NAG Library Routine Document

E04MWF

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

E04MWF writes data for sparse linear programming, mixed integer linear programming, quadratic programming or mixed integer quadratic programming problems to a file in MPS format.

2 Specification

```
SUBROUTINE E04MWF (OUTFILE, N, M, NNZC, NNZA, NCOLH, NNZH, LINTVAR,        &
                   IDXC, C, IOBJ, A, IROWA, ICCOLA, BL, BU, PNAMES,        &
                   NNAME, CRNAME, H, IROWH, ICCOLH, MINMAX, INTVAR,        &
                   IFAIL)
INTEGER            OUTFILE, N, M, NNZC, NNZA, NCOLH, NNZH, LINTVAR,        &
                   IDXC(NNZC), IOBJ, IROWA(NNZA), ICCOLA(N+1), NNAME,      &
                   IROWH(NNZH), ICCOLH(NCOLH+1), MINMAX,                   &
                   INTVAR(LINTVAR), IFAIL
REAL (KIND=nag_wp) C(NNZC), A(NNZA), BL(N+M), BU(N+M), H(NNZH)
CHARACTER(8)       PNAMES(5), CRNAME(NNAME)
```

3 Description

E04MWF writes data for linear programming (LP) or quadratic programming (QP) problems (or their mixed integer variants) from an optimization problem to a MPS output file, see Section 3.1 in E04MXF for the format description. The problem is expected in the form

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

Where n is the number of variables, m is the number of general linear constraints, A is the linear constraint matrix with dimension m by n, the vectors l and u are the lower and upper bounds, respectively. H is the Hessian matrix with dimension n by n, however, only leading NCOLH columns might contain nonzero elements and the rest is assumed to be zero.

Note that the linear term of the objective function c might be supplied either as C or via IOBJ. If C is supplied then IDXC contains the indices of the nonzero elements of sparse vector c, whereas if IOBJ is supplied (IOBJ > 0), row IOBJ of matrix A is a free row storing the nonzero elements of c.

Note: that this routine uses fixed MPS format, see IBM (1971).

4 References

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

5 Arguments

1: OUTFILE – INTEGER *Input*

On entry: the ID of the file to store the problem data as associated by a call to X04ACF.

Constraint: OUTFILE ≥ 0.

2: N – INTEGER *Input*

On entry: n, the number of variables in the problem.

Constraint: N ≥ 1.

3: M – INTEGER *Input*

On entry: m, the number of constraints in the problem. This is the number of rows in the linear constraint matrix A, including the free row (if any; see IOBJ).

Constraint: M ≥ 0.

4: NNZC – INTEGER *Input*

On entry: the number of nonzero elements in the sparse vector c.

If NNZC $= 0$, the vector c is considered empty and the arrays IDXC and C will not be referenced. In this case the linear term of the objective function, if any, might be provided via IOBJ.

Constraints:

NNZC ≥ 0;
if NNZC > 0, IOBJ $= 0$.

5: NNZA – INTEGER *Input*

On entry: the number of nonzero elements in matrix A.

If NNZA $= 0$, matrix A is considered empty, arrays A and IROWA will not be referenced, and ICCOLA should be the array of 1.

Constraint: NNZA ≥ 0.

6: NCOLH – INTEGER *Input*

On entry: the number of leading nonzero columns of the Hessian matrix H.

If NCOLH $= 0$, the quadratic term H of the objective function is considered zero (e.g., LP problems), and arrays H, IROWH and ICCOLH will not be referenced.

Constraint: $0 \leq$ NCOLH $\leq$ N.

7: NNZH – INTEGER *Input*

On entry: the number of nonzero elements of the Hessian matrix H.

Constraints:

if NCOLH > 0, NNZH > 0;
otherwise NNZH $= 0$.

8: LINTVAR – INTEGER *Input*

On entry: the number of integer variables in the problem.

If LINTVAR $= 0$, all variables are considered continuous and array INTVAR will not be referenced.

Constraint: LINTVAR ≥ 0.

