NAG Library Routine Document

G05ZMF

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

G05ZMF performs the setup required in order to simulate stationary Gaussian random fields in one dimension, for a user-defined variogram, using the *circulant embedding method*. Specifically, the eigenvalues of the extended covariance matrix (or embedding matrix) are calculated, and their square roots output, for use by G05ZPF, which simulates the random field.

2 Specification

```
SUBROUTINE G05ZMF (NS, XMIN, XMAX, MAXM, VAR, COV1, PAD, ICORR, LAM, XX,    &
                   M, APPROX, RHO, ICOUNT, EIG, IUSER, RUSER, IFAIL)
INTEGER            NS, MAXM, PAD, ICORR, M, APPROX, ICOUNT, IUSER(*),      &
                   IFAIL
REAL (KIND=nag_wp) XMIN, XMAX, VAR, LAM(MAXM), XX(NS), RHO, EIG(3),        &
                   RUSER(*)
EXTERNAL           COV1
```

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 positive semidefinite 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 and G05ZPF are used to simulate a one-dimensional stationary Gaussian random field, with mean function zero and variogram $\gamma(x)$, over an interval $[x_{\min}, x_{\max}]$, using an equally spaced set of N points on the interval. 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 positive semidefinite), 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 positive semidefinite, 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. We write $\Lambda = \Lambda_+ + \Lambda_-$, where Λ_+ and Λ_- contain the non-negative and negative eigenvalues of B respectively. Then B is replaced by ρB_+ where $B_+ = W\Lambda_+ W^*$ and $\rho \in (0, 1]$ is a scaling factor. The error ϵ in approximating the distribution of the random field is given by

$$\epsilon = \sqrt{\frac{(1-\rho)^2 \operatorname{trace} \Lambda + \rho^2 \operatorname{trace} \Lambda_-}{M}}.$$

Three choices for ρ are available, and are determined by the input parameter ICORR:

setting ICORR $= 0$ sets

$$\rho = \frac{\operatorname{trace} \Lambda}{\operatorname{trace} \Lambda_+},$$

setting ICORR $= 1$ sets

$$\rho = \sqrt{\frac{\operatorname{trace} \Lambda}{\operatorname{trace} \Lambda_+}},$$

setting ICORR $= 2$ sets $\rho = 1$.

G05ZMF finds a suitable positive semidefinite embedding matrix B and outputs its size, M, and the square roots of its eigenvalues in LAM. If approximation is used, information regarding the accuracy of the approximation is output. Note that only the first row (or column) of B is actually formed and stored.

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.

Constraint: NS ≥ 1.

2: XMIN – REAL (KIND=nag_wp) *Input*

On entry: the lower bound for the interval over which the random field is to be simulated.

Constraint: XMIN $<$ XMAX.

3: XMAX – REAL (KIND=nag_wp) *Input*

On entry: the upper bound for the interval over which the random field is to be simulated.

Constraint: XMIN $<$ XMAX.

4: MAXM – INTEGER *Input*

On entry: the maximum size of the circulant matrix to use. For example, if the embedding matrix is to be allowed to double in size three times before the approximation procedure is used, then choose MAXM $= 2^{k+2}$ where $k = 1 + \lceil \log_2 (\text{NS} - 1) \rceil$.

Suggested value: 2^{k+2} where $k = 1 + \lceil \log_2 (\mathrm{NS} - 1) \rceil$

Constraint: MAXM $\geq 2^k$, where k is the smallest integer satisfying $2^k \geq 2(\mathrm{NS} - 1)$.

5: VAR – REAL (KIND=nag_wp) *Input*

On entry: the multiplicative factor σ^2 of the variogram $\gamma(x)$.

Constraint: VAR ≥ 0.0.

6: COV1 – SUBROUTINE, supplied by the user. *External Procedure*

COV1 must evaluate the variogram $\gamma(x)$, without the multiplicative factor σ^2, for all $x \geq 0$. The value returned in GAMMA is multiplied internally by VAR.

