

NAG Library Routine Document

E04MFF/E04MFA

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.

Note: *this routine uses optional parameters to define choices in the problem specification and in the details of the algorithm. If you wish to use default settings for all of the optional parameters, you need only read Sections 1 to 10 of this document. If, however, you wish to reset some or all of the settings please refer to Section 11 for a detailed description of the algorithm, to Section 12 for a detailed description of the specification of the optional parameters and to Section 13 for a detailed description of the monitoring information produced by the routine.*

1 Purpose

E04MFF/E04MFA solves general linear programming problems. It is not intended for large sparse problems.

E04MFA is a version of E04MFF that has additional arguments in order to make it safe for use in multithreaded applications (see Section 5). The initialization routine E04WBF **must** have been called before calling E04MFA.

2 Specification

2.1 Specification for E04MFF

```
SUBROUTINE E04MFF (N, NCLIN, A, LDA, BL, BU, CVEC, ISTATE, X, ITER, OBJ,      &
                  AX, CLAMDA, IWORK, LIWORK, WORK, LWORK, IFAIL)
INTEGER          N, NCLIN, LDA, ISTATE(N+NCLIN), ITER,                      &
                  IWORK(LIWORK), LIWORK, LWORK, IFAIL
REAL (KIND=nag_wp) A(LDA,*), BL(N+NCLIN), BU(N+NCLIN), CVEC(*), X(N),      &
                  OBJ, AX(max(1,NCLIN)), CLAMDA(N+NCLIN), WORK(LWORK)
```

2.2 Specification for E04MFA

```
SUBROUTINE E04MFA (N, NCLIN, A, LDA, BL, BU, CVEC, ISTATE, X, ITER, OBJ,      &
                  AX, CLAMDA, IWORK, LIWORK, WORK, LWORK, LWSAV, IWSAV,      &
                  RWSAV, IFAIL)
INTEGER          N, NCLIN, LDA, ISTATE(N+NCLIN), ITER,                      &
                  IWORK(LIWORK), LIWORK, LWORK, IWSAV(610), IFAIL
REAL (KIND=nag_wp) A(LDA,*), BL(N+NCLIN), BU(N+NCLIN), CVEC(*), X(N),      &
                  OBJ, AX(max(1,NCLIN)), CLAMDA(N+NCLIN),                  &
                  WORK(LWORK), RWSAV(475)
LOGICAL          LWSAV(120)
```

Before calling E04MFA, or either of the option setting routines E04MGA or E04MHA, E04WBF **must** be called. The specification for E04WBF is:

```
SUBROUTINE E04WBF (RNAME, CWSAV, LCWSAV, LWSAV, LLWSAV, IWSAV, LIWSAV,      &
                  RWSAV, LRWSAV, IFAIL)
INTEGER          LCWSAV, LLWSAV, IWSAV(LIWSAV), LIWSAV, LRWSAV,            &
                  IFAIL
REAL (KIND=nag_wp) RWSAV(LRWSAV)
LOGICAL          LWSAV(LLWSAV)
CHARACTER(*)     RNAME
CHARACTER(80)    CWSAV(LCWSAV)
```

E04WBF should be called with RNAME = 'E04MFA'. LCWSAV, LLWSAV, LIWSAV and LRWSAV, the declared lengths of CWSAV, LWSAV, IWSAV and RWSAV respectively, must satisfy:

LCWSAV ≥ 1

LLWSAV ≥ 120

LIWSAV ≥ 610

LRWSAV ≥ 475

The contents of the arrays CWSAV, LWSAV, IWSAV and RWSAV **must not** be altered between calling routines E04MFA, E04MGA, E04MHA and E04WBF.

3 Description

E04MFF/E04MFA is designed to solve linear programming (LP) problems of the form

$$\underset{x \in R^n}{\text{minimize}} c^T x, \quad \text{subject to} \quad l \leq \begin{Bmatrix} x \\ Ax \end{Bmatrix} \leq u,$$

where c is an n -element vector and A is an m_L by n matrix.

This is the default type of problem, referred to as type LP. The optional parameter **Problem Type** may be used to specify an alternative problem type FP, in which the objective function is omitted and the routine attempts to find a feasible point for the set of constraints.

The constraints involving A are called the *general* constraints. Note that upper and lower bounds are specified for all the variables and for all the general constraints. An *equality* constraint can be specified by setting $l_i = u_i$. If certain bounds are not present, the associated elements of l or u can be set to special values that will be treated as $-\infty$ or $+\infty$. (See the description of the optional parameter **Infinite Bound Size**.)

You must supply an initial estimate of the solution.

The method used by E04MFF/E04MFA is described in detail in Section 11.

4 References

Gill P E, Hammarling S, Murray W, Saunders M A and Wright M H (1986) Users' guide for LSSOL (Version 1.0) *Report SOL 86-1* Department of Operations Research, Stanford University

Gill P E and Murray W (1978) Numerically stable methods for quadratic programming *Math. Programming* **14** 349–372

Gill P E, Murray W, Saunders M A and Wright M H (1984) Procedures for optimization problems with a mixture of bounds and general linear constraints *ACM Trans. Math. Software* **10** 282–298

Gill P E, Murray W, Saunders M A and Wright M H (1989) A practical anti-cycling procedure for linearly constrained optimization *Math. Programming* **45** 437–474

Gill P E, Murray W, Saunders M A and Wright M H (1991) Inertia-controlling methods for general quadratic programming *SIAM Rev.* **33** 1–36

Gill P E, Murray W and Wright M H (1981) *Practical Optimization* Academic Press

5 Arguments

- | | | |
|----|--|--------------|
| 1: | N – INTEGER | <i>Input</i> |
| | <i>On entry:</i> n , the number of variables. | |
| | <i>Constraint:</i> $N > 0$. | |
| 2: | NCLIN – INTEGER | <i>Input</i> |
| | <i>On entry:</i> m_L , the number of general linear constraints. | |
| | <i>Constraint:</i> $NCLIN \geq 0$. | |