9: IDXC(NNZC) – INTEGER array *Input*
10: C(NNZC) – REAL (KIND=nag_wp) array *Input*

On entry: the nonzero elements of sparse vector c. IDXC(i) must contain the index of C(i) in the vector, for $i = 1, 2, \ldots,$ NNZC.

The elements are stored in ascending order.

Constraints:

$1 \leq \text{IDXC}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZC}$;
$\text{IDXC}(i) < \text{IDXC}(i+1)$, for $i = 1, 2, \ldots, \text{NNZC}$.

11: IOBJ – INTEGER *Input*

On entry: if IOBJ > 0, row IOBJ of A is a free row containing the nonzero coefficients of the linear terms of the objective function. In this case NNZC is set to 0.

If IOBJ $= 0$, there is no free row in A, and the linear terms might be supplied in array C.

Constraint: if IOBJ > 0, NNZC $= 0$.

12: A(NNZA) – REAL (KIND=nag_wp) array *Input*
13: IROWA(NNZA) – INTEGER array *Input*
14: ICCOLA(N + 1) – INTEGER array *Input*

On entry: the nonzero elements of matrix A in compressed column storage (see Section 2.1.3 in the F11 Chapter Introduction). Arrays IROWA and A store the row indices and the values of the nonzero elements, respectively. The elements are sorted by columns and within each column in nondecreasing order. Duplicate entries are not allowed. ICCOLA contains the (one-based) indices to the beginning of each column in A and IROWA.

If NNZA $= 0$, A and IROWA are not referenced.

Constraints:

$1 \leq \text{IROWA}(i) \leq \text{M}$, for $i = 1, 2, \ldots, \text{NNZA}$;
$\text{ICCOLA}(1) = 1$;
$\text{ICCOLA}(i) \leq \text{ICCOLA}(i+1)$, for $i = 1, 2, \ldots, \text{N}$;
$\text{ICCOLA}(\text{N}+1) = \text{NNZA} + 1$.

15: BL(N + M) – REAL (KIND=nag_wp) array *Input*
16: BU(N + M) – REAL (KIND=nag_wp) array *Input*

On entry: BL and BU contains the lower bounds l and the upper bounds u, respectively.

The first N elements refer to the bounds for the variables x and the rest to the bounds for the linear constraints (including the objective row IOBJ if present).

To specify a nonexistent lower bound (i.e., $l_j = -\text{inf}$), set $\text{BL}(j) \leq -10^{20}$; to specify a nonexistent upper bound, set $\text{BU}(j) \geq 10^{20}$.

Constraints:

$\text{BL}(j) \leq \text{BU}(j)$, for $j = 1, 2, \ldots, \text{N} + \text{M}$;
$\text{BL}(j) < 10^{20}$, for $j = 1, 2, \ldots, \text{N} + \text{M}$;
$\text{BU}(j) > -10^{20}$, for $j = 1, 2, \ldots, \text{N} + \text{M}$;
if IOBJ > 0, $\text{BL}(\text{IOBJ} + \text{N}) \leq -10^{20}$ and $\text{BU}(\text{IOBJ} + \text{N}) \geq 10^{20}$.

17: PNAMES(5) – CHARACTER(8) array *Input*

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

The names can be composed only from 'printable' characters (ASCII codes between 32 and 126).

If any of the names are blank, the default name is used.

PNAMES(1)
Contains the name of the problem.

PNAMES(2)
Contains the name of the objective row if the objective is provided in C instead of IOBJ and all names CRNAME are given. The name must be nonempty and unique. In all other cases PNAMES(2) is not used.

PNAMES(3)
Contains the name of the RHS set.

PNAMES(4)
Contains the name of the RANGE.

PNAMES(5)
Contains the name of the BOUNDS.

18: NNAME – INTEGER *Input*

On entry: the number of column (i.e., variable) and row names supplied in the array CRNAME.

If NNAME $= 0$, the names are automatically generated and the array CRNAME is not referenced.

Constraint: NNAME $= 0$ or N + M.

19: CRNAME(NNAME) – CHARACTER(8) array *Input*

On entry: the names of all the variables and constraints in the problem in that order.

The names can be composed only from 'printable' characters and must be unique.

