

## NAG Library Function Document

### nag\_rand\_field\_1d\_predef\_setup (g05znc)

#### 1 Purpose

`nag_rand_field_1d_predef_setup` (g05znc) 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 `nag_rand_field_1d_generate` (g05zpc), which simulates the random field.

#### 2 Specification

```
#include <nag.h>
#include <nagg05.h>

void nag_rand_field_1d_predef_setup (Integer ns, double xmin, double xmax,
    Integer maxm, double var, Nag_Variogram cov, Integer np,
    const double params[], Nag_EmbedPad pad, Nag_EmbedScale corr,
    double lam[], double xx[], Integer *m, Integer *approx, double *rho,
    Integer *icount, double eig[], NagError *fail)
```

#### 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  $x_1, \dots, x_n \in \mathbb{R}$ , the random vector  $[Z(x_1), \dots, Z(x_n)]^T$  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  $\gamma$  of one variable:  $C(x, y) = \gamma(x - y)$ .  $\gamma$  is known as a variogram (or more correctly, a semivariogram) and includes the multiplicative factor  $\sigma^2$  representing the variance such that  $\gamma(0) = \sigma^2$ .

The functions `nag_rand_field_1d_predef_setup` (g05znc) and `nag_rand_field_1d_generate` (g05zpc) 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  $W$ ),  $\Lambda$  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  $\Lambda$  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  $\Lambda_+$  and  $\Lambda_-$  contain the non-negative and negative eigenvalues of  $B$  respectively. Then  $B$  is replaced by  $\rho B_+$  where  $B_+ = W\Lambda_+W^*$  and  $\rho \in (0, 1]$  is a scaling factor. The error  $\epsilon$  in approximating the distribution of the random field is given by

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

Three choices for  $\rho$  are available, and are determined by the input argument **corr**:

setting **corr** = Nag\_EmbedScaleTraces sets

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

setting **corr** = Nag\_EmbedScaleSqrtTraces sets

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

setting **corr** = Nag\_EmbedScaleOne sets  $\rho = 1$ .

nag\_rand\_field\_1d\_predef\_setup (g05znc) 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**  $\geq$  1.

2: **xmin** – double *Input*

*On entry:* the lower bound for the interval over which the random field is to be simulated. Note that if **cov** = Nag\_VgmBrownian (for simulating fractional Brownian motion), **xmin** is not referenced and the lower bound for the interval is set to zero.

*Constraint:* if **cov**  $\neq$  Nag\_VgmBrownian, **xmin** < **xmax**.

3: **xmax** – double *Input*

*On entry:* the upper bound for the interval over which the random field is to be simulated. Note that if **cov** = Nag\_VgmBrownian (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 **cov**  $\neq$  Nag\_VgmBrownian, **xmin** < **xmax**;  
if **cov** = Nag\_VgmBrownian, **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  $k = 1 + \lceil \log_2(\mathbf{ns} - 1) \rceil$ .

*Suggested value:*  $2^{k+2}$  where  $k = 1 + \lceil \log_2(\mathbf{ns} - 1) \rceil$ .

*Constraint:* **maxm**  $\geq 2^k$ , where  $k$  is the smallest integer satisfying  $2^k \geq 2(\mathbf{ns} - 1)$ .

5: **var** – double

Input

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

*Constraint:* **var**  $\geq 0.0$ .

