# NAG Toolbox

# nag_rand_quasi_init (g05yl)

## 1 Purpose

nag_rand_quasi_init (g05yl) initializes a quasi-random generator prior to calling nag_rand_quasi_normal (g05yj), nag_rand_quasi_lognormal (g05yk) or nag_rand_quasi_uniform (g05ym).

## 2 Syntax

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

## 3 Description

nag_rand_quasi_init (g05yl) selects a quasi-random number generator through the input value of **genid** and initializes the **iref** communication array for use by the functions nag_rand_quasi_normal (g05yj), nag_rand_quasi_lognormal (g05yk) or nag_rand_quasi_uniform (g05ym).

One of three types of quasi-random generator may be chosen, allowing the low-discrepancy sequences proposed by Sobol, Faure or Niederreiter to be generated.

Two sets of Sobol sequences are supplied, the first, is based on the work of Joe and Kuo (2008). The second, referred to in the documentation as "Sobol (A659)", is based on Algorithm 659 of Bratley and Fox (1988) with the extension to 1111 dimensions proposed by Joe and Kuo (2003). Both sets of Sobol sequences should satisfy the so-called Property A, up to 1111 dimensions, but the first set should have better two-dimensional projections than those produced using Algorithm 659.

## 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

Joe S and Kuo F Y (2003) Remark on Algorithm 659: implementing Sobol's quasirandom sequence generator *ACM Trans. Math. Software (TOMS)* **29** 49–57

Joe S and Kuo F Y (2008) Constructing Sobol sequences with better two-dimensional projections *SIAM J. Sci. Comput.* **30** 2635–2654

## 5 Parameters

### 5.1 Compulsory Input Parameters

1: **genid** – INTEGER

Must identify the quasi-random generator to use.

**genid** = 1
Sobol generator.

**genid** = 2
Sobol (A659) generator.

**genid** = 3
Niederreiter generator.

**genid** $= 4$
Faure generator.

*Constraint*: **genid** $= 1,\ 2,\ 3$ or $4$.

2: **idim** – INTEGER

The number of dimensions required.

*Constraints*:

if **genid** $= 1,\ 1 \leq$ **idim** $\leq 10000$;
if **genid** $= 2,\ 1 \leq$ **idim** $\leq 1111$;
if **genid** $= 3,\ 1 \leq$ **idim** $\leq 318$;
if **genid** $= 4,\ 1 \leq$ **idim** $\leq 40$.

3: **iskip** – INTEGER

The number of terms of the sequence to skip on initialization for the Sobol and Niederreiter generators. If **genid** $= 4$, **iskip** is ignored.

*Constraint*: if **genid** $= 1,\ 2$ or $3,\ 0 \leq$ **iskip** $\leq 2^{30}$.

### 5.2 Optional Input Parameters

None.

### 5.3 Output Parameters

1: **iref**$(liref)$ – INTEGER array

Contains initialization information for use by the generator functions nag_rand_quasi_normal (g05yj), nag_rand_quasi_lognormal (g05yk) and nag_rand_quasi_uniform (g05ym). **iref** must not be altered in any way between initialization and calls of the generator functions.

2: **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: **genid** $= 1,\ 2,\ 3$ or $4$.

**ifail** $= 2$

Constraint: $1 \leq$ **idim** $\leq \langle value \rangle$.

**ifail** $= 4$

On entry, $liref$ is too small.

**ifail** $= 5$

On entry, **iskip** $< 0$ or **iskip** is too large.

**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

The primitive polynomials and direction numbers used for the Sobol generator (**genid** $= 1$) were calculated by Joe and Kuo (2008) using the search critera $D^{(6)}$.

## 9 Example

See Section 10 in nag_rand_quasi_uniform (g05ym).

### 9.1 Program Text

```
     function g05yl_example

fprintf('g05yl 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 variates
[quas, iref, ifail] = g05ym( ...
                             nag_int(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

```
     g05yl 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
```