1 Purpose
	nag_nearest_correlation_bounded (g02abc) computes the nearest correlation matrix, in the Frobenius norm or weighted Frobenius norm, and optionally with bounds on the eigenvalues, to a given square, input matrix.

2 Specification

```c
#include <nag.h>
#include <nagg02.h>
void nag_nearest_correlation_bounded (Nag_OrderType order, double g[],
        Integer pdg, Integer n, Nag_NearCorr_ProbType opt, double alpha,
        double w[], double errtol, Integer maxits, Integer maxit, double x[],
        Integer pdx, Integer *iter, Integer *feval, double *nrmgrd,
        NagError *fail)
```

3 Description

Finds the nearest correlation matrix $X$ by minimizing $\frac{1}{2} \|G - X\|^2$ where $G$ is an approximate correlation matrix.

The norm can either be the Frobenius norm or the weighted Frobenius norm $\frac{1}{2} \|W^{1/2}(G - X)W^{1/2}\|^2_F$.

You can optionally specify a lower bound on the eigenvalues, $\alpha$, of the computed correlation matrix, forcing the matrix to be positive definite, $0 < \alpha < 1$.

Note that if the weights vary by several orders of magnitude from one another the algorithm may fail to converge.

4 References


5 Arguments

1:  
   order – Nag_OrderType   

   *Input*

   *On entry*: the order argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by order = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

   *Constraint*: order = Nag_RowMajor or Nag_ColMajor.

2:  
   g[pdg x n] – double   

   *Input/Output*

   *Note*: the $(i,j)$th element of the matrix $G$ is stored in
   
   $g[(j - 1) \times pdg + i - 1]$ when order = Nag_ColMajor;
   
   $g[(i - 1) \times pdg + j - 1]$ when order = Nag_RowMajor.
On entry: \( G \), the initial matrix.

On exit: \( G \) is overwritten.

3: \textbf{pdg} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \textit{order}) in the array \( g \).

\textit{Constraint:} \( \text{pdg} \geq \text{n} \).

4: \textbf{n} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the order of the matrix \( G \).

\textit{Constraint:} \( n > 0 \).

5: \textbf{opt} – Nag_NearCorr_ProbType \hspace{1cm} \textit{Input}

\textit{On entry:} indicates the problem to be solved.

\( \text{opt} = \text{Nag_LowerBound} \)

The lower bound problem is solved.

\( \text{opt} = \text{Nag_WeightedNorm} \)

The weighted norm problem is solved.

\( \text{opt} = \text{Nag_Both} \)

Both problems are solved.

\textit{Constraint:} \( \text{opt} = \text{Nag_LowerBound}, \text{Nag_WeightedNorm} \) or \( \text{Nag_Both} \).

6: \textbf{alpha} – double \hspace{1cm} \textit{Input}

\textit{On entry:} the value of \( \alpha \).

If \( \text{opt} = \text{Nag_WeightedNorm} \), \( \text{alpha} \) need not be set.

\textit{Constraint:} \( 0.0 < \text{alpha} < 1.0 \).

7: \textbf{w[n]} – double \hspace{1cm} \textit{Input/Output}

\textit{On entry:} the square roots of the diagonal elements of \( W \), that is the diagonal of \( W^{\frac{1}{2}} \).

If \( \text{opt} = \text{Nag_LowerBound} \), \( \text{w} \) is not referenced and may be \textbf{NULL}.

\textit{On exit:} if \( \text{opt} = \text{Nag_WeightedNorm} \) or \( \text{Nag_Both} \), the array is scaled so \( 0 < \text{w}[i - 1] \leq 1 \), for \( i = 1, 2, \ldots, n \).

\textit{Constraint:} \( \text{w}[i - 1] > 0.0 \), for \( i = 1, 2, \ldots, n \).

8: \textbf{errtol} – double \hspace{1cm} \textit{Input}

\textit{On entry:} the termination tolerance for the Newton iteration. If \( \text{errtol} \leq 0.0 \) then \( n \times \sqrt{\text{machine precision}} \) is used.

9: \textbf{maxits} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} specifies the maximum number of iterations to be used by the iterative scheme to solve the linear algebraic equations at each Newton step.

