NAG Library Routine Document

H02BZF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

H02BZF extracts more information associated with the solution of an integer programming problem computed by H02BBF.

2 Specification

```
SUBROUTINE H02BZF (N, M, BL, BU, CLAMDA, ISTATE, IWORK, LIWORK, RWORK,          &
                   LRWORK, IFAIL)

INTEGER            N, M, ISTATE(N+M), IWORK(LIWORK), LIWORK, LRWORK, IFAIL
REAL (KIND=nag_wp) BL(N+M), BU(N+M), CLAMDA(N+M), RWORK(LRWORK)
```

3 Description

H02BZF extracts the following information associated with the solution of an integer programming problem computed by H02BBF. The upper and lower bounds used for the solution, the Lagrange-multipliers (costs), and the status of the variables at the solution.

In the branch and bound method employed by H02BBF, the arrays BL and BU are used to impose restrictions on the values of the integer variables in each sub-problem. That is, if the variable x_j is restricted to take value v_j in a particular sub-problem, then $\mathrm{BL}(j) = \mathrm{BU}(j) = v_j$ is set in the sub-problem. Thus, on exit from this routine, some of the elements of BL and BU which correspond to integer variables may contain these imposed values, rather than those originally supplied to H02BBF.

4 References

None.

5 Parameters

1: N – INTEGER *Input*

On entry: this **must** be the same parameter N as supplied to H02BBF.

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

2: M – INTEGER *Input*

On entry: this **must** be the same parameter M as supplied to H02BBF.

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

3: $\mathrm{BL}(\mathrm{N}+\mathrm{M})$ – REAL (KIND=nag_wp) array *Output*

On exit: if H02BBF exits with $\mathrm{IFAIL} = 0$, 7 or 9, the values in the array BL contain the lower bounds imposed on the integer solution for all the constraints. The first N elements contain the lower bounds on the variables, and the next M elements contain the lower bounds for the general linear constraints (if any).

4: BU(N + M) – REAL (KIND=nag_wp) array *Output*

On exit: if H02BBF exits with IFAIL = 0, 7 or 9, the values in the array BU contain the upper bounds imposed on the integer solution for all the constraints. The first N elements contain the upper bounds on the variables, and the next M elements contain the upper bounds for the general linear constraints (if any).

5: CLAMDA(N + M) – REAL (KIND=nag_wp) array *Output*

On exit: if H02BBF exits with IFAIL = 0, 7 or 9, the values in the array CLAMDA contain the values of the Lagrange-multipliers for each constraint with respect to the current working set. The first N elements contain the multipliers (reduced costs) for the bound constraints on the variables, and the next M elements contain the multipliers (shadow costs) for the general linear constraints (if any).

6: ISTATE(N + M) – INTEGER array *Output*

On exit: if H02BBF exits with IFAIL = 0, 7 or 9, the values in the array ISTATE indicate the status of the constraints in the working set at an integer solution. Otherwise, ISTATE indicates the composition of the working set at the final iterate. 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 TOLFES (the feasibility tolerance, see H02BBF).
-1	The constraint violates its upper bound by more than TOLFES.
0	The constraint is satisfied to within TOLFES, 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 an integer solution being declared with x_j being temporarily fixed at its current value. This value of ISTATE can occur only when IFAIL = 0, 7 or 9 on exit from H02BBF.

7: IWORK(LIWORK) – INTEGER array *Communication Array*

On entry: this **must** be the same parameter IWORK as supplied to H02BBF. It is used to pass information from H02BBF to H02BZF and therefore the contents of this array **must not** be changed before calling H02BZF.

8: LIWORK – INTEGER *Input*

On entry: the dimension of the array IWORK as declared in the (sub)program from which H02BZF is called.

9: RWORK(LRWORK) – REAL (KIND=nag_wp) array *Communication Array*

On entry: this **must** be the same parameter RWORK as supplied to H02BBF. It is used to pass information from H02BBF to H02BZF and therefore the contents of this array **must not** be changed before calling H02BZF.

10: LRWORK – INTEGER *Input*

On entry: the dimension of the array RWORK as declared in the (sub)program from which H02BZF is called.

11: IFAIL – INTEGER *Input/Output*

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

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this parameter, the recommended value is 0. **When the value −1 or 1 is used it is essential to test the value of IFAIL on exit.**

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

6 Error Indicators and Warnings

If on entry IFAIL = 0 or −1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL = 1

On entry, N ≤ 0,
or M < 0.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

One of the applications of integer programming is to the so-called diet problem. Given the nutritional content of a selection of foods, the cost of each food, the amount available of each food and the consumer's minimum daily nutritional requirements, the problem is to find the cheapest combination. This gives rise to the following problem:

minimize

$$c^{\mathrm{T}}x$$

subject to

$$Ax \geq b, 0 \leq x \leq u,$$

where

$$c = \begin{pmatrix} 3 & 24 & 13 & 9 & 20 & 19 \end{pmatrix}^{\mathrm{T}}, x = (x_1, x_2, x_3, x_4, x_5, x_6)^{\mathrm{T}}$$

is integer,

$$A = \begin{pmatrix} 110 & 205 & 160 & 160 & 420 & 260 \\ 4 & 32 & 13 & 8 & 4 & 14 \\ 2 & 12 & 54 & 285 & 22 & 80 \end{pmatrix}, \qquad b = \begin{pmatrix} 2000 \\ 55 \\ 800 \end{pmatrix}$$

and

$$u = \begin{pmatrix} 4 & 3 & 2 & 8 & 2 & 2 \end{pmatrix}^{\mathrm{T}}$$

The rows of A correspond to energy, protein and calcium and the columns of A correspond to oatmeal, chicken, eggs, milk, pie and bacon respectively.

