NAG Library Routine Document

E04MZF

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

E04MZF reads data for a sparse linear programming or quadratic programming problem from an external file which is in standard or compatible MPSX input format.

2 Specification

```
SUBROUTINE E04MZF (INFILE, MAXN, MAXM, MAXNNZ, XBLDEF, XBUDEF, MPSLST,     &
                   N, M, NNZ, IOBJ, NCOLH, A, HA, KA, BL, BU, START,       &
                   NAMES, NNAME, CRNAME, XS, ISTATE, IFAIL)
INTEGER            INFILE, MAXN, MAXM, MAXNNZ, N, M, NNZ, IOBJ, NCOLH,     &
                   HA(MAXNNZ), KA(MAXN+1), NNAME, ISTATE(MAXN+MAXM),       &
                   IFAIL
REAL (KIND=nag_wp) XBLDEF, XBUDEF, A(MAXNNZ), BL(MAXN+MAXM),               &
                   BU(MAXN+MAXM), XS(MAXN+MAXM)
LOGICAL            MPSLST
CHARACTER(1)       START
CHARACTER(8)       NAMES(5), CRNAME(MAXN+MAXM)
```

3 Description

E04MZF reads linear programming (LP) or quadratic programming (QP) problem data from an external file which is prepared in standard or compatible MPSX (see IBM (1971)) input format and then initializes n (the number of variables), m (the number of general linear constraints), the m by n matrix A, and the vectors l, u and c (stored in row IOBJ of A) for use with E04NKF, which is designed to solve problems of the form

$$\underset{x \in R^n}{\text{minimize}} c^{\mathrm{T}}x + \tfrac{1}{2}x^{\mathrm{T}}Hx \quad \text{subject to} \quad l \leq \begin{Bmatrix} x \\ Ax \end{Bmatrix} \leq u.$$

For LP problems, $H = 0$. For QP problems, you must set NCOLH > 0 (see Section 5) and provide a subroutine to E04NKF to compute Hx for any given vector x. (This is illustrated in Section 10.) The optional parameter **Maximize** may be used to specify an alternative problem in which the objective function is maximized (see Section 12.1 in E04NKF/E04NKA).

MPSX input format

The input file of data may only contain two types of lines:

1. Indicator lines (specifying the type of data which is to follow).
2. Data lines (specifying the actual data).

The input file must not contain any blank lines. Any characters beyond column 80 are ignored. Indicator lines must not contain leading blank characters (in other words they must begin in column 1). The following displays the order in which the indicator lines must appear in the file:

```
      NAME          user-supplied name
ROWS
        data line(s)
COLUMNS
        data line(s)
RHS
        data line(s)
RANGES              (optional)
        data line(s)
BOUNDS              (optional)
        data line(s)
ENDATA
```

The 'user-supplied name' specifies a name for the problem and must occupy columns 15−22. The name can either be blank or up to a maximum of 8 characters.

A data line follows the same fixed format made up of fields defined below. The contents of the fields may have different significance depending upon the section of data in which they appear.

	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
Columns	2−3	5−12	15−22	25−36	40−47	50−61
Contents	Code	Name	Name	Value	Name	Value

The names and codes consist of 'alphanumeric' characters (i.e., a–z, A–Z, 0–9, +, −, *, blank ( ), :, $ or full stop (.) only) and the names must not contain leading blank characters. Values are read using Fortran format E12.0. This allows values to be entered in several equivalent forms. For example, 1.2345678, 1.2345678E + 0, 123.45678E−2 and 12345678E−07 all represent the same number. It is safest to include an explicit decimal point.

Note that in order to ensure numeric values are interpreted as intended, they should be *right-justified* in the 12-character field, with no trailing blanks. This is because in some situations trailing blanks may be interpreted as zeros and this can dramatically affect the interpretation of the value. This is relevant if the value contains an exponent, or if it contains neither an exponent nor an explicit decimal point. For example, the fields

```
%%%%1.23E-2%
 %%%%%%%123%%
```

may be interpreted as 1.23E−20 and 12300 respectively (where % denotes a blank). The actual behaviour is system-dependent.

