NAG Library Routine Document

E04RNF

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

E04RNF is a part of the NAG optimization modelling suite and defines one or more linear matrix constraints of the problem.

2 Specification

```
SUBROUTINE E04RNF (HANDLE, NVAR, DIMA, NNZA, NNZASUM, IROWA, ICOLA, A,      &
                   NBLK, BLKSIZEA, IDBLK, IFAIL)
INTEGER            NVAR, DIMA, NNZA(NVAR+1), NNZASUM, IROWA(NNZASUM),      &
                   ICOLA(NNZASUM), NBLK, BLKSIZEA(NBLK), IDBLK, IFAIL
REAL (KIND=nag_wp) A(NNZASUM)
TYPE (C_PTR)       HANDLE
```

3 Description

After the initialization routine E04RAF has been called, E04RNF may be used to add one or more linear matrix inequalities

$$\sum_{i=1}^{n} x_i A_i - A_0 \succeq 0 \qquad (1)$$

to the problem definition. Here A_i are d by d symmetric matrices. The expression $S \succeq 0$ stands for a constraint on eigenvalues of a symmetric matrix S, namely, all the eigenvalues should be non-negative, i.e., the matrix S should be positive semidefinite.

Typically, this will be used in linear semidefinite programming problems (SDP)

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

or to define the linear part of bilinear matrix inequalities (3)(b) in (BMI-SDP)

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}x^{\mathrm{T}}Hx + c^{\mathrm{T}}x & \text{(a)} \\ \text{subject to} & \sum_{i,j=1}^{n} x_i x_j Q_{ij}^k + \sum_{i=1}^{n} x_i A_i^k - A_0^k \succeq 0, \quad k=1,\dots,m_A & \text{(b)} \\ & l_B \le Bx \le u_B & \text{(c)} \\ & l_x \le x \le u_x & \text{(d)} \end{array} \qquad (3)$$

E04RNF can be called repeatedly to accumulate more matrix inequalities. See E04RAF for more details.

3.1 Input data organization

All the matrices A_i, for $i = 0,1,\dots,n$, are symmetric and thus only their upper triangles are passed to the routine. They are stored in sparse coordinate storage format (see Section 2.1.1 in the F11 Chapter Introduction), i.e., every nonzero from the upper triangles is coded as a triplet of row index, column index and the numeric value. These triplets of all (upper triangle) nonzeros from all A_i matrices are passed to the routine in three arrays: IROWA for row indices, ICOLA for column indices and A for the values. No particular order of nonzeros within one matrix is enforced but all nonzeros from A_0 must be stored first, followed by all nonzero from A_1, followed by A_2, etc.

The number of stored nonzeros from each A_i matrix is given in NNZA$(i+1)$, thus this array indicates which section of arrays IROWA, ICOLA and A belongs to which A_i matrix. See Table 1 and the example in Section 9. See also E04RDF which uses the same data organization.

IROWA ICOLA A	upper triangle nonzeros $\underbrace{\text{from } A_0}_{\text{NNZA}(1)}$	upper triangle nonzeros $\underbrace{\text{from } A_1}_{\text{NNZA}(2)}$	$\cdots$	upper triangle nonzeros $\underbrace{\text{from } A_n}_{\text{NNZA}(n+1)}$

Table 1
Coordinate storage format of matrices $A_0, A_1, \ldots, A_n$ in input arrays

There are two possibilities for defining more matrix inequality constraints

$$\sum_{i=1}^{n} x_i A_i^k - A_0^k \succeq 0, \quad k = 1, 2, \ldots, m_A \tag{4}$$

to the problem. The first is to call E04RNF m_A times and define a single matrix inequality at a time. This might be more straightforward and therefore it is recommended. Alternatively, it is possible to merge all m_A constraints into one inequality and pass them in a single call to E04RNF. It is easy to see that (4) can be equivalently expressed as one bigger matrix inequality with the following block diagonal structure

$$\sum_{i=1}^{n} x_i \begin{pmatrix} A_i^1 & & & \\ & A_i^2 & & \\ & & \ddots & \\ & & & A_i^{m_A} \end{pmatrix} - \begin{pmatrix} A_0^1 & & & \\ & A_0^2 & & \\ & & \ddots & \\ & & & A_0^{m_A} \end{pmatrix} \succeq 0.$$

If d_k denotes the dimension of inequality k, the new merged inequality has dimension $d = \sum_{k=1}^{m_A} d_k$ and each of the A_i matrices is formed by $A_i^1, A_i^2, \ldots, A_i^{m_A}$ stored as m_A diagonal blocks. In such a case, NBLK is set to m_A and BLKSIZEA(k) to d_k, the size of the kth diagonal blocks. This might be useful in connection with E04RDF.

