NAG Library Routine Document

E04MXF

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

E04MXF reads data for sparse linear programming, mixed integer linear programming, quadratic programming or mixed integer quadratic programming problems from an external file which is in standard or compatible MPS input format.

2 Specification

```
SUBROUTINE E04MXF (INFILE, 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)

INTEGER            INFILE, MAXN, MAXM, MAXNNZ, MAXNCOLH, MAXNNZH,          &
                   MAXLINTVAR, MPSLST, N, M, NNZ, NCOLH, NNZH, LINTVAR,    &
                   IOBJ, IROWA(MAXNNZ), ICCOLA(MAXN+1), NNAME,             &
                   IROWH(MAXNNZH), ICCOLH(MAXNCOLH+1), MINMAX,             &
                   INTVAR(MAXLINTVAR), IFAIL
REAL (KIND=nag_wp) A(MAXNNZ), BL(MAXN+MAXM), BU(MAXN+MAXM), H(MAXNNZH)
CHARACTER(8)       PNAMES(5), CRNAME(MAXN+MAXM)
```

3 Description

E04MXF reads data for linear programming (LP) or quadratic programming (QP) problems (or their mixed integer variants) from an external file which is prepared in standard or compatible MPS (see IBM (1971)) input format. It then initializes n (the number of variables), m (the number of general linear constraints), the m by n matrix A, the vectors l, u, c (stored in row IOBJ of A) and the n by n Hessian matrix H for use with E04NKF/E04NKA and E04NQF. These routines are designed to solve problems of the form

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

3.1 MPS 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).

A *section* is a combination of an indicator line and its corresponding data line(s). 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	(optional)
OBJSENSE		(optional)
	data line	
OBJNAME		(optional)
	data line	
ROWS		
	data line(s)	
COLUMNS		
	data line(s)	
RHS		
	data line(s)	
RANGES		(optional)
	data line(s)	
BOUNDS		(optional)
	data line(s)	
QUADOBJ		(optional)
	data line(s)	
ENDATA		

A data line follows a fixed format, being made up of fields as 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

Each name and code must consist of 'printable' characters only; names and codes supplied must match the case used in the following descriptions. Values are read using a field width of 12. 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.

Lines with an asterisk (∗) in column 1 will be considered comment lines and will be ignored by the routine.

Columns outside the six fields must be blank, except for columns 72–80, whose contents are ignored by the routine. A non-blank character outside the predefined six fields and columns 72–80 is considered to be a major error (IFAIL = 16; see Section 6), unless it is part of a comment.

3.1.1 NAME Section (optional)

The NAME section is the only section where the data must be on the same line as the indicator. The 'user-supplied name' must be in field 3 but may be blank.

Field	**Required**	**Description**
3	No	Name of the problem

3.1.2 OBJSENSE Section (optional)

The data line in this section can be used to specify the sense of the objective function. If this section is present it must contain only one data line. If the section is missing or empty, minimization is assumed.

Field	**Required**	**Description**
2	No	Sense of the objective function

Field 2 may contain either `MIN`, `MAX`, `MINIMIZE` or `MAXIMIZE`.

3.1.3 OBJNAME Section (optional)

The data line in this section can be used to specify the name of a free row (see Section 3.1.4) that should be used as the objective function. If this section is present it must contain only one data line. If the section is missing or is empty, the first free row will be chosen instead. Alternatively, OBJNAME can be overridden by setting nonempty PNAMES(2) (see Section 5).

Field	Required	Description
2	No	Row name to be used as the objective function

Field 2 must contain a valid row name.

3.1.4 ROWS Section

The data lines in this section specify unique row (constraint) names and their inequality types (i.e., unconstrained, $=$, $\geq$ or $\leq$).

Field	Required	Description
1	Yes	Inequality key
2	Yes	Row name

The inequality key specifies each row's type. It must be `E`, `G`, `L` or `N` and can be in either column 2 or 3.

Inequality Key	Description	l	u
`N`	Free row	$-\infty$	∞
`G`	Greater than or equal to	finite	∞
`L`	Less than or equal to	$-\infty$	finite
`E`	Equal to	finite	l

Row type `N` stands for 'Not binding'. 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 $c^{\mathrm{T}}x$. If there is more than one free row, the first free row is chosen, unless another free row name is specified by OBJNAME (see Section 3.1.3) or PNAMES(2) (see Section 5). Note that c is assumed to be zero if either the chosen row does not appear in the COLUMNS section (i.e., has no nonzero elements) or there are no free rows defined in the ROWS section.