Comment lines are allowed in the data file. These must have an asterisk (*) in column 1 and any characters in columns 2–80. In any data line, a dollar sign ($) as the first character in Field 3 or 5 indicates that the information from that point through column 80 consists of comments.

Columns outside the six fields must be blank, except for columns 72–80, whose contents are ignored by the routine. These columns may be used to enter a sequence number. A non-blank character outside the predefined six fields and columns 72–80 is considered to be a major error (IFAIL = 13; see Section 6), unless it is part of a comment.

ROWS Data Lines

These lines specify row (constraint) names and their inequality types (i.e., $=$, $\geq$ or $\leq$).

Field 1:	defines the constraint type. It may be in column 2 or column 3.
N	free row, that is no constraint. It may be used to define the objective row.
G	greater than or equal to (i.e., $\geq$).
L	less than or equal to (i.e., $\leq$).
E	exactly equal to (i.e., $=$).
Field 2:	defines the row name.

Row type N stands for 'Not binding', also known as 'Free'. It can be used to define the objective row. The objective row is a free row that specifies the vector c in the linear objective term $c^{\mathrm{T}}x$. It is taken to

be the first free row, unless some other free row name is specified by the NAMES array (see Section 5). Note that c is assumed to be zero if (for example) the line

```
%N%%DUMMYROW
```

(where % denotes a blank) appears in the ROWS section of the MPSX data file, and the row name DUMMYROW is omitted from the COLUMNS section.

COLUMNS Data Lines

These lines specify the names to be assigned to the variables (columns) in the general linear constraint matrix A, and define, in terms of column vectors, the actual values of the corresponding matrix elements.

Field 1: blank (ignored).

Field 2: gives the name of the column associated with the elements specified in the following fields.

Field 3: contains the name of a row.

Field 4: used in conjunction with Field 3 contains the value of the matrix element.

Field 5: is optional (may be used like Field 3).

Field 6: is optional (may be used like Field 4).

Note that only the nonzero elements of A and c need to be specified in the COLUMNS section, as any zero elements of A are removed and any unspecified elements of c are assumed to be zero. In addition, any nonzero elements in the jth column of A must be grouped together before those in the $(j+1)$th column, for $j = 1, 2, \ldots, n-1$. Nonzero elements within a column may however appear in any order.

RHS Data Lines

This section specifies the right-hand side values of the general linear constraint matrix A (if any). The lines specify the name to be given to the right-hand side (RHS) vector along with the numerical values of the elements of the vector, which may appear in any order. The data lines have exactly the same format as the COLUMNS data lines, except that the column name is replaced by the RHS name. Only the nonzero elements need be specified. Note that this section may be empty, in which case the RHS vector is assumed to be zero.

RANGES Data Lines (optional)

Ranges are used for constraints of the form $l \leq Ax \leq u$, where both l and u are finite. The range of the constraint is $r = u - l$. Either l or u must be specified in the RHS section and r must be defined in this section. The data lines have exactly the same format as the COLUMNS data lines, except that the column name is replaced by the RANGES name.

BOUNDS Data Lines (optional)

These lines specify limits on the values of the variables (l and u in $l \leq x \leq u$). If the variable is not specified in the bound set then it is automatically assumed to lie between default lower and upper bounds (usually 0 and $+\infty$). Like an RHS column which is given a name, the set of variables in one bound set is also given a name.

Field 1:	specifies the type of bound or defines the variable type.
LO	lower bound
UP	upper bound
FX	fixed variable
FR	free variable ($-\infty$ to $+\infty$)
MI	lower bound is $-\infty$
PL	upper bound is $+\infty$. This is the default variable type.
Field 2:	identifies a name for the bound set.
Field 3:	identifies the column name of the variable belonging to this set.
Field 4:	identifies the value of the bound; this has a numerical value only in association with LO, UP, FX in Field 1, otherwise it is blank.

Field 5: is blank and ignored.
Field 6: is blank and ignored.

Note that if RANGES and BOUNDS sections are both present, the RANGES section must appear first.

4 References

IBM (1971) MPSX – Mathematical programming system *Program Number 5734 XM4* IBM Trade Corporation, New York

5 Arguments

1: INFILE – INTEGER *Input*

On entry: the unit number associated with the MPSX data file.

Constraint: $0 \leq \text{INFILE} \leq 99$.

