1$

On entry, NDIM < 1,
or NDIM > 20.

IFAIL $= 2$

On entry, NPTS < 1.

IFAIL $= 3$

On entry, NRAND < 1.

7 Accuracy

If NRAND > 1, an estimate of the absolute standard error is given by the value, on exit, of ERR.

8 Further Comments

This routine performs the same computation as the D01GCF. However, the interface has been modified so that it can perform more efficiently on machines with vector processing capabilities. In particular, the routines VECFUN and VECREG must calculate the integrand and limits of integration at a *set* of points. For some problems the amount of time spent in these two subroutines, which must be supplied by the user, may account for a significant part of the total computation time. For this reason it is vital that the user considers the possibilities for vectorization in the code supplied for these two subroutines.

The time taken will be approximately proportional to NRAND $\times p$, where p is the number of points used, but may depend significantly on the efficiency of the code provided by the user in subroutines VECFUN and VECREG.

The exact values of RES and ERR returned by D01GDF will depend (within statistical limits) on the sequence of random numbers generated within the routine by calls to G05CAF. To ensure that the results returned by D01GDF in separate runs are identical, users should call G05CBF immediately before calling D01GDF; to ensure that they are different, call G05CCF.

9 Example

This example calculates the integral

$$\int_0^1 \int_0^1 \int_0^1 \int_0^1 \cos(0.5 + 2(x_1 + x_2 + x_3 + x_4) - 4)\, dx_1\, dx_2\, dx_3\, dx_4.$$

9.1 Program Text

Note. The listing of the example program presented below uses bold italicised terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     D01GDF Example Program Text
*     Mark 14 Release.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NDIM
      PARAMETER        (NDIM=4)
*     .. Local Scalars ..
      real             ERR, RES
      INTEGER          IFAIL, ITRANS, NPTS, NRAND
*     .. Local Arrays ..
      real             VK(NDIM)
*     .. External Subroutines ..
      EXTERNAL         D01GDF, VECFUN, VECREG
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D01GDF Example Program Results'
      WRITE (NOUT,*)
      NPTS = 2
      ITRANS = 0
      NRAND = 4
      IFAIL = 0
*
      CALL D01GDF(NDIM,VECFUN,VECREG,NPTS,VK,NRAND,ITRANS,RES,ERR,IFAIL)
*
      WRITE (NOUT,99999) 'Result = ', RES, ', standard error = ', ERR
      STOP
*
99999 FORMAT (1X,A,F13.5,A,e10.2)
      END
```

```
*
      SUBROUTINE VECFUN(NDIM,X,FV,M)
*     .. Scalar Arguments ..
      INTEGER           M, NDIM
*     .. Array Arguments ..
      real              FV(M), X(M,NDIM)
*     .. Local Scalars ..
      INTEGER           I, J
*     .. Intrinsic Functions ..
      INTRINSIC         COS, real
*     .. Executable Statements ..
      DO 20 I = 1, M
         FV(I) = 0.0e0
   20 CONTINUE
      DO 60 J = 1, NDIM
         DO 40 I = 1, M
            FV(I) = FV(I) + X(I,J)
   40    CONTINUE
   60 CONTINUE
      DO 80 I = 1, M
         FV(I) = COS(0.5e0+2.0e0*FV(I)-real(NDIM))
   80 CONTINUE
      RETURN
      END
*
      SUBROUTINE VECREG(NDIM,X,J,C,D,M)
*     .. Scalar Arguments ..
      INTEGER           J, M, NDIM
*     .. Array Arguments ..
      real              C(M), D(M), X(M,NDIM)
*     .. Local Scalars ..
      INTEGER           I
*     .. Executable Statements ..
      DO 20 I = 1, M
         C(I) = 0.0e0
         D(I) = 1.0e0
   20 CONTINUE
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
D01GDF Example Program Results

Result =       0.43999, standard error =   0.18E-05
```