NAG Toolbox

nag_rand_quasi_uniform (g05ym)

1 Purpose

nag_rand_quasi_uniform (g05ym) generates a uniformly distributed low-discrepancy sequence as proposed by Sobol, Faure or Niederreiter. It must be preceded by a call to one of the initialization functions nag_rand_quasi_init (g05yl) or nag_rand_quasi_init_scrambled (g05yn).

2 Syntax

```
[quas, iref, ifail] = nag_rand_quasi_uniform(n, iref, 'rcord', rcord)
[quas, iref, ifail] = g05ym(n, iref, 'rcord', rcord)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 24: **rcord** was made optional.

3 Description

Low discrepancy (quasi-random) sequences are used in numerical integration, simulation and optimization. Like pseudorandom numbers they are uniformly distributed but they are not statistically independent, rather they are designed to give more even distribution in multidimensional space (uniformity). Therefore they are often more efficient than pseudorandom numbers in multidimensional Monte–Carlo methods.

nag_rand_quasi_uniform (g05ym) generates a set of points $x^1, x^2, \ldots, x^N$ with high uniformity in the S-dimensional unit cube $I^S = [0,1]^S$.

Let G be a subset of I^S and define the counting function $S_N(G)$ as the number of points $x^i \in G$. For each $x = (x_1, x_2, \ldots, x_S) \in I^S$, let G_x be the rectangular S-dimensional region

$$G_x = [0, x_1) \times [0, x_2) \times \cdots \times [0, x_S)$$

with volume $x_1, x_2, \ldots, x_S$. Then one measure of the uniformity of the points $x^1, x^2, \ldots, x^N$ is the discrepancy:

$$D_N^*\left(x^1, x^2, \ldots, x^N\right) = \sup_{x \in I^S} |S_N(G_x) - N x_1, x_2, \ldots, x_S|.$$

which has the form

$$D_N^*\left(x^1, x^2, \ldots, x^N\right) \leq C_S (\log N)^S + O\left((\log N)^{S-1}\right) \quad \text{for all} \quad N \geq 2.$$

The principal aim in the construction of low-discrepancy sequences is to find sequences of points in I^S with a bound of this form where the constant C_S is as small as possible.

The type of low-discrepancy sequence generated by nag_rand_quasi_uniform (g05ym) depends on the initialization function called and can include those proposed by Sobol, Faure or Niederreiter. If the initialization function nag_rand_quasi_init_scrambled (g05yn) was used then the sequence will be scrambled (see Section 3 in nag_rand_quasi_init_scrambled (g05yn) for details).

4 References

Bratley P and Fox B L (1988) Algorithm 659: implementing Sobol's quasirandom sequence generator *ACM Trans. Math. Software* **14(1)** 88–100

Fox B L (1986) Algorithm 647: implementation and relative efficiency of quasirandom sequence generators *ACM Trans. Math. Software* **12(4)** 362–376

5 Parameters

Note: the following variables are used in the parameter descriptions:

$idim$ = **idim**, the number of dimensions required, see nag_rand_quasi_init (g05yl) or nag_rand_quasi_init_scrambled (g05yn)

$liref$ = **liref**, the length of **iref** as supplied to the initialization function nag_rand_quasi_init (g05yl) or nag_rand_quasi_init_scrambled (g05yn)

$tdquas$ = **n** if **rcord** = 1; otherwise $tdquas = idim$.

5.1 Compulsory Input Parameters

1: **n** – INTEGER

The number of quasi-random numbers required.

Constraint: **n** ≥ 0 and **n** + previous number of generated values $\leq 2^{31} - 1$.

2: **iref**$(liref)$ – INTEGER array

Contains information on the current state of the sequence.

5.2 Optional Input Parameters

1: **rcord** – INTEGER

Default: 1

The order in which the generated values are returned.

Constraint: **rcord** = 1 or 2.

5.3 Output Parameters

1: **quas**$(ldquas, tdquas)$ – REAL (KIND=nag_wp) array

Contains the **n** quasi-random numbers of dimension $idim$.

If **rcord** = 1, **quas**(i, j) holds the jth value for the ith dimension.

If **rcord** = 2, **quas**(i, j) holds the ith value for the jth dimension.

2: **iref**$(liref)$ – INTEGER array

Contains updated information on the state of the sequence.

3: **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 $= 1$

Constraint: **n** ≥ 0.

On entry, value of **n** would result in too many calls to the generator.

ifail $= 2$

Constraint: **rcord** $= 1$ or 2.

ifail $= 4$

Constraint: if **rcord** $= 1$, $ldquas \geq idim$.

Constraint: if **rcord** $= 2$, $ldquas \geq$ **n**.

ifail $= 5$

On entry, **iref** has either not been initialized or has been corrupted.

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

Not applicable.

8 Further Comments

None.

9 Example

This example calls nag_rand_quasi_init (g05yl) and nag_rand_quasi_uniform (g05ym) to estimate the value of the integral

$$\int_0^1 \cdots \int_0^1 \prod_{i=1}^{s} |4x_i - 2| dx_1, \, dx_2, \ldots, \, dx_s = 1.$$

In this example the number of dimensions S is set to 8.

9.1 Program Text

```
function g05ym_example

fprintf('g05ym example results\n\n');

% Initialize the Sobol generator, skipping some variates
iskip = nag_int(1000);
idim  = nag_int(8);
genid = nag_int(1);
% Initialize the Sobol generator
```

```
[iref, ifail] = g05yl( ...
                      genid,idim,iskip);

% Number of variates
n = nag_int(200);

% Generate N quasi-random variates
[quas, iref, ifail] = g05ym( ...
                            n, iref);

% Evaluate the function, and sum
p(1:n) = prod(abs(4*quas(1:idim,:)-2));
fsum = sum(p);

% Convert sum to mean value
vsbl = fsum/double(n);
fprintf('Value of integral = %8.4f\n\n', vsbl);
fprintf('First 10 variates\n');
for i = 1:10
  fprintf(' %3d', i);
  fprintf(' %7.4f', quas(1:idim,i));
  fprintf('\n');
end
```

9.2 Program Results

```
     g05ym example results

Value of integral =   1.0410

First 10 variates
   1  0.7197  0.5967  0.0186  0.1768  0.7803  0.4072  0.5459  0.3994
   2  0.9697  0.3467  0.7686  0.9268  0.5303  0.1572  0.2959  0.1494
   3  0.4697  0.8467  0.2686  0.4268  0.0303  0.6572  0.7959  0.6494
   4  0.3447  0.4717  0.1436  0.3018  0.1553  0.7822  0.4209  0.0244
   5  0.8447  0.9717  0.6436  0.8018  0.6553  0.2822  0.9209  0.5244
   6  0.5947  0.2217  0.3936  0.0518  0.9053  0.0322  0.1709  0.7744
   7  0.0947  0.7217  0.8936  0.5518  0.4053  0.5322  0.6709  0.2744
   8  0.0635  0.1904  0.0498  0.4580  0.6240  0.2510  0.9521  0.8057
   9  0.5635  0.6904  0.5498  0.9580  0.1240  0.7510  0.4521  0.3057
  10  0.8135  0.4404  0.2998  0.2080  0.3740  0.5010  0.7021  0.0557
```