2: MAXN – INTEGER *Input*

On entry: an upper limit for the number of variables in the problem.

Constraint: $\text{MAXN} \geq 1$.

3: MAXM – INTEGER *Input*

On entry: an upper limit for the number of constraints (including the objective row) in the problem.

Constraint: $\text{MAXM} \geq 1$.

4: MAXNNZ – INTEGER *Input*

On entry: an upper limit for the number of nonzeros (including the objective row) in the problem.

Constraint: $\text{MAXNNZ} \geq 1$.

5: XBLDEF – REAL (KIND=nag_wp) *Input*

On entry: the default lower bound to be used for the variables in the problem when none is specified in the BOUNDS section of the MPSX data file. For a standard LP or QP problem XBLDEF would normally be set to zero.

6: XBUDEF – REAL (KIND=nag_wp) *Input*

On entry: the default upper bound to be used for the variables in the problem when none is specified in the BOUNDS section of the MPSX data file. For a standard LP or QP problem XBUDEF would normally be set to 'infinity' (i.e., $\text{XBUDEF} \geq 10^{20}$).

Constraint: XBUDEF $\geq$ XBLDEF.

7: MPSLST – LOGICAL *Input*

On entry: if MPSLST = .TRUE., then a listing of the input data is sent to the current advisory message unit (as defined by X04ABF). This can be useful for debugging the MPSX data file. If MPSLST = .FALSE., then no listing is produced.

8: N – INTEGER *Output*

On exit: n, the actual number of variables in the problem.

9: M – INTEGER *Output*

On exit: m, the actual number of general linear constraints in the problem (including the objective row).

10: NNZ – INTEGER *Output*

On exit: the actual number of nonzeros in the problem (including the objective row).

11: IOBJ – INTEGER *Output*

On exit: if IOBJ > 0, row IOBJ of A is a free row containing the nonzero coefficients of the vector c.

If IOBJ $= 0$, the coefficients of c are assumed to be zero.

If IOBJ $= -1$, no such row was found and the routine terminates with IFAIL $= 4$ or 5 (see Section 6).

12: NCOLH – INTEGER *Output*

On exit: NCOLH $= 0$. For QP problems, NCOLH is the number of leading nonzero columns of the Hessian matrix H and must therefore be set > 0 before calling E04NKF.

13: A(MAXNNZ) – REAL (KIND=nag_wp) array *Output*

On exit: the nonzero elements of A, ordered by increasing column index.

14: HA(MAXNNZ) – INTEGER array *Output*

On exit: the row indices of the nonzero elements stored in A.

15: KA(MAXN + 1) – INTEGER array *Output*

On exit: a set of pointers to the beginning of each column of A. More precisely, KA(i) contains the index in A of the start of the ith column, for $i = 1, 2, \ldots,$ N. Note that KA(1) $= 1$ and KA(N + 1) $=$ NNZ $+ 1$.

16: BL(MAXN + MAXM) – REAL (KIND=nag_wp) array *Output*

17: BU(MAXN + MAXM) – REAL (KIND=nag_wp) array *Output*

On exit: BL contains the vector l (the lower bounds) and BU contains the vector u (the upper bounds), for all the variables and constraints in the following order. The first N elements of each array contain the bounds on the variables x and the next M elements contain the bounds for the linear objective term $c^{\mathrm{T}}x$ and the general linear constraints Ax (if any). Note that an 'infinite' lower bound is indicated by BL(j) $= -1.0$E $+ 20$, an 'infinite' upper bound by BU(j) $= -1.0$E $+ 20$ and an equality constraint by BL(j) $=$ BU(j). (The lower bound for $c^{\mathrm{T}}x$, stored in BL(N + IOBJ), is set to $-$XBUDEF. The corresponding upper bound, stored in BU(N + IOBJ), is set to XBUDEF.)

Note that E04MZF uses an 'infinite' bound size of 10^{20} in the definition of l and u. In other words, any element of u greater than or equal to 10^{20} will be regarded as $+\infty$ (and similarly any element of l less than or equal to -10^{20} will be regarded as $-\infty$). If this value is deemed to be 'inappropriate', you are recommended to reset the value of the optional parameter **Infinite Bound Size** and make any necessary changes to BL and/or BU before calling E04NKF.