3.1.5 COLUMNS Section

Data lines in this section 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	Required	Description
2	Yes	Column name
3	Yes	Row name
4	Yes	Value
5	No	Row name
6	No	Value

Each data line in the COLUMNS section defines the nonzero elements of A or c. Any elements of A or c that are undefined are assumed to be zero. Nonzero elements of A must be grouped by column, that is to say that all of the nonzero elements in the jth column of A must be specified before those in the $j+1$th column, for $j = 1, 2, \ldots, n-1$. Rows may appear in any order within the column.

3.1.5.1 Integer Markers

For backward compatibility E04MXF allows you to define the integer variables within the COLUMNS section using integer markers, although this is not recommended as markers can be treated differently by different MPS readers; you should instead define any integer variables in the BOUNDS section (see below). Each marker line must have the following format:

Field	Required	Description
2	No	Marker ID
3	Yes	Marker tag
5	Yes	Marker type

The marker tag must be 'MARKER'. The marker type must be 'INTORG' to start reading integer variables and 'INTEND' to finish reading integer variables. This implies that a row cannot be named 'MARKER', 'INTORG' or 'INTEND'. Please note that both marker tag and marker type comprise of 8 characters as a ' is the mandatory first and last character in the string. You may wish to have several integer marker sections within the COLUMNS section, in which case each marker section must begin with an 'INTORG' marker and end with an 'INTEND' marker and there should not be another marker between them.

Field 2 is ignored by E04MXF. When an integer variable is declared it will keep its default bounds unless they are changed in the BOUNDS section. This may vary between different MPS readers.

3.1.6 RHS Section

This section specifies the right-hand side values (if any) of the general linear constraint matrix A.

Field	Required	Description
2	Yes	RHS name
3	Yes	Row name
4	Yes	Value
5	No	Row name
6	No	Value

The MPS file may contain several RHS sets distinguished by RHS name. If an RHS name is defined in PNAMES(3) (see Section 5) then E04MXF will read in only that RHS vector, otherwise the first RHS set will be used.

Only the nonzero RHS elements need to be specified. Note that if an RHS is given to the objective function it will be ignored by E04MXF. An RHS given to the objective function is dealt with differently by different MPS readers, therefore it is safer to not define an RHS of the objective function in your MPS file. Note that this section may be empty, in which case the RHS vector is assumed to be zero.

3.1.7 RANGES Section (optional)

Ranges are used to modify the interpretation of constraints defined in the ROWS section (see Section 3.1.4) to the form $l \leq Ax \leq u$, where both l and u are finite. The range of the constraint is $r = u - l$.

Field	Required	Description
2	Yes	Range name
3	Yes	Row name
4	Yes	Value
5	No	Row name
6	No	Value

The range of each constraint implies an upper and lower bound dependent on the inequality key of each constraint, on the RHS b of the constraint (as defined in the RHS section), and on the range r.

Inequality Key	Sign of r	l	u
E	$+$	b	$b + r$
E	$-$	$b + r$	b
G	$+/-$	b	$b + \lvert r \rvert$
L	$+/-$	$b - \lvert r \rvert$	b
N	$+/-$	$-\infty$	$+\infty$

If a range name is defined in PNAMES(4) (see Section 5) then the routine will read in only the range set of that name, otherwise the first set will be used.

3.1.8 BOUNDS Section (optional)

These lines specify limits on the values of the variables (the quantities l and u in $l \leq x \leq u$). If a variable is not specified in the bound set then it is automatically assumed to lie between 0 and $+\infty$.

Field	Required	Description
1	Yes	Bound type identifier
2	Yes	Bound name
3	Yes	Column name
4	Yes/No	Value

Note: field 4 is required only if the bound type identifier is one of `UP`, `LO`, `FX`, `UI` or `LI` in which case it gives the value k below. If the bound type identifier is `FR`, `MI`, `PL` or `BV`, field 4 is ignored and it is recommended to leave it blank.

The table below describes the acceptable bound type identifiers and how each determines the variables' bounds.