20: H(NNZH) – REAL (KIND=nag_wp) array *Input*

21: IROWH(NNZH) – INTEGER array *Input*

22: ICCOLH(NCOLH + 1) – INTEGER array *Input*

On entry: the nonzero elements of the Hessian matrix H in compressed column storage (see Section 2.1.3 in the F11 Chapter Introduction). The Hessian matrix, H, is symmetric and its elements are stored in a lower triangular matrix.

Arrays IROWH and H store the row indices and the values of the nonzero elements, respectively. The elements are sorted by columns and within each column in nondecreasing order. Duplicate entries are not allowed. ICCOLH contains the (one-based) indices to the beginning of each column in H and IROWH.

If NCOLH $= 0$, H is not referenced.

Constraints:

$1 \leq \text{IROWH}(i) \leq \text{NCOLH}$, for $i = 1, 2, \ldots, \text{NNZH}$;
$\text{ICCOLH}(1) = 1$;
$\text{ICCOLH}(i) \leq \text{ICCOLH}(i+1)$, for $i = 1, 2, \ldots, \text{NCOLH}$;
$\text{ICCOLH}(\text{NCOLH} + 1) = \text{NNZH} + 1$.

23: MINMAX – INTEGER *Input*

On entry: MINMAX defines the direction of optimization problem.

MINMAX $= -1$
Minimization.

MINMAX $= 1$
Maximization.

Constraint: MINMAX $= -1$ or 1.

24: INTVAR(LINTVAR) – INTEGER array *Input*

On entry: INTVAR contains the indices k of variables x_k which are defined as integers. Duplicate indices are not allowed.

If LINTVAR $= 0$, INTVAR is not referenced.

Constraint: $1 \leq \text{INTVAR}(j) \leq \text{N}$, for $j = 1, 2, \ldots, \text{LINTVAR}$.

25: 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 $= 1$

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

IFAIL $= 2$

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

On entry, N $= \langle value \rangle$.
Constraint: N ≥ 1.

IFAIL $= 3$

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

On entry, NNAME $= \langle value \rangle$, N $= \langle value \rangle$ and M $= \langle value \rangle$.
Constraint: NNAME $= 0$ or N $+$ M.

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

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

IFAIL $= 4$

On entry, NCOLH $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: $0 \leq$ NCOLH $\leq$ N.

On entry, NCOLH $= \langle value \rangle$ and NNZH $= \langle value \rangle$.
Constraint: if NCOLH > 0, NNZH > 0.

On entry, NCOLH $= \langle value \rangle$ and NNZH $= \langle value \rangle$.
Constraint: if NCOLH $= 0$, NNZH $= 0$.

IFAIL $= 5$

On entry, $j = \langle value \rangle$, IDXC$(j) = \langle value \rangle$ and IDXC$(j+1) = \langle value \rangle$.
Constraint: IDXC$(j) <$ IDXC$(j+1)$.

On entry, $j = \langle value \rangle$, IDXC$(j) = \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: $1 \leq$ IDXC$(j) \leq$ N.

IFAIL $= 6$

On entry, MINMAX $= \langle value \rangle$.
Constraint: MINMAX $= -1$ or 1.

IFAIL $= 7$

On entry, IOBJ $= \langle value \rangle$ and M $= \langle value \rangle$.
Constraint: $0 \leq$ IOBJ $\leq$ M.

On entry, IOBJ $= \langle value \rangle$ and NNZC $= \langle value \rangle$.
Constraint: at most one of IOBJ or NNZC may be nonzero.

IFAIL $= 8$

On entry, IOBJ $= \langle value \rangle$, BL$(j) = \langle value \rangle$ and BU$(j) = \langle value \rangle$, if IOBJ > 0 the bounds must be infinite.
Constraints: BL$(j) \leq -1\text{E} + 20$, BU$(j) \geq 1\text{E} + 20$.

On entry, $j = \langle value \rangle$, BL$(j) = \langle value \rangle$ and BU$(j) = \langle value \rangle$, the integer variable j requires at least one bound finite.
Constraint: at least one of the following conditions must be met for integer variable j: BL$(j) > -1\text{E} + 20$, BU$(j) < 1\text{E} + 20$.

