NAG Toolbox

nag_rand_bb_inc (g05xd)

1 Purpose

nag_rand_bb_inc (g05xd) computes scaled increments of sample paths of a free or non-free Wiener process, where the sample paths are constructed by a Brownian bridge algorithm. The initialization function nag_rand_bb_inc_init (g05xc) must be called prior to the first call to nag_rand_bb_inc (g05xd).

2 Syntax

```
[z, b, ifail] = nag_rand_bb_inc(npaths, diff, z, c, rcomm, 'rcord', rcord, 'd', d, 'a', a)
[z, b, ifail] = g05xd(npaths, diff, z, c, rcomm, 'rcord', rcord, 'd', d, 'a', a)
```

3 Description

For details on the Brownian bridge algorithm and the bridge construction order see Section 2.6 in the G05 Chapter Introduction and Section 3 in nag_rand_bb_inc_init (g05xc). Recall that the terms Wiener process (or free Wiener process) and Brownian motion are often used interchangeably, while a non-free Wiener process (also known as a Brownian bridge process) refers to a process which is forced to terminate at a given point.

Fix two times $t_0 < T$, let $(t_i)_{1 \leq i \leq N}$ be any set of time points satisfying $t_0 < t_1 < t_2 < \cdots < t_N < T$, and let X_{t_0}, $(X_{t_i})_{1 \leq i \leq N}$, X_T denote a d-dimensional Wiener sample path at these time points.

The Brownian bridge increments generator uses the Brownian bridge algorithm to construct sample paths for the (free or non-free) Wiener process X, and then uses this to compute the *scaled Wiener increments*

$$\frac{X_{t_1} - X_{t_0}}{t_1 - t_0}, \frac{X_{t_2} - X_{t_1}}{t_2 - t_1}, \ldots, \frac{X_{t_N} - X_{t_{N-1}}}{t_N - t_{N-1}}, \frac{X_T - X_{t_N}}{T - t_N}$$

The example program in Section 10 shows how these increments can be used to compute a numerical solution to a stochastic differential equation (SDE) driven by a (free or non-free) Wiener process.

4 References

Glasserman P (2004) *Monte Carlo Methods in Financial Engineering* Springer

5 Parameters

Note: the following variable is used in the parameter descriptions: $N =$ **ntimes**, the length of the array **times** passed to the initialization function nag_rand_bb_inc_init (g05xc).

5.1 Compulsory Input Parameters

1: **npaths** – INTEGER

The number of Wiener sample paths.

Constraint: **npaths** ≥ 1.

2: **dif**($\mathbf{d}$) – REAL (KIND=nag_wp) array

The difference between the terminal value and starting value of the Wiener process. If $\mathbf{a} = 0$, **diff** is ignored.

3: **z**($ldz, :$) – REAL (KIND=nag_wp) array

The first dimension, ldz, of the array **z** must satisfy

if **rcord** $= 1$, $ldz \geq \mathbf{d} \times (N + 1 - \mathbf{a})$;
if **rcord** $= 2$, $ldz \geq$ **npaths**.

The second dimension of the array **z** must be at least **npaths** if **rcord** $= 1$ and at least $\mathbf{d} \times (n + 1 - \mathbf{a})$ if **rcord** $= 2$.

The Normal random numbers used to construct the sample paths.

If quasi-random numbers are used, the $\mathbf{d} \times (N + 1 - \mathbf{a})$-dimensional quasi-random points should be stored in successive rows of Z.

4: **c**($ldc, :$) – REAL (KIND=nag_wp) array

The first dimension of the array **c** must be at least **d**.

The second dimension of the array **c** must be at least **d**.

The lower triangular Cholesky factorization C such that CC^{T} gives the covariance matrix of the Wiener process. Elements of C above the diagonal are not referenced.

5: **rcomm**($*$) – REAL (KIND=nag_wp) array

Note: the dimension of this array is dictated by the requirements of associated functions that must have been previously called. This array **must** be the same array passed as argument **rcomm** in the previous call to nag_rand_bb_inc_init (g05xc) or nag_rand_bb_inc (g05xd).

Communication array as returned by the last call to nag_rand_bb_inc_init (g05xc) or nag_rand_bb_inc (g05xd). This array **must not** be directly modified.

5.2 Optional Input Parameters

1: **rcord** – INTEGER

Default: 1

The order in which Normal random numbers are stored in **z** and in which the generated values are returned in **b**.

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

2: **d** – INTEGER

Default: the dimension of the array **diff** and the first dimension of the array **c**. (An error is raised if these dimensions are not equal.)

The dimension of each Wiener sample path.

Constraint: $\mathbf{d} \geq 1$.