Bound Type Identifier	*l*	*u*	Integer Variable?
`UP`	unchanged	k	No
`LO`	k	unchanged	No
`FX`	k	k	No
`FR`	$-\infty$	∞	No
`MI`	$-\infty$	unchanged	No
`PL`	unchanged	∞	No
`BV`	0	1	Yes
`UI`	unchanged	k	Yes
`LI`	k	unchanged	Yes

If a bound name is defined in PNAMES(5) (see Section 5) then the routine will read in only the bound set of that name, otherwise the first set will be used.

3.1.9 QUADOBJ Section (optional)

The QUADOBJ section defines nonzero elements of the upper or lower triangle of the Hessian matrix H.

Field	Required	Description
2	Yes	Column name (HColumn Index)
3	Yes	Column name (HRow Index)
4	Yes	Value
5	No	Column name (HRow Index)
6	No	Value

Each data line in the QUADOBJ section defines one (or optionally two) nonzero elements H_{ij} of the matrix H. Each element H_{ij} is given as a triplet of row index i, column index j and a value. The column names (as defined in the COLUMNS section) are used to link the names of the variables and the indices i and j. More precisely, the matrix H on output will have a nonzero element

$$H_{ij} = \text{Value}$$

where index j belongs to HColumn Index and index i to one of the HRow Indices such that

$$\text{CRNAME}(j) = \text{Column name (HColumn Index) and}$$

$$\text{CRNAME}(i) = \text{Column name (HRow Index).}$$

It is only necessary to define either the upper or lower triangle of the H matrix; either will suffice. Any elements that have been defined in the upper triangle of the matrix will be moved to the lower triangle of the matrix, then any repeated nonzeros will be summed.

Note: it is much more efficient for E04NKF/E04NKA and E04NQF to have the H matrix defined by the first NCOLH column names. If the nonzeros of H are defined by any columns that are not in the first NCOLH of N then E04MXF will rearrange the matrices A and H so that they are.

3.2 Query Mode

E04MXF offers a 'query mode' to quickly give upper estimates on the sizes of user arrays. In this mode any expensive checks of the data and of the file format are skipped, providing a prompt count of the number of variables, constraints and matrix nonzeros. This might be useful in the common case where the size of the problem is not known in advance.

You may activate query mode by setting any of the following: MAXN < 1, MAXM < 1, MAXNNZ < 1, MAXNCOLH < 0 or MAXNNZH < 0. If no major formatting error is detected in the data file, IFAIL $= 0$ is returned and the upper estimates are given as stated in Table 1. Alternatively, the routine switches to query mode while the file is being read if it is discovered that the provided space is insufficient (that is, if N $>$ MAXN, M $>$ MAXM, NNZ $>$ MAXNNZ, NCOLH $>$ MAXNCOLH, NNZH $>$ MAXNNZH or LINTVAR $>$ MAXLINTVAR). In this case IFAIL $= 2$ is returned.

Argument Name	Upper Estimate for
N	MAXN
M	MAXM
NNZ	MAXNNZ
NCOLH	MAXNCOLH
NNZH	MAXNNZH
LINTVAR	MAXLINTVAR

Table 1

The recommended practice is shown in Section 10, where the routine is invoked twice. The first call queries the array lengths required, after which the data arrays are allocated to be of these sizes. The second call reads the data using the sufficiently-sized arrays.

4 References

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

5 Parameters

1: INFILE – INTEGER *Input*

On entry: the ID of the MPSX data file to be read as returned by a call to X04ACF.

Constraint: INFILE ≥ 0.

2: MAXN – INTEGER *Input*

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

If MAXN < 1, E04MXF will start in query mode (see Section 3.2).

3: MAXM – INTEGER *Input*

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

If MAXM < 1, E04MXF will start in query mode (see Section 3.2).

4: MAXNNZ – INTEGER *Input*

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

If MAXNNZ < 1, E04MXF will start in query mode (see Section 3.2).

5: MAXNCOLH – INTEGER *Input*

On entry: an upper limit for the dimension of the matrix H.

If MAXNCOLH < 0, E04MXF will start in query mode (see Section 3.2).

6: MAXNNZH – INTEGER *Input*

On entry: an upper limit for the number of nonzeros of the matrix H.

If MAXNNZH < 0, E04MXF will start in query mode (see Section 3.2).

