NAG Library Routine Document

H02CBF

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

H02CBF solves general quadratic programming problems with integer constraints on the variables. It is not intended for large sparse problems.

2 Specification

```
SUBROUTINE H02CBF (N, NCLIN, A, LDA, BL, BU, CVEC, H, LDH, QPHESS,          &
                   INTVAR, LINTVR, MDEPTH, ISTATE, XS, OBJ, AX, CLAMDA,     &
                   STRTGY, IWRK, LIWRK, WRK, LWRK, MONIT, IFAIL)
INTEGER            N, NCLIN, LDA, LDH, INTVAR(LINTVR), LINTVR, MDEPTH,      &
                   ISTATE(N+NCLIN), STRTGY, IWRK(LIWRK), LIWRK, LWRK,       &
                   IFAIL
REAL (KIND=nag_wp) A(LDA,*), BL(N+NCLIN), BU(N+NCLIN), CVEC(*),             &
                   H(LDH,*), XS(N), OBJ, AX(max(1,NCLIN)),                  &
                   CLAMDA(N+NCLIN), WRK(LWRK)
EXTERNAL           QPHESS, MONIT
```

3 Description

H02CBF uses a 'Branch and Bound' algorithm in conjunction with E04NFF to try and determine integer solutions to a general quadratic programming problem. The problem is assumed to be stated in the following general form:

$$\underset{x\in R^n}{\mathrm{minimize}} f(x) \quad \text{subject to} \quad l \leq \begin{pmatrix} x \\ Ax \end{pmatrix} \leq u,$$

where A is an m_L by n matrix and $f(x)$ may be specified in a variety of ways depending upon the particular problem to be solved. The available forms for $f(x)$ are listed in Table 1, in which the prefixes FP, LP and QP stand for 'feasible point', 'linear programming' and 'quadratic programming' respectively and c is an n-element vector.

Problem type	$f(x)$	**Matrix *H***
FP	Not applicable	Not applicable
LP	$c^{\mathrm{T}}x$	Not applicable
QP1	$\frac{1}{2}x^{\mathrm{T}}Hx$	symmetric
QP2	$c^{\mathrm{T}}x+\frac{1}{2}x^{\mathrm{T}}Hx$	symmetric
QP3	$\frac{1}{2}x^{\mathrm{T}}H^{\mathrm{T}}Hx$	m by n upper trapezoidal
QP4	$c^{\mathrm{T}}x+\frac{1}{2}x^{\mathrm{T}}H^{\mathrm{T}}Hx$	m by n upper trapezoidal

Only when the problem is linear or the matrix H is positive definite can the technique be guaranteed to work; but often useful results can be obtained for a wider class of problems.

The default problem type is QP2 and other objective functions are selected by using the optional parameter **Problem Type**. For problems of type FP, the objective function is omitted and H02CBF attempts to find a feasible point for the set of constraints.

Branch and bound consists firstly of obtaining a solution without any of the variables $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ constrained to be integer. Suppose x_1 ought to be integer, but at the optimal value just computed $x_1 = 2.4$. A constraint $x_1 \leq 2$ is added to the system and the second problem solved. A constraint $x_1 \geq 3$ gives rise to a third sub-problem. In a similar manner a whole series of sub-problems may be generated, corresponding to integer constraints on the variables. The sub-problems are all solved using E04NFF.

In practice the routine tries to compute an integer solution as quickly as possible using a depth-first approach, since this helps determine a realistic cut-off value. If we have a cut-off value, say the value of the function at this first integer solution, and any sub-problem, W say, has a solution value greater than this cut-off value, then subsequent sub-problems of W must have solutions greater than the value of the solution at W and therefore need not be computed. Thus a knowledge of a good cut-off value can result in fewer sub-problems being solved and thus speed up the operation of the routine. (See the description of MONIT in Section 5 for details of how you can supply your own cut-off value.)

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

Pardalos P M and Schnitger G (1988) Checking local optimality in constrained quadratic programming is NP-hard *Operations Research Letters* **7** 33–35

5 Arguments

1: N – INTEGER *Input*

On entry: n, the number of variables.

Constraint: $\mathrm{N} > 0$.

2: NCLIN – INTEGER *Input*

On entry: m_L, the number of general linear constraints.

Constraint: $\mathrm{NCLIN} \geq 0$.

3: $\mathrm{A(LDA,*)}$ – REAL (KIND=nag_wp) array *Input*

Note: the second dimension of the array A must be at least N if $\mathrm{NCLIN} > 0$ and at least 1 if $\mathrm{NCLIN} = 0$.

On entry: the ith row of A must contain the coefficients of the ith general linear constraint, for $i = 1, 2, \ldots, m_L$.

If $\mathrm{NCLIN} = 0$, the array 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 H02CBF 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: BL 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 $\text{BL}(j) \leq -bigbnd$, and to specify a nonexistent upper bound (i.e., $u_j = +\infty$), set $\text{BU}(j) \geq bigbnd$; the default value of $bigbnd$ is 10^{20}, but this may be changed by the **Infinite Bound Size**. To specify the jth constraint as an *equality*, set $\text{BL}(j) = \text{BU}(j) = \beta$, say, where $|\beta| < bigbnd$.