3: **a** – INTEGER

Default: 1

If $\mathbf{a} = 0$, a free Wiener process is created and **diff** is ignored.

If $\mathbf{a} = 1$, a non-free Wiener process is created where **diff** is the difference between the terminal value and the starting value of the process.

Constraint: $\mathbf{a} = 0$ or 1.

5.3 Output Parameters

1: **z**$(ldz, :)$ – REAL (KIND=nag_wp) array

The first dimension, ldz, of the array **z** will be

if **rcord** $= 1$, $ldz =$ **d** $\times (N + 1 -$ **a**$)$;
if **rcord** $= 2$, $ldz =$ **npaths**.

The second dimension of the array **z** will be **npaths** if **rcord** $= 1$ and at least **d** $\times (n + 1 -$ **a**$)$ if **rcord** $= 2$.

The Normal random numbers premultiplied by **c**.

2: **b**$(ldb, :)$ – REAL (KIND=nag_wp) array

The first dimension, ldb, of the array **b** will be

if **rcord** $= 1$, $ldb =$ **d** $\times (N + 1)$;
if **rcord** $= 2$, $ldb =$ **npaths**.

The second dimension of the array **b** will be **npaths** if **rcord** $= 1$ and at least **d** $\times (n + 1)$ if **rcord** $= 2$.

The scaled Wiener increments.

Let $X_{p,i}^k$ denote the kth dimension of the ith point of the pth sample path where $1 \leq k \leq$ **d**, $1 \leq i \leq n+1$ and $1 \leq p \leq$ **npaths**. The increment $\dfrac{\left(X_{p,i}^k - X_{p,i-1}^k\right)}{(t_i - t_{i-1})}$ is stored at $B(p, k + (i-1) \times$ **d**$)$.

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 $= 1$

On entry, **rcomm** was not initialized or has been corrupted.

ifail $= 2$

Constraint: **npaths** ≥ 1.

ifail $= 3$

On entry, **rcord** $= \langle value \rangle$ was an illegal value.

ifail $= 4$

Constraint: **d** ≥ 1.

ifail $= 5$

Constraint: **a** $= 0$ or 1.

ifail $= 6$

Constraint: $ldz \geq$ **d** $\times$ (**ntimes** $+ 1 -$ **a**$)$.

Constraint: $ldz \geq$ **npaths**.

ifail $= 7$

ldc is too small.

ifail $= 8$

Constraint: $ldb \geq \mathbf{d} \times (\mathbf{ntimes} + 1)$.

Constraint: $ldb \geq \mathbf{npaths}$.

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

The scaled Wiener increments produced by this function can be used to compute numerical solutions to stochastic differential equations (SDEs) driven by (free or non-free) Wiener processes. Consider an SDE of the form

$$dY_t = f(t, Y_t)\mathrm{dt} + \sigma(t, Y_t)dX_t$$

on the interval $[t_0, T]$ where $(X_t)_{t_0 \leq t \leq T}$ is a (free or non-free) Wiener process and f and σ are suitable functions. A numerical solution to this SDE can be obtained by the Euler–Maruyama method. For any discretization $t_0 < t_1 < t_2 < \cdots < t_{N+1} = T$ of $[t_0, T]$, set

$$Y_{t_{i+1}} = Y_{t_i} + f(t_i, Y_{t_i})(t_{i+1} - t_i) + \sigma(t_i, Y_{t_i})\big(X_{t_{i+1}} - X_{t_i}\big)$$

for $i = 1, \ldots, N$ so that $Y_{t_{N+1}}$ is an approximation to Y_T. The scaled Wiener increments produced by nag_rand_bb_inc (g05xd) can be used in the Euler–Maruyama scheme outlined above by writing

$$Y_{t_{i+1}} = Y_{t_i} + (t_{i+1} - t_i)\left(f(t_i, Y_{t_i}) + \sigma(t_i, Y_{t_i})\left(\frac{X_{t_{i+1}} - X_{t_i}}{t_{i+1} - t_i}\right)\right).$$

The following example program uses this method to solve the SDE for geometric Brownian motion

$$dS_t = rS_t\mathrm{dt} + \sigma S_t dX_t$$

where X is a Wiener process, and compares the results against the analytic solution

$$S_T = S_0 \exp\big((r - \sigma^2/2)T + \sigma X_T\big).$$

Quasi-random variates are used to construct the Wiener increments.