- 3: A(LDA,*) – REAL (KIND=nag_wp) array Input
Note: the second dimension of the array A must be at least N if NCLIN > 0 and at least 1 if NCLIN = 0.
On entry: the i th row of A must contain the coefficients of the i th general linear constraint, for $i = 1, 2, \dots, m_L$.
 If NCLIN = 0, A is not referenced.
- 4: LDA – INTEGER Input
On entry: the first dimension of the array A as declared in the (sub)program from which E04MFF/E04MFA is called.
Constraint: LDA $\geq \max(1, \text{NCLIN})$.
- 5: BL(N + NCLIN) – REAL (KIND=nag_wp) array Input
 6: BU(N + NCLIN) – REAL (KIND=nag_wp) array Input
On entry: must contain the lower bounds and BU the upper bounds, for all the constraints in the following order. The first n elements of each array must contain the bounds on the variables, and the next m_L elements the bounds for the general linear constraints (if any). To specify a nonexistent lower bound (i.e., $l_j = -\infty$), set $\text{BL}(j) \leq -\text{bigbnd}$, and to specify a nonexistent upper bound (i.e., $u_j = +\infty$), set $\text{BU}(j) \geq \text{bigbnd}$; the default value of bigbnd is 10^{20} , but this may be changed by the optional parameter **Infinite Bound Size**. To specify the j th constraint as an equality, set $\text{BL}(j) = \text{BU}(j) = \beta$, say, where $|\beta| < \text{bigbnd}$.
Constraints:
 $\text{BL}(j) \leq \text{BU}(j)$, for $j = 1, 2, \dots, N + \text{NCLIN}$;
 if $\text{BL}(j) = \text{BU}(j) = \beta$, $|\beta| < \text{bigbnd}$.
- 7: CVEC(*) – REAL (KIND=nag_wp) array Input
Note: the dimension of the array CVEC must be at least N if the problem is of type LP (the default), and at least 1 otherwise.
On entry: the coefficients of the objective function when the problem is of type LP.
 If the problem is of type FP, CVEC is not referenced.
- 8: ISTATE(N + NCLIN) – INTEGER array Input/Output
On entry: need not be set if the (default) optional parameter **Cold Start** is used.
 If the optional parameter **Warm Start** has been chosen, ISTATE specifies the desired status of the constraints at the start of the feasibility phase. More precisely, the first n elements of ISTATE refer to the upper and lower bounds on the variables, and the next m_L elements refer to the general linear constraints (if any). Possible values for ISTATE(j) are as follows:
- | ISTATE(j) | Meaning |
|---------------|---|
| 0 | The corresponding constraint should <i>not</i> be in the initial working set. |
| 1 | The constraint should be in the initial working set at its lower bound. |
| 2 | The constraint should be in the initial working set at its upper bound. |
| 3 | The constraint should be in the initial working set as an equality. This value must not be specified unless $\text{BL}(j) = \text{BU}(j)$. |

The values -2 , -1 and 4 are also acceptable but will be reset to zero by the routine. If E04MFF/E04MFA has been called previously with the same values of N and NCLIN, ISTATE already contains satisfactory information. (See also the description of the optional parameter **Warm**

Start.) The routine also adjusts (if necessary) the values supplied in X to be consistent with ISTATE.

Constraint: $-2 \leq \text{ISTATE}(j) \leq 4$, for $j = 1, 2, \dots, N + \text{NCLIN}$.

On exit: the status of the constraints in the working set at the point returned in X. The significance of each possible value of $\text{ISTATE}(j)$ is as follows:

$\text{ISTATE}(j)$	Meaning
-2	The constraint violates its lower bound by more than the feasibility tolerance.
-1	The constraint violates its upper bound by more than the feasibility tolerance.
0	The constraint is satisfied to within the feasibility tolerance, but is not in the working set.
1	This inequality constraint is included in the working set at its lower bound.
2	This inequality constraint is included in the working set at its upper bound.
3	This constraint is included in the working set as an equality. This value of ISTATE can occur only when $\text{BL}(j) = \text{BU}(j)$.
4	This corresponds to optimality being declared with $X(j)$ being temporarily fixed at its current value. This value of ISTATE can occur only when $\text{IFAIL} = 1$ on exit.

9: X(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: an initial estimate of the solution.

On exit: the point at which E04MFF/E04MFA terminated. If $\text{IFAIL} = 0, 1$ or 4 , X contains an estimate of the solution.

10: ITER – INTEGER *Output*

On exit: the total number of iterations performed.

11: OBJ – REAL (KIND=nag_wp) *Output*

On exit: the value of the objective function at x if x is feasible, or the sum of infeasibilities at x otherwise. If the problem is of type FP and x is feasible, OBJ is set to zero.

12: AX(max(1, NCLIN)) – REAL (KIND=nag_wp) array *Output*

On exit: the final values of the linear constraints Ax .

If $\text{NCLIN} = 0$, AX is not referenced.

13: CLAMDA(N + NCLIN) – REAL (KIND=nag_wp) array *Output*

On exit: the values of the Lagrange multipliers for each constraint with respect to the current working set. The first n elements contain the multipliers for the bound constraints on the variables, and the next m_L elements contain the multipliers for the general linear constraints (if any). If $\text{ISTATE}(j) = 0$ (i.e., constraint j is not in the working set), $\text{CLAMDA}(j)$ is zero. If x is optimal, $\text{CLAMDA}(j)$ should be non-negative if $\text{ISTATE}(j) = 1$, non-positive if $\text{ISTATE}(j) = 2$ and zero if $\text{ISTATE}(j) = 4$.

14: IWORK(LIWORK) – INTEGER array *Workspace*

15: LIWORK – INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which E04MFF/E04MFA is called.

Constraint: $\text{LIWORK} \geq 2 \times N + 3$.

- 16: WORK(LWORK) – REAL (KIND=nag_wp) array Workspace
 17: LWORK – INTEGER Input

On entry: the dimension of the array WORK as declared in the (sub)program from which E04MFF/E04MFA is called.

Constraints:

if the problem is of type LP (the default),
 if NCLIN = 0, $LWORK \geq 7 \times N + 1$;
 if $NCLIN \geq N$, $LWORK \geq 2 \times N^2 + 7 \times N + 5 \times NCLIN$;
 otherwise $LWORK \geq 2 \times (NCLIN + 1)^2 + 7 \times N + 5 \times NCLIN$.;
 if the problem is of type FP,
 if NCLIN = 0, $LWORK \geq 6 \times N + 1$;
 if $NCLIN \geq N$, $LWORK \geq 2 \times N^2 + 6 \times N + 5 \times NCLIN$;
 otherwise $LWORK \geq 2 \times (NCLIN + 1)^2 + 6 \times N + 5 \times NCLIN$..

The amounts of workspace provided and required are (by default) output on the current advisory message unit (as defined by X04ABF). As an alternative to computing LIWORK and LWORK from the formulas given above, you may prefer to obtain appropriate values from the output of a preliminary run with LIWORK and LWORK set to 1. (E04MFF/E04MFA will then terminate with IFAIL = 6.)

- 18: IFAIL – INTEGER Input/Output

Note: for E04MFA, IFAIL does not occur in this position in the argument list. See the additional arguments described below.

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, because for this routine the values of the output arguments may be useful even if $IFAIL \neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

E04MFF/E04MFA returns with IFAIL = 0 if x is a strong local minimizer, i.e., the reduced gradient (Norm Gz; see Section 9.2) is negligible and the Lagrange multipliers (Lagr Mult; see Section 9.2) are optimal.

Note: the following are additional arguments for specific use with E04MFA. Users of E04MFF therefore need not read the remainder of this description.

