NAG Library Routine Document

G05ZQF

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

G05ZQF performs the setup required in order to simulate stationary Gaussian random fields in two dimensions, 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 G05ZSF, which simulates the random field.

2 Specification

```
SUBROUTINE G05ZQF (NS, XMIN, XMAX, YMIN, YMAX, MAXM, VAR, COV2, EVEN,      &
                   PAD, ICORR, LAM, XX, YY, M, APPROX, RHO, ICOUNT, EIG,    &
                   IUSER, RUSER, IFAIL)

INTEGER            NS(2), MAXM(2), EVEN, PAD, ICORR, M(2), APPROX,         &
                   ICOUNT, IUSER(*), IFAIL
REAL (KIND=nag_wp) XMIN, XMAX, YMIN, YMAX, VAR, LAM(MAXM(1)*MAXM(2)),      &
                   XX(NS(1)), YY(NS(2)), RHO, EIG(3), RUSER(*)
EXTERNAL           COV2
```

3 Description

A two-dimensional random field $Z(\mathbf{x})$ in $\mathbb{R}^2$ is a function which is random at every point $\mathbf{x} \in \mathbb{R}^2$, so $Z(\mathbf{x})$ is a random variable for each $\mathbf{x}$. The random field has a mean function $\mu(\mathbf{x}) = \mathbb{E}[Z(\mathbf{x})]$ and a symmetric positive semidefinite covariance function $C(\mathbf{x},\mathbf{y}) = \mathbb{E}[(Z(\mathbf{x}) - \mu(\mathbf{x}))(Z(\mathbf{y}) - \mu(\mathbf{y}))]$. $Z(\mathbf{x})$ is a Gaussian random field if for any choice of $n \in \mathbb{N}$ and $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^2$, the random vector $[Z(\mathbf{x}_1), \ldots, Z(\mathbf{x}_n)]^{\mathrm{T}}$ follows a multivariate Normal distribution, which would have a mean vector $\tilde{\boldsymbol{\mu}}$ with entries $\tilde{\mu}_i = \mu(\mathbf{x}_i)$ and a covariance matrix $\tilde{C}$ with entries $\tilde{C}_{ij} = C\left(\mathbf{x}_i, \mathbf{x}_j\right)$. A Gaussian random field $Z(\mathbf{x})$ is stationary if $\mu(\mathbf{x})$ is constant for all $\mathbf{x} \in \mathbb{R}^2$ and $C(\mathbf{x},\mathbf{y}) = C(\mathbf{x}+\mathbf{a},\mathbf{y}+\mathbf{a})$ for all $\mathbf{x},\mathbf{y},\mathbf{a} \in \mathbb{R}^2$ and hence we can express the covariance function $C(\mathbf{x},\mathbf{y})$ as a function γ of one variable: $C(\mathbf{x},\mathbf{y}) = \gamma(\mathbf{x}-\mathbf{y})$. 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 G05ZQF and G05ZSF are used to simulate a two-dimensional stationary Gaussian random field, with mean function zero and variogram $\gamma(\mathbf{x})$, over a domain $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, using an equally spaced set of $N_1 \times N_2$ points; N_1 points in the x-direction and N_2 points in the y-direction. The problem reduces to sampling a Normal random vector $\mathbf{X}$ of size $N_1 \times N_2$, with mean vector zero and a symmetric covariance matrix A, which is an N_2 by N_2 block Toeplitz matrix with Toeplitz blocks of size N_1 by N_1. 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 matrix B, which is an M_2 by M_2 block circulant matrix with circulant blocks of size M_1 by M_1, where $M_1 \geq 2(N_1 - 1)$ and $M_2 \geq 2(N_2 - 1)$. B can now be factorized as $B = W\Lambda W^* = R^*R$, where W is the two-dimensional 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 $M_1 \times M_2$, and so only the first row (or column) of B is needed – the whole matrix does not need to be formed.

