NAG Library Routine Document

E04RPF

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

E04RPF is a part of the NAG optimization modelling suite and defines bilinear matrix terms either in a new matrix constraint or adds them to an existing linear matrix inequality.

2 Specification

```
SUBROUTINE E04RPF (HANDLE, NQ, QI, QJ, DIMQ, NNZQ, NNZQSUM, IROWQ,          &
                   ICOLQ, Q, IDBLK, IFAIL)
INTEGER            NQ, QI(NQ), QJ(NQ), DIMQ, NNZQ(NQ), NNZQSUM,             &
                   IROWQ(NNZQSUM), ICOLQ(NNZQSUM), IDBLK, IFAIL
REAL (KIND=nag_wp) Q(NNZQSUM)
TYPE (C_PTR)       HANDLE
```

3 Description

After the initialization routine E04RAF has been called, E04RPF may be used to define bilinear matrix terms. It may be used in two ways, either to add to the problem formulation a new bilinear matrix inequality (BMI) which does not have linear terms:

$$\sum_{i,j=1}^{n} x_i x_j Q_{ij} \succeq 0 \tag{1}$$

or to extend an existing linear matrix inequality constraint by bilinear terms:

$$\sum_{i,j=1}^{n} x_i x_j Q_{ij}^k. \tag{2}$$

Here Q_{ij}^k are d by d (sparse) symmetric matrices and k, if present, is the number of the existing constraint. This routine will typically be used on semidefinite programming problems with bilinear matrix constraints (BMI-SDP)

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

The routine can be called multiple times to define an additional matrix inequality or to extend an existing one, but it cannot be called twice to extend the same matrix inequality. The argument IDBLK is used to distinguish whether a new matrix constraint should be added (IDBLK $= 0$) or if an existing linear matrix inequality should be extended (IDBLK > 0). In the latter case, IDBLK should be set to k, the number of the existing inequality. See E04RNF for details about formulation of linear matrix constraints and their numbering and a further description of IDBLK. For a generic description of the problem see E04RAF. In the further text, the index k will be omitted.

3.1 Input data organization

It is expected that only some of the matrices Q_{ij} will be nonzero therefore only their index pairs i, j are listed in arrays QI and QJ. Note that a pair i, j should not repeat, i.e., a matrix Q_{ij} should not be defined more than once. No particular ordering of pairs i, j is expected but other input arrays IROWQ, ICOLQ, Q and NNZQ need to respect the chosen order.

Note: the dimension of Q_{ij} must respect the size of the linear matrix inequality if they are supposed to expand it (case IDBLK > 0).

Matrices Q_{ij} 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. All these triplets from all Q_{ij} matrices are passed to the routine in three arrays: IROWQ for row indices, ICOLQ for column indices and Q for the values. No particular order of nonzeros within one matrix is enforced but all nonzeros belonging to one Q_{ij} matrix need to be stored next to each other. The first NNZQ(1) nonzeros belong to $Q_{i_1 j_1}$ where $i_1 = \mathrm{QI}(1)$, $j_1 = \mathrm{QJ}(1)$, the following NNZQ(2) nonzeros to the next one given by QI, QJ and so on. The array NNZQ thus splits arrays IROWQ, ICOLQ and Q into sections so that each section defines one Q_{ij} matrix. See Table 1 below. Routines E04RDF and E04RNF use the same data organization so further examples can be found there.