On the other hand, if there is no block structure and just one matrix inequality is provided, NBLK should be set to 1 and BLKSIZEA is not referenced.

3.2 Definition of Bilinear Matrix Inequalities (BMI)

E04RNF is designed to be used together with E04RPF to define bilinear matrix inequalities (3)(b). E04RNF sets the linear part of the constraint and E04RPF expands it by higher order terms. To distinquish which linear matrix inequality (or more precisely, which block) is to be expanded, E04RPF needs the number of the block, IDBLK. The blocks are numbered as they are added, starting from 1.

Whenever a matrix inequality (or a set of them expressed as diagonal blocks) is stored, the routine returns IDBLK of the last inequality added. IDBLK is just the order of the inequality amongst all matrix inequalities accumulated through the calls. The first inequality has IDBLK $= 1$, the second one IDBLK $= 2$, etc. Therefore if you call E04RNF for the very first time with NBLK $= 42$, it adds 42 inequalities with IDBLK from 1 to 42 and the routine returns IDBLK $= 42$ (the number of the last one). A subsequent call with NBLK $= 1$ would add only one inequality, this time with IDBLK $= 43$ which would be returned.

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: NVAR – INTEGER *Input*

On entry: n, the number of decision variables x in the problem. It must be unchanged from the value set during the initialization of the handle by E04RAF.

3: DIMA – INTEGER *Input*

On entry: d, the dimension of the matrices A_i, for $i = 0, 1, \ldots, \text{NVAR}$.

Constraint: DIMA > 0.

4: NNZA(NVAR + 1) – INTEGER array *Input*

On entry: NNZA$(i+1)$, for $i = 0, 1, \ldots, \text{NVAR}$, gives the number of nonzero elements in the upper triangle of matrix A_i. To define A_i as a zero matrix, set NNZA$(i+1) = 0$. However, there must be at least one matrix with at least one nonzero.

Constraints:

$$\text{NNZA}(i) \geq 0;$$
$$\sum_{i=1}^{n+1} \text{NNZA}(i) \geq 1.$$

5: NNZASUM – INTEGER *Input*

On entry: the dimension of the arrays IROWA, ICOLA and A, at least the total number of all nonzeros in all matrices A_i.

Constraints:

$$\text{NNZASUM} > 0;$$
$$\sum_{i=1}^{n+1} \text{NNZA}(i) \leq \text{NNZASUM}.$$

6: IROWA(NNZASUM) – INTEGER array *Input*
7: ICOLA(NNZASUM) – INTEGER array *Input*
8: A(NNZASUM) – REAL (KIND=nag_wp) array *Input*

On entry: nonzero elements in upper triangle of matrices A_i stored in coordinate storage. The first NNZA(1) elements belong to A_0, the following NNZA(2) elements belong to A_1, etc. See explanation above.

Constraints:

$1 \leq \text{IROWA}(i) \leq \text{DIMA}$, $\text{IROWA}(i) \leq \text{ICOLA}(i) \leq \text{DIMA}$;
IROWA and ICOLA match the block diagonal pattern set by BLKSIZEA.

9: NBLK – INTEGER *Input*

On entry: m_A, number of diagonal blocks in A_i matrices. As explained above it is equivalent to the number of matrix inequalities supplied in this call.

Constraint: NBLK ≥ 1.

10: BLKSIZEA(NBLK) – INTEGER array *Input*

On entry: if NBLK > 1, sizes d_k of the diagonal blocks.

If NBLK $= 1$, BLKSIZEA is not referenced.

Constraints:

BLKSIZEA$(i) \geq 1$;

$\sum_{i=1}^{m_A} \text{BLKSIZEA}(i) = \text{DIMA}$.

11: IDBLK – INTEGER *Input/Output*

On entry: if IDBLK $= 0$, new matrix inequalities are created. This is the only value allowed at the moment; nonzero values are reserved for future releases of the NAG Library.

Constraint: IDBLK $= 0$.

On exit: the number of the last matrix inequality added. By definition, it is the number of the matrix inequalities already defined plus NBLK.

