NAG Library Routine Document

H03ABF

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

H03ABF solves the classical Transportation ('Hitchcock') problem.

2 Specification

```
SUBROUTINE H03ABF (KOST, LDKOST, MA, MB, M, K15, MAXIT, K7, K9, NUMIT,     &
                   K6, K8, K11, K12, Z, IFAIL)
INTEGER            KOST(LDKOST,MB), LDKOST, MA, MB, M, K15(M), MAXIT,      &
                   K7(M), K9(M), NUMIT, K6(M), K8(M), K11(M), K12(M),      &
                   IFAIL
REAL (KIND=nag_wp) Z
```

3 Description

H03ABF solves the Transportation problem by minimizing

$$z = \sum_{i}^{m_a} \sum_{j}^{m_b} c_{ij} x_{ij}.$$

subject to the constraints

$$\sum_{j}^{m_b} x_{ij} = A_i \quad \text{(Availabilities)}$$
$$\sum_{i}^{m_a} \sum_{i} x_{ij} = B_j \quad \text{(Requirements)}$$

where the x_{ij} can be interpreted as quantities of goods sent from source i to destination j, for $i = 1, 2, \ldots, m_a$ and $j = 1, 2, \ldots, m_b$, at a cost of c_{ij} per unit, and it is assumed that $\sum_{i}^{m_a} A_i = \sum_{j}^{m_b} \sum_{j} B_j$ and $x_{ij} \geq 0$.

H03ABF uses the 'stepping stone' method, modified to accept degenerate cases.

4 References

Hadley G (1962) *Linear Programming* Addison–Wesley

5 Parameters

1: KOST(LDKOST, MB) – INTEGER array *Input*

On entry: the coefficients c_{ij}, for $i = 1, 2, \ldots, m_a$ and $j = 1, 2, \ldots, m_b$.

2: LDKOST – INTEGER *Input*

On entry: the first dimension of the array KOST as declared in the (sub)program from which H03ABF is called.

Constraint: LDKOST $\geq$ MA.

3: MA – INTEGER *Input*

On entry: the number of sources, m_a.

Constraint: MA ≥ 1.

4: MB – INTEGER *Input*

On entry: the number of destinations, m_b.

Constraint: MB ≥ 1.

5: M – INTEGER *Input*

On entry: the value of $m_a + m_b$.

6: K15(M) – INTEGER array *Input/Output*

On entry: K15(i) must be set to the availabilities A_i, for $i = 1, 2, \ldots, m_a$; and K15($m_a + j$) must be set to the requirements B_j, for $j = 1, 2, \ldots, m_b$.

On exit: the contents of K15 are undefined.

7: MAXIT – INTEGER *Input*

On entry: the maximum number of iterations allowed.

Constraint: MAXIT ≥ 1.

8: K7(M) – INTEGER array *Workspace*

9: K9(M) – INTEGER array *Workspace*

10: NUMIT – INTEGER *Output*

On exit: the number of iterations performed.

11: K6(M) – INTEGER array *Output*

On exit: K6(k), for $k = 1, 2, \ldots, m_a + m_b - 1$, contains the source indices of the optimal solution (see K11).

12: K8(M) – INTEGER array *Output*

On exit: K8(k), for $k = 1, 2, \ldots, m_a + m_b - 1$, contains the destination indices of the optimal solution (see K11).

13: K11(M) – INTEGER array *Output*

On exit: K11(k), for $k = 1, 2, \ldots, m_a + m_b - 1$, contains the optimal quantities x_{ij} which, sent from source $i =$ K6(k) to destination $j =$ K8(k), minimize z.

14: K12(M) – INTEGER array *Output*

On exit: K12(k), for $k = 1, 2, \ldots, m_a + m_b - 1$, contains the unit cost c_{ij} associated with the route from source $i =$ K6(k) to destination $j =$ K8(k).

15: Z – REAL (KIND=nag_wp) *Output*

On exit: the value of the minimized total cost.

16: 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, the sum of the availabilities does not equal the sum of the requirements.

IFAIL $= 2$

During computation MAXIT has been exceeded.

IFAIL $= 3$

On entry, MAXIT < 1.

IFAIL $= 4$

On entry, MA < 1,
or MB < 1,
or M $\neq$ MA + MB,
or MA $>$ LDKOST.

IFAIL $= -99$

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

All operations are performed in integer arithmetic so that there are no rounding errors.

8 Parallelism and Performance

Not applicable.

9 Further Comments

An accurate estimate of the run time for a particular problem is difficult to achieve.

10 Example

A company has three warehouses and three stores. The warehouses have a surplus of 12 units of a given commodity divided among them as follows:

Warehouse	Surplus
1	1
2	5
3	6

The stores altogether need 12 units of commodity, with the following requirements:

Store	Requirement
1	4
2	4
3	4

Costs of shipping one unit of the commodity from warehouse i to store j are displayed in the following matrix:

		Store		
		1	2	3
	1	8	8	11
Warehouse	2	5	8	14
	3	4	3	10

It is required to find the units of commodity to be moved from the warehouses to the stores, such that the transportation costs are minimized. The maximum number of iterations allowed is 200.

10.1 Program Text

```
    Program h03abfe

!     H03ABF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: h03abf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: z
      Integer                          :: i, ifail, j, ldkost, m, ma, maxit,   &
                                          mb, numit
!     .. Local Arrays ..
      Integer, Allocatable             :: k11(:), k12(:), k15(:), k6(:),       &
                                          k7(:), k8(:), k9(:), kost(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'H03ABF Example Program Results'

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

      Read (nin,*) ma, mb
      m = ma + mb
      ldkost = ma
      Allocate (kost(ldkost,mb),k15(m),k7(m),k9(m),k6(m),k8(m),k11(m),k12(m))

      Read (nin,*)(k15(i),i=1,m)
```

```
      Do i = 1, ma
        Read (nin,*)(kost(i,j),j=1,mb)
      End Do

      maxit = 200

      ifail = 0
      Call h03abf(kost,ldkost,ma,mb,m,k15,maxit,k7,k9,numit,k6,k8,k11,k12,z, &
        ifail)

      Write (nout,*)
      Write (nout,99999) 'Total cost = ', z
      Write (nout,*)
      Write (nout,*) 'Goods from  to'
      Write (nout,*)
      Write (nout,99998)(k11(i),k6(i),k8(i),i=1,m-1)

99999 Format (1X,A,F5.1)
99998 Format (1X,I3,I6,I5)
    End Program h03abfe
```

10.2 Program Data

```
H03ABF Example Program Data
    3    3
    1    5    6    4    4    4
    8    8   11
    5    8   14
    4    3   10
```

10.3 Program Results

```
 H03ABF Example Program Results

 Total cost =  77.0

 Goods from  to

   4     3    2
   2     3    3
   1     2    3
   1     1    3
   4     2    1
```