IROWQ ICOLQ Q	$\underbrace{\text{upper triangle nonzeros from } Q_{i_1 j_1}}$	$\underbrace{\text{upper triangle nonzeros from } Q_{i_2 j_2}}$	$\cdots$	$\underbrace{\text{upper triangle nonzeros from } Q_{i_{\mathrm{NQ}} j_{\mathrm{NQ}}}}$
	NNZQ(1) $i_1 = \mathrm{QI}(1)$ $j_1 = \mathrm{QJ}(1)$	NNZQ(2) $i_2 = \mathrm{QI}(2)$ $j_2 = \mathrm{QJ}(2)$		NNZQ(NQ) $i_{\mathrm{NQ}} = \mathrm{QI}(\mathrm{NQ})$ $j_{\mathrm{NQ}} = \mathrm{QJ}(\mathrm{NQ})$

Table 1
Coordinate storage format of Q_{ij} matrices in input arrays

4 References

Syrmos V L, Abdallah C T, Dorato P and Grigoriadis K (1997) Static output feedback – a survey *Journal Automatica (Journal of IFAC) (Volume 33)* **2** 125–137

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: NQ – INTEGER *Input*

On entry: the number of index pairs i, j of the nonzero matrices Q_{ij}.

Constraint: NQ > 0.

3: QI(NQ) – INTEGER array *Input*
4: QJ(NQ) – INTEGER array *Input*

On entry: the index pairs i, j of the nonzero matrices Q_{ij} in any order.

Constraint: $1 \leq i, j \leq n$ where n is the number of decision variables in the problem set during the initialization of the handle by E04RAF. The pairs do not repeat.

5: DIMQ – INTEGER *Input*

On entry: d, the dimension of matrices Q_{ij}.

Constraints:

DIMQ > 0;
if IDBLK > 0, DIMQ needs to be identical to the dimension of matrices of the constraint k.

6: NNZQ(NQ) – INTEGER array *Input*

On entry: the numbers of nonzero elements in the upper triangles of Q_{ij} matrices.

Constraint: NNZQ$(i) > 0$.

7: NNZQSUM – INTEGER *Input*

On entry: the dimension of the arrays IROWQ, ICOLQ and Q, at least the total number of all nonzeros in all Q_{ij} matrices.

Constraints:

NNZQSUM > 0;

NNZQSUM $\geq \sum_{k=1}^{\mathrm{NQ}} \mathrm{NNZQ}(k)$.

8: IROWQ(NNZQSUM) – INTEGER array *Input*
9: ICOLQ(NNZQSUM) – INTEGER array *Input*
10: Q(NNZQSUM) – REAL (KIND=nag_wp) array *Input*

On entry: the nonzero elements of the upper triangles of matrices Q_{ij} stored in coordinate storage format. The first NNZQ(1) elements belong to the first $Q_{i_1j_1}$, the following NNZQ(2) to $Q_{i_2j_2}$, etc.

Constraint: $1 \leq$ IROWQ$(i) \leq$ DIMQ, IROWQ$(i) \leq$ ICOLQ$(i) \leq$ DIMQ.

11: IDBLK – INTEGER *Input/Output*

On entry: if IDBLK $= 0$, a new matrix constraint is created; otherwise IDBLK $= k > 0$, the number of the existing linear matrix constraint to be expanded with the bilinear terms.

Constraint: IDBLK ≥ 0.

On exit: if IDBLK $= 0$ on entry, the number of the new matrix constraint is returned. By definition, it is the number of the matrix inequalities already defined plus one. Otherwise, IDBLK > 0 stays unchanged.

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 $= 3$

On entry, IDBLK $= \langle value \rangle$.
Bilinear terms of the matrix inequality block with the given IDBLK have already been defined.

IFAIL $= 4$

On entry, IDBLK $= \langle value \rangle$.
The given IDBLK does not match with any existing matrix inequality block.
The maximum IDBLK is currently $\langle value \rangle$.

On entry, IDBLK $= \langle value \rangle$.
The given IDBLK refers to a nonexistent matrix inequality block.
No matrix inequalities have been added yet.

IFAIL $= 5$

On entry, DIMQ $= \langle value \rangle$, IDBLK $= \langle value \rangle$.
The correct dimension of the given IDBLK is $\langle value \rangle$.
Constraint: DIMQ must match the dimension of the block supplied earlier.

