

## NAG Library Function Document

### nag\_nearest\_correlation\_shrinking (g02anc)

#### 1 Purpose

nag\_nearest\_correlation\_shrinking (g02anc) computes a correlation matrix, subject to preserving a leading principal submatrix and applying the smallest relative perturbation to the remainder of the approximate input matrix.

#### 2 Specification

```
#include <nag.h>
#include <nagg02.h>

void nag_nearest_correlation_shrinking (double g[], Integer pdg, Integer n,
    Integer k, double errtol, double eigtol, double x[], Integer pdx,
    double *alpha, Integer *iter, double *eigmin, double *norm,
    NagError *fail)
```

#### 3 Description

nag\_nearest\_correlation\_shrinking (g02anc) finds a correlation matrix,  $X$ , starting from an approximate correlation matrix,  $G$ , with positive definite leading principal submatrix of order  $k$ . The returned correlation matrix,  $X$ , has the following structure:

$$X = \alpha \begin{pmatrix} A & 0 \\ 0 & I \end{pmatrix} + (1 - \alpha)G$$

where  $A$  is the  $k$  by  $k$  leading principal submatrix of the input matrix  $G$  and positive definite, and  $\alpha \in [0, 1]$ .

nag\_nearest\_correlation\_shrinking (g02anc) utilizes a shrinking method to find the minimum value of  $\alpha$  such that  $X$  is positive definite with unit diagonal.

#### 4 References

Higham N J, Strabić N and Šego V (2014) Restoring definiteness via shrinking, with an application to correlation matrices with a fixed block *MIMS EPrint 2014.54* Manchester Institute for Mathematical Sciences, The University of Manchester, UK

#### 5 Arguments

- 1: **g**[**pdg** × **n**] – double *Input/Output*  
**Note:** the  $(i, j)$ th element of the matrix  $G$  is stored in **g**[( $j - 1$ ) × **pdg** +  $i - 1$ ].  
*On entry:*  $G$ , the initial matrix.  
*On exit:* a symmetric matrix  $\frac{1}{2}(G + G^T)$  with the diagonal set to  $I$ .
- 2: **pdg** – Integer *Input*  
*On entry:* the stride separating matrix row elements in the array **g**.  
**Constraint:** **pdg** ≥ **n**.

- 3: **n** – Integer *Input*  
*On entry:* the order of the matrix  $G$ .  
*Constraint:*  $\mathbf{n} > 0$ .
- 4: **k** – Integer *Input*  
*On entry:*  $k$ , the order of the leading principal submatrix  $A$ .  
*Constraint:*  $\mathbf{n} \geq \mathbf{k} > 0$ .
- 5: **errtol** – double *Input*  
*On entry:* the termination tolerance for the iteration.  
 If  $\mathbf{errtol} \leq 0$ ,  $\sqrt{\text{machine precision}}$  is used. See Section 7 for further details.
- 6: **eigtol** – double *Input*  
*On entry:* the tolerance used in determining the definiteness of  $A$ .  
 If  $\lambda_{\min}(A) > \mathbf{n} \times \lambda_{\max}(A) \times \mathbf{eigtol}$ , where  $\lambda_{\min}(A)$  and  $\lambda_{\max}(A)$  denote the minimum and maximum eigenvalues of  $A$  respectively,  $A$  is positive definite.  
 If  $\mathbf{eigtol} \leq 0$ , *machine precision* is used.
- 7: **x[pdx × n]** – double *Output*  
**Note:** the  $(i, j)$ th element of the matrix  $X$  is stored in  $\mathbf{x}[(j - 1) \times \mathbf{pdx} + i - 1]$ .  
*On exit:* contains the matrix  $X$ .
- 8: **pdx** – Integer *Input*  
*On entry:* the stride separating matrix row elements in the array  $\mathbf{x}$ .  
*Constraint:*  $\mathbf{pdx} \geq \mathbf{n}$ .
- 9: **alpha** – double \* *Output*  
*On exit:*  $\alpha$ .
- 10: **iter** – Integer \* *Output*  
*On exit:* the number of iterations taken.
- 11: **eigmin** – double \* *Output*  
*On exit:* the smallest eigenvalue of the leading principal submatrix  $A$ .
- 12: **norm** – double \* *Output*  
*On exit:* the value of  $\|G - X\|_F$  after the final iteration.
- 13: **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 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument  $\langle value \rangle$  had an illegal value.

### NE\_EIGENPROBLEM

Failure to solve intermediate eigenproblem. This should not occur. Please contact NAG.

### NE\_INT

On entry,  $n = \langle value \rangle$ .

Constraint:  $n > 0$ .

### NE\_INT\_2

On entry,  $k = \langle value \rangle$  and  $n = \langle value \rangle$ .

Constraint:  $n \geq k > 0$ .

On entry,  $pdg = \langle value \rangle$  and  $n = \langle value \rangle$ .

Constraint:  $pdg \geq n$ .

On entry,  $pdx = \langle value \rangle$  and  $n = \langle value \rangle$ .

Constraint:  $pdx \geq n$ .

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

### NE\_MAT\_NOT\_POS\_DEF

The  $k$  by  $k$  principal leading submatrix of the initial matrix  $G$  is not positive definite.

### NE\_NO\_LICENCE

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

See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

## 7 Accuracy

The algorithm uses a bisection method. It is terminated when the computed  $\alpha$  is within **errtol** of the minimum value. The positive definiteness of  $X$  is such that it can be successfully factorized with a call to nag\_dpotrf (f07fdc).

