NAG Library Routine Document

E04RJF

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

E04RJF is a part of the NAG optimization modelling suite and defines the block of linear constraints of the problem.

2 Specification

```
SUBROUTINE E04RJF (HANDLE, NCLIN, BL, BU, NNZB, IROWB, ICOLB, B, IDLC,     &
                   IFAIL)

INTEGER             NCLIN, NNZB, IROWB(NNZB), ICOLB(NNZB), IDLC, IFAIL
REAL (KIND=nag_wp) BL(NCLIN), BU(NCLIN), B(NNZB)
TYPE (C_PTR)        HANDLE
```

3 Description

After the initialization routine E04RAF has been called, E04RJF may be used to define the linear constraints $l_B \leq Bx \leq u_B$ of the problem unless the linear constraints have already been defined. This will typically be used for problems, such as quadratic programming (QP)

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

nonlinear programming (NLP)

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) & \text{(a)} \\ \text{subject to} & l_g \leq g(x) \leq u_g & \text{(b)} \\ & l_B \leq Bx \leq u_B & \text{(c)} \\ & l_x \leq x \leq u_x & \text{(d)} \end{array} \tag{2}$$

linear semidefinite programming (SDP)

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & c^{\mathrm{T}}x & \text{(a)} \\ \text{subject to} & \displaystyle\sum_{i=1}^{n} x_i A_i^k - A_0^k \succeq 0, \quad k = 1,\ldots,m_A & \text{(b)} \\ & l_B \leq Bx \leq u_B & \text{(c)} \\ & l_x \leq x \leq u_x & \text{(d)} \end{array} \tag{3}$$

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

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}x^{\mathrm{T}}Hx + c^{\mathrm{T}}x & \text{(a)} \\ \text{subject to} & \displaystyle\sum_{i,j=1}^{n} x_i x_j Q_{ij}^k + \sum_{i=1}^{n} x_i A_i^k - A_0^k \succeq 0, \quad k = 1,\ldots,m_A & \text{(b)} \\ & l_B \leq Bx \leq u_B & \text{(c)} \\ & l_x \leq x \leq u_x & \text{(d)} \end{array} \tag{4}$$

where n is the number of decision variables, B is a general $m_B \times n$ rectangular matrix and l_B and u_B are m_B-dimensional vectors. Note that upper and lower bounds are specified for all the constraints. This form allows full generality in specifying various types of constraint. In particular, the jth constraint may be defined as an equality by setting $l_j = u_j$. If certain bounds are not present, the associated elements of

l_B or u_B may be set to special values that are treated as $-\infty$ or $+\infty$. See the description of the optional parameter **Infinite Bound Size** of the solvers in the suite, E04STF and E04SVF. Its value is denoted as $bigbnd$ further in this text. Note that the bounds are interpreted based on its value at the time of calling this routine and any later alterations to **Infinite Bound Size** will not affect these constraints.

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: NCLIN – INTEGER *Input*

On entry: m_B, the number of linear constraints (number of rows of the matrix B).

If NCLIN $= 0$, no linear constraints will be defined and BL, BU, NNZB, IROWB, ICOLB and B will not be referenced.

Constraint: NCLIN ≥ 0.

3: BL(NCLIN) – REAL (KIND=nag_wp) array *Input*

4: BU(NCLIN) – REAL (KIND=nag_wp) array *Input*

On entry: BL and BU define lower and upper bounds of the linear constraints, l_B and u_B, respectively. To define the jth constraint as equality, set BL(j) = BU$(j) = \beta$, where $|\beta| < bigbnd$. To specify a nonexistent lower bound (i.e., $l_j = -\infty$), set BL$(j) \leq -bigbnd$; to specify a nonexistent upper bound, set BU$(j) \geq bigbnd$.