IFAIL $= 6$

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

On entry, $i = \langle value \rangle$ and NNZQ$(i) = \langle value \rangle$.
Constraint: NNZQ$(i) > 0$.

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

On entry, NNZQSUM $= \langle value \rangle$ and sum(NNZQ) $= \langle value \rangle$.
Constraint: NNZQSUM $\geq$ sum(NNZQ).

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

IFAIL $= 8$

On entry, an error occurred in matrix Q_{ij} of index $k = \langle value \rangle$, QI$(k) = \langle value \rangle$, QJ$(k) = \langle value \rangle$.
For $j = \langle value \rangle$, ICOLQ$(j) = \langle value \rangle$ and DIMQ $= \langle value \rangle$.
Constraint: $1 \leq$ ICOLQ$(j) \leq$ DIMQ.

On entry, an error occurred in matrix Q_{ij} of index $k = \langle value \rangle$, QI$(k) = \langle value \rangle$, QJ$(k) = \langle value \rangle$.
For $j = \langle value \rangle$, IROWQ$(j) = \langle value \rangle$ and DIMQ $= \langle value \rangle$.
Constraint: $1 \leq$ IROWQ$(j) \leq$ DIMQ.

On entry, an error occurred in matrix Q_{ij} of index $k = \langle value \rangle$, $\mathrm{QI}(k) = \langle value \rangle$, $\mathrm{QJ}(k) = \langle value \rangle$.
For $j = \langle value \rangle$, $\mathrm{IROWQ}(j) = \langle value \rangle$ and $\mathrm{ICOLQ}(j) = \langle value \rangle$.
Constraint: $\mathrm{IROWQ}(j) \leq \mathrm{ICOLQ}(j)$ (elements within the upper triangle).

On entry, an error occurred in matrix Q_{ij} of index $k = \langle value \rangle$, $\mathrm{QI}(k) = \langle value \rangle$, $\mathrm{QJ}(k) = \langle value \rangle$.
More than one element of Q_{ij} has row index $\langle value \rangle$ and column index $\langle value \rangle$.
Constraint: each element of Q_{ij} must have a unique row and column index.

IFAIL = 9

On entry, index pair with QI = $\langle value \rangle$ and QJ = $\langle value \rangle$ repeats.
Constraint: each index pair QI, QJ must be unique.

On entry, $k = \langle value \rangle$, $\mathrm{QI}(k) = \langle value \rangle$ and $n = \langle value \rangle$.
Constraint: $1 \leq \mathrm{QI}(k) \leq n$.

On entry, $k = \langle value \rangle$, $\mathrm{QJ}(k) = \langle value \rangle$ and $n = \langle value \rangle$.
Constraint: $1 \leq \mathrm{QJ}(k) \leq n$.

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

E04RPF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example demonstrates how semidefinite programming can be used in control theory. See also Section 10 in E04RAF for links to further examples in the suite.

The problem, from static output feedback (SOF) control Syrmos *et al.* (1997), solved here is the linear time-invariant (LTI) 'test' system

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{4}$$

subject to static output feedback

$$u = Ky. \tag{5}$$

Here $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ and $C \in \mathbb{R}^{p \times n}$ are given matrices, $x \in \mathbb{R}^n$ is the vector of state variables, $u \in \mathbb{R}^m$ is the vector of control inputs, $y \in \mathbb{R}^p$ is the vector of system outputs, and $K \in \mathbb{R}^{m \times p}$ is the unknown feedback gain matrix.