18: START – CHARACTER(1) *Output*

On exit: START $=$ 'C' and an internal Crash procedure will be used by E04NKF to choose an initial basis.

19: NAMES(5) – CHARACTER(8) array *Input/Output*

On entry: a set of names associated with the MPSX form of the problem.

NAMES(1)
Must contain either the name of the problem or be blank.

NAMES(2)
Must contain either the name of the objective row or be blank (in which case the first objective free row is used).

NAMES(3)
Must contain either the name of the RHS set to be used or be blank (in which case the first RHS set is used).

NAMES(4)
Must contain either the name of the RANGE set to be used or be blank (in which case the first RANGE set (if any) is used).

NAMES(5)
Must contain either the name of the BOUNDS set to be used or be blank (in which case the first BOUNDS set (if any) is used).

On exit: a set of names associated with the problem as defined in the MPSX data file as follows:

NAMES(1)
Contains the name of the problem (or blank if none).

NAMES(2)
Contains the name of the objective row (or blank if none).

NAMES(3)
Contains the name of the RHS set (or blank if none).

NAMES(4)
Contains the name of the RANGE set (or blank if none).

NAMES(5)
Contains the name of the BOUNDS set (or blank if none).

20: NNAME – INTEGER *Output*

On exit: $n+m$, the total number of variables and constraints in the problem.

21: CRNAME(MAXN + MAXM) – CHARACTER(8) array *Output*

On exit: the MPSX names of all the variables and constraints in the problem in the following order. The first N elements contain the MPSX names for the variables and the next M elements contain the MPSX names for the objective row and general linear constraints (if any). Note that the MPSX name for the objective row is stored in CRNAME(N + IOBJ).

22: XS(MAXN + MAXM) – REAL (KIND=nag_wp) array *Output*

On exit: a set of initial values for the variables and constraints in the problem. More precisely, $\mathrm{XS}(j) = \min(\max(0.0, \mathrm{BL}(j)), \mathrm{BU}(j))$, for $j = 1, 2, \ldots, \mathrm{NNAME}$.

23: ISTATE(MAXN + MAXM) – INTEGER array *Output*

On exit: a set of initial states for the variables and constraints in the problem. More precisely, $\mathrm{ISTATE}(j) = 1$ if $\mathrm{XS}(j) = \mathrm{BU}(j)$ and 0 otherwise, for $j = 1, 2, \ldots, \mathrm{NNAME}$.

24: 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, if you are not familiar with this argument, the recommended value is 0. **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 $= 1$

There are too many rows present in the data file. Increase MAXM by at least (M – MAXM) and rerun E04MZF.

IFAIL $= 2$

There are too many columns present in the data file. Increase MAXN by at least (N – MAXN) and rerun E04MZF.

IFAIL $= 3$

There are too many nonzeros present in the data file. Increase MAXNNZ by at least (NNZ – MAXNNZ) and rerun E04MZF.

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.

The following error exits (apart from IFAIL $= 17$) are caused by having either a corrupt or a nonstandard MPSX data file. Refer to Section 3 for a detailed description of the MPSX format which can be read by E04MZF. If MPSLST = .TRUE., the last line of printed output refers to the line in the MPSX data file which contains the reported error.

IFAIL $= 4$

The objective row was not found. There must be at least one row in the ROWS section with row type `N` for the objective row.

IFAIL $= 5$

An unknown objective row name was detected in the ROWS section.

IFAIL $= 6$

There are no rows specified in the ROWS section.

IFAIL $= 7$

An illegal constraint type was detected in the ROWS section. The constraint type must be either `N`, `L`, `G` or `E`.

IFAIL $= 8$

An illegal row name was detected in the ROWS section. Names must be made up of 'alphanumeric' characters (see Section 3) with no leading blanks.

IFAIL $= 9$

An illegal column name was detected in the COLUMNS section. Names must be made up of 'alphanumeric' characters (see Section 3) with no leading blanks.

IFAIL $= 10$

An illegal bound type was detected in the BOUNDS section. The bound type must be either `LO`, `UP`, `FX`, `FR`, `MI` or `PL`.