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

On entry, $j = \langle value \rangle$ and BL$(j) = \langle value \rangle$, BL$(j)$ is incorrect.
Constraint: BL$(j) < 1\text{E} + 20$.

On entry, $j = \langle value \rangle$ and BU$(j) = \langle value \rangle$, BU$(j)$ is incorrect.
Constraint: BU$(j) > -1\text{E} + 20$.

IFAIL $= 9$

On entry, CRNAME(j) for $j = \langle value \rangle$ has been already used.
Constraint: the names in CRNAME must be unique.

On entry, CRNAME(j) for $j = \langle value \rangle$ is incorrect.
Constraint: the names in CRNAME must consist only of printable characters.

On entry, PNAMES(j) for $j = \langle value \rangle$ is incorrect.
Constraint: the names in PNAMES must consist only of printable characters.

The name specified in PNAMES(2) is empty or has been already used among row names.
Constraint: the names in PNAMES(2) must be unique and nonempty if CRNAME is provided and NNZC > 0.

IFAIL $= 10$

On entry, INTVAR($\langle value \rangle$) $=$ INTVAR($\langle value \rangle$) $= \langle value \rangle$.
Constraint: all entries in INTVAR must be unique.

On entry, $j = \langle value \rangle$, INTVAR$(j) = \langle value \rangle$ and LINTVAR $= \langle value \rangle$.
Constraint: $1 \leq$ INTVAR$(j) \leq$ LINTVAR.

IFAIL $= 11$

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

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

IFAIL $= 12$

On entry, ICCOLA$(1) = \langle value \rangle$.
Constraint: ICCOLA$(1) = 1$.

On entry, ICCOLA$(\mathrm{N}+1) = \langle value \rangle$ and NNZA $= \langle value \rangle$.
Constraint: ICCOLA$(\mathrm{N}+1) = \mathrm{NNZA}+1$.

On entry, $j = \langle value \rangle$, ICCOLA$(j) = \langle value \rangle$ and ICCOLA$(j+1) = \langle value \rangle$, the values of ICCOLA must be nondecreasing.
Constraint: ICCOLA$(j) \leq$ ICCOLA$(j+1)$.

IFAIL $= 13$

On entry, $j = \langle value \rangle$, $i = \langle value \rangle$, NCOLH $= \langle value \rangle$ and IROWH$(i) = \langle value \rangle$
Constraint: $j \leq$ IROWH$(i) \leq$ NCOLH (within the lower triangle).

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

IFAIL $= 14$

On entry, ICCOLH$(1) = \langle value \rangle$.
Constraint: ICCOLH$(1) = 1$.

On entry, ICCOLH$(\mathrm{NCOLH}+1) = \langle value \rangle$ and NNZH $= \langle value \rangle$.
Constraint: ICCOLH$(\mathrm{NCOLH}+1) = \mathrm{NNZH}+1$.

On entry, $j = \langle value \rangle$, ICCOLH$(j) = \langle value \rangle$ and ICCOLH$(j+1) = \langle value \rangle$, the values of ICCOLH must be nondecreasing.
Constraint: ICCOLH$(j) \leq$ ICCOLH$(j+1)$.

IFAIL $= 15$

An error occurred when writing to file.

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

E04MWF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example shows how to store an optimization problem to a file in MPS format after it has been solved by E04NQF. The problem is a minimization of the quadratic function $f(x) = c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Hx$, where

$$c = (-200.0, -2000.0, -2000.0, -2000.0, -2000.0, 400.0, 400.0)^{\mathrm{T}}$$

$$H = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 2 & 2 \end{pmatrix}$$

subject to the bounds

$$\begin{aligned} 0 &\leq x_1 \leq 200 \\ 0 &\leq x_2 \leq 2500 \\ 400 &\leq x_3 \leq 800 \\ 100 &\leq x_4 \leq 700 \\ 0 &\leq x_5 \leq 1500 \\ 0 &\leq x_6 \\ 0 &\leq x_7 \end{aligned}$$