- 19: LWSAV(120) – LOGICAL array Communication Array
 20: IWSAV(610) – INTEGER array Communication Array
 21: RWSAV(475) – REAL (KIND=nag_wp) array Communication Array

The arrays LWSAV, IWSAV and RWSAV **must not** be altered between calls to any of the routines E04MFA, E04MGA, E04MHA or E04WBF.

- 22: IFAIL – INTEGER Input/Output

Note: see the argument description for IFAIL above.

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).

Note: E04MFF/E04MFA may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL = 1

x is a weak local minimum (the projected gradient is negligible and the Lagrange multipliers are optimal but there is a small multiplier). This means that the solution is not unique.

IFAIL = 2

The solution appears to be unbounded, i.e., the objective function is not bounded below in the feasible region. This value of IFAIL occurs if a step larger than **Infinite Step Size** (default value = 10^{20}) would have to be taken in order to continue the algorithm, or the next step would result in an element of x having magnitude larger than optional parameter **Infinite Bound Size** (default value = 10^{20}).

IFAIL = 3

No feasible point was found, i.e., it was not possible to satisfy all the constraints to within the feasibility tolerance. In this case, the constraint violations at the final x will reveal a value of the tolerance for which a feasible point will exist – for example, when the feasibility tolerance for each violated constraint exceeds its Slack (see Section 9.2) at the final point. The modified problem (with an altered feasibility tolerance) may then be solved using a **Warm Start**. You should check that there are no constraint redundancies. If the data for the constraints are accurate only to the absolute precision σ , you should ensure that the value of the optional parameter **Feasibility Tolerance** (default value = $\sqrt{\epsilon}$, where ϵ is the *machine precision*) is *greater* than σ . For example, if all elements of A are of order unity and are accurate only to three decimal places, the **Feasibility Tolerance** should be at least 10^{-3} .

IFAIL = 4

The limiting number of iterations was reached before normal termination occurred.

The value of the optional parameter **Iteration Limit** (default value = $\max(50, 5(n + m_L))$) may be too small. If the method appears to be making progress (e.g., the objective function is being satisfactorily reduced), either rerun E04MFF/E04MFA with a larger value of **Iteration Limit** or, alternatively, rerun E04MFF/E04MFA using the **Warm Start** facility to specify the initial working set.

IFAIL = 5

Not used by this routine.

IFAIL = 6

An input argument is invalid.

IFAIL = 7

The designated problem type was not FP or LP. Rerun E04MFF/E04MFA with the optional parameter **Problem Type** set to one of these values.

Overflow

If the printed output before the overflow error contains a warning about serious ill-conditioning in the working set when adding the j th constraint, it may be possible to avoid the difficulty by increasing the magnitude of the **Feasibility Tolerance** (default value = $\sqrt{\epsilon}$, where ϵ is the

machine precision) and rerunning the program. If the message recurs even after this change, the offending linearly dependent constraint (with index ‘*j*’) must be removed from the problem.

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

E04MFF/E04MFA implements a numerically stable active set strategy and returns solutions that are as accurate as the condition of the problem warrants on the machine.

8 Parallelism and Performance

E04MFF/E04MFA 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

This section contains some comments on scaling and a description of the printed output.

9.1 Scaling

Sensible scaling of the problem is likely to reduce the number of iterations required and make the problem less sensitive to perturbations in the data, thus improving the condition of the problem. In the absence of better information it is usually sensible to make the Euclidean lengths of each constraint of comparable magnitude. See the E04 Chapter Introduction and Gill *et al.* (1981) for further information and advice.

9.2 Description of the Printed Output

This section describes the intermediate printout and final printout produced by E04MFF/E04MFA. The intermediate printout is a subset of the monitoring information produced by the routine at every iteration (see Section 13). You can control the level of printed output (see the description of the optional parameter **Print Level**). Note that the intermediate printout and final printout are produced only if **Print Level** ≥ 10 (the default for E04MFF, by default no output is produced by E04MFA).

The following line of summary output (< 80 characters) is produced at every iteration. In all cases, the values of the quantities printed are those in effect *on completion* of the given iteration.

Itn is the iteration count.

Step is the step taken along the computed search direction. If a constraint is added during the current iteration, Step will be the step to the nearest constraint. When

	the problem is of type LP, the step can be greater than one during the optimality phase.
Ninf	is the number of violated constraints (infeasibilities). This will be zero during the optimality phase.
Sinf/Objective	is the value of the current objective function. If x is not feasible, Sinf gives a weighted sum of the magnitudes of constraint violations. If x is feasible, Objective is the value of the objective function of (1). The output line for the final iteration of the feasibility phase (i.e., the first iteration for which Ninf is zero) will give the value of the true objective at the first feasible point. During the optimality phase the value of the objective function will be nonincreasing. During the feasibility phase the number of constraint infeasibilities will not increase until either a feasible point is found or the optimality of the multipliers implies that no feasible point exists. Once optimal multipliers are obtained the number of infeasibilities can increase, but the sum of infeasibilities will either remain constant or be reduced until the minimum sum of infeasibilities is found.
Norm Gz	is $\ Z_R^T g_{FR}\ $, the Euclidean norm of the reduced gradient with respect to Z_R . During the optimality phase, this norm will be approximately zero after a unit step. (See Sections 11.2 and 11.4.)
The final printout includes a listing of the status of every variable and constraint.	
The following describes the printout for each variable. A full stop (.) is printed for any numerical value that is zero.	
Varbl	gives the name (v) and index j , for $j = 1, 2, \dots, n$, of the variable.
State	gives the state of the variable (FR if neither bound is in the working set, EQ if a fixed variable, LL if on its lower bound, UL if on its upper bound, TF if temporarily fixed at its current value). If Value lies outside the upper or lower bounds by more than the Feasibility Tolerance , State will be ++ or -- respectively. A key is sometimes printed before State. A <i>Alternative optimum possible</i> . The variable is active at one of its bounds, but its Lagrange multiplier is essentially zero. This means that if the variable were allowed to start moving away from its bound then there would be no change to the objective function. The values of the other free variables <i>might</i> change, giving a genuine alternative solution. However, if there are any degenerate variables (labelled D), the actual change might prove to be zero, since one of them could encounter a bound immediately. In either case the values of the Lagrange multipliers might also change. D <i>Degenerate</i> . The variable is free, but it is equal to (or very close to) one of its bounds. I <i>Infeasible</i> . The variable is currently violating one of its bounds by more than the Feasibility Tolerance .
Value	is the value of the variable at the final iteration.
Lower Bound	is the lower bound specified for the variable. None indicates that $BL(j) \leq -bigbnd$.
Upper Bound	is the upper bound specified for the variable. None indicates that $BU(j) \geq bigbnd$.
Lagr Mult	is the Lagrange multiplier for the associated bound. This will be zero if State is FR unless $BL(j) \leq -bigbnd$ and $BU(j) \geq bigbnd$, in which case the entry will be blank. If x is optimal, the multiplier should be non-negative if State is LL and non-positive if State is UL.