If \( \text{maxits} \leq 0 \), \( 2 \times n \) is used.

10: \textbf{maxit} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} specifies the maximum number of Newton iterations.

If \( \text{maxit} \leq 0 \), 200 is used.
Note: the \((i,j)\)th element of the matrix \(X\) is stored in
\[x[(j - 1) \times \text{pdx} + i - 1]\] when \text{order} = \text{Nag\_ColMajor};
\[x[(i - 1) \times \text{pdx} + j - 1]\] when \text{order} = \text{Nag\_RowMajor}.

On exit: contains the nearest correlation matrix.

12: \text{pdx} \quad \text{Integer} \\
\text{Input} \\
\text{On entry: the stride separating row or column elements (depending on the value of} \text{order}) \text{in the array} \text{x.}
\text{Constraint: } \text{pdx} \geq n.

13: \text{iter} \quad \text{Integer} * \\
\text{Output} \\
\text{On exit: the number of Newton steps taken.}

14: \text{feval} \quad \text{Integer} * \\
\text{Output} \\
\text{On exit: the number of function evaluations of the dual problem.}

15: \text{nrmgrd} \quad \text{double} * \\
\text{Output} \\
\text{On exit: the norm of the gradient of the last Newton step.}

16: \text{fail} \quad \text{NagError} * \\
\text{Input/Output} \\
The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \quad \text{Error Indicators and Warnings}

\text{NE\_ALLOC\_FAIL}
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\text{NE\_BAD\_PARAM}
On entry, argument <\text{value}> had an illegal value.

\text{NE\_CONVERGENCE}
Newton iteration fails to converge in <\text{value}> iterations. Increase \text{maxit} or check the call to the function.
The \text{machine precision} is limiting convergence. In this instance the returned value of \text{x} may be useful.

\text{NE\_EIGENPROBLEM}
An intermediate eigenproblem could not be solved. This should not occur. Please contact NAG with details of your call.

\text{NE\_INT}
On entry, \(n = <\text{value}>\).
Constraint: \(n > 0\).

\text{NE\_INT\_2}
On entry, \(pdg = <\text{value}>\) and \(n = <\text{value}>\).
Constraint: \(pdg \geq n\).
On entry, \( pdx = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
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 3.6.6 in the Essential Introduction 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 the Essential Introduction for further information.

**NE_REAL**
On entry, \( \alpha = \langle \text{value} \rangle \).
Constraint: \( 0 < \alpha < 1 \).

**NE_WEIGHTS_NOT_POSITIVE**
On entry, all elements of \( w \) were not positive.

7 **Accuracy**
The returned accuracy is controlled by \( \texttt{errtol} \) and limited by \textit{machine precision}.

8 **Parallelism and Performance**
\texttt{nag_nearest_correlation_bounded (g02abc)} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
\texttt{nag_nearest_correlation_bounded (g02abc)} 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 \texttt{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 \texttt{nag_nearest_correlation_bounded (g02abc)}. The total size of these arrays is \( 12 \times n + 3 \times n \times n + \max(2 \times n \times n + 6 \times n + 1, 120 + 9 \times n) \) double elements and \( 5 \times n + 3 \) Integer elements. All allocated memory is freed before return of \texttt{nag_nearest_correlation_bounded (g02abc)}.

10 **Example**
This example finds the nearest correlation matrix to:

\[
G = \begin{pmatrix}
2 & -1 & 0 & 0 \\
-1 & 2 & -1 & 0 \\
0 & -1 & 2 & -1 \\
0 & 0 & -1 & 2
\end{pmatrix}
\]

weighted by \( W^2 = \text{diag}(100, 20, 20, 20) \) with minimum eigenvalue 0.02.
10.1 Program Text

/* nag_nearest_correlation_bounded (g02abc) Example Program. *
* Copyright 2014 Numerical Algorithms Group.
* Mark 23, 2011. */

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

int main(void) {

    /* Scalars */
    Integer exit_status = 0;
    double alpha, errtol, nrmgrd;
    Integer feval, i, iter, j, maxit, maxits, n, pdeig, pdg, pdx;