12: 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 $= 4$

On entry, IDBLK $= \langle$*value*$\rangle$.
Constraint: IDBLK $= 0$.

On entry, NVAR $= \langle$*value*$\rangle$, expected value $= \langle$*value*$\rangle$.
Constraint: NVAR must match the value given during initialization of HANDLE.

IFAIL $= 6$

On entry, DIMA $= \langle$*value*$\rangle$.
Constraint: DIMA > 0.

On entry, $i = \langle$*value*$\rangle$ and NNZA$(i) = \langle$*value*$\rangle$.
Constraint: NNZA$(i) \geq 0$.

On entry, NNZASUM $= \langle$*value*$\rangle$ and sum(NNZA) $= \langle$*value*$\rangle$.
Constraint: NNZASUM $\geq$ sum(NNZA).

On entry, $\mathrm{sum}\,(\mathrm{NNZA}) = \langle value \rangle$.
Constraint: $\mathrm{sum}\,(\mathrm{NNZA}) \geq 1$.

IFAIL $= 7$

On entry, DIMA $= \langle value \rangle$ and $\mathrm{sum}\,(\mathrm{BLKSIZEA}) = \langle value \rangle$.
Constraint: $\mathrm{sum}\,(\mathrm{BLKSIZEA}) = \mathrm{DIMA}$.

On entry, $i = \langle value \rangle$ and $\mathrm{BLKSIZEA}(i) = \langle value \rangle$.
Constraint: $\mathrm{BLKSIZEA}(i) \geq 1$.

On entry, NBLK $= \langle value \rangle$.
Constraint: NBLK > 0.

IFAIL $= 8$

An error occurred in matrix A_i, $i = \langle value \rangle$ (counting indices $1 \ldots \mathrm{NVAR} + 1$).
On entry, $j = \langle value \rangle$, $\mathrm{ICOLA}(j) = \langle value \rangle$ and DIMA $= \langle value \rangle$.
Constraint: $1 \leq \mathrm{ICOLA}(j) \leq \mathrm{DIMA}$.

An error occurred in matrix A_i, $i = \langle value \rangle$ (counting indices $1 \ldots \mathrm{NVAR} + 1$).
On entry, $j = \langle value \rangle$, $\mathrm{IROWA}(j) = \langle value \rangle$ and DIMA $= \langle value \rangle$.
Constraint: $1 \leq \mathrm{IROWA}(j) \leq \mathrm{DIMA}$.

An error occurred in matrix A_i, $i = \langle value \rangle$ (counting indices $1 \ldots \mathrm{NVAR} + 1$).
On entry, $j = \langle value \rangle$, $\mathrm{IROWA}(j) = \langle value \rangle$ and $\mathrm{ICOLA}(j) = \langle value \rangle$.
Constraint: $\mathrm{IROWA}(j) \leq \mathrm{ICOLA}(j)$ (elements within the upper triangle).

An error occurred in matrix A_i, $i = \langle value \rangle$ (counting indices $1 \ldots \mathrm{NVAR} + 1$).
On entry, $j = \langle value \rangle$, $\mathrm{IROWA}(j) = \langle value \rangle$ and $\mathrm{ICOLA}(j) = \langle value \rangle$. Maximum column index in this row given by the block structure defined by BLKSIZEA is $\langle value \rangle$.
Constraint: all elements of A_i must respect the block structure given by BLKSIZEA.

An error occurred in matrix A_i, $i = \langle value \rangle$ (counting indices $1 \ldots \mathrm{NVAR} + 1$).
On entry, more than one element of A_i has row index $\langle value \rangle$ and column index $\langle value \rangle$.
Constraint: each element of A_i 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

E04RNF is not threaded in any implementation.

9 Further Comments

The following example demonstrates how the elements of the A_i^k matrices are organized within the input arrays. Let us assume that there are two blocks defined (NBLK $= 2$). The first has dimension 3 by 3 (BLKSIZEA$(1) = 3$) and the second 2 by 2 (BLKSIZEA$(2) = 2$). For simplicity, the number of variables is 2. Please note that the values were chosen to ease orientation rather than to define a valid problem.