Constraints:

$\text{BL}(j) \leq \text{BU}(j)$, for $j = 1, 2, \ldots, \text{N} + \text{NCLIN}$;
if $\text{BL}(j) = \text{BU}(j) = \beta$, $|\beta| < bigbnd$.

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, QP2 (the default) or QP4, and at least 1 otherwise.

On entry: the coefficients of the explicit linear term of the objective function when the problem is of type LP, QP2 (the default) and QP4.

If the problem is of type FP, QP1, or QP3, CVEC is not referenced.

8: H(LDH, $*$) – REAL (KIND=nag_wp) array *Input*

Note: the second dimension of the array H must be at least N if it is to be used to store H explicitly, and at least 1 otherwise.

On entry: may be used to store the quadratic term H of the QP objective function if desired. In some cases, you need not use H to store H explicitly (see the specification of QPHESS). The elements of H are referenced only by QPHESS. The number of rows of H is denoted by m, whose default value is n. (The **Hessian Rows** may be used to specify a value of $m < n$.)

If the default version of QPHESS is used and the problem is of type QP1 or QP2 (the default), the first m rows and columns of H must contain the leading m by m rows and columns of the symmetric Hessian matrix H. Only the diagonal and upper triangular elements of the leading m rows and columns of H are referenced. The remaining elements need not be assigned.

If the default version of QPHESS is used and the problem is of type QP3 or QP4, the first m rows of H must contain an m by n upper trapezoidal factor of the symmetric Hessian matrix $H^{\mathrm{T}}H$. The factor need not be of full rank, i.e., some of the diagonal elements may be zero. However, as a general rule, the larger the dimension of the leading nonsingular sub-matrix of H, the fewer iterations will be required. Elements outside the upper trapezoidal part of the first m rows of H need not be assigned.

In other situations, it may be desirable to compute Hx or $H^{\mathrm{T}}Hx$ without accessing H – for example, if H or $H^{\mathrm{T}}H$ is sparse or has special structure. The arguments H and LDH may then refer to any convenient array.

If the problem is of type FP or LP, H is not referenced.

9: LDH – INTEGER *Input*

On entry: the first dimension of the array H as declared in the (sub)program from which H02CBF is called.

Constraints:

if the problem is of type QP1, QP2 (the default), QP3 or QP4, LDH $\geq$ N or at least the value of the optional parameter **Hessian Rows** (default value $= n$);
if the problem is of type FP or LP, LDH ≥ 1.

10: QPHESS – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

In general, you need not provide a version of QPHESS, because a 'default' subroutine with name E04NFU is included in the Library. However, the algorithm of H02CBF requires only the product of H or H^TH and a vector x; and in some cases you may obtain increased efficiency by providing a version of QPHESS that avoids the need to define the elements of the matrices H or H^TH explicitly. QPHESS is not referenced if the problem is of type FP or LP, in which case QPHESS may be the routine E04NFU.

The specification of QPHESS is:

```
SUBROUTINE QPHESS (N, JTHCOL, H, LDH, X, HX)

INTEGER             N, JTHCOL, LDH
REAL (KIND=nag_wp) H(LDH,*), X(N), HX(N)
```

1: N – INTEGER *Input*

On entry: this is the same argument N as supplied to H02CBF.

2: JTHCOL – INTEGER *Input*

On entry: specifies whether or not the vector x is a column of the identity matrix.

JTHCOL $= j > 0$
The vector x is the jth column of the identity matrix, and hence Hx or H^THx is the jth column of H or H^TH, respectively, which may in some cases require very little computation and QPHESS may be coded to take advantage of this. However special code is not necessary because x is always stored explicitly in the array X.

JTHCOL $= 0$
x has no special form.

3: H(LDH, $*$) – REAL (KIND=nag_wp) array *Input*

On entry: this is the same argument H as supplied to H02CBF.

4: LDH – INTEGER *Input*

On entry: this is the same argument LDH as supplied to H02CBF.

5: X(N) – REAL (KIND=nag_wp) array *Input*

On entry: the vector x.

6: HX(N) – REAL (KIND=nag_wp) array *Output*

On exit: the product Hx if the problem is of type QP1 or QP2 (the default), or the product H^THx if the problem is of type QP3 or QP4.

QPHESS must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which H02CBF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

11: INTVAR(LINTVR) – INTEGER array *Input*

On entry: INTVAR(i) must contain the index of the solution vector x which is required to be integer. For example, if x_1 and x_3 are constrained to take integer values then INTVAR(1) might be set to 1 and INTVAR(2) to 3. The order in which the indices are specified is important, since this determines the order in which the sub-problems are generated. As a rule-of-thumb, the important variables should always be specified first. Thus, in the above example, if x_3 relates to a more important quantity than x_1, then it might be advantageous to set INTVAR$(1) = 3$ and INTVAR$(2) = 1$. If k is the smallest integer such that INTVAR(k) is less than or equal to zero then H02CBF assumes that $k - 1$ variables are constrained to be integer; components INTVAR$(k + 1)$, ..., INTVAR(LINTVR) are *not* referenced.

