NAG Library Function Document

nag_nearest_correlation_target (g02apc)

1 Purpose

nag_nearest_correlation_target (g02apc) computes a correlation matrix, by using a positive definite **target** matrix derived from weighting the approximate input matrix, with an optional bound on the minimum eigenvalue.

2 Specification

```
#include <nag.h>
#include <nagg02.h>
void nag_nearest_correlation_target (double g[], Integer pdg, Integer n,
     double theta, double h[], Integer pdh, double errtol, double eigtol,
     double x[], Integer pdx, double *alpha, Integer *iter, double *eigmin,
     double *norm, NagError *fail)
```

3 Description

Starting from an approximate correlation matrix, G, nag_nearest_correlation_target (g02apc) finds a correlation matrix, X, which has the form

$$X = \alpha T + (1 - \alpha)G,$$

where $\alpha \in [0, 1]$ and $T = H \circ G$ is a target matrix. $C = A \circ B$ denotes the matrix C with elements $C_{ij} = A_{ij} \times B_{ij}$. H is a matrix of weights that defines the target matrix. The target matrix must be positive definite and thus have off-diagonal elements less than 1 in magnitude. A value of 1 in H essentially fixes an element in G so it is unchanged in X.

nag_nearest_correlation_target (g02apc) utilizes a shrinking method to find the minimum value of α such that X is positive definite with unit diagonal and with a smallest eigenvalue of at least $\theta \in [0, 1)$ times the smallest eigenvalue of the target matrix.

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**$[(j - 1) \times$ **pdg** $+ i - 1]$.

On entry: G, the initial matrix.

On exit: a symmetric matrix $\frac{1}{2}(G + G^{\mathrm{T}})$ with the diagonal elements set to 1.0.

2: **pdg** – Integer *Input*

On entry: the stride separating matrix row elements in the array **g**.

Constraint: **pdg** $\geq$ **n**.

3: **n** – Integer *Input*

On entry: the order of the matrix G.

Constraint: **n** > 0.

4: **theta** – double *Input*

On entry: the value of θ. If **theta** < 0.0, 0.0 is used.

Constraint: **theta** < 1.0.

5: **h**[**pdh** $\times$ **n**] – double *Input/Output*

Note: the (i,j)th element of the matrix H is stored in **h**$[(j-1) \times$ **pdh** $+ i - 1]$.

On entry: the matrix of weights H.

On exit: a symmetric matrix $\frac{1}{2}(H + H^{\mathrm{T}})$ with its diagonal elements set to 1.0.

6: **pdh** – Integer *Input*

On entry: the stride separating matrix row elements in the array **h**.

Constraint: **pdh** $\geq$ **n**.

7: **errtol** – double *Input*

On entry: the termination tolerance for the iteration.

If **errtol** ≤ 0, $\sqrt{\textit{machine precision}}$ is used. See Section 7 for further details.

8: **eigtol** – double *Input*

On entry: the tolerance used in determining the definiteness of the target matrix $T = H \circ G$.

If $\lambda_{\min}(T) >$ **n** $\times \lambda_{\max}(T) \times$ **eigtol**, where $\lambda_{\min}(T)$ and $\lambda_{\max}(T)$ denote the minimum and maximum eigenvalues of T respectively, T is positive definite.

If **eigtol** ≤ 0, ***machine precision*** is used.

9: **x**[**pdx** $\times$ **n**] – double *Output*

Note: the (i,j)th element of the matrix X is stored in **x**$[(j-1) \times$ **pdx** $+ i - 1]$.

On exit: contains the matrix X.

10: **pdx** – Integer *Input*

On entry: the stride separating matrix row elements in the array **x**.

Constraint: **pdx** $\geq$ **n**.

11: **alpha** – double * *Output*

On exit: the constant α used in the formation of X.

12: **iter** – Integer * *Output*

On exit: the number of iterations taken.

13: **eigmin** – double * *Output*

On exit: the smallest eigenvalue of the target matrix T.

14: **norm** – double * *Output*

On exit: the value of $\|G - X\|_F$ after the final iteration.

15: **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_EIGENPROBLEM

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

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** > 0.

NE_INT_2

On entry, **pdg** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **pdg** $\geq$ **n**.

On entry, **pdh** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **pdh** $\geq$ **n**.

On entry, **pdx** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
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 3.6.6 in How to Use the NAG Library and its Documentation for further information.

NE_MAT_NOT_POS_DEF

The target matrix is not positive definite.

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, **theta** = ⟨*value*⟩.
Constraint: **theta** < 1.0.

7 Accuracy

The algorithm uses a bisection method. It is terminated when the computed α is within **errtol** of the minimum value.