The following program solves the above problem to obtain the optimal integer solution and then examines the effect of increasing the energy required to 2200 units.

9.1 Program Text

```
!   H02BZF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module h02bzfe_mod

!     H02BZF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                    :: nin = 5, nout = 6
    Contains
      Subroutine outsol(n,m,a,lda,bl,bu,x,istate,clamda,bigbnd,names,nout)

!       .. Use Statements ..
        Use nag_library, Only: ddot
!       .. Parameters ..
        Character (2), Parameter            :: lstate(-2:4) = (/              &
                                               '  ','  ','FR','LL','UL',      &
                                               'EQ','TF' /)
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)     :: bigbnd
        Integer, Intent (In)                :: lda, m, n, nout
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In)     :: a(lda,*), bl(n+m), bu(n+m),    &
                                               clamda(n+m), x(n)
        Integer, Intent (In)                :: istate(n+m)
        Character (8), Intent (In)          :: names(n+m)
!       .. Local Scalars ..
        Real (Kind=nag_wp)                  :: b1, b2, res, res2, v, wlam
        Integer                             :: is, j, k
        Character (80)                      :: rec
!       .. Intrinsic Procedures ..
        Intrinsic                           :: abs
!       .. Executable Statements ..
        Write (nout,99999)

        Do j = 1, n + m
          b1 = bl(j)
          b2 = bu(j)
          wlam = clamda(j)
          is = istate(j)

          If (j<=n) Then

!           The variables  x.

            k = j
            v = x(j)
          Else

!           The linear constraints  A*x.

            If (j==n+1) Then
              Write (nout,99998)
            End If

            k = j - n
!           The NAG name equivalent of ddot is f06eaf
            v = ddot(n,a(k,1),lda,x,1)
          End If

!         Print a line for the j-th variable or constraint.

          res = v - b1
          res2 = b2 - v
```

```
          If (abs(res)>abs(res2)) Then
            res = res2
          End If

          Write (rec,99997) names(j), lstate(is), v, b1, b2, wlam, res

          If (b1<=-bigbnd) Then
            rec(29:42) = '    None     '
          End If

          If (b2>=bigbnd) Then
            rec(43:56) = '    None     '
          End If

          Write (nout,'(A)') rec
        End Do

        Return

99999   Format (//1X,'Varbl',3X,'State',5X,'Value',5X,'Lower Bound',3X, &
          'Upper Bound',4X,'Lagr Mult',3X,'Residual'/)
99998   Format (//1X,'L Con',3X,'State',5X,'Value',5X,'Lower Bound',3X, &
          'Upper Bound',4X,'Lagr Mult',3X,'Residual'/)
99997   Format (1X,A8,2X,A2,1X,1P,3G14.4,1P,G12.4,1P,G12.4)
      End Subroutine outsol
    End Module h02bzfe_mod
    Program h02bzfe

!     H02BZF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: h02bbf, h02bzf, nag_wp
      Use h02bzfe_mod, Only: nin, nout, outsol
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                  :: bigbnd, inival, objmip, tolfes,  &
                                             toliv
      Integer                             :: i, ifail, intfst, itmax, j, lda, &
                                             liwork, lrwork, m, maxdpt,       &
                                             maxnod, msglvl, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable     :: a(:,:), bl(:), bu(:), clamda(:), &
                                             cvec(:), rwork(:), x(:)
      Integer, Allocatable                :: intvar(:), istate(:), iwork(:)
      Character (8), Allocatable          :: names(:)
!     .. Intrinsic Procedures ..
      Intrinsic                           :: min
!     .. Executable Statements ..
      Write (nout,*) 'H02BZF Example Program Results'

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

      Read (nin,*) n, m
      lda = m
      Allocate (a(lda,n),bl(n+m),bu(n+m),clamda(n+m),cvec(n),x(n),intvar(n), &
        istate(n+m),names(n+m))

      Read (nin,*) itmax, msglvl
      Read (nin,*) maxnod
      Read (nin,*) intfst, maxdpt
      Read (nin,*) tolfes, toliv
      Read (nin,*) cvec(1:n)
      Read (nin,*)(names(j),a(1:m,j),j=1,n)
      Read (nin,*) bigbnd
      Read (nin,*) bl(1:n)
      Read (nin,*)(names(n+i),bl(n+i),i=1,m)
      Read (nin,*) bu(1:n+m)
      Read (nin,*) intvar(1:n)
```

```
      Read (nin,*) x(1:n)

      liwork = (25+n+m)*maxdpt + 5*n + m + 4
      lrwork = maxdpt*(n+1) + 2*min(n,m+1)**2 + 14*n + 12*m
      Allocate (iwork(liwork),rwork(lrwork))

!     Solve the IP problem using H02BBF

      ifail = -1
      Call h02bbf(itmax,msglvl,n,m,a,lda,bl,bu,intvar,cvec,maxnod,intfst, &
        maxdpt,toliv,tolfes,bigbnd,x,objmip,iwork,liwork,rwork,lrwork,ifail)

      Select Case (ifail)
      Case (0,7,9)
        Write (nout,99999) 'IP objective value = ', objmip

!       Get information about the solution

        ifail = 0
        Call h02bzf(n,m,bl,bu,clamda,istate,iwork,liwork,rwork,lrwork,ifail)

!       Print the solution

        Call outsol(n,m,a,lda,bl,bu,x,istate,clamda,bigbnd,names,nout)

!       Increase the energy requirements and solve the modified IP
!       problem using the current IP solution as the starting point

        inival = bl(n+1)
        Read (nin,*) bl(n+1)
        Write (nout,99998) 'Increase the energy requirements from', inival, &
          'to', bl(n+1)

        ifail = -1
        Call h02bbf(itmax,msglvl,n,m,a,lda,bl,bu,intvar,cvec,maxnod,intfst, &
          maxdpt,toliv,tolfes,bigbnd,x,objmip,iwork,liwork,rwork,lrwork,ifail)

        Select Case (ifail)
        Case (0,7,9)
          Write (nout,99999) 'IP objective value = ', objmip

!         Get information about the solution

          ifail = 0
          Call h02bzf(n,m,bl,bu,clamda,istate,iwork,liwork,rwork,lrwork,ifail)

!         Print the solution

          Call outsol(n,m,a,lda,bl,bu,x,istate,clamda,bigbnd,names,nout)

        End Select

      End Select

99999 Format (//1X,A,1P,G16.4)
99998 Format (//1X,A,1X,1P,G11.4,2X,A,1X,1P,G11.4)
    End Program h02bzfe
```