Constraints:

BL$(j) \leq$ BU(j), for $j = 1, 2, \ldots,$ NCLIN;
BL$(j) < bigbnd$, for $j = 1, 2, \ldots,$ NCLIN;
BU$(j) > -bigbnd$, for $j = 1, 2, \ldots,$ NCLIN;
if BL$(j) =$ BU(j), $|$BL$(j)| < bigbnd$, for $j = 1, 2, \ldots,$ NCLIN.

5: NNZB – INTEGER *Input*

On entry: NNZB gives the number of nonzeros in matrix B.

Constraint: if NCLIN > 0, NNZB > 0.

6: IROWB(NNZB) – INTEGER array *Input*

7: ICOLB(NNZB) – INTEGER array *Input*

8: B(NNZB) – REAL (KIND=nag_wp) array *Input*

On entry: arrays IROWB, ICOLB and B store NNZB nonzeros of the sparse matrix B in coordinate storage (CS) format (see Section 2.1.1 in the F11 Chapter Introduction). The matrix B has dimensions $m_B \times n$, where n, the number of variables in the problem, was set in NVAR during the initialization of the handle by E04RAF. IROWB specifies one-based row indices, ICOLB specifies one-based column indices and B specifies the values of the nonzero elements in such a way that $b_{ij} =$ B(l) where $i =$ IROWB(l) and $j =$ ICOLB(l), for $l = 1, 2, \ldots,$ NNZB. No particular order of elements is expected, but elements should not repeat.

Constraint: $1 \leq$ IROWB$(l) \leq$ NCLIN, $1 \leq$ ICOLB$(l) \leq n$, for $l = 1, 2, \ldots,$ NNZB.

9: IDLC – INTEGER *Input/Output*

Note: IDLC is reserved for future releases of the NAG Library.

On entry: if IDLC $= 0$, new linear constraints are added to the problem definition. This is the only value allowed at the moment.

Constraint: IDLC $= 0$.

On exit: the number of the last linear constraint added, thus NCLIN.

10: 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 $= 3$

A set of linear constraints has already been defined.

IFAIL $= 4$

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

IFAIL $= 6$

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

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

IFAIL $= 8$

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

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

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

IFAIL $= 10$

On entry, $j = \langle value \rangle$, BL$(j) = \langle value \rangle$, $bigbnd = \langle value \rangle$.
Constraint: BL$(j) <$ *bigbnd*.

On entry, $j = \langle value \rangle$, BL$(j) = \langle value \rangle$ and BU$(j) = \langle value \rangle$.
Constraint: BL$(j) \leq$ BU(j).

On entry, $j = \langle value \rangle$, BU$(j) = \langle value \rangle$, $bigbnd = \langle value \rangle$.
Constraint: BU$(j) > -$*bigbnd*.

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

E04RJF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example demonstrates how to use the MPS file reader E04MXF and this suite of routines to define and solve a QP problem. E04MXF uses a different output format to the one required by E04RJF, in particular, it uses the compressed column storage (CCS) (see Section 2.1.3 in the F11 Chapter Introduction) instead of the coordinate storage and the linear objective vector is included in the system matrix. Therefore a simple transformation is needed before calling E04RJF as demonstrated in the example program.

The data file stores the following problem:

$$\text{minimize} \quad c^{\mathrm{T}}x + \tfrac{1}{2}x^{\mathrm{T}}Hx \quad \text{subject to} \quad \begin{array}{lll} l_B & \leq Bx & \leq u_B, \\ -2 & \leq x & \leq 2, \end{array}$$

where

$$c = \begin{pmatrix} -4.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -0.1 \\ -0.3 \end{pmatrix}, \quad H = \begin{pmatrix} 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$B = \begin{pmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 4.0 \\ 1.0 & 2.0 & 3.0 & 4.0 & -2.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & -1.0 & 1.0 & -1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{pmatrix},$$

$$l_B = \begin{pmatrix} -2.0 \\ -2.0 \\ -2.0 \end{pmatrix} \quad \text{and} \quad u_B = \begin{pmatrix} 1.5 \\ 1.5 \\ 4.0 \end{pmatrix}.$$