The symmetry of A as a block matrix, and the symmetry of each block of A, depends on whether the variogram γ is even or not. γ is even in its first coordinate if $\gamma\left(\left[-x_1, x_2\right]^{\mathrm{T}}\right) = \gamma\left(\left[x_1, x_2\right]^{\mathrm{T}}\right)$, even in its second coordinate if $\gamma\left(\left[x_1, -x_2\right]^{\mathrm{T}}\right) = \gamma\left(\left[x_1, x_2\right]^{\mathrm{T}}\right)$, and even if it is even in both coordinates (in two

dimensions it is impossible for γ to be even in one coordinate and uneven in the other). If γ is even then A is a symmetric block matrix and has symmetric blocks; if γ is uneven then A is not a symmetric block matrix and has non-symmetric blocks. In the uneven case, M_1 and M_2 are set to be odd in order to guarantee symmetry in B.

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 $\mathbf{Y}$ which has M_2 blocks of size M_1. Two samples of $\mathbf{Y}$ 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_1 elements of the first N_2 blocks 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 $\rho = 1$.

G05ZQF finds a suitable positive semidefinite embedding matrix B and outputs its sizes in the vector 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 Arguments

1: NS(2) – INTEGER array *Input*

On entry: the number of sample points to use in each direction, with NS(1) sample points in the x-direction, N_1 and NS(2) sample points in the y-direction, N_2. The total number of sample points on the grid is therefore NS(1) $\times$ NS(2).

Constraints:

$\text{NS}(1) \geq 1$;
$\text{NS}(2) \geq 1$.

2: XMIN – REAL (KIND=nag_wp) *Input*

On entry: the lower bound for the x-coordinate, for the region in 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 x-coordinate, for the region in which the random field is to be simulated.

Constraint: XMIN $<$ XMAX.

4: YMIN – REAL (KIND=nag_wp) *Input*

On entry: the lower bound for the y-coordinate, for the region in which the random field is to be simulated.

Constraint: YMIN $<$ YMAX.

5: YMAX – REAL (KIND=nag_wp) *Input*

On entry: the upper bound for the y-coordinate, for the region in which the random field is to be simulated.

Constraint: YMIN $<$ YMAX.

6: MAXM(2) – INTEGER array *Input*

On entry: determines the maximum size of the circulant matrix to use – a maximum of MAXM(1) elements in the x-direction, and a maximum of MAXM(2) elements in the y-direction. The maximum size of the circulant matrix is thus MAXM(1)$\times$MAXM(2).

Constraints:

if EVEN $= 1$, $\text{MAXM}(i) \geq 2^k$, where k is the smallest integer satisfying $2^k \geq 2(\text{NS}(i) - 1)$, for $i = 1, 2$;
if EVEN $= 0$, $\text{MAXM}(i) \geq 3^k$, where k is the smallest integer satisfying $3^k \geq 2(\text{NS}(i) - 1)$, for $i = 1, 2$.

7: VAR – REAL (KIND=nag_wp) *Input*

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

Constraint: VAR ≥ 0.0.

8: COV2 – SUBROUTINE, supplied by the user. *External Procedure*

COV2 must evaluate the variogram $\gamma(\mathbf{x})$ for all $\mathbf{x}$ if EVEN $= 0$, and for all $\mathbf{x}$ with non-negative entries if EVEN $= 1$. The value returned in GAMMA is multiplied internally by VAR.

The specification of COV2 is:

```
SUBROUTINE COV2 (X, Y, GAMMA, IUSER, RUSER)
INTEGER            IUSER(*)
REAL (KIND=nag_wp) X, Y, GAMMA, RUSER(*)
```

1: X – REAL (KIND=nag_wp) *Input*

On entry: the coordinate x at which the variogram $\gamma(\mathbf{x})$ is to be evaluated.

2: Y – REAL (KIND=nag_wp) *Input*

On entry: the coordinate y at which the variogram $\gamma(\mathbf{x})$ is to be evaluated.

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

On exit: the value of the variogram $\gamma(\mathbf{x})$.

4: IUSER(∗) – INTEGER array *User Workspace*
5: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

COV2 is called with the arguments IUSER and RUSER as supplied to G05ZQF. You should use the arrays IUSER and RUSER to supply information to COV2.

COV2 must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which G05ZQF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

9: EVEN – INTEGER *Input*

On entry: indicates whether the covariance function supplied is even or uneven.

EVEN = 0
The covariance function is uneven.

EVEN = 1
The covariance function is even.