6: **cov** – Nag\_Variogram

Input

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

**cov** = Nag\_VgmSymmStab  
Symmetric stable variogram

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

where

$$\ell = \mathbf{params}[0], \ell > 0,$$

$$\nu = \mathbf{params}[1], 0 \leq \nu \leq 2.$$

**cov** = Nag\_VgmCauchy  
Cauchy variogram

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

where

$$\ell = \mathbf{params}[0], \ell > 0,$$

$$\nu = \mathbf{params}[1], \nu > 0.$$

**cov** = Nag\_VgmDifferential  
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

$$\ell = \mathbf{params}[0], \ell > 0.$$

**cov** = Nag\_VgmExponential  
Exponential variogram

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

where

$$\ell = \mathbf{params}[0], \ell > 0.$$

**cov** = Nag\_VgmGauss  
Gaussian variogram

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

where

$$\ell = \mathbf{params}[0], \ell > 0.$$

**cov** = Nag\_VgmNugget  
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 **cov**.

**cov** = Nag\_VgmSpherical  
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

$$\ell = \mathbf{params}[0], \ell > 0.$$

**cov** = Nag\_VgmBessel  
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,

$$\ell = \mathbf{params}[0], \ell > 0,$$

$$\nu = \mathbf{params}[1], \nu \geq -0.5.$$

**cov** = Nag\_VgmHole  
Hole effect variogram

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

where

$$\ell = \mathbf{params}[0], \ell > 0.$$

**cov** = Nag\_VgmWhittleMatern  
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,

$$\ell = \mathbf{params}[0], \ell > 0,$$

$$\nu = \mathbf{params}[1], \nu > 0.$$

**cov** = Nag\_VgmContParam

Continuously parameterised variogram with compact support

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

where

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

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

$$\ell = \mathbf{params}[0], \ell > 0,$$

$$s = \mathbf{params}[1], s > 0 \text{ (second correlation length),}$$

$$\nu = \mathbf{params}[2], \nu > 0.$$

**cov** = Nag\_VgmGenHyp

Generalized hyperbolic distribution variogram

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

where

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

$$\ell = \mathbf{params}[0], \ell > 0,$$

$$\lambda = \mathbf{params}[1], \text{ no constraint on } \lambda$$

$$\delta = \mathbf{params}[2], \delta > 0,$$

$$\kappa = \mathbf{params}[3], \kappa > 0.$$

**cov** = Nag\_VgmCosine

Cosine variogram

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

where

$$\ell = \mathbf{params}[0], \ell > 0.$$

**cov** = Nag\_VgmBrownian

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

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

where

$$x = t_j - t_i,$$

$$H = \mathbf{params}[0], 0 < H < 1, H \text{ is the Hurst parameter,}$$

$$\delta = \mathbf{params}[1], \delta > 0, \text{ normally } \delta = t_i - t_{i-1} \text{ is the (fixed) stepsize.}$$

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

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

The increments  $X(i)$  can then be simulated using `nag_rand_field_1d_generate` (g05zpc), then multiplied by  $\delta^H$  to obtain the original increments  $\tilde{X}(i)$  for the fractional Brownian motion.

*Constraint:* **cov** = Nag\_VgmSymmStab, Nag\_VgmCauchy, Nag\_VgmDifferential, Nag\_VgmExponential, Nag\_VgmGauss, Nag\_VgmNugget, Nag\_VgmSpherical, Nag\_VgmBessel, Nag\_VgmHole, Nag\_VgmWhittleMatern, Nag\_VgmContParam, Nag\_VgmGenHyp, Nag\_VgmCosine or Nag\_VgmBrownian.

- 7: **np** – Integer *Input*  
*On entry:* the number of parameters to be set. Different variograms need a different number of parameters.  
**cov** = Nag\_VgmNugget  
**np** must be set to 0.  
**cov** = Nag\_VgmDifferential, Nag\_VgmExponential, Nag\_VgmGauss, Nag\_VgmSpherical, Nag\_VgmHole or Nag\_VgmCosine  
**np** must be set to 1.  
**cov** = Nag\_VgmSymmStab, Nag\_VgmCauchy, Nag\_VgmBessel, Nag\_VgmWhittleMatern or Nag\_VgmBrownian  
**np** must be set to 2.  
**cov** = Nag\_VgmContParam  
**np** must be set to 3.  
**cov** = Nag\_VgmGenHyp  
**np** must be set to 4.
- 8: **params[np]** – const double *Input*  
*On entry:* the parameters set for the variogram.  
*Constraint:* see **cov** for a description of the individual parameter constraints.
- 9: **pad** – Nag\_EmbedPad *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** = Nag\_EmbedPadZeros  
The embedding matrix is padded with zeros.  
**pad** = Nag\_EmbedPadValues  
The embedding matrix is padded with values of the variogram.  
*Suggested value:* **pad** = Nag\_EmbedPadValues.  
*Constraint:* **pad** = Nag\_EmbedPadZeros or Nag\_EmbedPadValues.
- 10: **corr** – Nag\_EmbedScale *Input*  
*On entry:* determines which approximation to implement if required, as described in Section 3.  
*Suggested value:* **corr** = Nag\_EmbedScaleTraces.  
*Constraint:* **corr** = Nag\_EmbedScaleTraces, Nag\_EmbedScaleSqrtTraces or Nag\_EmbedScaleOne.
- 11: **lam[maxm]** – double *Output*  
*On exit:* contains the square roots of the eigenvalues of the embedding matrix.