Note: when θ is zero X is still positive definite, in that it can be successfully factorized with a call to nag_dpotrf (f07fdc).

The number of iterations taken for the bisection will be:

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

8 Parallelism and Performance

nag_nearest_correlation_target (g02apc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_nearest_correlation_target (g02apc) 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' Notefor your implementation for any additional implementation-specific information.

9 Further Comments

Arrays are internally allocated by nag_nearest_correlation_target (g02apc). 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_target (g02apc).

10 Example

This example finds the smallest α such that $\alpha(H \circ G) + (1-\alpha)G$ is a correlation matrix. The 2 by 2 leading principal submatrix of the input is preserved, and the last 2 by 2 diagonal block is weighted to give some emphasis to the off diagonal elements.

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

and

$$H = \begin{pmatrix} 1.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 1.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.5000 \\ 0.0000 & 0.0000 & 0.0000 & 0.5000 & 1.0000 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_nearest_correlation_target (g02apc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagg02.h>
#include <nagx04.h>
```

```
int main(void)
{

#define G(I,J) g[(J-1)*pdg + I-1]
#define H(I,J) h[(J-1)*pdh + I-1]

  /*  Scalars */
  Integer exit_status = 0;
  Integer one = 1;
  double alpha, eigmin, eigtol, errtol, norm, theta;
  Integer i, j, iter, n, pdg, pdh, pdx;

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

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

  INIT_FAIL(fail);

  /* Output preamble */
  printf("nag_nearest_correlation_target (g02apc)");
  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 theta */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%lf%*[^\n] ", &n, &theta);
#else
  scanf("%" NAG_IFMT "%lf%*[^\n] ", &n, &theta);
#endif

  pdg = n;
  pdh = n;
  pdx = n;
  if (!(eig = NAG_ALLOC(n, double)) ||
      !(g = NAG_ALLOC(pdg * n, double)) ||
      !(h = NAG_ALLOC(pdh * 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

  /* Read in the matrix h */
  for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
```

```
#ifdef _WIN32
      scanf_s("%lf", &H(i, j));
#else
      scanf("%lf", &H(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_target (g02apc).
   * Calculate nearest correlation matrix using target matrix
   */
  nag_nearest_correlation_target(g, pdg, n, theta, h, pdh, errtol,
  eigtol, x, pdx, &alpha, &iter, &eigmin,
              &norm, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_nearest_correlation_target (g02apc).\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 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 %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 %34.4f \n\n", "theta: ", theta);
  printf("%s %15.4f \n", "Smallest eigenvalue of A: ", eigmin);

  /*
   * nag_dsyev (f08fac).
   * Computes all eigenvalues and, optionally, eigenvectors of a real
   * symmetric matrix
   */
  nag_dsyev(order, Nag_EigVals, Nag_Upper, n, x, pdx, eig, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dsyev (f08fac).\n%s\n", fail.message);
    exit_status = 4;
    goto END;
```

```
  }

  printf("\n");
  fflush(stdout);
  /*
   * nag_gen_real_mat_print (x04cac).
   * Print real general matrix (easy-to-use)
   */
  nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, one, n,
                         eig, one, "Eigenvalues of x", NULL, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 5;
    goto END;
  }

END:
  NAG_FREE(eig);
  NAG_FREE(g);
  NAG_FREE(h);
  NAG_FREE(x);
  return exit_status;

}
```

10.2 Program Data

```
nag_nearest_correlation_target (g02apc) Example Program Data
 5  0.1                                    :: n, theta
  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
  1.0000  1.0000  0.0000  0.0000  0.0000
  1.0000  1.0000  0.0000  0.0000  0.0000
  0.0000  0.0000  1.0000  0.0000  0.0000
  0.0000  0.0000  0.0000  1.0000  0.5000
  0.0000  0.0000  0.0000  0.5000  1.0000 :: End of h
```

10.3 Program Results

```
nag_nearest_correlation_target (g02apc) 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 Correlation Matrix X
         1       2       3       4       5
 1   1.0000 -0.0991  0.3799 -0.3791 -0.2308
 2  -0.0991  1.0000 -0.2865  0.5683  0.3336
 3   0.3799 -0.2865  1.0000 -0.1232 -0.0392
 4  -0.3791  0.5683 -0.1232  1.0000 -0.2266
 5  -0.2308  0.3336 -0.0392 -0.2266  1.0000

Number of iterations taken:        27

alpha:                              0.3294

norm value:                         0.6526

theta:                              0.1000
```

```
Smallest eigenvalue of A:           0.8643

 Eigenvalues of x
            1          2          3          4          5
 1      0.0864     0.7431     1.0044     1.2018     1.9642
```