    /* Arrays */
    char nag_enum_arg[100];
    double *eig = 0, *g = 0, *w = 0, *x = 0;

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

    INIT_FAIL(fail);

    #ifdef NAG_COLUMN_MAJOR
    #define G(I, J) g[(J-1)*pdg + I-1]
    #define X(I, J) x[(J-1)*pdx + I-1]
    order = Nag_ColMajor;
    #else
    #define G(I, J) g[(I-1)*pdg + J-1]
    #define X(I, J) x[(I-1)*pdx + J-1]
    order = Nag_RowMajor;
    #endif

    /* Output preamble */
    printf("nag_nearest_correlation_bounded (g02abc)\nExample 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, opt and alpha */
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT"", &n);
    #else
    scanf("%"NAG_IFMT"", &n);
    #endif

    "nag_enum_name_to_value (x04nac). "
    "Converts NAG enum member name to value */
    opt = (Nag_NearCorr_ProbType) nag_enum_name_to_value(nag_enum_arg);
```c
#ifdef _WIN32
    scanf_s("%lf%*[\n]", &alpha);
#else
    scanf("%lf%*[\n]", &alpha);
#endif

pdg = n;
pdx = n;
if (order == Nag_ColMajor)
    pdeig = 1;
else
    pdeig = n;

if (! (g = NAG_ALLOC((pdg)*(n), double)) ||
    !(w = NAG_ALLOC((n), double)) ||
    !(x = NAG_ALLOC((pdx)*(n), double)) ||
    !(eig = NAG_ALLOC((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 vector w */
for (i = 0; i < n; i++)
#ifdef _WIN32
    scanf_s("%lf", &w[i]);
#else
    scanf("%lf", &w[i]);
#endif
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

/* Use the defaults for errtol, maxits and maxit */
errtol = 0.0;
maxits = 0;
maxit = 0;

/* nag_nearest_correlation_bounded (g02abc).
   Computes the nearest correlation matrix incorporating weights
   and/or bounds */
nag_nearest_correlation_bounded(order, g, pdg, n, opt, alpha, w, errtol,
    maxits, maxit, x, pdx, &iter, &feval,
    &nrmgrd, &fail);
if (fail.code != NE_NOERROR)
{
    printf("%s\n", fail.message);
    exit_status = 1;
    goto END;
}
```

/*
 * nag_gen_real_mat_print (x04cac).
 * Print real general matrix (easy-to-use)
 */

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("%s\n", fail.message);
    exit_status = 1;
    goto END;
}

printf("\nNumber of Newton steps taken: %11"NAG_IFMT"\n", iter);
printf("Number of function evaluations: %9"NAG_IFMT"\n\n", feval);
printf("\nalpha: %37.3f \n\n", alpha);
fflush(stdout);

/* 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("%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_gen_real_mat_print (x04cac). *
 * Print real general matrix (easy-to-use) *
 */
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, 1, n,
eig, pdeig, "Eigenvalues of x", NULL, &fail);
if (fail.code != NE_NOERROR)
{
    printf("%s\n", fail.message);
    exit_status = 1;
    goto END;
}

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

10.2 Program Data

nag_nearest_correlation_bounded (g02abc) Example Program Data
4 Nag_Both 0.02 :: n, opt, alpha
   2.0  -1.0  0.0  0.0
   -1.0  2.0  -1.0  0.0
    0.0  -1.0  2.0  -1.0
    0.0   0.0  -1.0  2.0 :: End of g
100.0  20.0  20.0  20.0 :: w

10.3 Program Results

nag_nearest_correlation_bounded (g02abc) Example Program Results

Nearest Correlation Matrix x
   1  2  3  4
1  1.0000 -0.9187  0.0257  0.0086
2 -0.9187  1.0000 -0.3008  0.2270
3  0.0257 -0.3008  1.0000 -0.8859
4  0.0086  0.2270 -0.8859  1.0000
Number of Newton steps taken: 5
Number of function evaluations: 6
alpha: 0.020

Eigenvalues of x

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0392</td>
<td>0.1183</td>
<td>1.6515</td>
<td>2.1910</td>
</tr>
</tbody>
</table>