The problem is to find K such that (4) is time-stable when subject to (5), i.e., all eigenvalues of the closed-loop system matrix $A + BKC$ belong to the left half-plane. From the Lyapunov stability theory, this holds if and only if there exists a symmetric positive definite matrix P such that

$$(A + BKC)^{\mathrm{T}} P + P(A + BKC) \prec 0.$$

Hence, by introducing the new variable, the Lyapunov matrix P, we can formulate the SOF problem as a feasibility BMI-SDP problem in variables K and P. As we cannot formulate the problem with sharp matrix inequalities, we can solve the following system instead (note that the objective function is added to bound matrix P):

$$\begin{array}{ll} \underset{K,P}{\text{minimize}} & \text{trace}(P) \\ \text{subject to} & (A + BKC)^{\mathrm{T}} P + P(A + BKC) \preceq -I \\ & P \succeq I. \end{array} \qquad (6)$$

For $n = p = 2$, $m = 1$,

$$A = \begin{pmatrix} -1 & 2 \\ -3 & -4 \end{pmatrix}, \quad B = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \quad C = I$$

and the unknown matrices expressed as

$$P = \begin{pmatrix} x_1 & x_2 \\ x_2 & x_3 \end{pmatrix}, \quad K = \begin{pmatrix} x_4 & x_5 \end{pmatrix},$$

the problem (6) can be rewritten in the form (3) as follows:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^5}{\text{minimize}} & x_1 + x_3 \\ \text{subject to} & \begin{pmatrix} 2x_1x_4 + 2x_2x_4 & x_1x_5 + x_2x_4 + x_2x_5 + x_3x_4 \\ \text{sym.} & 2x_2x_5 + 2x_3x_5 \end{pmatrix} + \\ & \begin{pmatrix} 2x_1 + 6x_2 & -2x_1 + 5x_2 + 3x_3 \\ \text{sym.} & -4x_2 + 8x_3 \end{pmatrix} - I \succeq 0 \\ & \begin{pmatrix} x_1 & x_2 \\ \text{sym.} & x_3 \end{pmatrix} - I \succeq 0. \end{array}$$

This formulation has been stored in a generic BMI-SDP data file which is processed and solved by the example program.

