NAG Toolbox

nag_rand_bb (g05xb)

1 Purpose

nag_rand_bb (g05xb) uses a Brownian bridge algorithm to construct sample paths for a free or non-free Wiener process. The initialization function nag_rand_bb_init (g05xa) must be called prior to the first call to nag_rand_bb (g05xb).

2 Syntax

```
[z, b, ifail] = nag_rand_bb(npaths, start, term, z, c, rcomm, 'rcord', rcord, 'd', d, 'a', a)
[z, b, ifail] = g05xb(npaths, start, term, 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_init (g05xa). 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.

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_init (g05xa).

5.1 Compulsory Input Parameters

1: **npaths** – INTEGER

The number of Wiener sample paths to create.

Constraint: **npaths** ≥ 1.

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

The starting value of the Wiener process.

3: **term**(**d**) – REAL (KIND=nag_wp) array

The terminal value at which the non-free Wiener process should end. If **a** $= 0$, **term** is ignored.

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

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

if **rcord** $= 1$, $ldz \geq$ **d** $\times (n + 1 -$ **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 **d** $\times (n + 1 -$ **a**$)$ if **rcord** $= 2$.

The Normal random numbers used to construct the sample paths.

If **rcord** $= 1$ and quasi-random numbers are used, the **d** $\times (n + 1 -$ **a**$)$, where $n =$ nint **rcomm**(2)-dimensional quasi-random points should be stored in successive columns of **z**.

If **rcord** $= 2$ and quasi-random numbers are used, the **d** $\times (n + 1 -$ **a**$)$, where $n =$ nint **rcomm**(2)-dimensional quasi-random points should be stored in successive rows of **z**.

5: **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.

6: **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_init (g05xa) or nag_rand_bb (g05xb).

Communication array as returned by the last call to nag_rand_bb_init (g05xa) or nag_rand_bb (g05xb). 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 arrays **term**, **start** 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: **d** ≥ 1.

3: **a** – INTEGER

Default: 1

If **a** $= 0$, a free Wiener process is created beginning at **start** and **term** is ignored.

If **a** $= 1$, a non-free Wiener process is created beginning at **start** and ending at **term**.

Constraint: **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 values of the Wiener sample paths.

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**.

If **rcord** $= 1$, the point $X_{p,i}^k$ will be stored at **b**$(k + (i - 1) \times$ **d**$, p)$.

If **rcord** $= 2$, the point $X_{p,i}^k$ will be stored at **b**$(p, k + (i - 1) \times$ **d**$)$.

The starting value **start** is never stored, whereas the terminal value is always stored.

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: $ldb \geq$ **d** $\times$ (**ntimes** $+ 1$).

Constraint: $ldb \geq$ **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

This example calls nag_rand_bb (g05xb), nag_rand_bb_init (g05xa) and nag_rand_bb_make_bridge_order (g05xe) to generate two sample paths of a three-dimensional non-free Wiener process. The process starts at zero and each sample path terminates at the point $(1.0, 0.5, 0.0)$. Quasi-random numbers are used to construct the sample paths.

See Section 10 in nag_rand_bb_init (g05xa) and nag_rand_bb_make_bridge_order (g05xe) for additional examples.

9.1 Program Text

```
function g05xb_example

fprintf('g05xb example results\n\n');

% Get information required to set up the bridge
[bgord,t0,tend,ntimes,intime,nmove,move] = get_bridge_init_data();

% Make the bridge construction bgord
[times, ifail] = g05xe( ...
                        t0, tend, intime, move, 'bgord', bgord);

% Initialize the Brownian bridge generator
[rcomm, ifail] = g05xa( ...
                        t0, tend, times);

% Get additional information required by the bridge generator
[npaths, d, start, term, c] = get_bridge_gen_data();

% Generate the Z values
[z] = get_z(npaths, d, ntimes);

% Call the Brownian bridge generator routine
[z, b, ifail] = g05xb( ...
                       npaths, start, term, z, c, rcomm);

% Display the results
for i = 1:npaths
  fprintf('Weiner Path %d, %d time steps, %d dimensions\n', i, ntimes+1, d);
  w = transpose(reshape(b(:,i), d, ntimes+1));

  ifail = x04ca('G', ' ', w, '');
```

```
  fprintf('\n');
end

function [bgord,t0,tend,ntimes,intime,nmove,move] = get_bridge_init_data()
  % Set the basic parameters for a Wiener process
  t0 = 0;
  n = 10;
  ntimes = nag_int(n);

  % We want to generate the Wiener process at these time points
  intime = 1:n + t0;
  tend = t0 + n + 1;

  nmove= nag_int(0);
  move = zeros(nmove, 1, nag_int_name);
  bgord = nag_int(3);

function [npaths,d,start,term,c] = get_bridge_gen_data();
  % Set the basic parameters for a non-free Wiener process
  npaths = nag_int(2);
  d = 3;
  start = zeros(d, 1);
  term  = [1, 0.5, 0];

  % We want the following covariance matrix
  c = [ 6,   1,   -0.2;
        1,   5,    0.3;
       -0.2, 0.3,  4];

  % Cholesky factorize the covariance matrix c
  [c, info] = f07fd('l', c);

function [z] = get_z(npaths, d, ntimes)
  % Non-free Wiener process (the default)
  a = nag_int(1);
  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

```
     g05xb example results

Weiner Path 1, 11 time steps, 3 dimensions
              1          2          3
  1     -1.0602    -2.8701    -0.9415
  2     -3.0575    -1.9502     0.2596
  3     -6.8274    -2.4434     0.4597
  4     -5.2855    -3.4475     0.0795
  5     -8.1784    -5.2296    -0.0921
  6     -4.6874    -5.0220     1.4862
  7     -3.0959    -4.8623    -4.4076
  8     -2.9605    -1.8936    -3.9539
  9     -5.4685    -2.3856    -3.2031
 10      0.1205    -5.0520    -1.0385
 11      1.0000     0.5000     0.0000

Weiner Path 2, 11 time steps, 3 dimensions
              1          2          3
  1      0.6564     3.5142     1.5911
  2     -2.3773     3.1618     3.0316
  3      0.3020     6.8815     2.0875
  4     -0.2169     4.6026     1.1982
  5     -2.0684     4.1503     2.4758
  6     -5.1075     3.7303     2.7563
  7     -3.8497     3.6682     2.4827
  8     -1.8292     4.4153     0.1916
  9     -2.0649     0.6952    -2.1201
 10      0.1962     1.7769    -5.7685
 11      1.0000     0.5000     0.0000
```