12: LINTVR – INTEGER *Input*

On entry: the dimension of the array INTVAR as declared in the (sub)program from which H02CBF is called. Often LINTVR is the number of variables that are constrained to be integer.

Constraint: LINTVR > 0.

13: MDEPTH – INTEGER *Input*

On entry: the maximum depth (i.e., number of extra constraints) that H02CBF may insert before admitting failure.

Suggested value: MDEPTH $= 3 \times$ N/2.

Constraint: MDEPTH ≥ 1.

14: 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 *not* 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 BL$(j) =$ BU(j).

The values -2, -1 and 4 are also acceptable but will be reset to zero by the routine. If H02CBF 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 XS to be consistent with ISTATE.

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

On exit: the status of the constraints in the working set at the point returned in XS. The significance of each possible value of ISTATE(j) is as follows:

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 BL$(j) =$ BU(j).
4	This corresponds to optimality being declared with XS(j) being temporarily fixed at its current value. This value of ISTATE can occur only when IFAIL $= 1$ on exit.

15: XS(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: an initial estimate of the solution.

On exit: the point at which H02CBF terminated. If IFAIL $= 0$, 1 or 3, XS contains an estimate of the solution.

16: 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.

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

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

If NCLIN $= 0$, AX is not referenced.

18: 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 ISTATE$(j) = 0$ (i.e., constraint j is not in the working set), CLAMDA(j) is zero. If x is optimal, CLAMDA(j) should be non-negative if ISTATE$(j) = 1$, non-positive if ISTATE$(j) = 2$ and zero if ISTATE$(j) = 4$.

19: STRTGY – INTEGER *Input*

On entry: determines a branching strategy to be used throughout the computation, as follows:

STRTGY	**Meaning**
0	Always left branch first, i.e., impose an upper bound constraint on the variable first.
1	Always right branch first, i.e., impose a lower bound constraint on the variable first.
2	Branch towards the nearest integer, i.e., if $x_k = 2.4$ then impose an upper bound constraint $x_k \leq 2$, whereas if $x_k = 2.6$ then impose the lower bound constraint $x_k \geq 3.0$.
3	A random choice is made between a left-hand and a right-hand branch.

Constraint: STRTGY $= 0$, 1, 2 or 3.

20: IWRK(LIWRK) – INTEGER array *Workspace*
21: LIWRK – INTEGER *Input*

On entry: the dimension of the array IWRK as declared in the (sub)program from which H02CBF is called.

Constraint: LIWRK $\geq 2 \times \mathrm{N} + 3 + 2 \times \mathrm{MDEPTH}$.

22: WRK(LWRK) – REAL (KIND=nag_wp) array *Workspace*
23: LWRK – INTEGER *Input*

On entry: the dimension of the array WRK as declared in the (sub)program from which H02CBF is called.

Constraints:

if the problem type is QP2 (the default) or QP4,

if NCLIN > 0, LWRK $\geq 2 \times \mathrm{N}^2 + 9 \times \mathrm{N} + 5 \times \mathrm{NCLIN} + 4 \times \mathrm{MDEPTH}$;
if NCLIN $= 0$, LWRK $\geq \mathrm{N}^2 + 9 \times \mathrm{N} + 4 \times \mathrm{MDEPTH}$.;

if the problem type is QP1 or QP3,

$\quad$ if $\mathrm{NCLIN} > 0$, $\mathrm{LWRK} \geq 2 \times \mathrm{N}^2 + 8 \times \mathrm{N} + 5 \times \mathrm{NCLIN} + 4 \times \mathrm{MDEPTH}$;
$\quad$ if $\mathrm{NCLIN} = 0$, $\mathrm{LWRK} \geq \mathrm{N}^2 + 8 \times \mathrm{N} + 4 \times \mathrm{MDEPTH}$.;

if the problem type is LP,

$\quad$ if $\mathrm{NCLIN} = 0$, $\mathrm{LWRK} \geq 9 \times \mathrm{N} + 1 + 4 \times \mathrm{MDEPTH}$;
$\quad$ if $\mathrm{NCLIN} \geq \mathrm{N}$, $\mathrm{LWRK} \geq 2 \times \mathrm{N}^2 + 9 \times \mathrm{N} + 5 \times \mathrm{NCLIN} + 4 \times \mathrm{MDEPTH}$;
$\quad$ otherwise $\mathrm{LWRK} \geq 2 \times (\mathrm{NCLIN} + 1)^2 + 9 \times \mathrm{N} + 5 \times \mathrm{NCLIN} + 4 \times \mathrm{MDEPTH}$.;

if the problem type is FP,

$\quad$ if $\mathrm{NCLIN} = 0$, $\mathrm{LWRK} \geq 8 \times \mathrm{N} + 1 + 4 \times \mathrm{MDEPTH}$;
$\quad$ if $\mathrm{NCLIN} \geq \mathrm{N}$, $\mathrm{LWRK} \geq 2 \times \mathrm{N}^2 + 8 \times \mathrm{N} + 5 \times \mathrm{NCLIN} + 4 \times \mathrm{MDEPTH}$;
$\quad$ otherwise $\mathrm{LWRK} \geq 2 \times (\mathrm{NCLIN} + 1)^2 + 8 \times \mathrm{N} + 5 \times \mathrm{NCLIN} + 4 \times \mathrm{MDEPTH}$..