IFAIL $= 11$

An unknown column name was detected in the BOUNDS section. All the column names must be specified in the COLUMNS section.

IFAIL $= 12$

The last line in the data file does not contain the ENDATA line indicator.

IFAIL $= 13$

An illegal data line was detected in the file. This line is neither a comment line nor a valid data line.

IFAIL $= 14$

An unknown row name was detected in COLUMNS, RHS or RANGES section. All the row names must be specified in the ROWS section.

IFAIL $= 15$

There were no columns specified in the COLUMNS section.

IFAIL $= 16$

The name of the RHS, RANGES or BOUNDS set to be used was not found in the data file.

IFAIL $= 17$

On entry, INFILE < 0,
or INFILE > 99,
or MAXN < 1,
or MAXM < 1,
or MAXNNZ < 1,
or XBLDEF $>$ XBUDEF.

7 Accuracy

Not applicable.

8 Parallelism and Performance

E04MZF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example solves the quadratic programming problem

$$\text{minimize} \quad c^{\mathrm{T}}x + \tfrac{1}{2}x^{\mathrm{T}}Hx \quad \text{subject to} \quad \begin{array}{rcl} l & \leq Ax & \leq u, \\ -2 & \leq x & \leq 2, \end{array}$$

where

$$c = \begin{pmatrix} -4.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -0.1 \\ -0.3 \end{pmatrix}, \quad H = \begin{pmatrix} 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \text{NCOLH} = 5,$$

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

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

The optimal solution (to five figures) is

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

Three bound constraints and two general linear constraints are active at the solution. Note that, although the Hessian matrix is positive semidefinite, the point x^* is unique.

The MPSX representation of the problem is given in Section 10.2.