Slack is the difference between the variable Value and the nearer of its (finite) bounds BL(j) and BU(j). A blank entry indicates that the associated variable is not bounded (i.e., BL(j) \leq -bigbnd and BU(j) \geq bigbnd).

The meaning of the printout for general constraints is the same as that given above for variables, with ‘variable’ replaced by ‘constraint’, BL(j) and BU(j) are replaced by BL($n + j$) and BU($n + j$) respectively, and with the following change in the heading:

L Con gives the name (L) and index j , for $j = 1, 2, \dots, n_L$, of the linear constraint.

Note that movement off a constraint (as opposed to a variable moving away from its bound) can be interpreted as allowing the entry in the Slack column to become positive.

Numerical values are output with a fixed number of digits; they are not guaranteed to be accurate to this precision.

10 Example

This example minimizes the function

$$-0.02x_1 - 0.2x_2 - 0.2x_3 - 0.2x_4 - 0.2x_5 + 0.04x_6 + 0.04x_7$$

subject to the bounds

$$\begin{aligned} -0.01 &\leq x_1 \leq 0.01 \\ -0.1 &\leq x_2 \leq 0.15 \\ -0.01 &\leq x_3 \leq 0.03 \\ -0.04 &\leq x_4 \leq 0.02 \\ -0.1 &\leq x_5 \leq 0.05 \\ -0.01 &\leq x_6 \\ -0.01 &\leq x_7 \end{aligned}$$

and the general constraints

$$\begin{array}{rcll} & x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & + & x_6 & + & x_7 & = & -0.13 \\ 0.15x_1 & + & 0.04x_2 & + & 0.02x_3 & + & 0.04x_4 & + & 0.02x_5 & + & 0.01x_6 & + & 0.03x_7 & \leq & -0.0049 \\ 0.03x_1 & + & 0.05x_2 & + & 0.08x_3 & + & 0.02x_4 & + & 0.06x_5 & + & 0.01x_6 & & & \leq & -0.0064 \\ 0.02x_1 & + & 0.04x_2 & + & 0.01x_3 & + & 0.02x_4 & + & 0.02x_5 & & & & & \leq & -0.0037 \\ 0.02x_1 & + & 0.03x_2 & & & & & & 0.01x_5 & & & & & \leq & -0.0012 \\ -0.0992 & \leq & 0.70x_1 & + & 0.75x_2 & + & 0.80x_3 & + & 0.75x_4 & + & 0.80x_5 & + & 0.97x_6 & & & \\ -0.003 & \leq & 0.02x_1 & + & 0.06x_2 & + & 0.08x_3 & + & 0.12x_4 & + & 0.02x_5 & + & 0.01x_6 & + & 0.97x_7 & \leq & 0.002 \end{array}$$

The initial point, which is infeasible, is

$$x_0 = (-0.01, -0.03, 0.0, -0.01, -0.1, 0.02, 0.01)^T.$$

The optimal solution (to five figures) is

$$x^* = (-0.01, -0.1, 0.03, 0.02, -0.067485, -0.0022801, -0.00023453)^T.$$

Four bound constraints and three general constraints are active at the solution.

The document for E04MGF/E04MGA includes an example program to solve the same problem using some of the optional parameters described in Section 12.

10.1 Program Text

the following program illustrates the use of E04MFF. An equivalent program illustrating the use of E04MFA is available with the supplied Library and is also available from the NAG web site.

```

Program e04mffe

!      E04MFF Example Program Text

!      Mark 26 Release. NAG Copyright 2016.

!      .. Use Statements ..
```

```

      Use nag_library, Only: e04mff, e04mhf, nag_wp
!      .. Implicit None Statement ..
      Implicit None
!      .. Parameters ..
      Integer, Parameter          :: nin = 5, nout = 6
!      .. Local Scalars ..
      Real (Kind=nag_wp)          :: obj
      Integer                     :: i, ifail, iter, j, lda, liwork,      &
                                   lwork, n, nclin, sda
      Logical                     :: verbose_output
!      .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable :: a(:, :), ax(:, :), bl(:, :), bu(:, :),      &
                                   clamda(:, :), cvec(:, :), work(:, :), x(:, :),
      Integer, Allocatable         :: istate(:, :), iwork(:, :),
!      .. Intrinsic Procedures ..
      Intrinsic                   :: max
!      .. Executable Statements ..
      Write (nout,*) 'E04MFF Example Program Results'

!      Skip heading in data file

      Read (nin,*)
      Read (nin,*) n, nclin
      liwork = 2*n + 3

!      The minimum LWORK for an LP problem:

      If (0 < nclin .And. nclin < n) Then
         lwork = 2*(nclin+1)**2 + 7*n + 5*nclin
      Else If (nclin >= n) Then
         lwork = 2*n**2 + 7*n + 5*nclin
      Else
         lwork = 7*n + 1
      End If

      lda = max(1, nclin)

      If (nclin > 0) Then
         sda = n
      Else
         sda = 1
      End If

      Allocate (istate(n+nclin), iwork(liwork), a(lda, sda), bl(n+nclin),      &
               bu(n+nclin), cvec(n), x(n), ax(max(1, nclin)), clamda(n+nclin), work(lwork))

      Read (nin,*) cvec(1:n)
      Read (nin,*) (a(i, 1:sda), i=1, nclin)
      Read (nin,*) bl(1:(n+nclin))
      Read (nin,*) bu(1:(n+nclin))
      Read (nin,*) x(1:n)

!      Set this to .True. to cause e04nqf to produce intermediate
!      progress output
      verbose_output = .False.

      If (.Not. verbose_output) Then
!         Turn off intermediate output from e04mff - it is on by default
         Call e04mhf('Nolist')
         Call e04mhf('Print Level = 0')
      End If

!      Solve the problem

      ifail = 0
      Call e04mff(n, nclin, a, lda, bl, bu, cvec, istate, x, iter, obj, ax, clamda, iwork,      &
                 liwork, work, lwork, ifail)

      Select Case (ifail)
      Case (0:5, 7:)
         Write (nout,*)

```

```

Write (nout,99999)
Do i = 1, n
  Write (nout,99998) i, istate(i), x(i), clamda(i)
End Do

If (nclin>0) Then
  Write (nout,*)
  Write (nout,99997)
  Do i = n + 1, n + nclin
    j = i - n
    Write (nout,99996) j, istate(i), ax(j), clamda(i)
  End Do
End If

Write (nout,*)
Write (nout,99995) obj
End Select

99999 Format (1X,'Varbl',3X,'Istate',4X,'Value',9X,'Lagr Mult')
99998 Format (1X,'V',2(1X,I3),4X,1P,E14.3,2X,1P,E12.3)
99997 Format (1X,'L Con',3X,'Istate',4X,'Value',9X,'Lagr Mult')
99996 Format (1X,'L',2(1X,I3),4X,1P,E14.3,2X,1P,E12.3)
99995 Format (1X,'Final objective value = ',1P,E15.3)
End Program e04mffe