7: MAXLINTVAR – INTEGER *Input*

On entry: if MAXLINTVAR ≥ 0, an upper limit for the number of integer variables.

If MAXLINTVAR < 0, E04MXF will treat all integer variables in the file as continuous variables.

8: MPSLST – INTEGER *Input*

On entry: if MPSLST $\neq 0$, summary messages are sent to the current advisory message unit (as defined by X04ABF) as E04MXF reads through the data file. This can be useful for debugging the file. If MPSLST $= 0$, then no summary is produced.

9: N – INTEGER *Output*

On exit: if E04MXF was run in query mode (see Section 3.2), or returned with IFAIL $= 2$, an upper estimate of the number of variables of the problem. Otherwise, n, the actual number of variables in the problem.

10: M – INTEGER *Output*

On exit: if E04MXF was run in query mode (see Section 3.2), or returned with IFAIL $= 2$, an upper estimate of the number of general linear constraints in the problem (including the objective row). Otherwise m, the actual number of general linear constaints of the problem.

11: NNZ – INTEGER *Output*

On exit: if E04MXF was run in query mode (see Section 3.2), or returned with IFAIL $= 2$, an upper estimate of the number of nonzeros in the problem (including the objective row). Otherwise the actual number of nonzeros in the problem (including the objective row).

12: NCOLH – INTEGER *Output*

On exit: if E04MXF was run in query mode (see Section 3.2), or returned with IFAIL $= 2$, an upper estimate of the value of NCOLH required by E04NKF/E04NKA and E04NQF. In this context NCOLH is the number of leading nonzero columns of the Hessian matrix H. Otherwise, the actual dimension of the matrix H.

13: NNZH – INTEGER *Output*

On exit: if E04MXF was run in query mode (see Section 3.2), or returned with IFAIL $= 2$, an upper estimate of the number of nonzeros of the matrix H. Otherwise, the actual number of nonzeros of the matrix H.

14: LINTVAR – INTEGER *Output*

On exit: if on entry MAXLINTVAR < 0, all integer variables are treated as continuous and LINTVAR $= -1$.

If E04MXF was run in query mode (see Section 3.2), or returned with IFAIL $= 2$, an upper estimate of the number of integer variables of the problem. Otherwise, the actual number of integer variables of the problem.

15: 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 E04MXF is run in query mode (see Section 3.2) IOBJ is not referenced.

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

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

If E04MXF is run in query mode (see Section 3.2), A is not referenced.

17: IROWA(MAXNNZ) – INTEGER array *Output*

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

If E04MXF is run in query mode (see Section 3.2), IROWA is not referenced.

18: ICCOLA(MAXN + 1) – INTEGER array *Output*

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

If E04MXF is run in query mode (see Section 3.2), ICCOLA is not referenced.

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

20: 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 contains the bounds on the variables x and the next M elements contains the bounds for the linear objective term $c^{\mathrm{T}}x$ and for the general linear constraints Ax (if any). Note that an 'infinite' lower bound is indicated by BL(j) $= -1.0\mathrm{E}+20$ and an 'infinite' upper bound by BU(j) $= +1.0\mathrm{E}+20$. 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', before calling E04NKF/E04NKA or E04NQF you are recommended to reset the value of its optional parameter E04NKF/E04NKA and E04NQF and make any necessary changes to BL and/or BU.

If E04MXF is run in query mode (see Section 3.2), BL and BU are not referenced.

21: PNAMES(5) – CHARACTER(8) array *Input/Output*

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

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

PNAMES(2)
: Must either be blank or contain the name of the objective row (in which case it overrides OBJNAME section and the default choice of the first objective free row).

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

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

PNAMES(5)
Must either contain 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:

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

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

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

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

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

If E04MXF is run in query mode (see Section 3.2), PNAMES is not referenced.

22: NNAME – INTEGER *Output*

On exit: $n+m$, the total number of variables and constraints in the problem (including the objective row).

If E04MXF was run in query mode (see Section 3.2), or returned with IFAIL $= 2$, NNAME is not set.

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

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

If E04MXF is run in query mode (see Section 3.2), CRNAME is not referenced.

24: H(MAXNNZH) – REAL (KIND=nag_wp) array *Output*

On exit: the NNZH nonzero elements of H, arranged by increasing column index.