and to the linear constraints

$$\begin{array}{rcrcrcrcrcrcrcr} & & x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & + & x_6 & + & x_7 & = & 2000 \\ & & 0.15x_1 & + & 0.04x_2 & + & 0.02x_3 & + & 0.04x_4 & + & 0.02x_5 & + & 0.01x_6 & + & 0.03x_7 & \leq & 60 \\ & & 0.03x_1 & + & 0.05x_2 & + & 0.08x_3 & + & 0.02x_4 & + & 0.06x_5 & + & 0.01x_6 & & & \leq & 100 \\ & & 0.02x_1 & + & 0.04x_2 & + & 0.01x_3 & + & 0.02x_4 & + & 0.02x_5 & & & & & \leq & 40 \\ & & 0.02x_1 & + & 0.03x_2 & & & & & + & 0.01x_5 & & & & & \leq & 30 \\ 1500 & \leq & 0.70x_1 & + & 0.75x_2 & + & 0.80x_3 & + & 0.75x_4 & + & 0.80x_5 & + & 0.97x_6 & & & & \\ 250 & \leq & 0.02x_1 & + & 0.06x_2 & + & 0.08x_3 & + & 0.12x_4 & + & 0.02x_5 & + & 0.01x_6 & + & 0.97x_7 & \leq & 300 \end{array}$$

The initial point, which is infeasible, is

$$x_0 = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)^{\mathrm{T}}.$$

The optimal solution (to five figures) is

$$x^* = (0.0, 349.40, 648.85, 172.85, 407.52, 271.36, 150.02)^{\mathrm{T}}.$$

The generated file is called e04mwfe.mps.