```

10.2 Program Data

E04MFF Example Program Data

```

7 7 :Values of N and NCLIN
-0.02 -0.20 -0.20 -0.20 -0.20 0.04 0.04 :End of CVEC
1.00 1.00 1.00 1.00 1.00 1.00 1.00
0.15 0.04 0.02 0.04 0.02 0.01 0.03
0.03 0.05 0.08 0.02 0.06 0.01 0.00
0.02 0.04 0.01 0.02 0.02 0.00 0.00
0.02 0.03 0.00 0.00 0.01 0.00 0.00
0.70 0.75 0.80 0.75 0.80 0.97 0.00
0.02 0.06 0.08 0.12 0.02 0.01 0.97 :End of matrix A
-0.01 -0.10 -0.01 -0.04 -0.10 -0.01 -0.01
-0.13 -1.0D+25 -1.0D+25 -1.0D+25 -1.0D+25 -9.92D-02 -3.0D-03 :End of BL
0.01 0.15 0.03 0.02 0.05 1.0D+25 1.0D+25
-0.13 -4.9D-03 -6.4D-03 -3.7D-03 -1.2D-03 1.0D+25 2.0D-03 :End of BU
-0.01 -0.03 0.00 -0.01 -0.10 0.02 0.01 :End of X

```

10.3 Program Results

E04MFF Example Program Results

Varbl	Istate	Value	Lagr Mult
V 1	1	-1.000E-02	3.301E-01
V 2	1	-1.000E-01	1.438E-02
V 3	2	3.000E-02	-9.100E-02
V 4	2	2.000E-02	-7.661E-02
V 5	0	-6.749E-02	0.000E+00
V 6	0	-2.280E-03	0.000E+00
V 7	0	-2.345E-04	0.000E+00

L Con	Istate	Value	Lagr Mult
L 1	3	-1.300E-01	-1.431E+00
L 2	0	-5.480E-03	0.000E+00
L 3	0	-6.572E-03	0.000E+00
L 4	0	-4.850E-03	0.000E+00
L 5	0	-3.875E-03	0.000E+00
L 6	1	-9.920E-02	1.501E+00
L 7	1	-3.000E-03	1.517E+00

Final objective value = 2.360E-02

Note: the remainder of this document is intended for more advanced users. Section 11 contains a detailed description of the algorithm which may be needed in order to understand Sections 12 and 13. Section 12 describes the optional parameters which may be set by calls to E04MGF/E04MGA and/or E04MHF/E04MHA. Section 13 describes the quantities which can be requested to monitor the course of the computation.

11 Algorithmic Details

This section contains a detailed description of the method used by E04MFF/E04MFA.

11.1 Overview

E04MFF/E04MFA is based on an inertia-controlling method due to Gill and Murray (1978), and is described in detail by Gill *et al.* (1991). Here we briefly summarise the main features of the method. Where possible, explicit reference is made to the names of variables that are arguments of E04MFF/E04MFA or appear in the printed output. E04MFF/E04MFA has two phases: finding an initial feasible point by minimizing the sum of infeasibilities (the *feasibility phase*), and minimizing the linear objective function within the feasible region (the *optimality phase*). The computations in both phases are performed by the same subroutines. The two-phase nature of the algorithm is reflected by changing the function being minimized from the sum of infeasibilities to the linear objective function. The feasibility phase does *not* perform the standard simplex method (i.e., it does not necessarily find a vertex), except in the case when $m_L \leq n$. Once any iterate is feasible, all subsequent iterates remain feasible.

In general, an iterative process is required to solve a linear program. (For simplicity, we shall always consider a typical iteration and avoid reference to the index of the iteration.) Each new iterate \bar{x} is defined by

$$\bar{x} = x + \alpha p, \quad (1)$$

where the *step length* α is a non-negative scalar, and p is called the *search direction*.

At each point x , a *working set* of constraints is defined to be a linearly independent subset of the constraints that are satisfied ‘exactly’ (to within the tolerance defined by the optional parameter **Feasibility Tolerance**). The working set is the current prediction of the constraints that hold with equality at a solution of an LP problem. The search direction is constructed so that the constraints in the working set remain *unaltered* for any value of the step length. For a bound constraint in the working set, this property is achieved by setting the corresponding element of the search direction to zero. Thus, the associated variable is *fixed*, and specification of the working set induces a partition of x into *fixed* and *free* variables. During a given iteration, the fixed variables are effectively removed from the problem; since the relevant elements of the search direction are zero, the columns of A corresponding to fixed variables may be ignored.

Let m_W denote the number of general constraints in the working set and let n_{FX} denote the number of variables fixed at one of their bounds (m_W and n_{FX} are the quantities Lin and Bnd in the monitoring file output from E04MFF/E04MFA; see Section 13). Similarly, let n_{FR} ($n_{FR} = n - n_{FX}$) denote the number of free variables. At every iteration, *the variables are reordered so that the last n_{FX} variables are fixed*, with all other relevant vectors and matrices ordered accordingly.

11.2 Definition of Search Direction

Let A_{FR} denote the m_W by n_{FR} sub-matrix of general constraints in the working set corresponding to the free variables, and let p_{FR} denote the search direction with respect to the free variables only. The general constraints in the working set will be unaltered by any move along p if

$$A_{FR} p_{FR} = 0. \quad (2)$$

In order to compute p_{FR} , the *TQ factorization* of A_{FR} is used:

$$A_{FR} Q_{FR} = (0 \quad T), \quad (3)$$

where T is a nonsingular m_W by m_W upper triangular matrix (i.e., $t_{ij} = 0$ if $i > j$), and the nonsingular n_{FR} by n_{FR} matrix Q_{FR} is the product of orthogonal transformations (see Gill *et al.* (1984)). If the

columns of Q_{FR} are partitioned so that

$$Q_{FR} = (Z \ Y),$$

where Y is n_{FR} by m_W , then the n_Z ($n_Z = n_{FR} - m_W$) columns of Z form a basis for the null space of A_{FR} . Let n_R be an integer such that $0 \leq n_R \leq n_Z$, and let Z_R denote a matrix whose n_R columns are a subset of the columns of Z . (The integer n_R is the quantity `Zr` in the monitoring file output from E04MFF/E04MFA. In many cases, Z_R will include *all* the columns of Z .) The direction p_{FR} will satisfy (2) if

$$p_{FR} = Z_R p_R \quad (4)$$

where p_R is any n_R -vector.