The optimal solution (to five figures) is

$$x^* = (2.0, -0.23333, -0.26667, -0.3, -0.1, 2.0, 2.0, -1.7777, -0.45555)^{\mathrm{T}}.$$

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

10.1 Program Text

```
    Program e04rjfe

!     E04RJF Example Program Text

!     Read in LP/QP problem stored in a MPS file, formulated it
!     as a handle and pass it to the solver.

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: e04mxf, e04raf, e04rff, e04rhf, e04rjf, e04rzf,   &
                             e04svf, e04zmf, nag_wp, x04acf, x04adf
      Use, Intrinsic                   :: iso_c_binding, Only: c_null_ptr,     &
                                          c_ptr
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: mpslst = 1, nin = 7, nout = 6
      Character (*), Parameter         :: fname_default = 'e04rjfe.opt'
!     .. Local Scalars ..
      Type (c_ptr)                     :: handle
      Integer                          :: idlc, idx, idx_c, idx_dest, ifail,   &
                                          inform, iobj, j, lintvar, m,         &
                                          maxlintvar, maxm, maxn, maxncolh,    &
                                          maxnnz, maxnnzh, minmax, mode, n,    &
                                          nargs, ncolh, nname, nnz, nnzc,      &
                                          nnzh, nnzu, nnzua, nnzuc
      Character (256)                  :: fname
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), bl(:), bu(:), c(:), h(:),      &
                                          u(:), ua(:), uc(:), x(:)
      Real (Kind=nag_wp)               :: rinfo(32), stats(32)
      Integer, Allocatable             :: iccola(:), iccolh(:), icola(:),      &
                                          icolh(:), idxc(:), intvar(:),        &
                                          irowa(:), irowh(:)
      Character (8), Allocatable       :: crname(:)
      Character (8)                    :: pnames(5)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: command_argument_count, count,       &
                                          get_command_argument, trim
```

```
!     .. Executable Statements ..
      Continue

      Write (nout,*) 'E04RJF Example Program Results'
      Write (nout,*)

!     Use the first command line argument as the filename or
!     choose default hard-coded filename in 'fname_default'.
      nargs = command_argument_count()
      If (nargs>=1) Then
        Call get_command_argument(1,fname)
      Else
        fname = fname_default
      End If

      Write (nout,*) 'Reading MPS file:  ', trim(fname)
      Flush (nout)

!     Read the input MPS file.
      pnames(1:5) = '        '
      maxm = 0
      maxn = 0
      maxnnz = 0
      maxnnzh = 0
      maxncolh = 0
      maxlintvar = -1

!     Open the data file for reading.
      mode = 0
      ifail = 0
      Call x04acf(nin,fname,mode,ifail)

!     Call E04MXF in query mode to obtain an approximate problem size.
      Allocate (a(maxnnz),irowa(maxnnz),iccola(maxn+1),bl(maxn+maxm),          &
        bu(maxn+maxm),crname(maxn+maxm),h(maxnnzh),irowh(maxnnzh),             &
        iccolh(maxncolh+1),intvar(maxlintvar))
      ifail = 0
      Call e04mxf(nin,maxn,maxm,maxnnz,maxncolh,maxnnzh,maxlintvar,mpslst,n,m, &
        nnz,ncolh,nnzh,lintvar,iobj,a,irowa,iccola,bl,bu,pnames,nname,crname,  &
        h,irowh,iccolh,minmax,intvar,ifail)
      Deallocate (a,irowa,iccola,bl,bu,crname,h,irowh,iccolh)

!     Close the data file.
      ifail = 0
      Call x04adf(nin,ifail)

!     Set maximal problem size.
      maxm = m
      maxn = n
      maxnnz = nnz
      maxnnzh = nnzh
      maxncolh = ncolh

      Allocate (irowa(maxnnz),iccola(maxn+1),a(maxnnz),bl(maxn+maxm),          &
