NAG Library Routine Document

E04RFF

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

E04RFF is a part of the NAG optimization modelling suite and defines the linear or the quadratic objective function of the problem.

2 Specification

```
SUBROUTINE E04RFF (HANDLE, NNZC, IDXC, C, NNZH, IROWH, ICOLH, H, IFAIL)
INTEGER              NNZC, IDXC(NNZC), NNZH, IROWH(NNZH), ICOLH(NNZH),        &
                     IFAIL
REAL (KIND=nag_wp) C(NNZC), H(NNZH)
TYPE (C_PTR)         HANDLE
```

3 Description

After the initialization routine E04RAF has been called, E04RFF may be used to define the objective function of the problem as a quadratic function $c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Hx$ or a sparse linear function $c^{\mathrm{T}}x$ unless the objective function has been defined previously by E04REF, E04RFF or by E04RGF. This objective function will typically be used for quadratic programming problems (QP)

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}x^{\mathrm{T}}Hx + c^{\mathrm{T}}x & \text{(a)} \\ \text{subject to} & l_B \leq Bx \leq u_B & \text{(b)} \\ & l_x \leq x \leq u_x & \text{(c)} \end{array} \qquad (1)$$

or for semidefinite programming problems with bilinear matrix inequalities (BMI-SDP)

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}x^{\mathrm{T}}Hx + c^{\mathrm{T}}x & \text{(a)} \\ \text{subject to} & \sum_{i,j=1}^{n} x_i x_j Q_{ij}^k + \sum_{i=1}^{n} x_i A_i^k - A_0^k \succeq 0, \quad k = 1, \dots, m_A & \text{(b)} \\ & l_B \leq Bx \leq u_B & \text{(c)} \\ & l_x \leq x \leq u_x & \text{(d)} \end{array} \qquad (2)$$

The matrix H is a sparse symmetric n by n matrix. It does not need to be positive definite. See E04RAF for more details.

4 References

None.

5 Arguments

1: HANDLE – TYPE (C_PTR) *Input*

On entry: the handle to the problem. It needs to be initialized by E04RAF and **must not** be changed.

2: NNZC – INTEGER *Input*

On entry: the number of nonzero elements in the sparse vector c.

If NNZC $= 0$, c is considered to be zero and the arrays IDXC and C will not be referenced.

Constraint: NNZC ≥ 0.

3: IDXC(NNZC) – INTEGER array *Input*
4: C(NNZC) – REAL (KIND=nag_wp) array *Input*

On entry: the nonzero elements of the sparse vector c. IDXC(i) must contain the index of C(i) in the vector, for $i = 1, 2, \ldots,$ NNZC. The elements are stored in ascending order. Note that n, the number of variables in the problem, was set in NVAR during the initialization of the handle by E04RAF.

Constraints:

$1 \leq$ IDXC$(i) \leq n$, for $i = 1, 2, \ldots,$ NNZC;
IDXC$(i) <$ IDXC$(i + 1)$, for $i = 1, 2, \ldots,$ NNZC $- 1$.

5: NNZH – INTEGER *Input*

On entry: the number of nonzero elements in the upper triangle of the matrix H.

If NNZH $= 0$, the matrix H is considered to be zero, the objective function is linear and IROWH, ICOLH and H will not be referenced.

Constraint: NNZH ≥ 0.

6: IROWH(NNZH) – INTEGER array *Input*
7: ICOLH(NNZH) – INTEGER array *Input*
8: H(NNZH) – REAL (KIND=nag_wp) array *Input*

On entry: arrays IROWH, ICOLH and H store the nonzeros of the upper triangle of the matrix H in coordinate storage (CS) format (see Section 2.1.1 in the F11 Chapter Introduction). IROWH specifies one-based row indices, ICOLH specifies one-based column indices and H specifies the values of the nonzero elements in such a way that $h_{ij} =$ H(l) where $i =$ IROWH(l), $j =$ ICOLH(l), for $l = 1, 2, \ldots,$ NNZH. No particular order is expected, but elements should not repeat.

Constraint: $1 \leq$ IROWH$(l) \leq$ ICOLH$(l) \leq n$, for $l = 1, 2, \ldots,$ NNZH.

9: 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, the recommended value is -1. **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 $= 1$

The supplied HANDLE does not define a valid handle to the data structure for the NAG optimization modelling suite. It has not been initialized by E04RAF or it has been corrupted.

IFAIL $= 2$

The problem cannot be modified in this phase any more, the solver has already been called.

IFAIL $= 3$

The objective function has already been defined.

IFAIL $= 6$

On entry, NNZC $= \langle value \rangle$.
Constraint: NNZC ≥ 0.

On entry, NNZH $= \langle value \rangle$.
Constraint: NNZH ≥ 0.