24: MONIT – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

MONIT may be used to print out intermediate output and to affect the course of the computation. Specifically, it allows you to specify a realistic value for the cut-off value (see Section 3) and to terminate the algorithm. If you do not require any intermediate output, have no estimate of the cut-off value and require an exhaustive tree search then MONIT may be the dummy routine H02CBU.

The specification of MONIT is:

```
SUBROUTINE MONIT (INTFND, NODES, DEPTH, OBJ, X, BSTVAL, BSTSOL,      &
                  BL, BU, N, HALT, COUNT)

INTEGER             INTFND, NODES, DEPTH, N, COUNT
REAL (KIND=nag_wp) OBJ, X(N), BSTVAL, BSTSOL(N), BL(N), BU(N)
LOGICAL             HALT
```

1: INTFND – INTEGER *Input*

On entry: specifies the number of integer solutions obtained so far.

2: NODES – INTEGER *Input*

On entry: specifies the number of nodes (sub-problems) solved so far.

3: DEPTH – INTEGER *Input*

On entry: specifies the depth in the tree of sub-problems the algorithm has now reached.

4: OBJ – REAL (KIND=nag_wp) *Input*

On entry: specifies the value of the objective function of the end of the latest sub-problem.

5: X(N) – REAL (KIND=nag_wp) array *Input*

On entry: specifies the values of the independent variables at the end of the latest sub-problem.

6: BSTVAL – REAL (KIND=nag_wp) *Input/Output*

On entry: normally specifies the value of the best integer solution found so far.

On exit: may be set a cut-off value if you are an experienced user as follows. Before an integer solution has been found BSTVAL will be set by H02CBF to the largest machine representable number (see X02ALF). If you know that the solution being sought is a much smaller number, then BSTVAL may be set to this number as a cut-off value (see Section 3). Beware of setting BSTVAL too small, since then no integer solutions will be discovered. Also make sure that BSTVAL is set using a statement of the form

```
IF (INTFND.EQ.0) BSTVAL = cut-off value
```

on entry to MONIT. This statement will not prevent the normal operation of the algorithm when subsequent integer solutions are found. It would be a grievous mistake to unconditionally set BSTVAL and if you have any doubts whatsoever about the correct use of this argument then you are strongly recommended to leave it unchanged.

7: BSTSOL(N) – REAL (KIND=nag_wp) array *Input*

On entry: specifies the solution vector which gives rise to the best integer solution value so far discovered.

8: BL(N) – REAL (KIND=nag_wp) array *Input*

On entry: BL(i) specifies the current lower bounds on the variable x_i.

9: BU(N) – REAL (KIND=nag_wp) array *Input*

On entry: BU(i) specifies the current upper bounds on the variable x_i.

10: N – INTEGER *Input*

On entry: specifies the number of variables.

11: HALT – LOGICAL *Input/Output*

On entry: will have the value .FALSE..

On exit: if HALT is set to .TRUE., E04NFF/E04NFA will be brought to a halt with IFAIL $= -1$. This facility may be useful if you are content with *any* integer solution, or with any integer solution that fits certain criteria. Under these circumstances setting HALT = .TRUE. can save considerable unnecessary computation.

12: COUNT – INTEGER *Input/Output*

On entry: unchanged from previous call.

On exit: may be used by you to save the last value of INTFND. If a subsequent call of MONIT has a value of INTFND which is greater than COUNT, then you know that a new integer solution has been found at this node.

MONIT must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which H02CBF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

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 $= -1$

Algorithm terminated at your request (HALT = .TRUE.).

IFAIL $= 1$

Input argument error immediately detected.

IFAIL $= 2$

No integer solution found. (Check that BSTVAL has not been set too small.)

IFAIL $= 3$

MDEPTH is too small. Increase the value of MDEPTH and re-enter H02CBF.

IFAIL $= 4$

The basic problem (without integer constraints) is unbounded.

IFAIL $= 5$

The basic problem is infeasible.

IFAIL $= 6$

The basic problem requires too many iterations.

IFAIL $= 7$

The basic problem has a reduced Hessian which exceeds its assigned dimension.

IFAIL $= 8$

The basic problem has an invalid argument setting.

IFAIL $= 9$

The basic problem, as defined, is not standard.

IFAIL $= 10$

LIWRK is too small.

IFAIL $= 11$

LWRK is too small.

IFAIL $= 12$

An internal error has occurred within the routine. Please contact NAG with details of the call to H02CBF.

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

H02CBF 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

H02CBF is not thread safe and should not be called from a multithreaded user program. Please see Section 3.12.1 in How to Use the NAG Library and its Documentation for more information on thread safety.

H02CBF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

H02CBF 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 Chapter E04 and Gill *et al.* (1981) for further information and advice.

9.2 Description of the Printed Output

This section describes the (default) intermediate printout and final printout produced by H02CBF. 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 **Print Level** in Section 12.1). Note that the intermediate printout and final printout are produced only if **Print Level** ≥ 10 (the default).