9.1 Program Text

```
     function g05xd_example

fprintf('g05xd example results\n\n');

% We wish to solve the stochastic differential equation (SDE)
%   dSt = r*St*dt + sigma*St*dXt
% where X is a one dimensional Wiener process.
% This means we have
%     a = 0,   d = 1,  c = 1
% Set the other parameters of the SDE and the Euler-Maruyama scheme

% Initial value of the process
s0    = 1;
r     = 0.05;
sigma = 0.12;

% Number of paths to simulate
npaths = nag_int(3);

% The time interval [t0,T] on which to solve the SDE
t0   = 0;
tend = 1;

% The time steps in the discretization of [t0,T]
tn = 20;
t = t0 + (1:tn)*(tend-t0)/(tn+1);

% Make the bridge construction order
bgord = nag_int(3);
move  = zeros(0, 1, nag_int_name);
[times, ifail] = g05xe( ...
                        t0, tend, t, move, 'bgord', bgord);

% Generate the Z values
d = 1;
a = nag_int(0);
[z] = get_z(npaths, d, a, tn);

% Initialize the generator
[rcomm, ifail] = g05xc( ...
                        t0, tend, times);

% Get the scaled increments of the Wiener process
dif = zeros(d, 1);
c   = ones(d, 1);
[z, b, ifail] = g05xd( ...
                       npaths, dif, z, c, rcomm, 'a', a);

% Do the Euler-Maruyama time stepping
st = zeros(tn+1, npaths);

% Do first time step
st(1,:) = s0 + (t(1)-t0)*(r*s0+sigma*s0*b(1, :));
v = st(1,:);
for i=2:tn
  v = v + (t(i)-t(i-1))*(r*v+sigma*v.*b(i,:));
  st(i,:) = v;
end
% Do last time step
st(tn+1,:) = v + (tend-t(tn))*(r*v+sigma*v.*b(tn+1,:));

% Compute the analytic solution:
%    ST = S0*exp( (r-sigma^2/2)T + sigma WT )
analytic = s0*exp((r-0.5*sigma^2)*tend+sigma*sqrt(tend-t0)*z(1,:));

% Display the results
fprintf('\nEuler-Maruyama solution for Geometric Brownian motion\n');
fprintf('        Path 1     Path 2     Path 3\n');
for i = 1:tn+1
```

```
  fprintf('%2d  %10.4f %10.4f %10.4f\n', i, st(i, :));
end
fprintf('\nAnalytic solution at final time step\n');
fprintf('        Path 1     Path 2     Path 3\n');
fprintf('    %10.4f %10.4f %10.4f\n', analytic);

function [z] = get_z(npaths, d, a, ntimes)
  idim = d*(ntimes+1-a);

  % We now need to generate the input pseudorandom points

  % First initialize the base pseudorandom number generator
  state = initialize_prng(nag_int(6), nag_int(0), [nag_int(1023401)]);

  % Scrambled quasi-random sequences preserve the good discrepancy
  % properties of quasi-random sequences while counteracting the bias
  % some applications experience when using quasi-random sequences.
  % Initialize the scrambled quasi-random generator.
  [iref, state] = initialize_scrambled_qrng(nag_int(1), nag_int(2), ...
                                            idim, state);

  % Generate the quasi-random points from N(0,1)
  xmean = zeros(idim, 1);
  std   = ones(idim, 1);
  [z, iref, ifail] = g05yj( ...
                            xmean, std, npaths, iref);
  z = z';

function [state] = initialize_prng(genid, subid, seed)
  % Initialize the generator to a repeatable sequence
  [state, ifail] = g05kf( ...
                          genid, subid, seed);

function [iref, state] = initialize_scrambled_qrng(genid,stype,idim,state)
  iskip = nag_int(0);
  nsdigits = nag_int(32);
  [iref, state, ifail] = g05yn( ...
                                genid, stype, nag_int(idim), ...
                                iskip, nsdigits, state);
```

9.2 Program Results

```
     g05xd example results

Euler-Maruyama solution for Geometric Brownian motion
        Path 1     Path 2     Path 3
 1      0.9668     1.0367     0.9992
 2      0.9717     1.0254     1.0077
 3      0.9954     1.0333     1.0098
 4      0.9486     1.0226     0.9911
 5      0.9270     1.0113     1.0630
 6      0.8997     1.0127     1.0164
 7      0.8955     1.0138     1.0771
 8      0.8953     0.9953     1.0691
 9      0.8489     1.0462     1.0484
10      0.8449     1.0592     1.0429
11      0.8158     1.0233     1.0625
12      0.7997     1.0384     1.0729
13      0.8025     1.0138     1.0725
14      0.8187     1.0499     1.0554
15      0.8270     1.0459     1.0529
16      0.7914     1.0294     1.0783
17      0.8076     1.0224     1.0943
18      0.8208     1.0359     1.0773
19      0.8190     1.0326     1.0857
20      0.8217     1.0326     1.1095
```

```
21      0.8084     0.9695     1.1389

Analytic solution at final time step
        Path 1     Path 2     Path 3
        0.8079     0.9685     1.1389
```