10.1 Program Text

```
    Program e04rpfe

!     E04RPF Example Program Text

!     Read a 'generic' LMI/BMI SDP problem from the input file,
!     formulate the problem via a handle and pass it to the solver.

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: e04raf, e04ref, e04rff, e04rjf, e04rnf, e04rpf,   &
                             e04ryf, e04rzf, e04svf, nag_wp
      Use, Intrinsic                   :: iso_c_binding, Only: c_ptr
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Type (c_ptr)                     :: handle
```

```
      Integer                            :: blkidx, dimaq, idblk, idlc, idx,     &
                                            idxend, ifail, inform, midx, nblk,   &
                                            nclin, nnzasum, nnzb, nnzc, nnzh,    &
                                            nnzqsum, nnzu, nnzua, nnzuc, nq,     &
                                            nvar
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable    :: a(:), b(:), bl(:), bu(:), cvec(:),   &
                                            h(:), q(:), x(:)
      Real (Kind=nag_wp)                 :: rdummy(1), rinfo(32), stats(32)
      Integer, Allocatable               :: icola(:), icolb(:), icolh(:),        &
                                            icolq(:), idxc(:), irowa(:),         &
                                            irowb(:), irowh(:), irowq(:),        &
                                            nnza(:), nnzq(:), qi(:), qj(:)
      Integer                            :: idummy(1)
!     .. Executable Statements ..
      Continue

      Write (nout,*) 'E04RPF Example Program Results'
      Write (nout,*)
      Flush (nout)

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

!     Read the problem size.
      Read (nin,*) nvar
      Read (nin,*) nnzh
      Read (nin,*) nclin, nnzb
      Read (nin,*) nblk

!     Initialize handle to an empty problem.
      ifail = 0
      Call e04raf(handle,nvar,ifail)

!     Read the linear part of the objective function.
      Allocate (cvec(nvar))
      Read (nin,*) cvec(1:nvar)

      If (nnzh==0) Then
!       Add the linear objective function to the problem formulation.
        ifail = 0
        Call e04ref(handle,nvar,cvec,ifail)
        Deallocate (cvec)

      Else
!       The linear part of the objective was read in as dense, E04RFF needs
!       the sparse format.
        nnzc = nvar
        Allocate (idxc(nnzc))
        Do idx = 1, nvar
          idxc(idx) = idx
        End Do

!       Read nonzeros for H (quadratic part of the objective) if present.
        Allocate (irowh(nnzh),icolh(nnzh),h(nnzh))
        Do idx = 1, nnzh
          Read (nin,*) h(idx), irowh(idx), icolh(idx)
        End Do

!       Add the quadratic objective function to the problem formulation.
        ifail = 0
        Call e04rff(handle,nnzc,idxc,cvec,nnzh,irowh,icolh,h,ifail)
        Deallocate (idxc,cvec,irowh,icolh,h)
      End If

!     Read a block of linear constraints and its bounds if present.
      If (nclin>0 .And. nnzb>0) Then
        Allocate (irowb(nnzb),icolb(nnzb),b(nnzb),bl(nclin),bu(nclin))
        Do idx = 1, nnzb
          Read (nin,*) b(idx), irowb(idx), icolb(idx)
        End Do
```

```
          Read (nin,*) bl(1:nclin)
          Read (nin,*) bu(1:nclin)

!         Add the block of linear constraints.
          idlc = 0
          ifail = 0
          Call e04rjf(handle,nclin,bl,bu,nnzb,irowb,icolb,b,idlc,ifail)
          Deallocate (irowb,icolb,b,bl,bu)
        End If

!       Read all matrix inequalities.
        Do blkidx = 1, nblk
          Read (nin,*) dimaq
          Read (nin,*) nnzasum, nnzqsum
          idblk = 0

          If (nnzasum>0) Then
!           Read a linear matrix inequality composed of (NVAR+1) matrices.
            Allocate (nnza(nvar+1),irowa(nnzasum),icola(nnzasum),a(nnzasum))
            idx = 1
            Do midx = 1, nvar + 1
!             Read matrix A_{midx-1}.
              Read (nin,*) nnza(midx)
              idxend = idx + nnza(midx) - 1
              Do idx = idx, idxend
                Read (nin,*) a(idx), irowa(idx), icola(idx)
              End Do
            End Do

!           Add the linear matrix inequality to the problem formulation.
            idblk = 0
            ifail = 0
            Call e04rnf(handle,nvar,dimaq,nnza,nnzasum,irowa,icola,a,1,idummy,   &
              idblk,ifail)
            Deallocate (nnza,irowa,icola,a)
          End If

          If (nnzqsum>0) Then
!           Read bilinear part of the matrix inequality composed of NQ matrices.
            Read (nin,*) nq
            Allocate (qi(nq),qj(nq),nnzq(nq),irowq(nnzqsum),icolq(nnzqsum),      &
              q(nnzqsum))
            idx = 1
            Do midx = 1, nq
!             Read matrix Q_ij where i=QI(midx), j=QJ(midx).
              Read (nin,*) qi(midx), qj(midx)
              Read (nin,*) nnzq(midx)
              idxend = idx + nnzq(midx) - 1
              Do idx = idx, idxend
                Read (nin,*) q(idx), irowq(idx), icolq(idx)
              End Do
            End Do

!           Expand the existing linear matrix inequality with the bilinear terms
!           or (if linear part was not present) create a new matrix inequality.
            ifail = 0
            Call e04rpf(handle,nq,qi,qj,dimaq,nnzq,nnzqsum,irowq,icolq,q,idblk,  &
              ifail)
            Deallocate (qi,qj,nnzq,irowq,icolq,q)
          End If

        End Do

!       Problem was successfully decoded.
        Write (nout,*) 'SDP problem was read, passing it to the solver.'
        Write (nout,*)
        Flush (nout)

!       Print overview of the handle.
        ifail = 0
        Call e04ryf(handle,nout,'Overview,Matrix Constraints',ifail)
```

```
!     Allocate memory for the solver.
      Allocate (x(nvar))

!     Call the solver, ignore Lagrangian multipliers.
      nnzu = 0
      nnzuc = 0
      nnzua = 0
      inform = 0
      x(:) = 0.0_nag_wp

      ifail = 0
      Call e04svf(handle,nvar,x,nnzu,rdummy,nnzuc,rdummy,nnzua,rdummy,rinfo,   &
        stats,inform,ifail)

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

    End Program e04rpfe
```