$$A_0^1 = \begin{pmatrix} 0.1 & 0 & 0.3 \\ 0 & 0.2 & 0.4 \\ 0.3 & 0.4 & 0 \end{pmatrix}, \qquad A_1^1 \text{ empty} \qquad A_2^1 = \begin{pmatrix} 2.1 & 0 & 0 \\ 0 & 2.2 & 0 \\ 0 & 0 & 2.3 \end{pmatrix},$$

$$A_0^2 = \begin{pmatrix} 0 & -0.1 \\ -0.1 & 0 \end{pmatrix}, \quad A_1^2 = \begin{pmatrix} -1.1 & 0 \\ 0 & -1.2 \end{pmatrix}, \quad A_2^2 = \begin{pmatrix} -2.1 & -2.2 \\ -2.2 & -2.3 \end{pmatrix}.$$

Both inequalities will be passed in a single call to E04RNF, therefore the matrices are merged into the following block diagonal form:

$$A_0 = \begin{pmatrix} 0.1 & 0 & 0.3 & & \\ 0 & 0.2 & 0.4 & & \\ 0.3 & 0.4 & 0 & & \\ & & & 0 & -0.1 \\ & & & -0.1 & 0 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 0 & 0 & 0 & & \\ 0 & 0 & 0 & & \\ 0 & 0 & 0 & & \\ & & & -1.1 & 0 \\ & & & 0 & -1.2 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 2.1 & 0 & 0 & & \\ 0 & 2.2 & 0 & & \\ 0 & 0 & 2.3 & & \\ & & & -2.1 & -2.2 \\ & & & -2.2 & -2.3 \end{pmatrix}.$$

All matrices are symmetric and therefore only the upper triangles are passed to the routine. The coordinate storage format is used. Note that elements within the same matrix do not need to be in any specific order. The table below shows one of the ways the arrays could be populated.

IROWA	2	2	4	1	1	4	5	1	2	3	4	4	5
ICOLA	2	3	5	1	3	4	5	1	2	3	4	5	5
A	0.2	0.4	−0.1	0.1	0.3	−1.1	−1.2	2.1	2.2	2.3	−2.1	−2.2	−2.3
	A_0					A_1		A_2					
NNZA	5					2		6					

10 Example

There are various problems which can be successfully reformulated and solved as an SDP problem. The following example shows how a maximization of the minimal eigenvalue of a matrix depending on certain parameters can be utilized in statistics.

For further examples, please refer to Section 10 in E04RAF.

Given a series of M vectors of length p, $\{v_i : i = 1, 2, \ldots, M\}$ this example solves the SDP problem:

$$\begin{array}{ll} \underset{\lambda_1,\ldots,\lambda_M,t}{\text{maximize}} & t \\ \text{subject to} & \sum_{i=1}^{M} \lambda_i v_i v_i^{\mathrm{T}} \succeq tI \\ & \sum_{i=1}^{M} \lambda_i = 1 \\ & \lambda_i \geq 0, \quad k = 1, \ldots, M. \end{array}$$

This formulation comes from an area of statistics called experimental design and corresponds to finding an approximate E optimal design for a linear regression.

A linear regression model has the form:

$$y = X\beta + \epsilon$$

where y is a vector of observed values, X is a design matrix of (known) independent variables and ϵ is a vector of errors. In experimental design it is assumed that each row of X is chosen from a set of M possible vectors, $\{v_i : i = 1, 2, \ldots, M\}$. The goal of experimental design is to choose the rows of X so that the error covariance is 'small'. For an E optimal design this is defined as the X that maximizes the minimum eigenvalue of $X^{\mathrm{T}}X$.

In this example we construct the E optimal design for a polynomial regression model of the form:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \epsilon$$

where $x \in \{1 - j \times 0.05 : j = 0, 1, \ldots, 40\}$.