11.3 Main Iteration

Let Q denote the n by n matrix

$$Q = \begin{pmatrix} Q_{FR} & \\ & I_{FX} \end{pmatrix},$$

where I_{FX} is the identity matrix of order n_{FX} . Let g_Q denote the *transformed gradient*

$$g_Q = Q^T c$$

and let the vector of the first n_R elements of g_Q be denoted by g_R . The quantity g_R is known as the *reduced gradient* of $c^T x$. If the reduced gradient is zero, x is a constrained stationary point in the subspace defined by Z . During the feasibility phase, the reduced gradient will usually be zero only at a vertex (although it may be zero at non-vertices in the presence of constraint dependencies). During the optimality phase, a zero reduced gradient implies that x minimizes the linear objective when the constraints in the working set are treated as equalities. At a constrained stationary point, Lagrange multipliers λ_C and λ_B for the general and bound constraints are defined from the equations

$$A_{FR}^T \lambda_C = g_{FR} \quad \text{and} \quad \lambda_B = g_{FX} - A_{FX}^T \lambda_C. \quad (5)$$

Given a positive constant δ of the order of the **machine precision**, a Lagrange multiplier λ_j corresponding to an inequality constraint in the working set is said to be *optimal* if $\lambda_j \leq \delta$ when the associated constraint is at its *upper bound*, or if $\lambda_j \geq -\delta$ when the associated constraint is at its *lower bound*. If a multiplier is nonoptimal, the objective function (either the true objective or the sum of infeasibilities) can be reduced by deleting the corresponding constraint (with index `Jdel`; see Section 13) from the working set.

If optimal multipliers occur during the feasibility phase and the sum of infeasibilities is nonzero, there is no feasible point, and you can force E04MFF/E04MFA to continue until the minimum value of the sum of infeasibilities has been found; see the discussion of the optional parameter **Minimum Sum of Infeasibilities**. At such a point, the Lagrange multiplier λ_j corresponding to an inequality constraint in the working set will be such that $-(1 + \delta) \leq \lambda_j \leq \delta$ when the associated constraint is at its *upper bound*, and $-\delta \leq \lambda_j \leq (1 + \delta)$ when the associated constraint is at its *lower bound*. Lagrange multipliers for equality constraints will satisfy $|\lambda_j| \leq 1 + \delta$.

If the reduced gradient is not zero, Lagrange multipliers need not be computed and the nonzero elements of the search direction p are given by $Z_R p_R$. The choice of step length is influenced by the need to maintain feasibility with respect to the satisfied constraints.

Each change in the working set leads to a simple change to A_{FR} : if the status of a general constraint changes, a *row* of A_{FR} is altered; if a bound constraint enters or leaves the working set, a *column* of A_{FR} changes. Explicit representations are recurred of the matrices T and Q_{FR} ; and of vectors $Q^T g$, and $Q^T c$.

One of the most important features of E04MFF/E04MFA is its control of the conditioning of the working set, whose nearness to linear dependence is estimated by the ratio of the largest to smallest diagonal elements of the TQ factor T (the printed value `Cond T`; see Section 13). In constructing the initial working set, constraints are excluded that would result in a large value of `Cond T`.

E04MFF/E04MFA includes a rigorous procedure that prevents the possibility of cycling at a point where the active constraints are nearly linearly dependent (see Gill *et al.* (1989)). The main feature of the anti-cycling procedure is that the feasibility tolerance is increased slightly at the start of every iteration. This not only allows a positive step to be taken at every iteration, but also provides, whenever possible, a *choice* of constraints to be added to the working set. Let α_M denote the maximum step at which $x + \alpha_M p$ does not violate any constraint by more than its feasibility tolerance. All constraints at a distance α ($\alpha \leq \alpha_M$) along p from the current point are then viewed as acceptable candidates for inclusion in the working set. The constraint whose normal makes the largest angle with the search direction is added to the working set.

11.4 Choosing the Initial Working Set

Let Z be partitioned as $Z = (Z_R Z_A)$. A working set for which Z_R defines the null space can be obtained by including the rows of Z_A^T as ‘artificial constraints’. Minimization of the objective function then proceeds within the subspace defined by Z_R , as described in Section 11.2.

The artificially augmented working set is given by

$$\bar{A}_{FR} = \begin{pmatrix} Z_A^T \\ A_{FR} \end{pmatrix}, \quad (6)$$

so that p_{FR} will satisfy $A_{FR} p_{FR} = 0$ and $Z_A^T p_{FR} = 0$. By definition of the TQ factorization, \bar{A}_{FR} automatically satisfies the following:

$$\bar{A}_{FR} Q_{FR} = \begin{pmatrix} Z_A^T \\ A_{FR} \end{pmatrix} Q_{FR} = \begin{pmatrix} Z_A^T \\ A_{FR} \end{pmatrix} (Z_R \ Z_A \ Y) = \begin{pmatrix} 0 & \bar{T} \end{pmatrix},$$

where

$$\bar{T} = \begin{pmatrix} I & 0 \\ 0 & T \end{pmatrix},$$

and hence the TQ factorization of (6) is available trivially from T and Q_{FR} without additional expense.

The matrix Z_A is not kept fixed, since its role is purely to define an appropriate null space; the TQ factorization can therefore be updated in the normal fashion as the iterations proceed. No work is required to ‘delete’ the artificial constraints associated with Z_A when $Z_R^T g_{FR} = 0$, since this simply involves repartitioning Q_{FR} . The ‘artificial’ multiplier vector associated with the rows of Z_A^T is equal to $Z_A^T g_{FR}$, and the multipliers corresponding to the rows of the ‘true’ working set are the multipliers that would be obtained if the artificial constraints were not present. If an artificial constraint is ‘deleted’ from the working set, an A appears alongside the entry in the Jdel column of the monitoring file output (see Section 13).

The number of columns in Z_A and Z_R and the Euclidean norm of $Z_R^T g_{FR}$, appear in the monitoring file output as Art, Zr and Norm Gz respectively (see Section 13).

Under some circumstances, a different type of artificial constraint is used when solving a linear program. Although the algorithm of E04MFF/E04MFA does not usually perform simplex steps (in the traditional sense), there is one exception: a linear program with fewer general constraints than variables (i.e., $m_L \leq n$). Use of the simplex method in this situation leads to savings in storage. At the starting point, the ‘natural’ working set (the set of constraints exactly or nearly satisfied at the starting point) is augmented with a suitable number of ‘temporary’ bounds, each of which has the effect of temporarily fixing a variable at its current value. In subsequent iterations, a temporary bound is treated as a standard constraint until it is deleted from the working set, in which case it is never added again. If a temporary bound is ‘deleted’ from the working set, an F (for ‘Fixed’) appears alongside the entry in the Jdel column of the monitoring file output (see Section 13).

