

NAG Toolbox

nag_quad_md_numth_vec (d01gd)

1 Purpose

nag_quad_md_numth_vec (d01gd) calculates an approximation to a definite integral in up to 20 dimensions, using the Korobov–Conroy number theoretic method. This function is designed to be particularly efficient on vector processors.

2 Syntax

```
[vk, res, err, ifail] = nag_quad_md_numth_vec(vecfun, vecreg, npts, vk, nrand,
'ndim', ndim, 'itrans', itrans)
```

```
[vk, res, err, ifail] = d01gd(vecfun, vecreg, npts, vk, nrand, 'ndim', ndim,
'itrans', itrans)
```

3 Description

nag_quad_md_numth_vec (d01gd) calculates an approximation to the integral

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

using the Korobov–Conroy number theoretic method (see Korobov (1957), Korobov (1963) and Conroy (1967)). 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

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

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

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

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 (1967).) 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

$$E \leq CKp^{-\alpha} \ln^{\alpha\beta} p \quad (3)$$

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

$$a_1 = 1 \quad \text{and} \quad a_i = a^{i-1} \pmod{p} \quad (4)$$

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

In this function the periodisation is achieved by the simple transformation

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

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 (1976) 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 $i = 1, 2, \dots, n$, 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 function provides built-in sets of optimal coefficients, corresponding to six different values of p . Alternatively, the optimal coefficients may be supplied by you. Functions `nag_quad_md_numth_coeff_prime` (d01gy) and `nag_quad_md_numth_coeff_2prime` (d01gz) compute the optimal coefficients for the cases where p is a prime number or p is a product of two primes, respectively.

This function is designed to be particularly efficient on vector processors, although it is very important that you also code `vecfun` and `vecreg` efficiently.

4 References

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

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

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

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

5 Parameters

5.1 Compulsory Input Parameters

- 1: `vecfun` – SUBROUTINE, supplied by the user.
`vecfun` must evaluate the integrand at a specified set of points.

```
[fv] = vecfun(ndim, x, m)
```

Input Parameters

- 1: `ndim` – INTEGER
 n , the number of dimensions of the integral.
- 2: `x(m, ndim)` – REAL (KIND=`nag_wp`) array
The coordinates of the m points at which the integrand must be evaluated. $\mathbf{x}(i, j)$ contains the j th coordinate of the i th point.
- 3: `m` – INTEGER
The number of points m at which the integrand is to be evaluated.

Output Parameters

- 1: `fv(m)` – REAL (KIND=`nag_wp`) array
 $\mathbf{fv}(i)$ must contain the value of the integrand of the i th point, i.e., $\mathbf{fv}(i) = f(\mathbf{x}(i, 1), \mathbf{x}(i, 2), \dots, \mathbf{x}(i, \mathbf{ndim}))$, for $i = 1, 2, \dots, \mathbf{m}$.

2: **vecreg** – SUBROUTINE, supplied by the user.

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

```
[c, d] = vecreg(ndim, x, j, m)
```

Input Parameters

1: **ndim** – INTEGER

n , the number of dimensions of the integral.

2: **x(m, ndim)** – REAL (KIND=nag_wp) array

For $i = 1, 2, \dots, m$, $\mathbf{x}(i, 1), \mathbf{x}(i, 2), \dots, \mathbf{x}(i, j - 1)$ contain the current values of the first $(j - 1)$ coordinates of the i th point, which may be used if necessary in calculating the m values of c_j and d_j .

3: **j** – INTEGER

The index j for which the limits of the range of integration are required.

4: **m** – INTEGER

The number of points m at which the limits of integration must be specified.

Output Parameters

1: **c(m)** – REAL (KIND=nag_wp) array

$\mathbf{c}(i)$ must be set to the lower limit of the range for $\mathbf{x}(i, j)$, for $i = 1, 2, \dots, m$.

2: **d(m)** – REAL (KIND=nag_wp) array

$\mathbf{d}(i)$ must be set to the upper limit of the range for $\mathbf{x}(i, j)$, for $i = 1, 2, \dots, m$.

3: **npts** – INTEGER

The Korobov rule to be used. There are two alternatives depending on the value of **npts**.