        bu(maxn+maxm),crname(maxn+maxm),irowh(maxnnzh),iccolh(maxncolh+1),     &
        h(maxnnzh),x(maxn),icolh(maxnnzh),icola(maxnnz))

!     Open the data file for reading.
      mode = 0
      ifail = 0
      Call x04acf(nin,fname,mode,ifail)

!     Call E04MXF to read the problem.
      ifail = 0
      Call e04mxf(nin,maxn,maxm,maxnnz,maxncolh,maxnnzh,maxlintvar,mpslst,n,m, &
        nnz,ncolh,nnzh,lintvar,iobj,a,irowa,iccola,bl,bu,pnames,nname,crname,  &
        h,irowh,iccolh,minmax,intvar,ifail)

      Write (nout,*) 'MPS/QPS file read'
      Flush (nout)
```

```
!     Close the data file.
      ifail = 0
      Call x04adf(nin,ifail)

!     Data has been read. Set up the problem to the solver.

!     Initialize handle.
      handle = c_null_ptr
      ifail = 0
      Call e04raf(handle,n,ifail)

!     Move linear objective from A to C.
      If (iobj>0) Then
!       Shift bounds.
        Do j = iobj, m - 1
          bl(n+j) = bl(n+j+1)
          bu(n+j) = bu(n+j+1)
        End Do
        m = m - 1
!       Extract row IOBJ.
!       Count how many nonzeros will be needed in C.
        nnzc = count(irowa(1:nnz)==iobj)
        Allocate (idxc(nnzc),c(nnzc))
        idx = 1
        idx_c = 1
        idx_dest = 1
        Do j = 1, n
          Do idx = idx, iccola(j+1) - 1
            If (irowa(idx)<iobj) Then
              a(idx_dest) = a(idx)
              irowa(idx_dest) = irowa(idx)
              idx_dest = idx_dest + 1
            Else If (irowa(idx)==iobj) Then
              idxc(idx_c) = j
              c(idx_c) = a(idx)
              idx_c = idx_c + 1
            Else
              a(idx_dest) = a(idx)
              irowa(idx_dest) = irowa(idx) - 1
              idx_dest = idx_dest + 1
            End If
          End Do
          iccola(j+1) = idx_dest
        End Do
        nnz = idx_dest - 1
      Else
!       There is no linear part of the objective function.
        nnzc = 0
        Allocate (idxc(nnzc),c(nnzc))
      End If
!     Convert (decompress) ICCOLA() to ICOLA().
      Do j = 1, n
        icola(iccola(j):iccola(j+1)-1) = j
      End Do

!     Add objective function to the problem formulation.
      If (nnzh==0) Then
!       The objective is a (sparse) linear function.
        ifail = 0
        Call e04rff(handle,nnzc,idxc,c,nnzh,irowh,icolh,h,ifail)
      Else
!       The objective is a quadratic function.
!       Transform (decompress) ICCOLH() -> ICOLH().
        Do j = 1, ncolh
          icolh(iccolh(j):iccolh(j+1)-1) = j
        End Do
!       E04MX returned L triangle, E04RFF needs U triangle -> swap.
        ifail = 0
        Call e04rff(handle,nnzc,idxc,c,nnzh,icolh,irowh,h,ifail)
      End If
```

```
!     Add box constraints to the formulation.
      ifail = 0
      Call e04rhf(handle,n,bl,bu,ifail)

!     Add linear constraints.
      idlc = 0
      ifail = 0
      Call e04rjf(handle,m,bl(n+1:n+m),bu(n+1:n+m),nnz,irowa,icola,a,idlc,     &
        ifail)

      Write (nout,*) 'The problem was set-up'
      Flush (nout)

!     Call the solver.

!     Set optional arguments.
      ifail = 0
      Call e04zmf(handle,'Print Options = No',ifail)

!     Set up a starting point and call the solver.
!     Let's ignore Lagrangian multipliers U/UA.
      x(:) = 0.0_nag_wp
      nnzu = 0
      nnzuc = 0
      nnzua = 0
      Allocate (u(nnzu),uc(nnzuc),ua(nnzua))

      ifail = 0
      Call e04svf(handle,n,x,nnzu,u,nnzuc,uc,nnzua,ua,rinfo,stats,inform,      &
        ifail)

      Write (nout,*)
      Write (nout,*) 'Optimal solution:'
      Write (nout,99999) x(1:n)
99999 Format (1X,'X = ',3F9.2)
      Flush (nout)

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

    End Program e04rjfe
```