10.1 Program Text

```
!   E04MWF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module e04mwfe_mod

!     E04MWF 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                        :: icol, idx, iend, irow, istart
!       .. Executable Statements ..
        hx(1:ncolh) = 0.0E0_nag_wp

        Do icol = 1, ncolh
          istart = iuser(icol)
          iend = iuser(icol+1) - 1

          Do idx = istart, iend
            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 e04mwfe_mod

    Program e04mwfe

!     E04MWF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e04mwf, e04npf, e04nqf, e04ntf, x04acf, x04adf
      Use e04mwfe_mod, Only: qphx
      Use nag_precisions, Only: wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: lencw = 600, leniw = 600,           &
                                          lenrw = 600, nin = 5, nout = 6,     &
                                          outfile = 42
      Character (*), Parameter         :: outfile_name = 'e04mwfe.mps'
!     .. Local Scalars ..
      Real (Kind=wp)                   :: obj, objadd, sinf
      Integer                          :: i, icol, ierr, ifail, iobj, jcol,   &
                                          lenc, lintvar, m, minmax, n, ncolh, &
                                          ne, ninf, nname, nnzh, ns
      Logical                          :: verbose_output
      Character (1)                    :: istart
      Character (8)                    :: prob
!     .. Local Arrays ..
      Real (Kind=wp), Allocatable      :: acol(:), bl(:), bu(:), c(:), h(:),  &
                                          pi(:), rc(:), ruser(:), x(:)
      Real (Kind=wp)                   :: rw(lenrw)
      Integer, Allocatable             :: helast(:), hs(:), iccolh(:),        &
                                          idxc(:), inda(:), intvar(:),        &
                                          irowh(:), iuser(:), loca(:)
      Integer                          :: iw(leniw)
      Character (8)                    :: cuser(1), cw(lencw), pnames(5)
      Character (8), Allocatable       :: names(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max
```

```
!     .. Executable Statements ..
      Write (nout,*) 'E04MWF Example Program Results'

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

      Read (nin,*) n, m
      Read (nin,*) ne, iobj, ncolh, nnzh, nname
      Allocate (inda(ne),loca(n+1),helast(n+m),hs(n+m),acol(ne),bl(n+m),       &
        bu(n+m),x(n+m),pi(m),rc(n+m),names(nname),h(nnzh),irowh(nnzh),         &
        iccolh(ncolh+1))

!     Read array of names
      Read (nin,*) names(1:nname)

!     Read the matrix ACOL from data file. Set up LOCA.
      jcol = 1
      loca(jcol) = 1

      Do i = 1, ne
!       Element ( INDA( I ), ICOL ) is stored in ACOL( I ).
        Read (nin,*) acol(i), inda(i), icol

        If (icol<jcol) Then
!         Elements not ordered by increasing column index.
          Write (nout,99998) 'Element in column', icol,                        &
            ' found after element in column', jcol, '. Problem', ' abandoned.'
          Go To 100
        Else If (icol==jcol+1) Then
!         Index in ACOL of the start of the ICOL-th column equals I.
          loca(icol) = i
          jcol = icol
        Else If (icol>jcol+1) Then
!         Columns JCOL+1,JCOL+2,...,ICOL-1 are empty. Set the
!         corresponding elements of LOCA to I.
          loca((jcol+1):icol) = i
          jcol = icol
        End If
      End Do

      loca(n+1) = ne + 1

!     Columns N,N-1,...,ICOL+1 are empty. Set the corresponding
!     elements of LOCA accordingly.
      Do i = n, icol + 1, -1
        loca(i) = loca(i+1)
      End Do

!     Read the matrix H from data file. Set up ICCOLH.
      jcol = 1
      iccolh(1) = 1

      Do i = 1, nnzh
!       Element ( IROWH( I ), ICOL ) is stored in H( I ).
        Read (nin,*) h(i), irowh(i), icol

        If (icol<jcol) Then
!         Elements not ordered by increasing column index
          Write (nout,99998) 'Element in column', icol,                        &
            ' found after element in column', jcol, '. Problem', ' abandoned.'
          Go To 100
        Else If (icol==jcol+1) Then
!         Index in ICCOLH of the start of the ICOL-th column equals I.
          iccolh(icol) = i
          jcol = icol
        Else If (icol>jcol+1) Then
!         Columns JCOL+1,JCOL+2,...,ICOL-1 are empty. Set the
!         corresponding elements of ICCOLH to I.
          iccolh((jcol+1):icol) = i
          jcol = icol
        End If
```

```
      End Do

      iccolh(ncolh+1) = nnzh + 1

!     Columns N,N-1,...,ICOL+1 are empty. Set the corresponding
!     elements of ICCOLH accordingly.
      Do i = n, icol + 1, -1
        iccolh(i) = iccolh(i+1)
      End Do

!     Read lower and upper bounds
      Read (nin,*) bl(1:(n+m))
      Read (nin,*) bu(1:(n+m))

!     Set cold start
      istart = 'C'
      hs(1:(n+m)) = 0.0_wp

!     Read the initial point x_0
      Read (nin,*) x(1:n)

!     Print dimensions of the QP problem
      Write (nout,99999) n, m

!     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
!       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)
      End If

!     We have no explicit objective vector so set LENC = 0; the
!     objective vector is stored in row IOBJ of ACOL.
      lenc = 0
      Allocate (c(max(1,lenc)),iuser(ncolh+1+nnzh),ruser(nnzh))

      objadd = 0.0E0_wp
      prob = ' '

!     Do not allow any elastic variables (i.e. they cannot be
!     infeasible). If we'd set optional argument "Elastic mode" to 0,
!     we wouldn't need to set the individual elements of array HELAST.
      helast(1:(n+m)) = 0

      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(istart,qphx,m,n,ne,nname,lenc,ncolh,iobj,objadd,prob,acol,   &
        inda,loca,bl,bu,c,names,helast,hs,x,pi,rc,ns,ninf,sinf,obj,cw,lencw,   &
        iw,leniw,rw,lenrw,cuser,iuser,ruser,ifail)

!     Print objective function and optimal x*
      Write (nout,*)
      Write (nout,99997) obj
```

```
      Write (nout,99996) x(1:n)

!     Set the rest of inputs for e04mwf. Due to IOBJ > 0, LENC = 0 and
!     C and IDXC will not be referenced.
      lintvar = 0
      Allocate (idxc(lenc),intvar(lintvar))
      c(:) = 0
      idxc(:) = 0
      intvar(:) = 0
      pnames(:) = '        '
      pnames(1) = 'USRPNAME'
      pnames(2) = 'OBJ.....'
      minmax = -1

!     Open data file for writing
      ifail = 0
      Call x04acf(outfile,outfile_name,1,ifail)

!     Call e04mwf to store the problem in a MPS file with fixed format.
      ifail = 0
      Call e04mwf(outfile,n,m,lenc,ne,ncolh,nnzh,lintvar,idxc,c,iobj,acol,     &
        inda,loca,bl,bu,pnames,nname,names,h,irowh,iccolh,minmax,intvar,ifail)

      Write (nout,99995) outfile_name

!     Close data file
      ierr = 0
      Call x04adf(outfile,ierr)

100   Continue

99999 Format (1X,/,1X,'QP problem contains ',I3,' variables and ',I3,          &
        ' linear constraints')
99998 Format (1X,A,I5,A,I5,A,A)
99997 Format (1X,'Final objective value = ',1P,E11.3)
99996 Format (1X,'Optimal X = ',7F9.2)
99995 Format (1X,/,1X,'MPS file was written: ',A)

    End Program e04mwfe
```

