NAG Library Routine Document

G05ZPF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

G05ZPF produces realizations of a stationary Gaussian random field in one dimension, using the circulant embedding method. The square roots of the eigenvalues of the extended covariance matrix (or embedding matrix) need to be input, and can be calculated using G05ZMF or G05ZNF.

2 Specification

```
SUBROUTINE G05ZPF (NS, S, M, LAM, RHO, STATE, Z, IFAIL)

INTEGER              NS, S, M, STATE(*), IFAIL
REAL (KIND=nag_wp) LAM(M), RHO, Z(NS,S)
```

3 Description

A one-dimensional random field $Z(x)$ in $\mathbb{R}$ is a function which is random at every point $x \in \mathbb{R}$, so $Z(x)$ is a random variable for each x. The random field has a mean function $\mu(x) = \mathbb{E}[Z(x)]$ and a symmetric non-negative definite covariance function $C(x,y) = \mathbb{E}[(Z(x) - \mu(x))(Z(y) - \mu(y))]$. $Z(x)$ is a Gaussian random field if for any choice of $n \in \mathbb{N}$ and $x_1, \ldots, x_n \in \mathbb{R}$, the random vector $[Z(x_1), \ldots, Z(x_n)]^{\mathrm{T}}$ follows a multivariate Normal distribution, which would have a mean vector $\tilde{\boldsymbol{\mu}}$ with entries $\tilde{\mu}_i = \mu(x_i)$ and a covariance matrix $\tilde{C}$ with entries $\tilde{C}_{ij} = C(x_i, x_j)$. A Gaussian random field $Z(x)$ is stationary if $\mu(x)$ is constant for all $x \in \mathbb{R}$ and $C(x,y) = C(x+a, y+a)$ for all $x, y, a \in \mathbb{R}$ and hence we can express the covariance function $C(x,y)$ as a function γ of one variable: $C(x,y) = \gamma(x-y)$. γ is known as a variogram (or more correctly, a semivariogram) and includes the multiplicative factor σ^2 representing the variance such that $\gamma(0) = \sigma^2$.

The routines G05ZMF or G05ZNF, along with G05ZPF, are used to simulate a one-dimensional stationary Gaussian random field, with mean function zero and variogram $\gamma(x)$, over an interval $[x_{\mathrm{min}}, x_{\mathrm{max}}]$, using an equally spaced set of N points. The problem reduces to sampling a Normal random vector $\mathbf{X}$ of size N, with mean vector zero and a symmetric Toeplitz covariance matrix A. Since A is in general expensive to factorize, a technique known as the *circulant embedding method* is used. A is embedded into a larger, symmetric circulant matrix B of size $M \geq 2(N-1)$, which can now be factorized as $B = W\Lambda W^* = R^*R$, where W is the Fourier matrix (W^* is the complex conjugate of W), Λ is the diagonal matrix containing the eigenvalues of B and $R = \Lambda^{\frac{1}{2}}W^*$. B is known as the embedding matrix. The eigenvalues can be calculated by performing a discrete Fourier transform of the first row (or column) of B and multiplying by M, and so only the first row (or column) of B is needed – the whole matrix does not need to be formed.

As long as all of the values of Λ are non-negative (i.e., B is non-negative definite), B is a covariance matrix for a random vector $\mathbf{Y}$, two samples of which can now be simulated from the real and imaginary parts of $R^*(\mathbf{U} + i\mathbf{V})$, where $\mathbf{U}$ and $\mathbf{V}$ have elements from the standard Normal distribution. Since $R^*(\mathbf{U} + i\mathbf{V}) = W\Lambda^{\frac{1}{2}}(\mathbf{U} + i\mathbf{V})$, this calculation can be done using a discrete Fourier transform of the vector $\Lambda^{\frac{1}{2}}(\mathbf{U} + i\mathbf{V})$. Two samples of the random vector $\mathbf{X}$ can now be recovered by taking the first N elements of each sample of $\mathbf{Y}$ – because the original covariance matrix A is embedded in B, $\mathbf{X}$ will have the correct distribution.

If B is not non-negative definite, larger embedding matrices B can be tried; however if the size of the matrix would have to be larger than MAXM, an approximation procedure is used. See the documentation of G05ZMF or G05ZNF for details of the approximation procedure.

G05ZPF takes the square roots of the eigenvalues of the embedding matrix B, and its size M, as input and outputs S realizations of the random field in Z.

One of the initialization routines G05KFF (for a repeatable sequence if computed sequentially) or G05KGF (for a non-repeatable sequence) must be called prior to the first call to G05ZPF.

4 References

Dietrich C R and Newsam G N (1997) Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix *SIAM J. Sci. Comput.* **18** 1088–1107

Schlather M (1999) Introduction to positive definite functions and to unconditional simulation of random fields *Technical Report ST 99–10* Lancaster University

Wood A T A and Chan G (1994) Simulation of stationary Gaussian processes in $[0,1]^d$ *Journal of Computational and Graphical Statistics* **3(4)** 409–432

5 Parameters

1: NS – INTEGER *Input*

On entry: the number of sample points to be generated in realizations of the random field. This must be the same value as supplied to G05ZMF or G05ZNF when calculating the eigenvalues of the embedding matrix.