The specification of COV1 is:

```
SUBROUTINE COV1 (X, GAMMA, IUSER, RUSER)
INTEGER            IUSER(*)
REAL (KIND=nag_wp) X, GAMMA, RUSER(*)
```

1: X – REAL (KIND=nag_wp) *Input*

On entry: the value x at which the variogram $\gamma(x)$ is to be evaluated.

2: GAMMA – REAL (KIND=nag_wp) *Output*

On exit: the value of the variogram $\dfrac{\gamma(x)}{\sigma^2}$.

3: IUSER$(*)$ – INTEGER array *User Workspace*

4: RUSER$(*)$ – REAL (KIND=nag_wp) array *User Workspace*

COV1 is called with the parameters IUSER and RUSER as supplied to G05ZMF. You are free to use the arrays IUSER and RUSER to supply information to COV1 as an alternative to using COMMON global variables.

COV1 must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which G05ZMF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

7: PAD – INTEGER *Input*

On entry: determines whether the embedding matrix is padded with zeros, or padded with values of the variogram. The choice of padding may affect how big the embedding matrix must be in order to be positive semidefinite.

PAD $= 0$
The embedding matrix is padded with zeros.

PAD $= 1$
The embedding matrix is padded with values of the variogram.

Suggested value: PAD $= 1$.

Constraint: PAD $= 0$ or 1.

8: ICORR – INTEGER *Input*

On entry: determines which approximation to implement if required, as described in Section 3.

Suggested value: ICORR $= 0$.

Constraint: ICORR $= 0$, 1 or 2.

9: LAM(MAXM) – REAL (KIND=nag_wp) array *Output*

On exit: contains the square roots of the eigenvalues of the embedding matrix.

10: XX(NS) – REAL (KIND=nag_wp) array *Output*

On exit: the points at which values of the random field will be output.

11: M – INTEGER *Output*

On exit: the size of the embedding matrix.

12: APPROX – INTEGER *Output*

On exit: indicates whether approximation was used.

APPROX $= 0$
No approximation was used.

APPROX $= 1$
Approximation was used.

13: RHO – REAL (KIND=nag_wp) *Output*

On exit: indicates the scaling of the covariance matrix. RHO $= 1.0$ unless approximation was used with ICORR $= 0$ or 1.

14: ICOUNT – INTEGER *Output*

On exit: indicates the number of negative eigenvalues in the embedding matrix which have had to be set to zero.

15: EIG(3) – REAL (KIND=nag_wp) array *Output*

On exit: indicates information about the negative eigenvalues in the embedding matrix which have had to be set to zero. EIG(1) contains the smallest eigenvalue, EIG(2) contains the sum of the squares of the negative eigenvalues, and EIG(3) contains the sum of the absolute values of the negative eigenvalues.

16: IUSER(∗) – INTEGER array *User Workspace*

17: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by G05ZMF, but are passed directly to COV1 and may be used to pass information to this routine as an alternative to using COMMON global variables.

18: 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, XMIN $= \langle value \rangle$ and XMAX $= \langle value \rangle$.
Constraint: XMIN $<$ XMAX.

IFAIL $= 4$

On entry, MAXM $= \langle value \rangle$.
Constraint: the minimum calculated value for MAXM is $\langle value \rangle$.
Where the minimum calculated value is given by 2^k, where k is the smallest integer satisfying $2^k \geq 2(\text{NS} - 1)$.

IFAIL $= 5$

On entry, VAR $= \langle value \rangle$.
Constraint: VAR ≥ 0.0.

IFAIL $= 7$

On entry, PAD $= \langle value \rangle$.
Constraint: PAD $= 0$ or 1.

IFAIL $= 8$

On entry, ICORR $= \langle value \rangle$.
Constraint: ICORR $= 0$, 1 or 2.

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

If on exit APPROX $= 1$, see the comments in Section 3 regarding the quality of approximation; increase the value of MAXM to attempt to avoid approximation.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example calls G05ZMF to calculate the eigenvalues of the embedding matrix for 8 sample points of a random field characterized by the symmetric stable variogram:

$$\gamma(x) = \sigma^2 \exp\bigl(-(x')^{\nu}\bigr),$$

where $x' = \frac{x}{\ell}$, and ℓ and ν are parameters.