IFAIL $= 7$

On entry, $i = \langle value \rangle$, IDXC$(i) = \langle value \rangle$ and IDXC$(i+1) = \langle value \rangle$.
Constraint: IDXC$(i) <$ IDXC$(i+1)$ (ascending order).

On entry, $i = \langle value \rangle$, IDXC$(i) = \langle value \rangle$ and $n = \langle value \rangle$.
Constraint: $1 \leq$ IDXC$(i) \leq n$.

IFAIL $= 8$

On entry, $i = \langle value \rangle$, ICOLH$(i) = \langle value \rangle$ and $n = \langle value \rangle$.
Constraint: $1 \leq$ ICOLH$(i) \leq n$.

On entry, $i = \langle value \rangle$, IROWH$(i) = \langle value \rangle$ and ICOLH$(i) = \langle value \rangle$.
Constraint: IROWH$(i) \leq$ ICOLH(i) (elements within the upper triangle).

On entry, $i = \langle value \rangle$, IROWH$(i) = \langle value \rangle$ and $n = \langle value \rangle$.
Constraint: $1 \leq$ IROWH$(i) \leq n$.

On entry, more than one element of H has row index $\langle value \rangle$ and column index $\langle value \rangle$.
Constraint: each element of H must have a unique row and column index.

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

Not applicable.

8 Parallelism and Performance

E04RFF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example demonstrates how to use nonlinear semidefinite programming to find a nearest correlation matrix satisfying additional requirements. This is a viable alternative to routines G02AAF, G02ABF, G02AJF or G02ANF as it easily allows you to add further constraints on the correlation matrix. In this case a problem with a linear matrix inequality and a quadratic objective function is formulated to find the nearest correlation matrix in the Frobenius norm preserving the nonzero pattern of the original input matrix. However, additional box bounds (E04RHF) or linear constraints (E04RJF) can be readily added to further bind individual elements of the new correlation matrix or new matrix inequalities (E04RNF) to restrict its eigenvalues.

The problem is as follows (to simplify the notation only the upper triangular parts are shown). To a given m by m symmetric input matrix G

$$G = \begin{pmatrix} g_{11} & \cdots & g_{1m} \\ & \ddots & \vdots \\ & & g_{mm} \end{pmatrix}$$

find correction terms $x_1, \ldots, x_n$ which form symmetric matrix $\bar{G}$

$$\bar{G} = \begin{pmatrix} \bar{g}_{11} & \bar{g}_{12} & \cdots & \bar{g}_{1m} \\ & \bar{g}_{22} & \cdots & \bar{g}_{2m} \\ & & \ddots & \vdots \\ & & & \bar{g}_{mm} \end{pmatrix} = \begin{pmatrix} 1 & g_{12}+x_1 & g_{13}+x_2 & \cdots & & g_{1m}+x_i \\ & 1 & g_{23}+x_3 & & & \\ & & 1 & & & \vdots \\ & & & \ddots & & \\ & & & & 1 & g_{m-1m}+x_n \\ & & & & & 1 \end{pmatrix}$$

so that the following requirements are met:

(a) It is a correlation matrix, i.e., symmetric positive semidefinite matrix with a unit diagonal. This is achieved by the way $\bar{G}$ is assembled and by a linear matrix inequality

$$\bar{G} = \quad x_1 \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ & 0 & 0 & \cdots & 0 \\ & & 0 & \cdots & 0 \\ & & & \ddots & \vdots \\ & & & & 0 \end{pmatrix} + x_2 \begin{pmatrix} 0 & 0 & 1 & \cdots & 0 \\ & 0 & 0 & \cdots & 0 \\ & & 0 & \cdots & 0 \\ & & & \ddots & \vdots \\ & & & & 0 \end{pmatrix} + x_3 \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ & 0 & 1 & \cdots & 0 \\ & & 0 & \cdots & 0 \\ & & & \ddots & \vdots \\ & & & & 0 \end{pmatrix} + \cdots$$
$$+x_n \begin{pmatrix} 0 & \cdots & 0 & 0 & 0 \\ & \ddots & \vdots & \vdots & \vdots \\ & & 0 & 0 & 0 \\ & & & 0 & 1 \\ & & & & 0 \end{pmatrix} - \begin{pmatrix} -1 & -g_{12} & -g_{13} & \cdots & -g_{1m} \\ & -1 & -g_{23} & \cdots & -g_{2m} \\ & & -1 & \cdots & -g_{3m} \\ & & & \ddots & \vdots \\ & & & & -1 \end{pmatrix} \succeq 0.$$

(b) $\bar{G}$ is nearest to G in the Frobenius norm, i.e., it minimizes the Frobenius norm of the difference which is equivalent to:

$$\text{minimize } \frac{1}{2}\sum_{i\neq j}\left(\bar{g}_{ij} - g_{ij}\right)^2 = \sum_{i=1}^{n} x_i^2.$$

(c) $\bar{G}$ preserves the nonzero structure of G. This is met by defining x_i only for nonzero elements g_{ij}.