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. 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 $\left\|Z_R^{\mathrm{T}} g_{\mathrm{FR}}\right\|$, the Euclidean norm of the reduced gradient with respect to Z_R (see Sections 11.2 and 11.4). During the optimality phase, this norm will be approximately zero after a unit step.

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.

A key is sometimes printed before `State` to give some additional information about the state of a variable.

`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** (default value $= \sqrt{\epsilon}$, where ϵ is the ***machine precision***; see Section 12.1), `State` will be `++` or `--` respectively. `A` *Alternative optimum possible*. 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, there would be no change to the objective function. The values of the other free variables *might* 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` *Degenerate*. The variable is free, but it is equal to (or very close to) one of its bounds. `I` *Infeasible*. 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 iterate.
`Lower Bound`	is the lower bound specified for the variable. `None` indicates that $\mathrm{BL}(j) \leq -bigbnd$.
`Upper Bound`	is the upper bound specified for the variable. `None` indicates that $\mathrm{BU}(j) \geq bigbnd$.
`Slack`	is the difference between the variable `Value` and the nearer of its (finite) bounds $\mathrm{BL}(j)$ and $\mathrm{BU}(j)$. A blank entry indicates that the associated variable is not bounded (i.e., $\mathrm{BL}(j) \leq -bigbnd$ and $\mathrm{BU}(j) \geq bigbnd$).

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

`L Con` gives the name (`L`) and index j, for $j = 1, 2, \dots, m$, of the 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 quadratic function $f(x) = c^{\mathrm{T}} x + \frac{1}{2} x^{\mathrm{T}} H x$, where

$$c = (-0.02, -0.2, -0.2, -0.2, -0.2, 0.04, 0.04)^{\mathrm{T}}$$

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

subject to the bounds

$$\begin{array}{l} -0.01 \le x_1 \le 0.01 \\ -0.1 \le x_2 \le 0.15 \\ -0.01 \le x_3 \le 0.03 \\ -0.04 \le x_4 \le 0.02 \\ -0.1 \le x_5 \le 0.05 \\ -0.01 \le x_6 \\ -0.01 \le x_7 \end{array}$$

to the general constraints

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

and the variable x_4 is constrained to be integer.

The initial point, which is infeasible, is

$$x_0 = (-0.01, -0.03, 0.0, -0.01, -0.1, 0.02, 0.01)^{\mathrm{T}}.$$

The optimal solution (to five figures) is

$$x^* = (-0.01, -0.073328, -0.00025809, 0.0, -0.063354, 0.014109, 0.0028312)^{\mathrm{T}}.$$

The document for H02CCF includes an example program to solve the same problem using some of the optional parameters described in Section 12.

10.1 Program Text

```
    Program h02cbfe

!     H02CBF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: e04nfu, h02cbf, h02cbu, h02cdf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
```

```
      Integer, Parameter               :: lintvr = 1, mdepth = 30, nin = 5,    &
                                          nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: obj
      Integer                          :: i, ifail, j, lda, ldh, liwrk, lwrk,  &
                                          n, nclin, strtgy
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), ax(:), bl(:), bu(:),         &
                                          clamda(:), cvec(:), h(:,:), wrk(:),  &
                                          xs(:)
      Integer, Allocatable             :: intvar(:), istate(:), iwrk(:)
!     .. Executable Statements ..
      Write (nout,*) 'H02CBF Example Program Results'

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

      Read (nin,*) n, nclin
      lda = nclin
      ldh = n
      liwrk = 2*n + 3 + 2*mdepth

!     LWRK for default problem-type QP2

      If (nclin==0) Then
        lwrk = n**2 + 9*n + 4*mdepth
      Else
        lwrk = 2*n**2 + 9*n + 5*nclin + 4*mdepth
      End If

      Allocate (a(lda,n),ax(nclin),bl(n+nclin),bu(n+nclin),clamda(n+nclin),    &
        cvec(n),h(ldh,n),xs(n),intvar(lintvr),istate(n+nclin),iwrk(liwrk),     &
        wrk(lwrk))

      Read (nin,*)(cvec(i),i=1,n)
      Read (nin,*)((a(i,j),j=1,n),i=1,nclin)
      Read (nin,*)(bl(i),i=1,n+nclin)
      Read (nin,*)(bu(i),i=1,n+nclin)
      Read (nin,*)(xs(i),i=1,n)
      Read (nin,*)((h(i,j),j=1,n),i=1,n)

      strtgy = 2
      intvar(1) = 4

      Call h02cdf('Nolist')

      Call h02cdf('Print Level = 0')

!     Solve the problem

      ifail = 0
      Call h02cbf(n,nclin,a,lda,bl,bu,cvec,h,ldh,e04nfu,intvar,lintvr,mdepth,  &
        istate,xs,obj,ax,clamda,strtgy,iwrk,liwrk,wrk,lwrk,h02cbu,ifail)

!     Print out the best integer solution found

      Write (nout,99999) obj, (i,xs(i),i=1,n)

99999 Format (' Optimal Integer Value is = ',E20.8,/,' Components are ',       &
        7(/,' X(',I3,') = ',F15.8))
    End Program h02cbfe
```