10.2 Program Data

```
E04RPF Example Program Data
      5                 : no of variables
      0                 : no of nonzeros in H matrix
      0         0       : no of linear constraints and nnz in B
      2                 : no of matrix constraints

    1.000000    0.000000    1.000000    0.000000
    0.000000                                          : Linear obj. vector

    2                 : beginning of matrix constr 1, its dimension
    9    8            : no of nonzeroes in all A_i, Q_ij

       2                              : number of nonzeros in A_0
       1.000000       1       1       : Upper triangle of A_0
       1.000000       2       2       : End of matrix A_0

       2                              : number of nonzeros in A_1
       2.000000       1       1       : Upper triangle of A_1
      -2.000000       1       2       : End of matrix A_1

       3                              : number of nonzeros in A_2
       6.000000       1       1       : Upper triangle of A_2
       5.000000       1       2       : in coordinate storage
      -4.000000       2       2       : End of matrix A_2

       2                              : number of nonzeros in A_3
       3.000000       1       2       : Upper triangle of A_3
       8.000000       2       2       : End of matrix A_3

       0                              : number of nonzeros in A_4
       0                              : number of nonzeros in A_5
    6                 : number of Q_ij matrices

       1       4                      : indices giving i & j for Q_ij
       1                              : number of nonzeros in Q_{1,4}
       2.000000       1       1       : End of matrix Q_{1,4}

       2       4                      : indices giving i & j for Q_ij
       2                              : number of nonzeros in Q_{2,4}
       2.000000       1       1       : Upper triangle of Q_{2,4}
       1.000000       1       2       : End of matrix Q_{2,4}

       3       4                      : indices giving i & j for Q_ij
       1                              : number of nonzeros in Q_{3,4}
       1.000000       1       2       : End of matrix Q_{3,4}

       1       5                      : indices giving i & j for Q_ij
```

```
      1                             : number of nonzeros in Q_{1,5}
      1.000000       1       2      : End of matrix Q_{1,5}

      2       5                     : indices giving i & j for Q_ij
      2                             : number of nonzeros in Q_{2,5}
      1.000000       1       2      : Upper triangle of Q_{2,5}
      2.000000       2       2      : End of matrix Q_{2,5}

      3       5                     : indices giving i & j for Q_ij
      1                             : number of nonzeros in Q_{3,5}
      2.000000       2       2      : End of matrix Q_{3,5}

   2              : beginning of matrix constr 2, its dimension
   5    0         : no of nonzeroes in all A_i, Q_ij

      2                             : number of nonzeros in A_0
      1.000000       1       1      : Upper triangle of A_0
      1.000000       2       2      : End of matrix A_0

      1                             : number of nonzeros in A_1
      1.000000       1       1      : End of matrix A_1

      1                             : number of nonzeros in A_2
      1.000000       1       2      : End of matrix A_2

      1                             : number of nonzeros in A_3
      1.000000       2       2      : End of matrix A_3

      0                             : number of nonzeros in A_4
      0                             : number of nonzeros in A_5
```

