NAG Library Routine Document

G05ZNF

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

G05ZNF performs the setup required in order to simulate stationary Gaussian random fields in one dimension, for a preset 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 G05ZNF (NS, XMIN, XMAX, MAXM, VAR, ICOV1, NP, PARAMS, PAD,      &
                   ICORR, LAM, XX, M, APPROX, RHO, ICOUNT, EIG, IFAIL)
INTEGER            NS, MAXM, ICOV1, NP, PAD, ICORR, M, APPROX, ICOUNT,     &
                   IFAIL
REAL (KIND=nag_wp) XMIN, XMAX, VAR, PARAMS(NP), LAM(MAXM), XX(NS), RHO,     &
                   EIG(3)
```

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 G05ZNF 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. 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 argument 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 $\rho = 1$.

G05ZNF 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 (1997) Algorithm AS 312: An Algorithm for Simulating Stationary Gaussian Random Fields *Journal of the Royal Statistical Society, Series C (Applied Statistics) (Volume 46)* **1** 171–181

5 Arguments

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. Note that if ICOV1 $= 14$ (for simulating fractional Brownian motion), XMIN is not referenced and the lower bound for the interval is set to zero.

Constraint: if ICOV1 $\neq 14$, 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. Note that if ICOV1 $= 14$ (for simulating fractional Brownian motion), the lower bound for the interval is set to zero and so XMAX is required to be greater than zero.

Constraints:

if ICOV1 $\neq 14$, XMIN $<$ XMAX;
if ICOV1 $= 14$, XMAX > 0.0.

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 $k = 1 + \lceil \log_2 (\mathrm{NS} - 1) \rceil$.

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: ICOV1 – INTEGER *Input*

On entry: determines which of the preset variograms to use. The choices are given below. Note that $x' = \frac{|x|}{\ell}$, where ℓ is the correlation length and is a parameter for most of the variograms, and σ^2 is the variance specified by VAR.

ICOV1 $= 1$
Symmetric stable variogram

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

where

$\ell = \mathrm{PARAMS}(1)$, $\ell > 0$,

$\nu = \mathrm{PARAMS}(2)$, $0 \leq \nu \leq 2$.

ICOV1 $= 2$
Cauchy variogram

$$\gamma(x) = \sigma^2 \left(1 + (x')^2\right)^{-\nu},$$

where

$\ell = \mathrm{PARAMS}(1)$, $\ell > 0$,

$\nu = \mathrm{PARAMS}(2)$, $\nu > 0$.

ICOV1 $= 3$
Differential variogram with compact support

$$\gamma(x) = \begin{cases} \sigma^2 \left(1 + 8x' + 25(x')^2 + 32(x')^3\right)(1 - x')^8, & x' < 1, \\ 0, & x' \geq 1, \end{cases}$$

where

$\ell = \mathrm{PARAMS}(1)$, $\ell > 0$.

ICOV1 $= 4$
Exponential variogram

$$\gamma(x) = \sigma^2 \exp(-x'),$$

where

$\ell = \mathrm{PARAMS}(1)$, $\ell > 0$.

ICOV1 = 5
Gaussian variogram

$$\gamma(x) = \sigma^2 \exp\left(-(x')^2\right),$$

where

$\ell = \text{PARAMS}(1)$, $\ell > 0$.

ICOV1 = 6
Nugget variogram

$$\gamma(x) = \begin{cases} \sigma^2, & x = 0, \\ 0, & x \neq 0. \end{cases}$$

No parameters need be set for this value of ICOV1.

ICOV1 = 7
Spherical variogram

$$\gamma(x) = \begin{cases} \sigma^2\left(1 - 1.5x' + 0.5(x')^3\right), & x' < 1, \\ 0, & x' \geq 1, \end{cases}$$

where

$\ell = \text{PARAMS}(1)$, $\ell > 0$.

ICOV1 = 8
Bessel variogram

$$\gamma(x) = \sigma^2 \frac{2^\nu \Gamma(\nu+1) J_\nu(x')}{(x')^\nu},$$

where

$J_\nu(\cdot)$ is the Bessel function of the first kind,

$\ell = \text{PARAMS}(1)$, $\ell > 0$,

$\nu = \text{PARAMS}(2)$, $\nu \geq -0.5$.

ICOV1 = 9
Hole effect variogram

$$\gamma(x) = \sigma^2 \frac{\sin(x')}{x'},$$

where

$\ell = \text{PARAMS}(1)$, $\ell > 0$.

ICOV1 = 10
Whittle-Matérn variogram

$$\gamma(x) = \sigma^2 \frac{2^{1-\nu}(x')^\nu K_\nu(x')}{\Gamma(\nu)},$$

where

$K_\nu(\cdot)$ is the modified Bessel function of the second kind,

$\ell = \text{PARAMS}(1)$, $\ell > 0$,

$\nu = \text{PARAMS}(2)$, $\nu > 0$.

ICOV1 = 11

Continuously parameterised variogram with compact support

$$\gamma(x) = \begin{cases} \sigma^2 \frac{2^{1-\nu}(x')^{\nu} K_{\nu}(x')}{\Gamma(\nu)} \Big(1 + 8x'' + 25(x'')^2 + 32(x'')^3\Big)(1-x'')^8, & x'' < 1, \\ 0, & x'' \geq 1, \end{cases}$$

where

$x'' = \frac{x'}{s}$,

$K_{\nu}(\cdot)$ is the modified Bessel function of the second kind,

$\ell =$ PARAMS(1), $\ell > 0$,

$s =$ PARAMS(2), $s > 0$ (second correlation length),

$\nu =$ PARAMS(3), $\nu > 0$.

ICOV1 = 12

Generalized hyperbolic distribution variogram

$$\gamma(x) = \sigma^2 \frac{\Big(\delta^2 + (x')^2\Big)^{\frac{\lambda}{2}}}{\delta^{\lambda} K_{\lambda}(\kappa\delta)} K_{\lambda}\Big(\kappa\Big(\delta^2 + (x')^2\Big)^{\frac{1}{2}}\Big),$$

where

$K_{\lambda}(\cdot)$ is the modified Bessel function of the second kind,

$\ell =$ PARAMS(1), $\ell > 0$,

$\lambda =$ PARAMS(2), no constraint on λ

$\delta =$ PARAMS(3), $\delta > 0$,

$\kappa =$ PARAMS(4), $\kappa > 0$.

ICOV1 = 13

Cosine variogram

$$\gamma(x) = \sigma^2 \cos(x'),$$

where

$\ell =$ PARAMS(1), $\ell > 0$.

ICOV1 = 14