(i) $1 \leq \mathbf{npts} \leq 6$.

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.

(ii) **npts** > 6.

npts is the number of actual points to be used with corresponding optimal coefficients supplied in the array **vk**.

Constraint: **npts** ≥ 1 .

4: **vk(ndim)** – REAL (KIND=nag_wp) array

If **npts** > 6, **vk** must contain the n optimal coefficients (which may be calculated using nag_quad_md_numth_coeff_prime (d01gy) or nag_quad_md_numth_coeff_2prime (d01gz)).

If **npts** ≤ 6 , **vk** need not be set.

5: **nrand** – INTEGER

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 function is then the average of **nrand** calculations with different random origin shifts. If **npts** > 6, the total number of integrand evaluations will be **nrand** \times **npts**. If $1 \leq \mathbf{npts} \leq 6$, then the number of integrand evaluations will

be $\mathbf{nrand} \times p$, 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: $\mathbf{nrand} \geq 1$.

5.2 Optional Input Parameters

1: **ndim** – INTEGER

Default: the dimension of the array **vk**.

n , the number of dimensions of the integral.

Constraint: $1 \leq \mathbf{ndim} \leq 20$.

2: **itrans** – INTEGER

Suggested value: **itrans** = 0.

Default: 0

Indicates whether the periodising transformation is to be used.

itrans = 0

The transformation is to be used.

itrans \neq 0

The transformation is to be suppressed (to cover cases where the integrand may already be periodic or where you want to specify a particular transformation in the definition of **vecfun**).

5.3 Output Parameters

1: **vk(ndim)** – REAL (KIND=nag_wp) array

If **npts** > 6, **vk** is unchanged.

If **npts** \leq 6, **vk** contains the n optimal coefficients used by the preset rule.

2: **res** – REAL (KIND=nag_wp)

The approximation to the integral I .

3: **err** – REAL (KIND=nag_wp)

The standard error as computed from **nrand** sample values. If **nrand** = 1, then **err** contains zero.

4: **ifail** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, **ndim** < 1,
or **ndim** > 20.

ifail = 2

On entry, **npts** < 1.

ifail = 3

On entry, **nrand** < 1.

ifail = -99

An unexpected error has been triggered by this routine. Please contact NAG.

ifail = -399

Your licence key may have expired or may not have been installed correctly.

ifail = -999

Dynamic memory allocation failed.

7 Accuracy

If **nrand** > 1, an estimate of the absolute standard error is given by the value, on exit, of **err**.

8 Further Comments

`nag_quad_md_numth_vec` (d01gd) performs the same computation as `nag_quad_md_numth` (d01gc). However, the interface has been modified so that it can perform more efficiently on machines with vector processing capabilities. In particular, **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 functions, which must be supplied by you, may account for a significant part of the total computation time. For this reason it is vital that you consider the possibilities for vectorization in the code supplied for these two functions.

The time taken will be approximately proportional to **nrand** × *p*, where *p* is the number of points used, but may depend significantly on the efficiency of the code provided by you in **vecfun** and **vecreg**.

The exact values of **res** and **err** on return will depend (within statistical limits) on the sequence of random numbers generated within `nag_quad_md_numth_vec` (d01gd) by calls to `nag_rand_dist_uniform01` (g05sa). Separate runs will produce identical answers.

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

```
function d01gd_example
    fprintf('d01gd example results\n\n');

    npts = nag_int(2);
    vk = zeros(4,1);
    nrand = nag_int(4);

    [vk, res, err, ifail] = d01gd( ...
        @vecfun, @vecreg, npts, vk, nrand);

    fprintf('Result = %13.5f, Standard error = %10.2e\n', res, err);

    function fv = vecfun(ndim, x, m)
        fv = zeros(m,1);

        for i=1:m
            fv(i) = cos(0.5 + 2*sum(x(i,1:ndim)) - double(ndim));
```

```
end  
  
function [c,d] = vecreg(ndim, x, j, m)  
    c = zeros(m,1);  
    d = ones(m,1);
```

9.2 Program Results

d01gd example results

Result = 0.43999, Standard error = 1.89e-06