12 Optional Parameters

Several optional parameters in E04MFF/E04MFA define choices in the problem specification or the algorithm logic. In order to reduce the number of formal arguments of E04MFF/E04MFA these optional parameters have associated *default values* that are appropriate for most problems. Therefore, you need only specify those optional parameters whose values are to be different from their default values.

The remainder of this section can be skipped if you wish to use the default values for all optional parameters.

The following is a list of the optional parameters available. A full description of each optional parameter is provided in Section 12.1.

Check Frequency

Cold Start

Crash Tolerance

Defaults

Expand Frequency

Feasibility Tolerance

Infinite Bound Size

Infinite Step Size

Iteration Limit

Iters

Itns

List

Minimum Sum of Infeasibilities

Monitoring File

Nolist

Optimality Tolerance

Print Level

Problem Type

Warm Start

Optional parameters may be specified by calling one, or both, of the routines E04MGF/E04MGA and E04MHF/E04MHA before a call to E04MFF/E04MFA.

E04MGF/E04MGA reads options from an external options file, with *Begin* and *End* as the first and last lines respectively and each intermediate line defining a single optional parameter. For example,

```
Begin
  Print Level = 5
End
```

The call

```
CALL E04MGF (IOPTNS, INFORM)
```

can then be used to read the file on unit IOPTNS. INFORM will be zero on successful exit. E04MGF/E04MGA should be consulted for a full description of this method of supplying optional parameters.

E04MHF/E04MHA can be called to supply options directly, one call being necessary for each optional parameter. For example,

```
CALL E04MHF ('Print Level = 5')
```

E04MHF/E04MHA should be consulted for a full description of this method of supplying optional parameters.

All optional parameters not specified by you are set to their default values. Optional parameters specified by you are unaltered by E04MFF/E04MFA (unless they define invalid values) and so remain in effect for subsequent calls unless altered by you.

12.1 Description of the Optional Parameters

For each option, we give a summary line, a description of the optional parameter and details of constraints.

The summary line contains:

the keywords, where the minimum abbreviation of each keyword is underlined (if no characters of an optional qualifier are underlined, the qualifier may be omitted);

a parameter value, where the letters a , i and r denote options that take character, integer and real values respectively;

the default value, where the symbol ϵ is a generic notation for *machine precision* (see X02AJF).

Keywords and character values are case and white space insensitive.

Check Frequency i Default = 50

Every i th iteration, a numerical test is made to see if the current solution x satisfies the constraints in the working set. If the largest residual of the constraints in the working set is judged to be too large, the current working set is refactorized and the variables are recomputed to satisfy the constraints more accurately. If $i \leq 0$, the default value is used.

Cold Start Default
Warm Start

This option specifies how the initial working set is chosen. With a **Cold Start**, E04MFF/E04MFA chooses the initial working set based on the values of the variables and constraints at the initial point. Broadly speaking, the initial working set will include equality constraints and bounds or inequality constraints that violate or ‘nearly’ satisfy their bounds (to within **Crash Tolerance**).

With a **Warm Start**, you must provide a valid definition of every element of the array ISTATE. E04MFF/E04MFA will override your specification of ISTATE if necessary, so that a poor choice of the working set will not cause a fatal error. For instance, any elements of ISTATE which are set to -2 , -1 or 4 will be reset to zero, as will any elements which are set to 3 when the corresponding elements of BL and BU are not equal. A warm start will be advantageous if a good estimate of the initial working set is available – for example, when E04MFF/E04MFA is called repeatedly to solve related problems.

Crash Tolerance r Default = 0.01

This value is used in conjunction with the optional parameter **Cold Start** (the default value) when E04MFF/E04MFA selects an initial working set. If $0 \leq r \leq 1$, the initial working set will include (if possible) bounds or general inequality constraints that lie within r of their bounds. In particular, a constraint of the form $a_j^T x \geq l$ will be included in the initial working set if $|a_j^T x - l| \leq r(1 + |l|)$. If $r < 0$ or $r > 1$, the default value is used.

Defaults

This special keyword may be used to reset all optional parameters to their default values.

Expand Frequency i Default = 5

This option is part of an anti-cycling procedure designed to guarantee progress even on highly degenerate problems.

The strategy is to force a positive step at every iteration, at the expense of violating the constraints by a small amount. Suppose that the value of the optional parameter **Feasibility Tolerance** is δ . Over a period of i iterations, the feasibility tolerance actually used by E04MFF/E04MFA (i.e., the *working* feasibility tolerance) increases from 0.5δ to δ (in steps of $0.5\delta/i$).

At certain stages the following ‘resetting procedure’ is used to remove constraint infeasibilities. First, all variables whose upper or lower bounds are in the working set are moved exactly onto their bounds. A count is kept of the number of nontrivial adjustments made. If the count is positive, iterative

refinement is used to give variables that satisfy the working set to (essentially) *machine precision*. Finally, the working feasibility tolerance is reinitialized to 0.5δ .

If a problem requires more than i iterations, the resetting procedure is invoked and a new cycle of i iterations is started with i incremented by 10. (The decision to resume the feasibility phase or optimality phase is based on comparing any constraint infeasibilities with δ .)

The resetting procedure is also invoked when E04MFF/E04MFA reaches an apparently optimal, infeasible or unbounded solution, unless this situation has already occurred twice. If any nontrivial adjustments are made, iterations are continued.

If $i \leq 0$, the default value is used. If $i \geq 9999999$, no anti-cycling procedure is invoked.

Feasibility Tolerance r Default = $\sqrt{\epsilon}$

If $r \geq \epsilon$, r defines the maximum acceptable *absolute* violation in each constraint at a ‘feasible’ point. For example, if the variables and the coefficients in the general constraints are of order unity, and the latter are correct to about 6 decimal digits, it would be appropriate to specify r as 10^{-6} . If $0 \leq r < \epsilon$, the default value is used.

E04MFF/E04MFA attempts to find a feasible solution before optimizing the objective function. If the sum of infeasibilities cannot be reduced to zero, the optional parameter **Minimum Sum of Infeasibilities** can be used to find the minimum value of the sum. Let S_{inf} be the corresponding sum of infeasibilities. If S_{inf} is quite small, it may be appropriate to raise r by a factor of 10 or 100. Otherwise, some error in the data should be suspected.

Note that a ‘feasible solution’ is a solution that satisfies the current constraints to within the tolerance r .

Infinite Bound Size r Default = 10^{20}

If $r > 0$, r defines the ‘infinite’ bound *bigbnd* in the definition of the problem constraints. Any upper bound greater than or equal to *bigbnd* will be regarded as $+\infty$ (and similarly any lower bound less than or equal to $-bigbnd$ will be regarded as $-\infty$). If $r < 0$, the default value is used.

Infinite Step Size r Default = $\max(bigbnd, 10^{20})$

If $r > 0$, r specifies the magnitude of the change in variables that will be considered a step to an unbounded solution. (Note that an unbounded solution can occur only when the problem is of type LP.) If the change in x during an iteration would exceed the value of r , the objective function is considered to be unbounded below in the feasible region. If $r \leq 0$, the default value is used.