10.1 Program Text

```
    Program e04rnfe

!     E04RNF Example Program Text
!     Mark 26 Release. NAG Copyright 2016.

!     Compute E-optimal experiment design via semidefinite programming,
!     this can be done as follows
!       max {lambda_min(A) | A = sum x_i*v_i*v_i^T, x_i>=0, sum x_i = 1}
!     where v_i are given vectors.

!     Use nag_library

!     .. Use Statements ..
      Use nag_library, Only: e04raf, e04rff, e04rhf, e04rjf, e04rnf, e04rzf,   &
                             e04svf, e04zmf, nag_wp
      Use, Intrinsic                   :: iso_c_binding, Only: c_null_ptr,     &
                                          c_ptr
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: big = 1E+20_nag_wp
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Type (c_ptr)                     :: h
      Real (Kind=nag_wp)               :: tol
      Integer                          :: dima, i, idblk, idlc, idx, ifail,    &
                                          inform, j, k, m, nblk, nnzasum,      &
                                          nnzb, nnzc, nnzu, nnzua, nnzuc,      &
                                          nvar, p
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), b(:), bl(:), bu(:), c(:),      &
                                          v(:,:), x(:)
      Real (Kind=nag_wp)               :: rdummy(1), rinfo(32), stats(32)
      Integer, Allocatable             :: blksizea(:), icola(:), icolb(:),     &
                                          idxc(:), irowa(:), irowb(:), nnza(:)
      Integer                          :: idummy(1)
!     .. Intrinsic Procedures ..
```

```
      Intrinsic                          :: repeat
!     .. Executable Statements ..
      Continue

      Write (nout,*) 'E04RNF Example Program Results'
      Write (nout,*)
      Flush (nout)

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

!     Read in the number of vectors and their size.
      Read (nin,*) m
      Read (nin,*) p

      Allocate (v(p,m))

!     Read in the vectors v_j.
      Do j = 1, m
        Read (nin,*)(v(i,j),i=1,p)
      End Do

!     Initialize handle.
      h = c_null_ptr

!     Variables of the problem will be x_1, ..., x_m (weights of the vectors)
!     and t (artificial variable for minimum eigenvalue).
      nvar = m + 1

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

!     Add the objective function to the handle: max t.
      nnzc = 1
      Allocate (idxc(nnzc),c(nnzc))
      idxc(:) = (/m+1/)
      c(:) = (/1._nag_wp/)

      ifail = 0
      Call e04rff(h,nnzc,idxc,c,0,idummy,idummy,rdummy,ifail)

      Allocate (bl(nvar),bu(nvar))
      bl(1:m) = 0.0_nag_wp
      bl(m+1) = -big
      bu(1:m+1) = big

!     Add simple bounds on variables, x_i>=0.
      ifail = 0
      Call e04rhf(h,nvar,bl,bu,ifail)

      nnzb = m
      Allocate (irowb(nnzb),icolb(nnzb),b(nnzb))
      irowb(:) = 1
      icolb(:) = (/(j,j=1,m)/)
      b(:) = 1.0_nag_wp

!     Add the linear constraint: sum x_i = 1.
      idlc = 0
      ifail = 0
      Call e04rjf(h,1,(/1.0_nag_wp/),(/1.0_nag_wp/),nnzb,irowb,icolb,b,idlc,   &
        ifail)

!     Generate matrix constraint as:
!       sum_{i=1}^m x_i*v_i*v_i^T - t*I >=0

      nblk = 1
      dima = p

!     Total number of nonzeros
      nnzasum = p + m*(p+1)*p/2
```

```
      Allocate (nnza(nvar+1),irowa(nnzasum),icola(nnzasum),a(nnzasum),x(nvar))
!     A_0 is empty
      nnza(1) = 0
!     A_1, A_2, ..., A_m are v_i*v_i^T
      nnza(2:m+1) = (p+1)*p/2
      idx = 0
      Do k = 1, m
        Do i = 1, p
          Do j = i, p
            idx = idx + 1
            irowa(idx) = i
            icola(idx) = j
            a(idx) = v(i,k)*v(j,k)
          End Do
        End Do
      End Do
!     A_{m+1} is the -identity
      nnza(m+2) = p
      Do i = 1, p
        idx = idx + 1
        irowa(idx) = i
        icola(idx) = i
        a(idx) = -1.0_nag_wp
      End Do

!     Add the constraint to the problem formulation.
      Allocate (blksizea(nblk))
      blksizea(:) = (/dima/)

      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,'Task = Maximize',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)

!     Print results
      Write (nout,*)
      tol = 0.00001_nag_wp
      Write (nout,*) '  Weight        Row of design matrix'
      Write (nout,*) repeat('-',13+p*8)
      Do j = 1, m
        If (x(j)>tol) Then
          Write (*,99999) x(j), v(1:p,j)
        End If
      End Do
      Write (nout,99998) 'only those rows with a weight > ', tol, ' are shown'

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

99999 Format (1X,F7.2,5X,10(1X,F7.2))
99998 Format (1X,A,E8.1,A)
    End Program e04rnfe
```