10.3 Program Results

```
E04RPF Example Program Results

SDP problem was read, passing it to the solver.

Overview
  Status:                    Problem and option settings are editable.
  No of variables:           5
  Objective function:        linear
  Simple bounds:             not defined yet
  Linear constraints:        not defined yet
  Nonlinear constraints:     not defined yet
  Matrix constraints:        2
Matrix constraints
  IDblk =     1, size =      2 x     2, polynomial of order 2
  IDblk =     2, size =      2 x     2, linear
E04SV, NLP-SDP Solver (Pennon)
------------------------------
Number of variables             5                 [eliminated            0]
                           simple  linear  nonlin
(Standard) inequalities         0       0       0
(Standard) equalities                   0       0
Matrix inequalities                     1       1 [dense    2, sparse    0]
                                                  [max dimension         2]

Begin of Options
    Outer Iteration Limit         =                 100     * d
    Inner Iteration Limit         =                 100     * d
    Infinite Bound Size           =         1.00000E+20     * d
    Initial X                     =                User     * d
    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          =                   No     * S
    Dimacs Measures                =                   No     * S
    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                =            Goldstein     * S
    Inner Stop Tolerance           =          1.00000E-02     * d
    Inner Stop Criteria            =            Heuristic     * d
    Task                           =             Minimize     * d
    P Update Speed                 =                   12     * d
End of Options
--------------------------------------------------------------
 it|  objective |  optim  |   feas  |  compl  | pen min |inner
--------------------------------------------------------------
  0  0.00000E+00  1.82E+01  1.00E+00  4.00E+00  2.00E+00   0
  1  4.11823E+00  3.85E-03  0.00E+00  1.73E+00  2.00E+00   6
  2  2.58252E+00  5.36E-03  0.00E+00  4.93E-01  9.04E-01   4
  3  2.06132E+00  1.02E-03  0.00E+00  7.70E-02  4.08E-01   4
  4  2.00050E+00  3.00E-03  8.91E-03  1.78E-02  1.85E-01   3
  5  1.99929E+00  1.55E-03  3.16E-03  3.65E-03  8.34E-02   2
  6  1.99985E+00  1.03E-04  3.16E-04  7.19E-04  3.77E-02   4
  7  1.99997E+00  7.04E-04  5.76E-05  1.41E-04  1.70E-02   1
  8  2.00000E+00  1.32E-04  6.52E-06  2.76E-05  7.70E-03   1
  9  2.00000E+00  8.49E-06  7.86E-07  5.37E-06  3.48E-03   1
 10  2.00000E+00  5.88E-07  1.06E-07  1.04E-06  1.57E-03   1
 11  2.00000E+00  5.55E-08  4.87E-08  2.02E-07  7.11E-04   1
 12  2.00000E+00  5.34E-09  5.37E-09  3.93E-08  3.21E-04   1
 13  2.00000E+00  5.03E-10  5.45E-09  7.62E-09  1.45E-04   1
 14  2.00000E+00  4.45E-11  5.55E-09  1.48E-09  6.56E-05   1
--------------------------------------------------------------
 it|  objective |  optim  |   feas  |  compl  | pen min |inner
--------------------------------------------------------------
 15  2.00000E+00  4.36E-12  5.67E-09  2.87E-10  2.96E-05   1
 16  2.00000E+00  1.61E-11  5.82E-09  5.57E-11  1.34E-05   1
 17  2.00000E+00  3.13E-11  6.00E-09  1.08E-11  6.06E-06   1
 18  2.00000E+00  8.65E-11  6.22E-09  2.10E-12  2.74E-06   1
 19  2.00000E+00  1.31E-10  6.48E-09  4.07E-13  1.24E-06   1
--------------------------------------------------------------
Status: converged, an optimal solution found
--------------------------------------------------------------
Final objective value                2.000000E+00
Relative precision                   8.141636E-16
Optimality                           1.310533E-10
Feasibility                          6.484489E-09
Complementarity                      4.066867E-13
Iteration counts
  Outer iterations                             19
  Inner iterations                             36
  Linesearch steps                             56
Evaluation counts
  Augm. Lagr. values                           76
  Augm. Lagr. gradient                         56
  Augm. Lagr. hessian                          36
--------------------------------------------------------------
```