10.2 Program Data

```
E04MWF Example Program Data
 7  8                      : Values of N and M
48  8  7  9  15            : Values of NNZ, IOBJ, NCOLH, NNZH and NNAME

'...X1...'  '...X2...'  '...X3...'  '...X4...'  '...X5...'
'...X6...'  '...X7...'  '..ROW1..'  '..ROW2..'  '..ROW3..'
'..ROW4..'  '..ROW5..'  '..ROW6..'  '..ROW7..'  '..COST..' : End of array NAMES

    0.02   7   1   : Sparse matrix A, ordered by increasing column index;
    0.02   5   1   : each row contains ACOL(i), INDA(i), ICOL (= column index)
    0.03   3   1   : The row indices may be in any order. In this example
    1.00   1   1   : row 8 defines the linear objective term transpose(C)*X.
    0.70   6   1
    0.02   4   1
    0.15   2   1
 -200.00   8   1
    0.06   7   2
    0.75   6   2
    0.03   5   2
    0.04   4   2
    0.05   3   2
    0.04   2   2
    1.00   1   2
-2000.00   8   2
    0.02   2   3
    1.00   1   3
    0.01   4   3
    0.08   3   3
```

```
    0.08   7   3
    0.80   6   3
-2000.00   8   3
    1.00   1   4
    0.12   7   4
    0.02   3   4
    0.02   4   4
    0.75   6   4
    0.04   2   4
-2000.00   8   4
    0.01   5   5
    0.80   6   5
    0.02   7   5
    1.00   1   5
    0.02   2   5
    0.06   3   5
    0.02   4   5
-2000.00   8   5
    1.00   1   6
    0.01   2   6
    0.01   3   6
    0.97   6   6
    0.01   7   6
  400.00   8   6
    0.97   7   7
    0.03   2   7
    1.00   1   7
  400.00   8   7    : End of matrix A

  2.0      1   1    : Sparse matrix H, stored in a lower triangular matrix,
  2.0      2   2    : ordered by increasing column index; each row contains
  2.0      3   3    : H(i), IROWH(i), ICOL (= column index)
  2.0      4   3    : The row indices may be in any order.
  2.0      4   4
  2.0      5   5
  2.0      6   6
  2.0      7   6
  2.0      7   7    : End of matrix H

 0.0       0.0       4.0E+02   1.0E+02   0.0       0.0
 0.0       2.0E+03  -1.0E+25  -1.0E+25  -1.0E+25  -1.0E+25
 1.5E+03   2.5E+02  -1.0E+25             : End of lower bounds array BL

 2.0E+02   2.5E+03   8.0E+02   7.0E+02  1.5E+03  1.0E+25
 1.0E+25   2.0E+03   6.0E+01   1.0E+02  4.0E+01  3.0E+01
 1.0E+25   3.0E+02   1.0E+25             : End of upper bounds array BU

  0.0  0.0  0.0  0.0  0.0  0.0  0.0       : Initial vector X
```

10.3 Program Results

```
E04MWF Example Program Results

QP problem contains   7 variables and   8 linear constraints

Final objective value =  -1.848E+06
Optimal X =      0.00   349.40   648.85   172.85   407.52   271.36   150.02

MPS file was written: e04mwfe.mps
```