10.2 Program Data

```
E04RNF Example Program Data
41          : Number of vectors to choose from
5           : Length of vectors
 1.00000000 -1.00000000  1.00000000 -1.00000000  1.00000000
 1.00000000 -0.95000000  0.90250000 -0.85737500  0.81450625
 1.00000000 -0.90000000  0.81000000 -0.72900000  0.65610000
 1.00000000 -0.85000000  0.72250000 -0.61412500  0.52200625
 1.00000000 -0.80000000  0.64000000 -0.51200000  0.40960000
 1.00000000 -0.75000000  0.56250000 -0.42187500  0.31640625
 1.00000000 -0.70000000  0.49000000 -0.34300000  0.24010000
 1.00000000 -0.65000000  0.42250000 -0.27462500  0.17850625
 1.00000000 -0.60000000  0.36000000 -0.21600000  0.12960000
 1.00000000 -0.55000000  0.30250000 -0.16637500  0.09150625
 1.00000000 -0.50000000  0.25000000 -0.12500000  0.06250000
 1.00000000 -0.45000000  0.20250000 -0.09112500  0.04100625
 1.00000000 -0.40000000  0.16000000 -0.06400000  0.02560000
 1.00000000 -0.35000000  0.12250000 -0.04287500  0.01500625
 1.00000000 -0.30000000  0.09000000 -0.02700000  0.00810000
 1.00000000 -0.25000000  0.06250000 -0.01562500  0.00390625
 1.00000000 -0.20000000  0.04000000 -0.00800000  0.00160000
 1.00000000 -0.15000000  0.02250000 -0.00337500  0.00050625
 1.00000000 -0.10000000  0.01000000 -0.00100000  0.00010000
 1.00000000 -0.05000000  0.00250000 -0.00012500  0.00000625
 1.00000000  0.00000000  0.00000000  0.00000000  0.00000000
 1.00000000  0.05000000  0.00250000  0.00012500  0.00000625
 1.00000000  0.10000000  0.01000000  0.00100000  0.00010000
 1.00000000  0.15000000  0.02250000  0.00337500  0.00050625
 1.00000000  0.20000000  0.04000000  0.00800000  0.00160000
 1.00000000  0.25000000  0.06250000  0.01562500  0.00390625
 1.00000000  0.30000000  0.09000000  0.02700000  0.00810000
 1.00000000  0.35000000  0.12250000  0.04287500  0.01500625
 1.00000000  0.40000000  0.16000000  0.06400000  0.02560000
 1.00000000  0.45000000  0.20250000  0.09112500  0.04100625
 1.00000000  0.50000000  0.25000000  0.12500000  0.06250000
 1.00000000  0.55000000  0.30250000  0.16637500  0.09150625
 1.00000000  0.60000000  0.36000000  0.21600000  0.12960000
 1.00000000  0.65000000  0.42250000  0.27462500  0.17850625
 1.00000000  0.70000000  0.49000000  0.34300000  0.24010000
 1.00000000  0.75000000  0.56250000  0.42187500  0.31640625
 1.00000000  0.80000000  0.64000000  0.51200000  0.40960000
 1.00000000  0.85000000  0.72250000  0.61412500  0.52200625
 1.00000000  0.90000000  0.81000000  0.72900000  0.65610000
 1.00000000  0.95000000  0.90250000  0.85737500  0.81450625
 1.00000000  1.00000000  1.00000000  1.00000000  1.00000000
```