Iteration Limit i Default = $\max(50, 5(n + m_L))$
Iters
Itns

The value of i specifies the maximum number of iterations allowed before termination. With $i = 0$ and **Print Level** > 0 , the workspace needed will be computed and printed, but no iterations will be performed. If $i < 0$, the default value is used.

List Default for E04MFF = **List**
Nolist Default for E04MFA = **Nolist**

Normally each optional parameter specification is printed as it is supplied. Optional parameter **Nolist** may be used to suppress the printing and optional parameter **List** may be used to restore printing.

Minimum Sum of Infeasibilities **No** Default = **NO**

If no feasible point exists for the constraints, this option is used to control whether or not E04MFF/E04MFA will calculate a point that minimizes the constraint violations. If **Minimum Sum of Infeasibilities** = **NO**, E04MFF/E04MFA will terminate as soon as it is evident that no feasible point exists for the constraints. The final point will generally not be the point at which the sum of infeasibilities is minimized. If **Minimum Sum of Infeasibilities** = **YES**, E04MFF/E04MFA will continue until the sum of infeasibilities is minimized.

Monitoring File i

Default = -1

If $i \geq 0$ and **Print Level** ≥ 5 , monitoring information produced by E04MFF/E04MFA at every iteration is sent to a file with logical unit number i . If $i < 0$ and/or **Print Level** < 5 , no monitoring information is produced.

Optimality Tolerance r Default = $\epsilon^{0.8}$

If $r \geq \epsilon$, r defines the tolerance used to determine if the bounds and general constraints have the right 'sign' for the solution to be judged to be optimal.

If $0 \leq r < \epsilon$, the default value is used.

Print Level i

Default for E04MFF = 10

Default for E04MFA = 0

The value of i controls the amount of printout produced by E04MFF/E04MFA, as indicated below. A detailed description of the printed output is given in Section 9.2 (summary output at each iteration and the final solution) and Section 13 (monitoring information at each iteration).

The following printout is sent to the current advisory message unit (as defined by X04ABF):

i	Output
0	No output.
1	The final solution only.
5	One line of summary output (< 80 characters; see Section 9.2) for each iteration (no printout of the final solution).
≥ 10	The final solution and one line of summary output for each iteration.

The following printout is sent to the logical unit number by the optional parameter **Monitoring File**:

i	Output
< 5	No output.
≥ 5	One long line of output (> 80 characters; see Section 13) for each iteration (no printout of the final solution).
≥ 20	At each iteration, the Lagrange multipliers, the variables x , the constraint values Ax and the constraint status (see ISTATE).
≥ 30	At each iteration, the diagonal elements of the upper triangular matrix T associated with the TQ factorization (3) (see Section 11.2) of the working set.

If **Print Level** ≥ 5 and the unit number defined by the optional parameter **Monitoring File** is the same as that defined by X04ABF, then the summary output for each major iteration is suppressed.

Problem Type a

Default = LP

This option specifies the type of objective function to be minimized during the optimality phase. The following is the optional keyword and the dimensions of the array that must be specified in order to define the objective function:

LP CVEC(N) required.

For problems of type FP, the objective function is omitted and CVEC is not referenced. The following keywords are also acceptable. The minimum abbreviation of each keyword is underlined.

a	Option
<u>L</u> inear	LP
<u>F</u> easible	FP

13 Description of Monitoring Information

This section describes the long line of output (> 80 characters) which forms part of the monitoring information produced by E04MFF/E04MFA. (See also the description of the optional parameters **Monitoring File** and **Print Level**.) You can control the level of printed output.

To aid interpretation of the printed results, the following convention is used for numbering the constraints: indices 1 through n refer to the bounds on the variables, and indices $n + 1$ through $n + m_L$ refer to the general constraints. When the status of a constraint changes, the index of the constraint is printed, along with the designation L (lower bound), U (upper bound), E (equality), F (temporarily fixed variable) or A (artificial constraint).

When **Print Level** ≥ 5 and **Monitoring File** ≥ 0 , the following line of output is produced at every iteration on the unit number specified by optional parameter **Monitoring File**. In all cases, the values of the quantities printed are those in effect *on completion* of the given iteration.

Itn	is the iteration count.
Jdel	is the index of the constraint deleted from the working set. If Jdel is zero, no constraint was deleted.
Jadd	is the index of the constraint added to the working set. If Jadd is zero, no constraint was added.
Step	is the step taken along the computed search direction. If a constraint is added during the current iteration, Step will be the step to the nearest constraint. When the problem is of type LP, the step can be greater than one during the optimality phase.
Ninf	is the number of violated constraints (infeasibilities). This will be zero during the optimality phase.
Sinf/Objective	is the value of the current objective function. If x is not feasible, Sinf gives a weighted sum of the magnitudes of constraint violations. If x is feasible, Objective is the value of the objective function of (1). The output line for the final iteration of the feasibility phase (i.e., the first iteration for which Ninf is zero) will give the value of the true objective at the first feasible point. During the optimality phase the value of the objective function will be nonincreasing. During the feasibility phase the number of constraint infeasibilities will not increase until either a feasible point is found or the optimality of the multipliers implies that no feasible point exists. Once optimal multipliers are obtained the number of infeasibilities can increase, but the sum of infeasibilities will either remain constant or be reduced until the minimum sum of infeasibilities is found.
Bnd	is the number of simple bound constraints in the current working set.
Lin	is the number of general linear constraints in the current working set.
Art	is the number of artificial constraints in the working set, i.e., the number of columns of Z_A (see Section 11.4).
Zr	is the number of columns of Z_R (see Section 11.2). Zr is the dimension of the subspace in which the objective function is currently being minimized. The value of Zr is the number of variables minus the number of constraints in the working set; i.e., $Zr = n - (Bnd + Lin + Art)$. The value of n_Z , the number of columns of Z (see Section 11.2) can be calculated as $n_Z = n - (Bnd + Lin)$. A zero value of n_Z implies that x lies at a vertex of the feasible region.
Norm Gz	is $\ Z_R^T g_{FR}\ $, the Euclidean norm of the reduced gradient with respect to Z_R . During the optimality phase, this norm will be approximately zero after a unit step.

NOpt	is the number of nonoptimal Lagrange multipliers at the current point. NOpt is not printed if the current x is infeasible or no multipliers have been calculated. At a minimizer, NOpt will be zero.
Min Lm	is the value of the Lagrange multiplier associated with the deleted constraint. If Min Lm is negative, a lower bound constraint has been deleted, if Min Lm is positive, an upper bound constraint has been deleted. If no multipliers are calculated during a given iteration Min Lm will be zero.
Cond T	is a lower bound on the condition number of the working set.