9.2 Program Data

```
H02BZF Example Program Data
 6  3                                                :Values of N and M
 0  0                                                :Values of ITMAX and MSGLVL
 0                                                   :Value of MAXNOD
 0  9                                                :Values of INTFST and MAXDPT
 0.0  0.0                                            :Values of TOLFES and TOLIV
 3.0  24.0  13.0  9.0  20.0  19.0                    :End of CVEC
 'Oatmeal'    110.0    4.0      2.0
 'Chicken'    205.0   32.0     12.0
 'Eggs'       160.0   13.0     54.0
```

```
'Milk'       160.0    8.0    285.0
'Pie'        420.0    4.0     22.0
'Bacon'      260.0   14.0     80.0                      :End of matrix A
1.0E+20                                                 :Value of BIGBND
0.0  0.0  0.0  0.0  0.0  0.0
'Energy'  2000.0  'Protein'  55.0  'Calcium'  800.0     :End of BL
4.0  3.0  2.0  8.0  2.0  2.0  1.0E+20  1.0E+20  1.0E+20 :End of BU
1    1    1    1    1    1                              :End of INTVAR
0.0  0.0  0.0  0.0  0.0  0.0                            :End of X
2200.0                                        :Change 'Energy' in RHS
```

9.3 Program Results

```
H02BZF Example Program Results

IP objective value =        97.00

Varbl   State     Value     Lower Bound   Upper Bound    Lagr Mult   Residual

Oatmeal   EQ      4.000        4.000         4.000        3.000        0.000
Chicken   LL      0.000        0.000         3.000        24.00        0.000
Eggs      LL      0.000        0.000         2.000        13.00        0.000
Milk      LL      5.000        5.000         8.000        9.000        0.000
Pie       EQ      2.000        2.000         2.000        20.00        0.000
Bacon     LL      0.000        0.000         2.000        19.00        0.000

L Con   State     Value     Lower Bound   Upper Bound    Lagr Mult   Residual

Energy    FR      2080.        2000.         None         0.000        80.00
Protein   FR      64.00        55.00         None         0.000        9.000
Calcium   FR      1477.        800.0         None         0.000        677.0

Increase the energy requirements from   2000.      to   2200.

IP objective value =        106.0

Varbl   State     Value     Lower Bound   Upper Bound    Lagr Mult   Residual

Oatmeal   EQ      4.000        4.000         4.000        3.000        0.000
Chicken   LL      0.000        0.000         3.000        24.00        0.000
Eggs      LL      0.000        0.000         2.000        13.00        0.000
Milk      LL      6.000        6.000         8.000        9.000        0.000
Pie       EQ      2.000        2.000         2.000        20.00        0.000
Bacon     LL      0.000        0.000         2.000        19.00        0.000

L Con   State     Value     Lower Bound   Upper Bound    Lagr Mult   Residual

Energy    FR      2240.        2200.         None         0.000        40.00
Protein   FR      72.00        55.00         None         0.000        17.00
Calcium   FR      1762.        800.0         None         0.000        962.0
```