If E04MXF is run in query mode (see Section 3.2), H is not referenced.

25: IROWH(MAXNNZH) – INTEGER array *Output*

On exit: the NNZH row indices of the elements stored in H.

If E04MXF is run in query mode (see Section 3.2), IROWH is not referenced.

26: ICCOLH(MAXNCOLH + 1) – INTEGER array *Output*

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

If E04MXF is run in query mode (see Section 3.2), ICCOLH is not referenced.

27: MINMAX – INTEGER *Output*

On exit: MINMAX defines the direction of the optimization as read from the MPS file. By default the routine assumes the objective function should be minimized and will return MINMAX $= -1$. If the routine discovers in the OBJSENSE section that the objective function should be maximized it will return MINMAX $= 1$. If the routine discovers that there is neither the linear objective term

c (the objective row) nor the Hessian matrix H, the problem is considered as a feasible point problem and MINMAX $= 0$ is returned.

If E04MXF was run in query mode (see Section 3.2), or returned with IFAIL $= 2$, MINMAX is not set.

28: INTVAR(MAXLINTVAR) – INTEGER array *Output*

On exit: if MAXLINTVAR > 0 on entry, INTVAR contains pointers to the columns that are defined as integer variables. More precisely, INTVAR$(i) = k$, where k is the index of a column that is defined as an integer variable, for $i = 1, 2, \ldots,$ LINTVAR.

If MAXLINTVAR ≤ 0 on entry, or E04MXF was run in query mode (see Section 3.2), or it returned with IFAIL $= 2$, INTVAR is not set.

29: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 parameter, 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).

Note that if any of the relevant parameters are accidentally set to zero, or not set and assume zero values, then the routine will have executed in query mode. In this case only the size of the problem is returned and other parameters are not set. See Section 3.2.

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 $= 1$

Warning: MPS file not strictly fixed format, although the problem was read anyway. The data may have been read incorrectly. You should set MPSLST $= 1$ and repeat the call to E04MXF for more details.

IFAIL $= 2$

At least one of MAXM, MAXN, MAXNNZ, MAXNNZH, MAXNCOLH or MAXLINTVAR is too small. Suggested values are returned in M, N, NNZ, NNZH, NCOLH and LINTVAR respectively.

IFAIL $= 3$

Incorrect ordering of indicator lines.
OBJNAME indicator line found after ROWS indicator line.

IFAIL $= 4$

Incorrect ordering of indicator lines.
COLUMNS indicator line found before ROWS indicator line.

IFAIL $= 5$

Incorrect ordering of indicator lines.
RHS indicator line found before COLUMNS indicator line.

IFAIL $= 6$

Incorrect ordering of indicator lines.
RANGES indicator line found before RHS indicator line.

IFAIL $= 7$

Incorrect ordering of indicator lines.
BOUNDS indicator line found before COLUMNS indicator line.

IFAIL $= 8$

Incorrect ordering of indicator lines.
QUADOBJ indicator line found before BOUNDS indicator line.

IFAIL $= 9$

Incorrect ordering of indicator lines.
QUADOBJ indicator line found before COLUMNS indicator line.

IFAIL $= 10$

Unknown indicator line '⟨*value*⟩'.

IFAIL $= 12$

Indicator line '⟨*value*⟩' has been found more than once in the MPS file.

IFAIL $= 13$

End of file found before ENDATA indicator line.

IFAIL $= 14$

No indicator line found in file. It may be an empty file.

IFAIL $= 15$

At least one mandatory section not found in MPS file.

IFAIL $= 16$

An illegal line was detected in '⟨*value*⟩' section.
This is neither a comment nor a valid data line.

IFAIL $= 17$

Unknown inequality key '⟨*value*⟩' in ROWS section.
Expected '`N`', '`G`', '`L`' or '`E`'.

IFAIL $= 18$

Empty ROWS section.
Neither the objective row nor the constraints were defined.

IFAIL $= 19$

The supplied name, in PNAMES(2) or in OBJNAME, of the objective row was not found among the free rows in the ROWS section.

IFAIL $= 20$

The supplied name, in PNAMES(5), of the BOUNDS set to be used was not found in the BOUNDS section.

IFAIL $= 21$

The supplied name, in PNAMES(3), of the RHS set to be used was not found in the RHS section.