It should be noted that the symmetric stable variogram is one of the pre-defined variograms available in G05ZNF. It is used here purely for illustrative purposes.

10.1 Program Text

```
!   G05ZMF Example Program Text

!   Mark 25 Release. NAG Copyright 2014.

    Module g05zmfe_mod

!     G05ZMF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                                :: cov1
    Contains
      Subroutine cov1(t,gamma,iuser,ruser)

!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: gamma
        Real (Kind=nag_wp), Intent (In)      :: t
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout)   :: ruser(*)
        Integer, Intent (Inout)              :: iuser(*)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                   :: dummy, l, nu
!       .. Intrinsic Procedures ..
        Intrinsic                            :: abs, exp
!       .. Executable Statements ..
!       Correlation length in ruser(1)
        l = ruser(1)
!       Exponent in ruser(2)
        nu = ruser(2)

        If (t==0.0_nag_wp) Then
          gamma = 1.0_nag_wp
        Else
          dummy = (abs(t)/l)**nu
          gamma = exp(-dummy)
        End If

        Return

      End Subroutine cov1
    End Module g05zmfe_mod

    Program g05zmfe
```

```
!     G05ZMF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: g05zmf, nag_wp
      Use g05zmfe_mod, Only: cov1
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                  :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)                  :: l, nu, rho, var, xmax, xmin
      Integer                             :: approx, icorr, icount, ifail, m, &
                                             maxm, ns, pad
!     .. Local Arrays ..
      Real (Kind=nag_wp)                  :: eig(3), ruser(2)
      Real (Kind=nag_wp), Allocatable     :: lam(:), xx(:)
      Integer                             :: iuser(0)
!     .. Executable Statements ..
      Write (nout,*) 'G05ZMF Example Program Results'
      Write (nout,*)

!     Get problem specifications from data file
      Call read_input_data(l,nu,var,xmin,xmax,ns,maxm,icorr,pad)

!     Put covariance parameters in communication array
      ruser(1) = l
      ruser(2) = nu

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

!     Get square roots of the eigenvalues of the embedding matrix
      ifail = 0
      Call g05zmf(ns,xmin,xmax,maxm,var,cov1,pad,icorr,lam,xx,m,approx,rho, &
        icount,eig,iuser,ruser,ifail)

!     Output results
      Call display_results(approx,m,rho,eig,icount,lam)

    Contains
      Subroutine read_input_data(l,nu,var,xmin,xmax,ns,maxm,icorr,pad)

!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: l, nu, var, xmax, xmin
        Integer, Intent (Out)                :: icorr, maxm, ns, pad
!       .. Executable Statements ..
!       Skip heading in data file
        Read (nin,*)

!       Read in l and nu for cov1 function
        Read (nin,*) l, nu

!       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
```

```
        Return

      End Subroutine read_input_data

      Subroutine display_results(approx,m,rho,eig,icount,lam)

!       .. 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), lam(m)
!       .. 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

!       Display square roots of the eigenvalues of the embedding matrix
        Write (nout,*)
        Write (nout,*) 'Square roots of eigenvalues of embedding matrix:'
        Write (nout,*)
        Write (nout,99996) lam(1:m)

        Return

99999   Format (1X,A,I7)
99998   Format (1X,A,F10.5)
99997   Format (1X,A,3(F10.5,1X))
99996   Format (1X,4F10.5)

      End Subroutine display_results

    End Program g05zmfe
```

10.2 Program Data

```
G05ZMF Example Program Data
 0.1    1.2    : l, nu
 0.5           : var
-1.0    1.0    : xmin, xmax
 8             : ns
 2048          : maxm
 2             : icorr
 1             : pad
```

10.3 Program Results

```
 G05ZMF Example Program Results


 Size of embedding matrix =      16

 Approximation not required

 Square roots of eigenvalues of embedding matrix:
```

```
 0.74207   0.73932   0.73150   0.71991
 0.70639   0.69304   0.68184   0.67442
 0.67182   0.67442   0.68184   0.69304
 0.70639   0.71991   0.73150   0.73932
```