10.2 Program Data

```
H02CBF 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 XS
  2.00   0.00   0.00   0.00   0.00   0.00   0.00
  0.00   2.00   0.00   0.00   0.00   0.00   0.00
  0.00   0.00   2.00   2.00   0.00   0.00   0.00
  0.00   0.00   2.00   2.00   0.00   0.00   0.00
  0.00   0.00   0.00   0.00   2.00   0.00   0.00
  0.00   0.00   0.00   0.00   0.00  -2.00  -2.00
  0.00   0.00   0.00   0.00   0.00  -2.00  -2.00   :End of matrix H
```

10.3 Program Results

```
 H02CBF Example Program Results
 Optimal Integer Value is =       0.37469662E-01
 Components are
 X(  1) =     -0.01000000
 X(  2) =     -0.07332830
 X(  3) =     -0.00025809
 X(  4) =      0.00000000
 X(  5) =     -0.06335433
 X(  6) =      0.01410944
 X(  7) =      0.00283128
```

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 H02CCF and/or H02CDF. Section 13 describes the quantities which can be requested to monitor the course of the computation.*

11 Algorithmic Details

H02CBF implements a basic branch and bound algorithm (see Section 3) using E04NFF as its basic sub-problem solver. See below for details of its algorithm.

11.1 Overview

H02CBF is based on an inertia-controlling method that maintains a Cholesky factorization of the reduced Hessian (see below). The method is based on that of 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 H02CBF or appear in the printed output. H02CBF has two phases:

(i) finding an initial feasible point by minimizing the sum of infeasibilities (the *feasibility phase*), and

(ii) minimizing the quadratic 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 quadratic objective function. The feasibility phase does *not* perform the standard simplex method (i.e., it does not necessarily find a vertex), except in the LP case when $m_L \leq n$. Once any iterate is feasible, all subsequent iterates remain feasible.

H02CBF has been designed to be efficient when used to solve a *sequence* of related problems – for example, within a sequential quadratic programming method for nonlinearly constrained optimization (e.g., E04WDF). In particular, you may specify an initial working set (the indices of the constraints believed to be satisfied exactly at the solution); see the discussion of the **Warm Start** in Section 12.1.

In general, an iterative process is required to solve a quadratic 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

$$\bar{x} = x + \alpha p \tag{1}$$

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 **Feasibility Tolerance**; see Section 12.1). The working set is the current prediction of the constraints that hold with equality at the solution of a linearly constrained QP 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 H02CBF; see Section 13). Similarly, let n_{FR} ($n_{\mathrm{FR}} = n - n_{\mathrm{FX}}$) 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 the 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

$$A_{\mathrm{FR}} p_{\mathrm{FR}} = 0. \tag{2}$$

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

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

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

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

where Y is n_{FR} by m_{W}, then the n_Z ($n_Z = n_{\mathrm{FR}} - m_{\mathrm{W}}$) columns of Z form a basis for the null space of A_{FR}. Let n_R be an integer such that $0 \le n_R \le n_Z$, 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 output from H02CBF. In many cases, Z_R will include *all* the columns of Z.) The direction p_{FR} will satisfy (2) if

$$p_{\mathrm{FR}} = Z_R p_R, \tag{4}$$

where p_R is any n_R-vector.

Let Q denote the n by n matrix

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

where I_{FX} is the identity matrix of order n_{FX}. Let H_Q and g_Q denote the n by n *transformed Hessian* and *transformed gradient*

$$H_Q = Q^{\mathrm{T}} H Q \quad \text{and} \quad g_Q = Q^{\mathrm{T}}(c + Hx)$$

and let the matrix of first n_R rows and columns of H_Q be denoted by H_R and the vector of the first n_R elements of g_Q be denoted by g_R. The quantities H_R and g_R are known as the *reduced Hessian* and *reduced gradient* of $f(x)$, respectively. Roughly speaking, g_R and H_R describe the first and second derivatives of an *unconstrained* problem for the calculation of p_R.

At each iteration, a triangular factorization of H_R is available. If H_R is positive definite, $H_R = R^{\mathrm{T}}R$, where R is the upper triangular Cholesky factor of H_R. If H_R is not positive definite, $H_R = R^{\mathrm{T}}DR$, where $D = \mathrm{diag}(1, 1, \ldots, 1, \mu)$, with $\mu \leq 0$.

The computation is arranged so that the reduced-gradient vector is a multiple of e_R, a vector of all zeros except in the last (i.e., n_Rth) position. This allows the vector p_R in (4) to be computed from a single back-substitution

$$Rp_R = \gamma e_R \tag{5}$$

where γ is a scalar that depends on whether or not the reduced Hessian is positive definite at x. In the positive definite case, $x + p$ is the minimizer of the objective function subject to the constraints (bounds and general) in the working set treated as equalities. If H_R is not positive definite, p_R satisfies the conditions

$$p_R^{\mathrm{T}} H_R p_R < 0 \quad \text{and} \quad g_R^{\mathrm{T}} p_R \leq 0,$$

which allow the objective function to be reduced by any positive step of the form $x + \alpha p$.