IFAIL $= 22$

The supplied name, in PNAMES(4), of the RANGES set to be used was not found in the RANGES section.

IFAIL $= 23$

Illegal row name.
Row names must consist of printable characters only.

IFAIL $= 24$

Illegal column name.
Column names must consist of printable characters only.

IFAIL $= 25$

Row name '⟨*value*⟩' has been defined more than once in the ROWS section.

IFAIL $= 26$

Column '⟨*value*⟩' has been defined more than once in the COLUMNS section. Column definitions must be continuous. (See Section 3.1.5).

IFAIL $= 27$

Found '`INTORG`' marker within '`INTORG`' to '`INTEND`' range.

IFAIL $= 28$

Found '`INTEND`' marker without previous marker being '`INTORG`'.

IFAIL $= 29$

Found '`INTORG`' but not '`INTEND`' before the end of the COLUMNS section.

IFAIL $= 30$

Illegal marker type '⟨*value*⟩'.
Should be either '`INTORG`' or '`INTEND`'.

IFAIL $= 31$

Unknown row name '⟨*value*⟩' in ⟨*value*⟩ section.
All row names must be specified in the ROWS section.

IFAIL $= 32$

Unknown column name '⟨*value*⟩' in ⟨*value*⟩ section.
All column names must be specified in the COLUMNS section.

IFAIL $= 33$

Unknown bound type '⟨*value*⟩' in BOUNDS section.

IFAIL $= 34$

More than one nonzero of A has row name '$\langle value \rangle$' and column name '$\langle value \rangle$' in the COLUMNS section.

IFAIL $= 35$

Field $\langle value \rangle$ did not contain a number (see Section 3).

IFAIL $= 36$

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

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

E04MXF is not threaded by NAG in any implementation.

E04MXF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

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

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

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

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

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 only positive semidefinite, the point x^* is unique.