10.2 Program Data

```
NAME          E04RJ.EX
ROWS
 L  ..ROW1..
 L  ..ROW2..
 L  ..ROW3..
 N  ..COST..
COLUMNS
    ...X1...  ..ROW1..      1.0        ..ROW2..         1.0
    ...X1...  ..ROW3..      1.0        ..COST..        -4.0
    ...X2...  ..ROW1..      1.0        ..ROW2..         2.0
    ...X2...  ..ROW3..     -1.0        ..COST..        -1.0
    ...X3...  ..ROW1..      1.0        ..ROW2..         3.0
    ...X3...  ..ROW3..      1.0        ..COST..        -1.0
    ...X4...  ..ROW1..      1.0        ..ROW2..         4.0
    ...X4...  ..ROW3..     -1.0        ..COST..        -1.0
    ...X5...  ..ROW1..      1.0        ..ROW2..        -2.0
    ...X5...  ..ROW3..      1.0        ..COST..        -1.0
    ...X6...  ..ROW1..      1.0        ..ROW2..         1.0
    ...X6...  ..ROW3..      1.0        ..COST..        -1.0
    ...X7...  ..ROW1..      1.0        ..ROW2..         1.0
    ...X7...  ..ROW3..      1.0        ..COST..        -1.0
    ...X8...  ..ROW1..      1.0        ..ROW2..         1.0
    ...X8...  ..ROW3..      1.0        ..COST..        -0.1
    ...X9...  ..ROW1..      4.0        ..ROW2..         1.0
    ...X9...  ..ROW3..      1.0        ..COST..        -0.3
```

```
RHS
    RHS1      ..ROW1..      1.5
    RHS1      ..ROW2..      1.5
    RHS1      ..ROW3..      4.0
    RHS1      ..COST..      1000.0
RANGES
    RANGE1    ..ROW1..      3.5
    RANGE1    ..ROW2..      3.5
    RANGE1    ..ROW3..      6.0
BOUNDS
 LO BOUND     ...X1...     -2.0
 LO BOUND     ...X2...     -2.0
 LO BOUND     ...X3...     -2.0
 LO BOUND     ...X4...     -2.0
 LO BOUND     ...X5...     -2.0
 LO BOUND     ...X6...     -2.0
 LO BOUND     ...X7...     -2.0
 LO BOUND     ...X8...     -2.0
 LO BOUND     ...X9...     -2.0
 UP BOUND     ...X1...      2.0
 UP BOUND     ...X2...      2.0
 UP BOUND     ...X3...      2.0
 UP BOUND     ...X4...      2.0
 UP BOUND     ...X5...      2.0
 UP BOUND     ...X6...      2.0
 UP BOUND     ...X7...      2.0
 UP BOUND     ...X8...      2.0
 UP BOUND     ...X9...      2.0
QUADOBJ
    ...X1...  ...X1...  2.00000000E0   ...X2...  1.00000000E0
    ...X1...  ...X3...  1.00000000E0   ...X4...  1.00000000E0
    ...X1...  ...X5...  1.00000000E0
    ...X2...  ...X2...  2.00000000E0   ...X3...  1.00000000E0
    ...X2...  ...X4...  1.00000000E0   ...X5...  1.00000000E0
    ...X3...  ...X3...  2.00000000E0   ...X4...  1.00000000E0
    ...X3...  ...X5...  1.00000000E0
    ...X4...  ...X4...  2.00000000E0   ...X5...  1.00000000E0
    ...X5...  ...X5...  2.00000000E0
ENDATA
```