For the input matrix

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

the result is

$$\bar{G} = \begin{pmatrix} 1.0000 & -0.6823 & 0.0000 & 0.0000 \\ -0.6823 & 1.0000 & -0.5344 & 0.0000 \\ 0.0000 & -0.5344 & 1.0000 & -0.6823 \\ 0.0000 & 0.0000 & -0.6823 & 1.0000 \end{pmatrix}.$$

See also Section 10 in E04RAF for links to further examples in the suite.

10.1 Program Text

```
    Program e04rffe

!     E04RFF Example Program Text

!     Compute the nearest correlation matrix in Frobenius norm
!     using nonlinear semidefinite programming. By default,
!     preserve the nonzero structure of the input matrix
!     (preserve_structure = .True.).

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: e04raf, e04rff, e04rnf, e04rzf, e04svf, e04zmf,   &
                             nag_wp, x04caf
      Use, Intrinsic                   :: iso_c_binding, Only: c_null_ptr,     &
                                          c_ptr
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
      Logical, Parameter               :: preserve_structure = .True.
!     .. Local Scalars ..
      Type (c_ptr)                     :: h
      Integer                          :: dima, i, idblk, idx, ifail, inform,  &
                                          j, n, nblk, nnzasum, nnzc, nnzh,     &
                                          nnzu, nnzua, nnzuc, nvar
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), g(:,:), hmat(:), x(:)
      Real (Kind=nag_wp)               :: rdummy(1), rinfo(32), stats(32)
      Integer, Allocatable             :: blksizea(:), icola(:), icolh(:),     &
                                          irowa(:), irowh(:), nnza(:)
      Integer                          :: idummy(1)
!     .. Executable Statements ..
      Continue

      Write (nout,*) 'E04RFF Example Program Results'
      Write (nout,*)
      Flush (nout)

!     Skip heading in data file.
      Read (nin,*)

!     Read in the problem size.
      Read (nin,*) n

      Allocate (g(n,n))

!     Read in the matrix G.
      Read (nin,*)(g(i,1:n),i=1,n)

!     Symmetrize G:  G = (G + G')/2
```

```
      Do j = 2, n
        Do i = 1, j - 1
          g(i,j) = (g(i,j)+g(j,i))/2.0_nag_wp
          g(j,i) = g(i,j)
        End Do
      End Do

!     Initialize handle.
      h = c_null_ptr

!     There are as many variables as nonzeros above the main diagonal in
!     the input matrix. The variables are corrections of these elements.
      nvar = 0
      Do j = 2, n
        Do i = 1, j - 1
          If (.Not. preserve_structure .Or. g(i,j)/=0.0_nag_wp) Then
            nvar = nvar + 1
          End If
        End Do
      End Do
      Allocate (x(nvar))

!     Initialize an empty problem handle with NVAR variables.
      ifail = 0
      Call e04raf(h,nvar,ifail)

!     Set up the objective - minimize Frobenius norm of the corrections.
!     Our variables are stored as a vector thus, just minimize
!     sum of squares of the corrections --> H is identity matrix, c = 0.
      nnzc = 0
      nnzh = nvar
      Allocate (irowh(nnzh),icolh(nnzh),hmat(nnzh))
      Do i = 1, nvar
        irowh(i) = i
        icolh(i) = i
        hmat(i) = 1.0_nag_wp
      End Do

!     Add the quadratic objective to the handle.
      ifail = 0
      Call e04rff(h,nnzc,idummy,rdummy,nnzh,irowh,icolh,hmat,ifail)

!     Construct linear matrix inequality to request that
!     matrix G with corrections X is positive semidefinite.
!     (Don't forget the sign at A_0!)

!     How many nonzeros do we need? Full triangle for A_0 and
!     one nonzero element for each A_i.
      nnzasum = n*(n+1)/2 + nvar

      Allocate (nnza(nvar+1),irowa(nnzasum),icola(nnzasum),a(nnzasum))
      nnza(1) = n*(n+1)/2
      nnza(2:nvar+1) = 1

!     Copy G to A_0, only upper triangle with different sign (because -A_0)
!     and set the diagonal to 1.0 as that's what we want independently
!     of what was in G.
      idx = 1
      Do j = 1, n
        Do i = 1, j - 1
          irowa(idx) = i
          icola(idx) = j
          a(idx) = -g(i,j)
          idx = idx + 1
        End Do
!       Unit diagonal.
        irowa(idx) = j
        icola(idx) = j
        a(idx) = -1.0_nag_wp
        idx = idx + 1
      End Do
```

```
!     A_i has just one nonzero - it binds x_i with its position as
!     a correction.
      Do j = 2, n
        Do i = 1, j - 1
          If (.Not. preserve_structure .Or. g(i,j)/=0.0_nag_wp) Then
            irowa(idx) = i
            icola(idx) = j
            a(idx) = 1.0_nag_wp
            idx = idx + 1
          End If
        End Do
      End Do

!     Just one matrix inequality of the dimension of the original matrix.
      nblk = 1
      Allocate (blksizea(nblk))
      dima = n
      blksizea(:) = (/dima/)

!     Add the constraint to the problem formulation.
      idblk = 0
      ifail = 0
      Call e04rnf(h,nvar,dima,nnza,nnzasum,irowa,icola,a,nblk,blksizea,idblk,  &
        ifail)

!     Set optional arguments of the solver.
      ifail = 0
      Call e04zmf(h,'Print Options = No',ifail)
      ifail = 0
      Call e04zmf(h,'Initial X = Automatic',ifail)

!     Pass the handle to the solver, we are not interested in
!     Lagrangian multipliers.
      nnzu = 0
      nnzuc = 0
      nnzua = 0
      ifail = 0
      Call e04svf(h,nvar,x,nnzu,rdummy,nnzuc,rdummy,nnzua,rdummy,rinfo,stats,  &
        inform,ifail)

!     Destroy the handle.
      ifail = 0
      Call e04rzf(h,ifail)

!     Form the new nearest correlation matrix as the sum
!     of G and the correction X.
      idx = 1
      Do j = 1, n
        Do i = 1, j - 1
          If (.Not. preserve_structure .Or. g(i,j)/=0.0_nag_wp) Then
            g(i,j) = g(i,j) + x(idx)
            idx = idx + 1
          End If
        End Do
        g(j,j) = 1.0_nag_wp
      End Do

!     Print the matrix.
      ifail = 0
      Call x04caf('Upper','N',n,n,g,n,'Nearest Correlation Matrix',ifail)

    End Program e04rffe
```