- 12: **xx[ns]** – double *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** – double \* *Output*  
*On exit:* indicates the scaling of the covariance matrix. **rho** = 1.0 unless approximation was used with **corr** = Nag\_EmbedScaleTraces or Nag\_EmbedScaleSqrtTraces.
- 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]** – double *Output*  
*On exit:* indicates information about the negative eigenvalues in the embedding matrix which have had to be set to zero. **eig**[0] contains the smallest eigenvalue, **eig**[1] contains the sum of the squares of the negative eigenvalues, and **eig**[2] contains the sum of the absolute values of the negative eigenvalues.
- 18: **fail** – NagError \* *Input/Output*  
 The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument *⟨value⟩* had an illegal value.

### NE\_ENUM\_INT

On entry, **np** = *⟨value⟩*.

Constraint: for **cov** = *⟨value⟩*, **np** = *⟨value⟩*.

### NE\_ENUM\_REAL\_1

On entry, **cov** = Nag\_VgmBrownian and **xmax** = *⟨value⟩*.

Constraint: **xmax** > 0.0.

On entry, **params**[*⟨value⟩*] = *⟨value⟩*.

Constraint: dependent on **cov**.

**NE\_ENUM\_REAL\_2**

On entry, **cov**  $\neq$  Nag\_VgmBrownian, **xmin** =  $\langle value \rangle$  and **xmax** =  $\langle value \rangle$ .  
 Constraint: **xmin** < **xmax**.

**NE\_INT**

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  $2^k \geq 2(\mathbf{ns} - 1)$ .

On entry, **ns** =  $\langle value \rangle$ .  
 Constraint: **ns**  $\geq$  1.

**NE\_INTERNAL\_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.  
 See Section 3.6.6 in How to Use the NAG Library and its Documentation for further information.

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly.  
 See Section 3.6.5 in How to Use the NAG Library and its Documentation for further information.

**NE\_REAL**

On entry, **var** =  $\langle value \rangle$ .  
 Constraint: **var**  $\geq$  0.0.

**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**

nag\_rand\_field\_1d\_predef\_setup (g05znc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag\_rand\_field\_1d\_predef\_setup (g05znc) 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 function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

**9 Further Comments**

None.

**10 Example**

This example calls nag\_rand\_field\_1d\_predef\_setup (g05znc) to calculate the eigenvalues of the embedding matrix for 8 sample points of a random field characterized by the symmetric stable variogram (**cov** = Nag\_VgmSymmStab).



## 10.1 Program Text

```

/* nag_rand_field_ld_predef_setup (g05znc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>
#include <nagx04.h>

static void read_input_data(Nag_Variogram *cov, Integer *np, double *params,
                           double *var, double *xmin, double *xmax,
                           Integer *ns, Integer *maxm, Nag_EmbedScale *corr,
                           Nag_EmbedPad *pad);
static void display_results(Integer approx, Integer m, double rho,
                           double *eig, Integer icount, double *lam);

int main(void)
{
  Integer exit_status = 0;
  /* Scalars */
  double rho, var, xmax, xmin;
  Integer approx, icount, m, maxm, np, ns;
  /* Arrays */
  double eig[3], params[4], *lam = 0, *xx = 0;
  /* Nag types */
  Nag_Variogram cov;
  Nag_EmbedScale corr;
  Nag_EmbedPad pad;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_rand_field_ld_predef_setup (g05znc) Example Program Results\n\n");

  /* Get problem specifications from data file */
  read_input_data(&cov, &np, params, &var, &xmin, &xmax, &ns, &maxm, &corr,
                 &pad);
  if (!(lam = NAG_ALLOC((maxm), double)) || !(xx = NAG_ALLOC((ns), double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }
  /* Get square roots of the eigenvalues of the embedding matrix. These are
   * obtained from the setup for simulating one-dimensional random fields,
   * with a preset variogram, by the circulant embedding method using
   * nag_rand_field_ld_predef_setup (g05znc).
   */
  nag_rand_field_ld_predef_setup(ns, xmin, xmax, maxm, var, cov, np,
                                params, pad, corr, lam, xx, &m, &approx,
                                &rho, &icount, eig, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_field_ld_predef_setup (g05znc).\n%s\n",
          fail.message);
    exit_status = 1;
    goto END;
  }
  /* Output results */
  display_results(approx, m, rho, eig, icount, lam);
END:
  NAG_FREE(lam);
  NAG_FREE(xx);
  return exit_status;
}