10.3 Program Results

```
 E04RJF Example Program Results

 Reading MPS file:  e04rjfe.opt

 MPSX INPUT LISTING
 ------------------
 Searching for indicator line
 Line          1: Found NAME indicator line
                  Query mode - Ignoring NAME data.
 Line          2: Found ROWS indicator line
                  Query mode - Counting ROWS data.
 Line          7: Found COLUMNS indicator line
                  Query mode - Counting COLUMNS data.
 Line         26: Found RHS indicator line
                  Query mode - Ignoring RHS data.
 Line         31: Found RANGES indicator line
                  Query mode - Ignoring RANGES data.
 Line         35: Found BOUNDS indicator line
                  Query mode - Ignoring BOUNDS data.
 Line         54: Found QUADOBJ indicator line
                  Query mode - Counting QUADOBJ data.
                  Query mode - End of QUADOBJ data. Exit

 MPSX INPUT LISTING
 ------------------
 Searching for indicator line
 Line          1: Found NAME indicator line
 Line          2: Found ROWS indicator line
```

```
Line          7: Found COLUMNS indicator line
Line         26: Found RHS indicator line
Line         31: Found RANGES indicator line
Line         35: Found BOUNDS indicator line
Line         54: Found QUADOBJ indicator line
Line         64: Found ENDATA indicator line
MPS/QPS file read
The problem was set-up
E04SV, NLP-SDP Solver (Pennon)
------------------------------
Number of variables             9                 [eliminated            0]
                          simple  linear  nonlin
(Standard) inequalities       18       6       0
(Standard) equalities                  0       0
Matrix inequalities                    0       0 [dense    0, sparse    0]
                                                 [max dimension         0]

------------------------------------------------------------
 it|  objective |  optim  |   feas  |  compl  | pen min |inner
------------------------------------------------------------
  0  0.00000E+00  4.70E+00  0.00E+00  1.00E+01  1.00E+00   0
  1 -6.76762E+00  9.46E-04  0.00E+00  6.25E-01  1.00E+00   4
  2 -7.82467E+00  1.79E-03  3.69E-02  1.45E-01  4.65E-01   4
  3 -8.02059E+00  7.27E-03  2.27E-02  3.38E-02  2.16E-01   4
  4 -8.06187E+00  3.18E-03  5.75E-03  7.86E-03  1.01E-01   4
  5 -8.06653E+00  1.30E-03  1.13E-03  1.83E-03  4.68E-02   5
  6 -8.06739E+00  6.98E-03  1.42E-04  4.25E-04  2.18E-02   3
  7 -8.06775E+00  2.16E-04  2.80E-05  9.89E-05  1.01E-02   2
  8 -8.06778E+00  4.44E-05  6.94E-05  2.30E-05  4.71E-03   1
  9 -8.06778E+00  1.88E-06  1.15E-05  5.35E-06  2.19E-03   1
 10 -8.06778E+00  4.38E-08  1.52E-06  1.24E-06  1.02E-03   1
 11 -8.06778E+00  6.52E-10  1.74E-07  2.90E-07  4.74E-04   1
 12 -8.06778E+00  8.12E-12  1.90E-08  6.73E-08  2.21E-04   1
------------------------------------------------------------
Status: converged, an optimal solution found
------------------------------------------------------------
Final objective value               -8.067778E+00
Relative precision                   1.518278E-09
Optimality                           8.116995E-12
Feasibility                          1.900689E-08
Complementarity                      6.734260E-08
Iteration counts
  Outer iterations                             12
  Inner iterations                             31
  Linesearch steps                             43
Evaluation counts
  Augm. Lagr. values                           56
  Augm. Lagr. gradient                         44
  Augm. Lagr. hessian                          31
------------------------------------------------------------

Optimal solution:
X =      2.00    -0.23    -0.27
X =     -0.30    -0.10     2.00
X =      2.00    -1.78    -0.46
```