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

10.1 Program Text

```
!   E04MXF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module e04mxfe_mod

!     E04MXF 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
    Contains
      Subroutine qphx(ncolh,x,hx,nstate,cuser,iuser,ruser)

!       Subroutine to compute H*x.

!       Note: IUSER and RUSER contain the following data:
!       RUSER(1:NNZH) = H(1:NNZH)
!       IUSER(1:NCOLH+1) = ICCOLH(1:NCOLH+1)
!       IUSER(NCOLH+2:NNZH+NCOLH+1) = IROWH(1:NNZH)

!       .. 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(*)
```

```
!       .. Local Scalars ..
        Integer                              :: end, icol, idx, irow, start
!       .. Executable Statements ..
        hx(1:ncolh) = 0.0E0_nag_wp

        Do icol = 1, ncolh
          start = iuser(icol)
          end = iuser(icol+1) - 1

          Do idx = start, end
            irow = iuser(ncolh+1+idx)
            hx(irow) = hx(irow) + x(icol)*ruser(idx)
            If (irow/=icol) Then
              hx(icol) = hx(icol) + x(irow)*ruser(idx)
            End If

          End Do

        End Do

        Return
      End Subroutine qphx
    End Module e04mxfe_mod

    Program e04mxfe

!     .. Use Statements ..
      Use nag_library, Only: e04mxf, e04npf, e04nqf, e04nsf, e04ntf, nag_wp,   &
                             x04acf, x04adf
      Use e04mxfe_mod, Only: qphx
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: lencw = 600, leniw = 600,        &
                                              lenrw = 600, mpslst = 1,         &
                                              nin = 7, nout = 6
      Logical, Parameter                   :: readints = .False.
      Character (*), Parameter             :: fname = 'e04mxfe.opt'
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: obj, objadd, sinf
      Integer                              :: i, ifail, iobj, lenc, lintvar,   &
                                              m, maxlintvar, maxm, maxn,       &
                                              maxncolh, maxnnz, maxnnzh,       &
                                              minmax, mode, n, ncolh, ninf,    &
                                              nname, nnz, nnzh, ns
      Character (1)                        :: start
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: a(:), bl(:), bu(:), c(:), h(:),  &
                                              pi(:), rc(:), ruser(:), rw(:),   &
                                              x(:)
      Integer, Allocatable                 :: helast(:), hs(:), iccola(:),     &
                                              iccolh(:), intvar(:), irowa(:),  &
                                              irowh(:), iuser(:), iw(:)
      Character (8), Allocatable           :: crname(:), cw(:)
      Character (8)                        :: cuser(1), pnames(5)
!     .. Intrinsic Procedures ..
      Intrinsic                            :: max, min
!     .. Executable Statements ..
      Write (nout,*) 'E04MXF Example Program Results'
      Flush (nout)

!     Initialize
      pnames(1:5) = '        '
      maxm = 0
      maxn = 0
      maxnnz = 0
      maxnnzh = 0
      maxncolh = 0
      maxlintvar = 0

!     Open the data file for reading
```

```
      mode = 0
      ifail = 0
      Call x04acf(nin,fname,mode,ifail)

!     Call e04mxf in query mode
      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,intvar)

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

!     set maxm maxn and maxnnz
      maxm = m
      maxn = n
      maxnnz = nnz
      maxnnzh = nnzh
      maxncolh = ncolh
      If (readints) Then
        maxlintvar = lintvar
      Else
        maxlintvar = -1
      End If

!     Allocate memory
      Allocate (irowa(maxnnz),iccola(maxn+1),a(maxnnz),bl(maxn+maxm), &
        bu(maxn+maxm),crname(maxn+maxm),irowh(maxnnzh),iccolh(maxncolh+1), &
        h(maxnnzh),intvar(maxlintvar))

!     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)

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

!     Data has been read. Set up and run the solver

      Allocate (iw(leniw),rw(lenrw),cw(lencw))

!     Call e04npf to initialize workspace
      ifail = 0
      Call e04npf(cw,lencw,iw,leniw,rw,lenrw,ifail)

!     Call option setter e04nsf to change the direction of optimization.
!     Minimization is assumed by default.
      If (minmax==1) Then
        ifail = 0
        Call e04nsf('Maximize',cw,iw,rw,ifail)
      Else If (minmax==0) Then
        ifail = 0
        Call e04nsf('Feasible Point',cw,iw,rw,ifail)
      End If

!     By default E04NQF does not print monitoring
!     information. Set the print file unit or the summary
!     file unit to get information.
```

```
      ifail = 0
      Call e04ntf('Print file',nout,cw,iw,rw,ifail)


!     We have no explicit objective vector so set LENC = 0; the
!     objective vector is stored in row IOBJ of ACOL.
      lenc = 0
      objadd = 0.0E0_nag_wp
      start = 'C'

      Allocate (c(max(1,lenc)),helast(n+m),x(n+m),pi(m),rc(n+m),hs(n+m),iuser( &
        ncolh+1+nnzh),ruser(nnzh))

      helast(1:n+m) = 0
      hs(1:n+m) = 0
      Do i = 1, n + m
        x(i) = min(max(0.0E0_nag_wp,bl(i)),bu(i))
      End Do

      If (ncolh>0) Then
!       Store the non zeros of H in ruser for use by qphx
        ruser(1:nnzh) = h(1:nnzh)

!       Store iccolh and irowh in iuser for use by qphx
        iuser(1:ncolh+1) = iccolh(1:ncolh+1)
        iuser(ncolh+2:nnzh+ncolh+1) = irowh(1:nnzh)
      End If

!     Call e04nqf to solve the problem
      ifail = 0
      Call e04nqf(start,qphx,m,n,nnz,nname,lenc,ncolh,iobj,objadd,pnames(1),a, &
        irowa,iccola,bl,bu,c,crname,helast,hs,x,pi,rc,ns,ninf,sinf,obj,cw, &
        lencw,iw,leniw,rw,lenrw,cuser,iuser,ruser,ifail)

    End Program e04mxfe
```

10.2 Program Data

```
NAME          E04MX.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

```
 E04MXF Example Program Results

 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 - Counting 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

 Parameters
 ==========
```

```
Files
-----
Solution file.....         0       Old basis file ........         0       (Print file)...........         6
Insert file.............         0       New basis file ........         0       (Summary file).........         0
Punch file..............         0       Backup basis file......         0
Load file...............         0       Dump file..............         0

Frequencies
-----------
Print frequency........       100       Check frequency........        60       Save new basis map.....       100
Summary frequency......       100       Factorization frequency        50       Expand frequency.......     10000