10.2 Program Data

```
E04RFF Example Program Data
 4                         :: N
 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
```

10.3 Program Results

```
 E04RFF Example Program Results

 E04SV, NLP-SDP Solver (Pennon)
 ------------------------------
 Number of variables             3                 [eliminated            0]
                            simple  linear  nonlin
 (Standard) inequalities         0       0       0
 (Standard) equalities                   0       0
 Matrix inequalities                     1       0 [dense    1, sparse    0]
                                                   [max dimension         4]

 --------------------------------------------------------------
  it|  objective |  optim  |   feas  |  compl  | pen min |inner
 --------------------------------------------------------------
   0  0.00000E+00  0.00E+00  6.19E-01  6.63E+00  1.00E+00   0
   1  4.12017E-01  6.38E-04  0.00E+00  1.44E+00  1.00E+00   5
   2  3.29642E-01  7.76E-04  0.00E+00  4.96E-01  4.65E-01   2
   3  2.65315E-01  1.02E-04  0.00E+00  1.55E-01  2.16E-01   3
   4  2.33229E-01  1.03E-03  0.00E+00  4.71E-02  1.01E-01   3
   5  2.19082E-01  2.22E-03  0.00E+00  1.46E-02  4.68E-02   3
   6  2.13121E-01  2.12E-03  0.00E+00  4.72E-03  2.18E-02   3
   7  2.10698E-01  1.26E-03  0.00E+00  1.56E-03  1.01E-02   3
   8  2.09756E-01  4.90E-04  0.00E+00  4.85E-04  4.71E-03   3
   9  2.09413E-01  1.13E-04  0.00E+00  1.21E-04  2.19E-03   3
  10  2.09310E-01  1.95E-03  0.00E+00  1.63E-05  1.02E-03   2
  11  2.09297E-01  1.25E-05  0.00E+00  2.77E-06  4.74E-04   2
  12  2.09294E-01  2.68E-07  0.00E+00  3.89E-07  2.21E-04   2
  13  2.09294E-01  2.25E-09  0.00E+00  5.43E-08  1.03E-04   2
 --------------------------------------------------------------
 Status: converged, an optimal solution found
 --------------------------------------------------------------
 Final objective value                 2.092940E-01
 Relative precision                    2.759238E-07
 Optimality                            2.249294E-09
 Feasibility                           0.000000E+00
 Complementarity                       5.426796E-08
 Iteration counts
   Outer iterations                              13
   Inner iterations                              36
   Linesearch steps                              36
 Evaluation counts
   Augm. Lagr. values                            50
   Augm. Lagr. gradient                          50
   Augm. Lagr. hessian                           36
 --------------------------------------------------------------
 Nearest Correlation Matrix
          1       2       3       4
 1   1.0000 -0.6823  0.0000  0.0000
 2           1.0000 -0.5344  0.0000
 3                   1.0000 -0.6823
 4                           1.0000
```