Constraint: $\mathrm{NS} \geq 1$.

2: S – INTEGER *Input*

On entry: S, the number of realizations of the random field to simulate.

Constraint: $\mathrm{S} \geq 1$.

3: M – INTEGER *Input*

On entry: M, the size of the embedding matrix, as returned by G05ZMF or G05ZNF.

Constraint: $\mathrm{M} \geq \max(1, 2(\mathrm{NS} - 1))$.

4: LAM(M) – REAL (KIND=nag_wp) array *Input*

On entry: must contain the square roots of the eigenvalues of the embedding matrix, as returned by G05ZMF or G05ZNF.

Constraint: $\mathrm{LAM}(i) \geq 0, i = 1, 2, \ldots, \mathrm{M}$.

5: RHO – REAL (KIND=nag_wp) *Input*

On entry: indicates the scaling of the covariance matrix, as returned by G05ZMF or G05ZNF.

Constraint: $0.0 < \mathrm{RHO} \leq 1.0$.

6: STATE($*$) – INTEGER array *Communication Array*

Note: the actual argument supplied **must** be the array STATE supplied to the initialization routines G05KFF or G05KGF.

On entry: contains information on the selected base generator and its current state.

On exit: contains updated information on the state of the generator.

7: Z(NS, S) – REAL (KIND=nag_wp) array *Output*

On exit: contains the realizations of the random field. The jth realization, for the NS sample points, is stored in $\mathrm{Z}(i, j)$, for $i = 1, 2, \ldots, \mathrm{NS}$. The sample points are as returned in XX by G05ZMF or G05ZNF.

8: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, NS $= \langle value \rangle$.
Constraint: NS ≥ 1.

IFAIL $= 2$

On entry, S $= \langle value \rangle$.
Constraint: S ≥ 1.

IFAIL $= 3$

On entry, M $= \langle value \rangle$ and NS $= \langle value \rangle$.
Constraint: M $\geq \max(1, 2 \times (\text{NS} - 1))$.

IFAIL $= 4$

On entry, at least one element of LAM was negative.
Constraint: all elements of LAM must be non-negative.

IFAIL $= 5$

On entry, RHO $= \langle value \rangle$.
Constraint: $0.0 \leq \text{RHO} \leq 1.0$.

IFAIL $= 6$

On entry, STATE vector has been corrupted or not initialized.

IFAIL $= -99$

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

G05ZPF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

Because samples are generated in pairs, calling this routine k times, with S $= s$, say, will generate a different sequence of numbers than calling the routine once with S $= ks$, unless s is even.

10 Example

This example calls G05ZPF to generate 5 realizations of a random field on 8 sample points using eigenvalues calculated by G05ZNF for a symmetric stable variogram.