10.1 Program Text

```
!   E04MZF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.
    Module e04mzfe_mod

!     E04MZF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                             :: qphx
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter, Public :: xbldef = 0.0_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: xbudef = 1.0E+20_nag_wp
      Integer, Parameter, Public         :: iset = 1, lencw = 600, leniw = 600,  &
                                            lenrw = 600, maxm = 10000,           &
                                            maxn = 10000, maxnnz = 100000,       &
                                            nindat = 7, nout = 6
    Contains
      Subroutine qphx(ncolh,x,hx,nstate,cuser,iuser,ruser)
```

```
!       Routine to compute H*x. (In this version of QPHX, the Hessian
!       matrix H is not referenced explicitly.)

!       .. Scalar Arguments ..
        Integer, Intent (In)           :: ncolh, nstate
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: hx(ncolh)
        Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
        Real (Kind=nag_wp), Intent (In) :: x(ncolh)
        Integer, Intent (Inout)        :: iuser(*)
        Character (8), Intent (Inout)  :: cuser(*)
!       .. Executable Statements ..
        If (nstate==1) Then

!         First entry. You may perform any required one-time
!         operations here.

        End If

        hx(1) = 2.0_nag_wp*x(1) + x(2) + x(3) + x(4) + x(5)
        hx(2) = x(1) + 2.0_nag_wp*x(2) + x(3) + x(4) + x(5)
        hx(3) = x(1) + x(2) + 2.0_nag_wp*x(3) + x(4) + x(5)
        hx(4) = x(1) + x(2) + x(3) + 2.0_nag_wp*x(4) + x(5)
        hx(5) = x(1) + x(2) + x(3) + x(4) + 2.0_nag_wp*x(5)

        If (nstate>=2) Then

!         Final entry. You may perform any required clean up
!         operations here.

        End If

        Return

      End Subroutine qphx
    End Module e04mzfe_mod
    Program e04mzfe

!     E04MZF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e04mzf, e04npf, e04nqf, e04ntf, nag_wp, x04abf,   &
                             x04acf
      Use e04mzfe_mod, Only: iset, lencw, leniw, lenrw, maxm, maxn, maxnnz,    &
                             nindat, nout, qphx, xbldef, xbudef
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Character (*), Parameter         :: fname = 'e04mzfe.opt'
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: obj, objadd, sinf
      Integer                          :: i, ifail, infile, iobj, lenc, m,     &
                                          mode, n, ncolh, ninf, nname, nnz,    &
                                          ns, outchn
      Logical                          :: mpslst, verbose_output
      Character (8)                    :: prob
      Character (1)                    :: start
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), bl(:), bu(:), c(:), pi(:),     &
                                          rc(:), xs(:)
      Real (Kind=nag_wp)               :: ruser(1), rw(lenrw)
      Integer, Allocatable             :: ha(:), helast(:), istate(:), ka(:)
      Integer                          :: iuser(1), iw(leniw)
      Character (8), Allocatable       :: crname(:)
      Character (8)                    :: cuser(1), cw(lencw), names(5)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max
!     .. Executable Statements ..
      Write (nout,*) 'E04MZF Example Program Results'

      Allocate (ha(maxnnz),ka(maxn+1),istate(maxn+maxm),a(maxnnz),             &
```

```
        bl(maxn+maxm),bu(maxn+maxm),xs(maxn+maxm),crname(maxn+maxm))

!     Open the data file for reading

      mode = 0

      ifail = 0
      Call x04acf(nindat,fname,mode,ifail)

!     Initialize parameters.

      infile = nindat
      mpslst = .False.
      names(1:5) = '        '

!     Convert the MPSX data file for use by E04NQF.

      Write (nout,99999)

      ifail = 0
      Call e04mzf(infile,maxn,maxm,maxnnz,xbldef,xbudef,mpslst,n,m,nnz,iobj,   &
        ncolh,a,ha,ka,bl,bu,start,names,nname,crname,xs,istate,ifail)

      Write (nout,99998) n, m

!     Set the unit number for advisory messages to OUTCHN.

      outchn = nout
      Call x04abf(iset,outchn)

!     Set the value of NCOLH (the number of columns of the Hessian matrix)

      ncolh = 5

!     Call E04NPF to initialize E04NQF.

      ifail = 0
      Call e04npf(cw,lencw,iw,leniw,rw,lenrw,ifail)

!     Set this to .True. to cause e04nqf to produce intermediate
!     progress output
      verbose_output = .False.
      If (verbose_output) Then
        Call e04ntf('Print file',nout,cw,iw,rw,ifail)
      End If

!     We have no explicit objective vector so set LENC = 0; the
!     objective vector is stored in row IOBJ of A.

      lenc = 0
      Allocate (c(max(1,lenc)),helast(n+m),pi(m),rc(n+m))

      objadd = 0.0_nag_wp
      prob = ' '

!     Do not allow any elastic variables (i.e. they cannot be
!     infeasible).

      helast(1:(n+m)) = 0

      Write (nout,99997)
      ifail = 0
      Call e04nqf(start,qphx,m,n,nnz,nname,lenc,ncolh,iobj,objadd,prob,a,ha,   &
        ka,bl,bu,c,crname,helast,istate,xs,pi,rc,ns,ninf,sinf,obj,cw,lencw,iw, &
        leniw,rw,lenrw,cuser,iuser,ruser,ifail)

      Write (nout,99996) obj
      Write (nout,99995)
      Write (nout,99994)(i,xs(i),i=1,n)

99999 Format (1X,/,1X,'Reading MPSX file:')
```

```
99998 Format (1X,'MPSX file contains ',I3,' variables and ',I3,                &
        ' linear constraints')
99997 Format (1X,/,1X,'Solving QP problem:')
99996 Format (1X,'Optimal objective value: ',1P,E11.3)
99995 Format (1X,'Optimal X:')
99994 Format (1X,'  X(',I2,') =',F10.4)
    End Program e04mzfe
```

10.2 Program Data

Note: the MPSX data which is read by E04MZF begins with the **second** record of this data file; the first record is a caption which is read by the example program.

```
NAME          QP
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
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
ENDATA
```

10.3 Program Results

```
 E04MZF Example Program Results

 Reading MPSX file:
 MPSX file contains   9 variables and   4 linear constraints

 Solving QP problem:
 Optimal objective value:  -8.068E+00
 Optimal X:
   X( 1) =    2.0000
   X( 2) =   -0.2333
   X( 3) =   -0.2667
   X( 4) =   -0.3000
   X( 5) =   -0.1000
   X( 6) =    2.0000
   X( 7) =    2.0000
   X( 8) =   -1.7778
   X( 9) =   -0.4556
```