```

```

void read_input_data(Nag_Variogram *cov, Integer *np, double *params,
                    double *var, double *xmin, double *xmax, Integer *ns,
                    Integer *maxm, Nag_EmbedScale *corr, Nag_EmbedPad *pad)
{
    Integer j;
    char nag_enum_arg[40];

    /* Read in covariance function name and convert to value using
     * nag_enum_name_to_value (x04nac).
     */
#ifdef _WIN32
    scanf_s("%*[\n] %39s%*[\n]", nag_enum_arg,
            (unsigned)_countof(nag_enum_arg));
#else
    scanf("%*[\n] %39s%*[\n]", nag_enum_arg);
#endif
    *cov = (Nag_Variogram) nag_enum_name_to_value(nag_enum_arg);
    /* Read in parameters */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[\n]", np);
#else
    scanf("%" NAG_IFMT "%*[\n]", np);
#endif
    for (j = 0; j < *np; j++)
#ifdef _WIN32
        scanf_s("%lf", &params[j]);
#else
        scanf("%lf", &params[j]);
#endif
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
    /* Read in variance of random field. */
#ifdef _WIN32
    scanf_s("%lf%*[\n]", var);
#else
    scanf("%lf%*[\n]", var);
#endif
    /* Read in domain endpoints. */
#ifdef _WIN32
    scanf_s("%lf %lf%*[\n]", xmin, xmax);
#else
    scanf("%lf %lf%*[\n]", xmin, xmax);
#endif
    /* Read in number of sample points. */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[\n]", ns);
#else
    scanf("%" NAG_IFMT "%*[\n]", ns);
#endif
    /* Read in maximum size of embedding matrix. */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%*[\n]", maxm);
#else
    scanf("%" NAG_IFMT "%*[\n]", maxm);
#endif
    /* Read name of scaling in case of approximation and convert to value. */
#ifdef _WIN32
    scanf_s(" %39s%*[\n]", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
    scanf(" %39s%*[\n]", nag_enum_arg);
#endif
    *corr = (Nag_EmbedScale) nag_enum_name_to_value(nag_enum_arg);
    /* Read in choice of padding and convert name to value. */
#ifdef _WIN32
    scanf_s(" %39s%*[\n]", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
    scanf(" %39s%*[\n]", nag_enum_arg);

```

```

#endif
    *pad = (Nag_EmbedPad) nag_enum_name_to_value(nag_enum_arg);
}

void display_results(Integer approx, Integer m, double rho, double *eig,
                    Integer icount, double *lam)
{
    Integer j;
    /* Display size of embedding matrix */
    printf("\nSize of embedding matrix = %" NAG_IFMT "\n\n", m);
    /* Display approximation information if approximation used */
    if (approx == 1) {
        printf("Approximation required\n\n");
        printf("rho = %10.5f\n", rho);
        printf("eig = ");
        for (j = 0; j < 3; j++)
            printf("%10.5f ", eig[j]);
        printf("\nicount = %" NAG_IFMT "\n", icount);
    }
    else {
        printf("Approximation not required\n");
    }
    /* Display square roots of the eigenvalues of the embedding matrix. */
    printf("\nSquare roots of eigenvalues of embedding matrix:\n\n");
    for (j = 0; j < m; j++)
        printf("%10.5f%s", lam[j], j % 4 == 3 ? "\n" : "");
    printf("\n");
}

```

## 10.2 Program Data

```

nag_rand_field_ld_predef_setup (g05znc) Example Program Data
Nag_VgmSymmStab      : cov
2                    : np (2 parameters for Nag_VgmSymmStab)
0.1    1.2          : params (c and nu)
0.5                : var
-1     1            : xmin, xmax
8                    : ns
64                : maxm
Nag_EmbedScaleOne   : corr
Nag_EmbedPadValues  : pad

```

## 10.3 Program Results

```

nag_rand_field_ld_predef_setup (g05znc) 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

---