Used for simulating fractional Brownian motion $B^H(t)$. Fractional Brownian motion itself is not a stationary Gaussian random field, but its increments $\tilde{X}(i) = B^H(t_i) - B^H(t_{i-1})$ can be simulated in the same way as a stationary random field. The variogram for the so-called 'increment process' is

$$C\big(\tilde{X}(t_i), \tilde{X}\big(t_j\big)\big) = \tilde{\gamma}(x) = \frac{\delta^{2H}}{2}\Big(\Big|\frac{x}{\delta} - 1\Big|^{2H} + \Big|\frac{x}{\delta} + 1\Big|^{2H} - 2\Big|\frac{x}{\delta}\Big|^{2H}\Big),$$

where

$x = t_j - t_i$,

$H =$ PARAMS(1), $0 < H < 1$, H is the Hurst parameter,

$\delta =$ PARAMS(2), $\delta > 0$, normally $\delta = t_i - t_{i-1}$ is the (fixed) stepsize.

We scale the increments to set $\gamma(0) = 1$; let $X(i) = \frac{\tilde{X}(i)}{\delta^{-H}}$, then

$$C\big(X(t_i), X\big(t_j\big)\big) = \gamma(x) = \frac{1}{2}\Big(\Big|\frac{x}{\delta} - 1\Big|^{2H} + \Big|\frac{x}{\delta} + 1\Big|^{2H} - 2\Big|\frac{x}{\delta}\Big|^{2H}\Big).$$

The increments $X(i)$ can then be simulated using G05ZPF, then multiplied by δ^H to obtain the original increments $\tilde{X}(i)$ for the fractional Brownian motion.

Constraint: ICOV1 = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 or 14.

7: NP – INTEGER *Input*

On entry: the number of parameters to be set. Different variograms need a different number of parameters.

ICOV1 = 6
NP must be set to 0.

ICOV1 = 3, 4, 5, 7, 9 or 13
NP must be set to 1.

ICOV1 = 1, 2, 8, 10 or 14
NP must be set to 2.

ICOV1 = 11
NP must be set to 3.

ICOV1 = 12
NP must be set to 4.

8: PARAMS(NP) – REAL (KIND=nag_wp) array *Input*

On entry: the parameters set for the variogram.

Constraint: see ICOV1 for a description of the individual parameter constraints.

9: 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.

10: 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.

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

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

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

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

13: M – INTEGER *Output*

On exit: the size of the embedding matrix.

14: APPROX – INTEGER *Output*

On exit: indicates whether approximation was used.

APPROX = 0
No approximation was used.

APPROX = 1
Approximation was used.

15: 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.

16: ICOUNT – INTEGER *Output*

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

17: 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.

18: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, −1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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 argument, 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 = ⟨*value*⟩.
Constraint: NS ≥ 1.

IFAIL = 2

On entry, ICOV1 ≠ 14, XMIN = ⟨*value*⟩ and XMAX = ⟨*value*⟩.
Constraint: XMIN < XMAX.

IFAIL = 3

On entry, ICOV1 = 14 and XMAX = ⟨*value*⟩.
Constraint: XMAX > 0.0.

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 $= 6$

On entry, ICOV1 $= \langle value \rangle$.
Constraint: ICOV1 ≥ 1 and ICOV1 ≤ 14.

IFAIL $= 7$

On entry, NP $= \langle value \rangle$.
Constraint: for ICOV1 $= \langle value \rangle$, NP $= \langle value \rangle$.

IFAIL $= 8$

On entry, PARAMS($\langle value \rangle$) $= \langle value \rangle$.
Constraint: dependent on ICOV1.

IFAIL $= 9$

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

IFAIL $= 10$

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 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation 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

G05ZNF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

G05ZNF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

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

None.

10 Example

This example calls G05ZNF to calculate the eigenvalues of the embedding matrix for 8 sample points of a random field characterized by the symmetric stable variogram (ICOV1 = 1).

10.1 Program Text

```
!   G05ZNF Example Program Text

!   Mark 26 Release. NAG Copyright 2016.

    Program g05znfe

!     G05ZNF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: g05znf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: 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
!     .. Local Arrays ..
      Real (Kind=nag_wp)               :: eig(3), params(npmax)
      Real (Kind=nag_wp), Allocatable  :: lam(:), xx(:)
!     .. Executable Statements ..
      Write (nout,*) 'G05ZNF Example Program Results'
      Write (nout,*)

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

      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)

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

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

!       .. 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
!       .. 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

        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 g05znfe
```

10.2 Program Data

```
G05ZNF 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
```

10.3 Program Results

```
 G05ZNF 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
```