11.3 The Main Iteration

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 quadratic 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

$$A_{\mathrm{FR}}^{\mathrm{T}} \lambda_C = g_{\mathrm{FR}} \quad \text{and} \quad \lambda_B = g_{\mathrm{FX}} - A_{\mathrm{FX}}^{\mathrm{T}} \lambda_C. \tag{6}$$

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 H02CBF to continue until the minimum value of the sum of infeasibilities has been found; see the discussion of the **Minimum Sum of Infeasibilities** in Section 12.1. 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 $Z_R p_R$ (see (4) and (5)). The choice of step length is influenced by the need to maintain feasibility with respect to the satisfied constraints. If H_R is positive definite and $x + p$ is feasible, α will be taken as unity. In this case, the reduced gradient at $\bar{x}$ will be zero, and Lagrange-multipliers are computed. Otherwise, α is set to α_{M}, the step to the 'nearest' constraint (with index `Jadd`; see Section 13), which is added to the working set at the next iteration.

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, Q_{FR} and R; and of vectors $Q^{\mathrm{T}}g$, and $Q^{\mathrm{T}}c$. The triangular factor R associated with the reduced Hessian is only updated during the optimality phase.

One of the most important features of H02CBF 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`.

H02CBF 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 $x + \alpha_{\mathrm{M}} p$ does not violate any constraint by more than its feasibility tolerance. All constraints at a distance α ($\alpha \leq \alpha_{\mathrm{M}}$) 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

At the start of the optimality phase, a positive definite H_R can be defined if enough constraints are included in the initial working set. (The matrix with no rows and columns is positive definite by definition, corresponding to the case when A_{FR} contains n_{FR} constraints.) The idea is to include as many general constraints as necessary to ensure that the reduced Hessian is positive definite.

Let H_Z denote the matrix of the first n_Z rows and columns of the matrix $H_Q = Q^{\mathrm{T}} H Q$ at the beginning of the optimality phase. A partial Cholesky factorization is used to find an upper triangular matrix R that is the factor of the largest positive definite leading sub-matrix of H_Z. The use of interchanges during the factorization of H_Z tends to maximize the dimension of R. (The condition of R may be controlled using the **Rank Tolerance**. Let Z_R denote the columns of Z corresponding to R, and let Z be partitioned as $Z = \begin{pmatrix} Z_R & Z_A \end{pmatrix}$. 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

$$\bar{A}_{\mathrm{FR}} = \begin{pmatrix} Z_A^{\mathrm{T}} \\ A_{\mathrm{FR}} \end{pmatrix}, \tag{7}$$

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

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

where

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

and hence the TQ factorization of (7) 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 $Z_R^{\mathrm{T}} g_{\mathrm{FR}} = 0$, since this simply involves repartitioning Q_{FR}. The 'artificial' multiplier vector associated with the rows of Z_A^{T} is equal to $Z_A^{\mathrm{T}} g_{\mathrm{FR}}$, 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, the Euclidean norm of $Z_R^{\mathrm{T}} g_{\mathrm{FR}}$, and the condition estimator of R appear in the monitoring file output as `Art`, `Zr`, `Norm Gz` and `Cond Rz` respectively (see Section 13).

Under some circumstances, a different type of artificial constraint is used when solving a linear program. Although the algorithm of H02CBF 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 H02CBF define choices in the problem specification or the algorithm logic. In order to reduce the number of formal arguments of H02CBF 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 Phase Iteration Limit
Feasibility Tolerance
Hessian Rows
Infinite Bound Size
Infinite Step Size
Iteration Limit
Iters
Itns
List
Maximum Degrees of Freedom
Minimum Sum of Infeasibilities
Monitoring File
Nolist
Optimality Phase Iteration Limit
Optimality Tolerance
Print Level
Problem Type
Rank Tolerance
Warm Start

Optional parameters may be specified by calling one, or both, of the routines H02CCF and H02CDF prior to a call to H02CBF.

H02CCF 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 H02CCF(IOPTNS, INFORM)
```

can then be used to read the file on unit IOPTNS. INFORM will be zero on successful exit. H02CCF should be consulted for a full description of this method of supplying optional parameters.

H02CDF can be called to supply options directly, one call being necessary for each optional parameter. For example,