LP/QP Parameters
----------------
Minimize...............                 QPsolver Cholesky......                 Cold start.............
Scale tolerance........     0.900       Feasibility tolerance..  1.00E-06       Iteration limit........     10000
Scale option...........         2       Optimality tolerance...  1.00E-06       Print level............         1
Crash tolerance........     0.100       Pivot tolerance........  2.04E-11       Partial price..........         1
Crash option...........         3       Elastic weight.........  1.00E+00       Prtl price section ( A)         9
Elastic mode...........         1       Elastic objective......         1       Prtl price section (-I)         4

QP objective
------------
Objective variables....         5       Hessian columns........         5       Superbasics limit......         6
Nonlin Objective vars..         5       Unbounded step size....  1.00E+20
Linear Objective vars..         0

Miscellaneous
-------------
LU factor tolerance....      3.99       LU singularity tol.....  2.04E-11       Timing level...........         0
LU update tolerance....      3.99       LU swap tolerance......  1.03E-04       Debug level............         0
LU partial  pivoting...                 eps (machine precision)  1.11E-16       System information.....        No



Matrix statistics
-----------------
               Total      Normal        Free       Fixed     Bounded
Rows               4           0           1           0           3
Columns            9           0           0           0           9

No. of matrix elements                   36     Density     100.000
Biggest                           4.0000E+00  (excluding fixed columns,
Smallest                          1.0000E+00   free rows, and RHS)

No. of objective coefficients             9
Biggest                           4.0000E+00  (excluding fixed columns)
Smallest                          1.0000E-01

Nonlinear constraints       0     Linear constraints       4
Nonlinear variables         5     Linear variables         4
Jacobian  variables         0     Objective variables      5
Total constraints           4     Total variables          9


Itn      0: Feasible linear constraints

E04NQT EXIT   0 -- finished successfully
E04NQT INFO   1 -- optimality conditions satisfied

Problem name                 E04MX.EX
No. of iterations                  11   Objective value     -8.0677777778E+00
No. of Hessian products            25   Objective row       -1.0785555556E+01
                                        Quadratic objective  2.7177777778E+00
No. of superbasics                  4   No. of basic nonlinears             2
No. of degenerate steps             2   Percentage                      18.18
Max x       (scaled)        1 1.3E+00   Max pi      (scaled)        4 1.0E+00
Max x                       1 2.0E+00   Max pi                      4 1.0E+00
Max Prim inf(scaled)        0 0.0E+00   Max Dual inf(scaled)        0 0.0E+00
Max Primal infeas           0 0.0E+00   Max Dual infeas             0 0.0E+00
```

Name E04MX.EX Objective Value -8.0677777778E+00

Status Optimal Soln Iteration 11 Superbasics 4

Section 1 - Rows

Number	...Row..	State	...Activity...	Slack Activity	..Lower Limit.	..Upper Limit.	.Dual Activity	..i
10	..ROW1..	UL	1.50000	.	-2.00000	1.50000	-0.06667	1
11	..ROW2..	UL	1.50000	.	-2.00000	1.50000	-0.03333	2
12	..ROW3..	SBS	3.93333	-0.06667	-2.00000	4.00000	.	3
13	..COST..	BS	-10.78556	-10.78556	None	None	-1.0	4

Section 2 - Columns

Number	.Column.	State	...Activity...	.Obj Gradient.	..Lower Limit.	..Upper Limit.	Reduced Gradnt	m+j
1	...X1...	UL	2.00000	-0.90000	-2.00000	2.00000	-0.80000	5
2	...X2...	SBS	-0.23333	-0.13333	-2.00000	2.00000	.	6
3	...X3...	BS	-0.26667	-0.16667	-2.00000	2.00000	.	7
4	...X4...	BS	-0.30000	-0.20000	-2.00000	2.00000	.	8
5	...X5...	SBS	-0.10000	.	-2.00000	2.00000	.	9
6	...X6...	UL	2.00000	-1.0	-2.00000	2.00000	-0.90000	10
7	...X7...	UL	2.00000	-1.0	-2.00000	2.00000	-0.90000	11
8	...X8...	SBS	-1.77778	-0.10000	-2.00000	2.00000	.	12
9	...X9...	BS	-0.45556	-0.30000	-2.00000	2.00000	.	13