Constraint: EVEN = 0 or 1.

10: 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.

11: 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.

12: LAM(MAXM(1) × MAXM(2)) – REAL (KIND=nag_wp) array *Output*

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

13: XX(NS(1)) – REAL (KIND=nag_wp) array *Output*

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

14: YY(NS(2)) – REAL (KIND=nag_wp) array *Output*

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

15: M(2) – INTEGER array *Output*

On exit: M(1) contains M_1, the size of the circulant blocks and M(2) contains M_2, the number of blocks, resulting in a final square matrix of size $M_1 \times M_2$.

16: APPROX – INTEGER *Output*

On exit: indicates whether approximation was used.

APPROX = 0
No approximation was used.

APPROX = 1
Approximation was used.

17: 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.

18: ICOUNT – INTEGER *Output*

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

19: 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.

20: IUSER(∗) – INTEGER array *User Workspace*
21: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by G05ZQF, but are passed directly to COV2 and should be used to pass information to this routine.

22: 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*⟩, ⟨*value*⟩].
Constraint: NS(1) ≥ 1, NS(2) ≥ 1.

IFAIL $= 2$

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

IFAIL $= 4$

On entry, YMIN $= \langle value \rangle$ and YMAX $= \langle value \rangle$.
Constraint: YMIN $<$ YMAX.

IFAIL $= 6$

On entry, MAXM $= [\langle value \rangle, \langle value \rangle]$.
Constraint: the minima for MAXM are $[\langle value \rangle, \langle value \rangle]$.
Where, if EVEN $= 1$, the minimum calculated value of MAXM(i) is given by 2^k, where k is the smallest integer satisfying $2^k \geq 2(\text{NS}(i) - 1)$, and if EVEN $= 0$, the minimum calculated value of MAXM(i) is given by 3^k, where k is the smallest integer satisfying $3^k \geq 2(\text{NS}(i) - 1)$, for $i = 1, 2$.

IFAIL $= 7$

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

IFAIL $= 9$

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

IFAIL $= 10$

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

IFAIL $= 11$

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 values in MAXM to attempt to avoid approximation.

8 Parallelism and Performance

G05ZQF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

G05ZQF 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 G05ZQF to calculate the eigenvalues of the embedding matrix for 25 sample points on a 5 by 5 grid of a two-dimensional random field characterized by the symmetric stable variogram:

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

where $x' = \left|\frac{x}{\ell_1} + \frac{y}{\ell_2}\right|$, and ℓ_1, ℓ_2 and ν are parameters.

It should be noted that the symmetric stable variogram is one of the pre-defined variograms available in G05ZRF. It is used here purely for illustrative purposes.