10.1 Program Text

```
!   G05ZPF Example Program Text

!   Mark 25 Release. NAG Copyright 2014.

    Program g05zpfe

!     G05ZPF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: g05znf, g05zpf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                 :: lenst = 17, nin = 5, nout = 6,   &
                                            npmax = 4
!     .. Local Scalars ..
      Real (Kind=nag_wp)                 :: rho, var, xmax, xmin
      Integer                            :: approx, icorr, icount, icov1,    &
                                            ifail, m, maxm, np, ns, pad, s
!     .. Local Arrays ..
      Real (Kind=nag_wp)                 :: eig(3), params(npmax)
      Real (Kind=nag_wp), Allocatable    :: lam(:), xx(:), z(:,:)
      Integer                            :: state(lenst)
!     .. Executable Statements ..
      Write (nout,*) 'G05ZPF Example Program Results'
      Write (nout,*)
      Flush (nout)

!     Get problem specifications from data file
      Call read_input_data(icov1,np,params,var,xmin,xmax,ns,maxm,icorr,pad,s)

      Allocate (lam(maxm),xx(ns))

!     Get square roots of the eigenvalues of the embedding matrix
```

```
      ifail = 0
      Call g05znf(ns,xmin,xmax,maxm,var,icov1,np,params,pad,icorr,lam,xx,m, &
        approx,rho,icount,eig,ifail)

      Call display_embedding_results(approx,m,rho,eig,icount)

!     Initialize state array
      Call initialize_state(state)

      Allocate (z(ns,s))

!     Compute s random field realisations.
      Call g05zpf(ns,s,m,lam,rho,state,z,ifail)

      Call display_realizations(ns,s,xx,z)

    Contains
      Subroutine read_input_data(icov1,np,params,var,xmin,xmax,ns,maxm,icorr, &
        pad,s)

!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: var, xmax, xmin
        Integer, Intent (Out)                :: icorr, icov1, maxm, np, ns,    &
                                                pad, s
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: params(npmax)
!       .. Executable Statements ..
!       Skip heading in data file
        Read (nin,*)

!       Read in covariance function number
        Read (nin,*) icov1

!       Read in number of parameters
        Read (nin,*) np

!       Read in parameters
        If (np>0) Then
          Read (nin,*) params(1:np)
        End If

!       Read in variance of random field
        Read (nin,*) var

!       Read in domain endpoints
        Read (nin,*) xmin, xmax

!       Read in number of sample points
        Read (nin,*) ns

!       Read in maximum size of embedding matrix
        Read (nin,*) maxm

!       Read in choice of scaling in case of approximation
        Read (nin,*) icorr

!       Read in choice of padding
        Read (nin,*) pad

!       Read in number of realization samples to be generated
        Read (nin,*) s

        Return

      End Subroutine read_input_data

      Subroutine display_embedding_results(approx,m,rho,eig,icount)

!       .. Implicit None Statement ..
```

```
        Implicit None
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: rho
        Integer, Intent (In)                 :: approx, icount, m
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: eig(3)
!       .. Executable Statements ..
!       Display size of embedding matrix
        Write (nout,*)
        Write (nout,99999) 'Size of embedding matrix = ', m

!       Display approximation information if approximation used
        Write (nout,*)
        If (approx==1) Then
          Write (nout,*) 'Approximation required'
          Write (nout,*)
          Write (nout,99998) 'RHO = ', rho
          Write (nout,99997) 'EIG = ', eig(1:3)
          Write (nout,99999) 'ICOUNT = ', icount
        Else
          Write (nout,*) 'Approximation not required'
        End If

        Return

99999   Format (1X,A,I7)
99998   Format (1X,A,F10.5)
99997   Format (1X,A,3(F10.5,1X))

      End Subroutine display_embedding_results

      Subroutine initialize_state(state)

!       .. Use Statements ..
        Use nag_library, Only: g05kff
!       .. Implicit None Statement ..
        Implicit None
!       .. Parameters ..
        Integer, Parameter                   :: genid = 1, inseed = 14965,     &
                                                lseed = 1, subid = 1
!       .. Array Arguments ..
        Integer, Intent (Out)                :: state(lenst)
!       .. Local Scalars ..
        Integer                              :: ifail, lstate
!       .. Local Arrays ..
        Integer                              :: seed(lseed)
!       .. Executable Statements ..
!       Initialize the generator to a repeatable sequence
        lstate = lenst
        seed(1) = inseed
        ifail = 0
        Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

      End Subroutine initialize_state

      Subroutine display_realizations(ns,s,xx,z)

!       .. Use Statements ..
        Use nag_library, Only: x04cbf
!       .. Implicit None Statement ..
        Implicit None
!       .. Parameters ..
        Integer, Parameter                   :: indent = 0, ncols = 80
        Character (1), Parameter             :: charlab = 'C', intlab = 'I',   &
                                                matrix = 'G', unit = 'n'
        Character (5), Parameter             :: form = 'F10.5'
!       .. Scalar Arguments ..
        Integer, Intent (In)                 :: ns, s
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: xx(ns), z(ns,s)
!       .. Local Scalars ..
```

```
        Integer                              :: i, ifail
        Character (26)                       :: title
!       .. Local Arrays ..
        Character (1)                        :: clabs(0)
        Character (10), Allocatable          :: rlabs(:)
!       .. Executable Statements ..
        Allocate (rlabs(ns))

!       Set row labels to grid points (column label is realization number).
        Do i = 1, ns
          Write (rlabs(i),99999) xx(i)
        End Do

!       Display random field results
        title = 'Random field realisations:'
        Write (nout,*)
        ifail = 0
        Call x04cbf(matrix,unit,ns,s,z,ns,form,title,charlab,rlabs,intlab, &
          clabs,ncols,indent,ifail)

99999   Format (F10.5)

      End Subroutine display_realizations

    End Program g05zpfe
```

10.2 Program Data

```
G05ZPF Example Program Data
 1             : icov1  (icov=1, symmetric stable)
 2             : np     (icov=1, 2 parameters)
 0.1   1.2     : params (icov=1, l and nu)
 0.5           : var
-1     1       : xmin, xmax
 8             : ns
 2048          : maxm
 2             : icorr
 1             : pad
 5             : s
```

10.3 Program Results

```
 G05ZPF Example Program Results


 Size of embedding matrix =      16

 Approximation not required

 Random field realisations:
                    1         2         3         4         5
 -0.87500    -0.41663  -0.81847  -0.97692   0.67410  -0.67616
 -0.62500     0.01457   1.45384   0.02481   0.52178   1.94664
 -0.37500    -0.55557   0.29127  -0.08534   0.42145  -0.13891
 -0.12500    -0.55678   0.31985  -0.60936   0.20194   0.90846
  0.12500    -0.04230   0.04860   1.45897   0.36077  -0.52877
  0.37500    -0.28057  -0.79688   0.23301   0.13351   0.40119
  0.62500     0.92981  -0.39561  -0.84545  -0.27487   0.52703
  0.87500     0.32217   1.52273  -2.16445   0.17941   1.19373
```