```
      CALL H02CDF ('Print Level = 5')
```

H02CDF 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 H02CBF (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 ith 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**, H02CBF 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 (see Section 5 for the definition of this array). H02CBF 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 H02CBF 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 H02CBF 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 $a_j^{\mathrm{T}}x \geq l$ will be included in the initial working set if $\left|a_j^{\mathrm{T}}x - l\right| \leq r(1 + |l|)$. 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 H02CBF (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 H02CBF 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 Phase Iteration Limit i_1 Default $= \max(50, 5(n + m_L))$
Optimality Phase Iteration Limit i_2 Default $= \max(50, 5(n + m_L))$

The scalars i_1 and i_2 specify the maximum number of iterations allowed in the feasibility and optimality phases. **Optimality Phase Iteration Limit** is equivalent to **Iteration Limit**. Setting $i_1 = 0$ and **Print Level** > 0 means that the workspace needed will be computed and printed, but no iterations will be performed. If $i_1 < 0$ or $i_2 < 0$, the default value is used.

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.

H02CBF attempts to find a feasible solution before optimizing the objective function. If the sum of infeasibilities cannot be reduced to zero, the **Minimum Sum of Infeasibilities** can be used to find the minimum value of the sum. Let `Sinf` be the corresponding sum of infeasibilities. If `Sinf` 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.

Hessian Rows i Default $= n$

Note that this option does not apply to problems of type FP or LP.

This specifies m, the number of rows of the Hessian matrix H. The default value of m is n, the number of variables of the problem.

If the problem is of type QP, m will usually be n, the number of variables. However, a value of m less than n is appropriate for QP3 or QP4 if H is an upper trapezoidal matrix with m rows. Similarly, m may be used to define the dimension of a leading block of nonzeros in the Hessian matrices of QP1 or QP2, in which case the last $n - m$ rows and columns of H are assumed to be zero. In the QP case, m should not be greater than n; if it is, the last $m - n$ rows of H are ignored.

If $i < 0$ or $i > n$, the default value is used.

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

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

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 Hessian is not positive definite.) 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

See optional parameter **Feasibility Phase Iteration Limit**.

List Default
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.

Maximum Degrees of Freedom i Default $= n$

Note that this option does not apply to problems of type FP or LP.

This places a limit on the storage allocated for the triangular factor R of the reduced Hessian H_R. Ideally, i should be set slightly larger than the value of n_R expected at the solution. It need not be larger than $m_{\mathrm{N}} + 1$, where m_{N} is the number of variables that appear nonlinearly in the quadratic objective function. For many problems it can be much smaller than m_{N}.

For quadratic problems, a minimizer may lie on any number of constraints, so that n_R may vary between 1 and n. The default value of i is therefore the number of variables n. If **Hessian Rows** m is specified, the default value of i is the same number, m.

Minimum Sum of Infeasibilities Default $=$ NO

If no feasible point exists for the constraints, this option is used to control whether or not H02CBF will calculate a point that minimizes the constraint violations. If **Minimum Sum of Infeasibilities** $=$ NO, H02CBF 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, H02CBF 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 H02CBF 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 $= 10$

The value of i controls the amount of printout produced by H02CBF, 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). If $i < 0$, the default value is used.

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 defined by the **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.
≥ 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, and the diagonal elements of the upper triangular matrix R.

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

Problem Type a Default $=$ QP2

This option specifies the type of objective function to be minimized during the optimality phase. The following are the five optional keywords and the dimensions of the arrays that must be specified in order to define the objective function:

LP	H not referenced, CVEC(N) required;
QP1	H(LDH,$*$) symmetric, CVEC not referenced;
QP2	H(LDH,$*$) symmetric, CVEC(N) required;
QP3	H(LDH,$*$) upper trapezoidal, CVEC not referenced;
QP4	H(LDH,$*$) upper trapezoidal, CVEC(N) required.

For problems of type FP, the objective function is omitted and neither H nor CVEC are referenced.

The following keywords are also acceptable. The minimum abbreviation of each keyword is underlined.

a	**Option**
Quadratic	QP2
Linear	LP
Feasible	FP

In addition, the keyword QP is equivalent to the default option QP2.

If H $= 0$, i.e., the objective function is purely linear, the efficiency of H02CBF may be increased by specifying a as LP.

Rank Tolerance r Default $= 100\epsilon$

Note that this option does not apply to problems of type FP or LP.

This parameter enables you to control the condition number of the triangular factor R (see Section 11). If ρ_i denotes the function $\rho_i = \max\{|R_{11}|, |R_{22}|, \ldots, |R_{ii}|\}$, the dimension of R is defined to be smallest index i such that $\left|R_{i+1,i+1}\right| \leq \sqrt{r}|\rho_{i+1}|$. If $r \leq 0$, the default value is used.

13 Description of Monitoring Information

This section describes the long line of output ($>$ 80 characters) which forms part of the monitoring information produced by H02CBF. (See also the description of the optional parameters **Monitoring File** and **Print Level** in Section 12.1.) 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 **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. 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., $\texttt{Zr} = n - (\texttt{Bnd} + \texttt{Lin} + \texttt{Art})$. The value of n_Z, the number of columns of Z (see Section 11.2) can be calculated as $n_Z = n - (\texttt{Bnd} + \texttt{Lin})$. A zero value of n_Z implies that x lies at a vertex of the feasible region.

`Norm Gz`	is $\left\|Z_R^{\mathrm{T}} g_{\mathrm{FR}}\right\|$, the Euclidean norm of the reduced gradient with respect to Z_R (see Sections 11.2 and 11.4). 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.
`Cond Rz`	is a lower bound on the condition number of the triangular factor R (the Cholesky factor of the current reduced Hessian; see Section 11.2). If the problem is specified to be of type LP, `Cond Rz` is not printed.
`Rzz`	is the last diagonal element μ of the matrix D associated with the $R^{\mathrm{T}}DR$ factorization of the reduced Hessian H_R (see Section 11.2). `Rzz` is only printed if H_R is not positive definite (in which case $\mu \neq 1$). If the printed value of `Rzz` is small in absolute value, then H_R is approximately singular. A negative value of `Rzz` implies that the objective function has negative curvature on the current working set.