10.1 Program Text

```
!   G05ZQF Example Program Text

!   Mark 26 Release. NAG Copyright 2016.

    Module g05zqfe_mod

!     G05ZQF 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                           :: cov2
!     .. Parameters ..
      Integer, Parameter, Public       :: even = 1
    Contains
      Subroutine cov2(t1,t2,gamma,iuser,ruser)

!       .. Implicit None Statement ..
        Implicit None
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: gamma
        Real (Kind=nag_wp), Intent (In) :: t1, t2
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
        Integer, Intent (Inout)        :: iuser(*)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: l1, l2, nu, rnorm, tl1, tl2
        Integer                        :: norm
!       .. Intrinsic Procedures ..
        Intrinsic                      :: abs, exp, sqrt
!       .. Executable Statements ..
```

```
!       Covariance parameters stored in ruser array.
        norm = iuser(1)
        l1 = ruser(1)
        l2 = ruser(2)
        nu = ruser(3)

        tl1 = abs(t1)/l1
        tl2 = abs(t2)/l2
        If (norm==1) Then
          rnorm = tl1 + tl2
        Else If (norm==2) Then
          rnorm = sqrt(tl1**2+tl2**2)
        End If

        gamma = exp(-(rnorm**nu))

        Return

      End Subroutine cov2
    End Module g05zqfe_mod

    Program g05zqfe

!     G05ZQF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: g05zqf, nag_wp
      Use g05zqfe_mod, Only: cov2, even
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: l1, l2, nu, rho, var, xmax, xmin,    &
                                         ymax, ymin
      Integer                         :: approx, icorr, icount, ifail, norm,  &
                                         pad
!     .. Local Arrays ..
      Real (Kind=nag_wp)              :: eig(3), ruser(3)
      Real (Kind=nag_wp), Allocatable :: lam(:), xx(:), yy(:)
      Integer                         :: iuser(1), m(2), maxm(2), ns(2)
!     .. Executable Statements ..
      Write (nout,*) 'G05ZQF Example Program Results'
      Write (nout,*)

!     Get problem specifications from data file
      Call read_input_data(norm,l1,l2,nu,var,xmin,xmax,ymin,ymax,ns,maxm,      &
        icorr,pad)

!     Put covariance parameters in communication arrays
      iuser(1) = norm
      ruser(1) = l1
      ruser(2) = l2
      ruser(3) = nu

      Allocate (lam(maxm(1)*maxm(2)),xx(ns(1)),yy(ns(2)))

!     Get square roots of the eigenvalues of the embedding matrix
      ifail = 0
      Call g05zqf(ns,xmin,xmax,ymin,ymax,maxm,var,cov2,even,pad,icorr,lam,xx,  &
        yy,m,approx,rho,icount,eig,iuser,ruser,ifail)

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

    Contains
      Subroutine read_input_data(norm,l1,l2,nu,var,xmin,xmax,ymin,ymax,ns,     &
        maxm,icorr,pad)

!       .. Implicit None Statement ..
        Implicit None
```

```
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: l1, l2, nu, var, xmax, xmin, ymax, &
                                            ymin
        Integer, Intent (Out)          :: icorr, norm, pad
!       .. Array Arguments ..
        Integer, Intent (Out)          :: maxm(2), ns(2)
!       .. Executable Statements ..
!       Skip heading in data file
        Read (nin,*)

!       Read in norm, l1, l2 and nu for cov2 function
        Read (nin,*) norm, l1, l2, nu

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

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

!       Read in number of sample points in each direction
        Read (nin,*) ns(1), ns(2)

!       Read in maximum size of embedding matrix
        Read (nin,*) maxm(1), maxm(2)

!       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
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: eig(3)
        Integer, Intent (In)           :: m(2)
        Real (Kind=nag_wp), Intent (In) :: lam(m(1),m(2))
!       .. Local Scalars ..
        Integer                        :: i
!       .. Executable Statements ..
!       Display size of embedding matrix
        Write (nout,*)
        Write (nout,99999) 'Size of embedding matrix = ', m(1)*m(2)

!       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,*)
        Do i = 1, m(1)
```

```
          Write (nout,99996) lam(i,1:m(2))
        End Do

        Return

99999   Format (1X,A,I7)
99998   Format (1X,A,F10.5)
99997   Format (1X,A,3(F10.5,1X))
99996   Format (1X,8F8.4)

      End Subroutine display_results

    End Program g05zqfe
```

10.2 Program Data

```
G05ZQF Example Program Data
  2    0.1   0.15 1.2 : norm, l1, l2, nu
  0.5                 : var
 -1    1              : xmin, xmax
 -0.5  0.5            : ymin, ymax
  5    5              : ns
 81   81              : maxm
  2                   : icorr
  1                   : pad
```

10.3 Program Results

```
 G05ZQF Example Program Results


 Size of embedding matrix =      64

 Approximation not required

 Square roots of eigenvalues of embedding matrix:

   0.8966  0.8234  0.6810  0.5757  0.5391  0.5757  0.6810  0.8234
   0.8940  0.8217  0.6804  0.5756  0.5391  0.5756  0.6804  0.8217
   0.8877  0.8175  0.6792  0.5754  0.5391  0.5754  0.6792  0.8175
   0.8813  0.8133  0.6780  0.5751  0.5390  0.5751  0.6780  0.8133
   0.8787  0.8116  0.6774  0.5750  0.5390  0.5750  0.6774  0.8116
   0.8813  0.8133  0.6780  0.5751  0.5390  0.5751  0.6780  0.8133
   0.8877  0.8175  0.6792  0.5754  0.5391  0.5754  0.6792  0.8175
   0.8940  0.8217  0.6804  0.5756  0.5391  0.5756  0.6804  0.8217
```