The number of iterations taken for the bisection will be:

$$\left\lceil \log_2 \left( \frac{1}{\text{errtol}} \right) \right\rceil.$$

## 8 Parallelism and Performance

nag\_nearest\_correlation\_shrinking (g02anc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag\_nearest\_correlation\_shrinking (g02anc) 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

Arrays are internally allocated by nag\_nearest\_correlation\_shrinking (g02anc). The total size of these arrays does not exceed  $2 \times n^2 + 3 \times n$  real elements. All allocated memory is freed before return of nag\_nearest\_correlation\_shrinking (g02anc).

## 10 Example

This example finds the smallest uniform perturbation  $\alpha$  to  $G$ , such that the output is a correlation matrix and the  $k$  by  $k$  leading principal submatrix of the input is preserved,

$$G = \begin{pmatrix} 1.0000 & -0.0991 & 0.5665 & -0.5653 & -0.3441 \\ -0.0991 & 1.0000 & -0.4273 & 0.8474 & 0.4975 \\ 0.5665 & -0.4273 & 1.0000 & -0.1837 & -0.0585 \\ -0.5653 & 0.8474 & -0.1837 & 1.0000 & -0.2713 \\ -0.3441 & 0.4975 & -0.0585 & -0.2713 & 1.0000 \end{pmatrix}.$$

### 10.1 Program Text

```
/* nag_nearest_correlation_shrinking (g02anc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg02.h>
#include <nagx04.h>

int main(void)
{
#define G(I,J) g[(J-1)*pdg + I-1]

    /* Scalars */
    Integer exit_status = 0;
    double alpha, eigmin, eigtol, errtol, norm;
    Integer i, j, iter, k, n, pdg, pdx;

    /* Arrays */
    double *g = 0, *x = 0;

    /* Nag Types */
    Nag_OrderType order;
    NagError fail;

    INIT_FAIL(fail);

    /* Output preamble */
    printf("nag_nearest_correlation_shrinking (g02anc)");
    printf(" Example Program Results\n\n");
    fflush(stdout);
```

```

/* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

/* Read in the problem size and k */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &n, &k);
#else
    scanf("%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &n, &k);
#endif

    pdg = n;
    pdx = n;
    if (!(g = NAG_ALLOC(pdg * n, double)) || !(x = NAG_ALLOC(pdx * n, double))
        )
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

/* Read in the matrix g */
for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
#ifdef _WIN32
        scanf_s("%lf", &G(i, j));
#else
        scanf("%lf", &G(i, j));
#endif
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

/* Use the defaults for ERRTOL and EIGTOL */
errtol = -1.0;
eigtol = -1.0;

/*
 * nag_nearest_correlation_shrinking (g02anc).
 * Calculate nearest perturbed correlation matrix with preserved leading block
 */
nag_nearest_correlation_shrinking(g, pdg, n, k, errtol, eigtol, x, pdx,
                                   &alpha, &iter, &eigmin, &norm, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_nearest_correlation_shrinking (g02anc).\n%s\n",
          fail.message);
    exit_status = 1;
    goto END;
}

/* Display results */

order = Nag_ColMajor;
/*
 * nag_gen_real_mat_print (x04cac).
 * Prints real general matrix
 */
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n, g,
                       pdg, "Symmetrised Input Matrix G", NULL, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 2;
    goto END;
}

printf("\n");

```

```

fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n, x,
                        pdx, "Nearest Perturbed Correlation Matrix X", NULL,
                        &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 3;
    goto END;
}

printf("\n%s %34" NAG_IFMT " \n\n", "k:", k);
printf("%s %9" NAG_IFMT " \n\n", "Number of iterations taken:", iter);
printf("%s %34.4f \n\n", "alpha: ", alpha);
printf("%s %29.4f \n\n", "norm value: ", norm);
printf("%s %15.4f \n", "Smallest eigenvalue of A: ", eigmin);

END:
    NAG_FREE(g);
    NAG_FREE(x);
    return exit_status;
}

```

## 10.2 Program Data

nag\_nearest\_correlation\_shrinking (g02anc) Example Program Data

```

5 3 :: n, k
1.0000 -0.0991 0.5665 -0.5653 -0.3441
-0.0991 1.0000 -0.4273 0.8474 0.4975
0.5665 -0.4273 1.0000 -0.1837 -0.0585
-0.5653 0.8474 -0.1837 1.0000 -0.2713
-0.3441 0.4975 -0.0585 -0.2713 1.0000 :: End of g

```

## 10.3 Program Results

nag\_nearest\_correlation\_shrinking (g02anc) Example Program Results

```

Symmetrised Input Matrix G
      1      2      3      4      5
1  1.0000 -0.0991 0.5665 -0.5653 -0.3441
2 -0.0991 1.0000 -0.4273 0.8474 0.4975
3  0.5665 -0.4273 1.0000 -0.1837 -0.0585
4 -0.5653 0.8474 -0.1837 1.0000 -0.2713
5 -0.3441 0.4975 -0.0585 -0.2713 1.0000

Nearest Perturbed Correlation Matrix X
      1      2      3      4      5
1  1.0000 -0.0991 0.5665 -0.3826 -0.2329
2 -0.0991 1.0000 -0.4273 0.5735 0.3367
3  0.5665 -0.4273 1.0000 -0.1243 -0.0396
4 -0.3826 0.5735 -0.1243 1.0000 -0.1836
5 -0.2329 0.3367 -0.0396 -0.1836 1.0000

k:                                     3

Number of iterations taken:           27

alpha:                                0.3232

norm value:                           0.5624

Smallest eigenvalue of A:              0.3359

```

---