10.3 Program Results

```
 E04RNF Example Program Results

 E04SV, NLP-SDP Solver (Pennon)
 ------------------------------
 Number of variables           42                 [eliminated            0]
                           simple  linear  nonlin
 (Standard) inequalities       41       2       0
 (Standard) equalities                  0       0
 Matrix inequalities                    1       0 [dense    1, sparse    0]
                                                  [max dimension         5]

 Begin of Options
     Outer Iteration Limit         =                 100     * d
     Inner Iteration Limit         =                 100     * d
     Infinite Bound Size           =         1.00000E+20     * d
     Initial X                     =           Automatic     * U
     Initial U                     =           Automatic     * d
     Initial P                     =           Automatic     * d
     Hessian Density               =               Dense     * S
     Init Value P                  =         1.00000E+00     * d
     Init Value Pmat               =         1.00000E+00     * d
```

```
    Presolve Block Detect         =                 Yes     * d
    Print File                    =                   6     * d
    Print Level                   =                   2     * d
    Print Options                 =                 Yes     * d
    Monitoring File               =                  -1     * d
    Monitoring Level              =                   4     * d
    Monitor Frequency             =                   0     * d
    Stats Time                    =                  No     * d
    P Min                         =         1.05367E-08     * d
    Pmat Min                      =         1.05367E-08     * d
    U Update Restriction          =         5.00000E-01     * d
    Umat Update Restriction       =         3.00000E-01     * d
    Preference                    =               Speed     * d
    Transform Constraints         =          Equalities     * S
    Dimacs Measures               =               Check     * d
    Stop Criteria                 =                Soft     * d
    Stop Tolerance 1              =         1.00000E-06     * d
    Stop Tolerance 2              =         1.00000E-07     * d
    Stop Tolerance Feasibility    =         1.00000E-07     * d
    Linesearch Mode               =            Fullstep     * S
    Inner Stop Tolerance          =         1.00000E-02     * d
    Inner Stop Criteria           =           Heuristic     * d
    Task                          =            Maximize     * U
    P Update Speed                =                  12     * d
End of Options
--------------------------------------------------------------
 it|  objective |  optim  |   feas  |  compl  | pen min |inner
--------------------------------------------------------------
  0  0.00000E+00  4.80E+01  5.90E-01  2.37E+00  1.00E+00   0
  1 -2.25709E+00  2.53E-03  7.15E-01  2.76E+00  1.00E+00   6
  2 -9.90666E-01  1.29E-03  1.38E-02  1.25E+00  4.65E-01   5
  3 -3.96590E-01  1.52E-03  2.07E-02  5.42E-01  2.16E-01   5
  4 -1.52400E-01  6.63E-04  1.42E-02  2.26E-01  1.01E-01   5
  5 -5.45545E-02  5.47E-03  9.33E-03  8.91E-02  4.68E-02   5
  6 -1.62316E-02  1.05E-02  3.18E-03  3.33E-02  2.18E-02   5
  7 -2.39571E-03  6.74E-03  3.90E-04  1.22E-02  1.01E-02   5
  8  3.39831E-03  5.41E-04  4.33E-05  4.43E-03  4.71E-03   6
  9  6.27924E-03  2.25E-03  3.47E-06  1.64E-03  2.19E-03   5
 10  7.23641E-03  4.07E-03  4.79E-07  5.77E-04  1.02E-03   4
 11  7.56230E-03  5.26E-04  1.76E-05  2.08E-04  4.74E-04   4
 12  7.67523E-03  1.18E-02  2.18E-06  7.69E-05  2.21E-04   3
 13  7.71758E-03  4.26E-03  2.51E-07  2.94E-05  1.03E-04   3
 14  7.73491E-03  4.34E-06  2.95E-08  1.11E-05  4.77E-05   4
--------------------------------------------------------------
 it|  objective |  optim  |   feas  |  compl  | pen min |inner
--------------------------------------------------------------
 15  7.74186E-03  8.50E-07  2.29E-09  3.96E-06  2.22E-05   4
 16  7.74450E-03  7.25E-08  1.58E-10  1.29E-06  1.03E-05   4
 17  7.74545E-03  2.51E-09  8.39E-12  3.32E-07  4.81E-06   4
 18  7.74574E-03  5.19E-10  3.49E-13  4.73E-08  2.24E-06   4
--------------------------------------------------------------
Status: converged, an optimal solution found
--------------------------------------------------------------
Final objective value                 7.745738E-03
Relative precision                    2.815426E-07
Optimality                            5.188682E-10
Feasibility                           3.486927E-13
Complementarity                       4.732416E-08
DIMACS error 1                        2.594341E-10
DIMACS error 2                        0.000000E+00
DIMACS error 3                        0.000000E+00
DIMACS error 4                        1.743464E-13
DIMACS error 5                        4.676597E-08
DIMACS error 6                        4.662494E-08
Iteration counts
  Outer iterations                              18
  Inner iterations                              81
  Linesearch steps                             186
Evaluation counts
  Augm. Lagr. values                           100
  Augm. Lagr. gradient                         100
```

```
  Augm. Lagr. hessian                         81
--------------------------------------------------------------

  Weight        Row of design matrix
-----------------------------------------------------
   0.09         1.00   -1.00    1.00   -1.00    1.00
   0.25         1.00   -0.70    0.49   -0.34    0.24
   0.32         1.00    0.00    0.00    0.00    0.00
   0.25         1.00    0.70    0.49    0.34    0.24
   0.09         1.00    1.00    1.00    1.00    1.00
only those rows with a weight >  0.1E-04 are shown
```