NAG Toolbox

nag_mip_transportation (h03ab)

1 Purpose

nag_mip_transportation (h03ab) solves the classical Transportation ('Hitchcock') problem.

2 Syntax

```
[k15, numit, k6, k8, k11, k12, z, ifail] = nag_mip_transportation(kost, k15, maxit, 'ma', ma, 'mb', mb, 'm', m)
[k15, numit, k6, k8, k11, k12, z, ifail] = h03ab(kost, k15, maxit, 'ma', ma, 'mb', mb, 'm', m)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:
At Mark 22: **ma** was made optional.

3 Description

nag_mip_transportation (h03ab) 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 $x_{ij} \geq 0$.

nag_mip_transportation (h03ab) uses the 'stepping stone' method, modified to accept degenerate cases.

4 References

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

5 Parameters

5.1 Compulsory Input Parameters

1: **kost**($ldkost$, **mb**) – INTEGER array

$ldkost$, the first dimension of the array, must satisfy the constraint $ldkost \geq$ **ma**.

The coefficients c_{ij}, for $i = 1, 2, \ldots, m_a$ and $j = 1, 2, \ldots, m_b$.

2: **k15**(**m**) – INTEGER array

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 $j = 1, 2, \ldots, m_b$.

3: **maxit** – INTEGER

The maximum number of iterations allowed.

Constraint: **maxit** ≥ 1.

5.2 Optional Input Parameters

1: **ma** – INTEGER

Default: the first dimension of the array **kost**.

The number of sources, m_a.

Constraint: **ma** ≥ 1.

2: **mb** – INTEGER

Default: the second dimension of the array **kost**.

The number of destinations, m_b.

Constraint: **mb** ≥ 1.

3: **m** – INTEGER

Default: the dimension of the array **k15**.

The value of $m_a + m_b$.

5.3 Output Parameters

1: **k15**(**m**) – INTEGER array

The contents of **k15** are undefined.

2: **numit** – INTEGER

The number of iterations performed.

3: **k6**(**m**) – INTEGER array

k6(k), for $k = 1, 2, \ldots, m_a + m_b - 1$, contains the source indices of the optimal solution (see **k11**).

4: **k8**(**m**) – INTEGER array

k8(k), for $k = 1, 2, \ldots, m_a + m_b - 1$, contains the destination indices of the optimal solution (see **k11**).

5: **k11**(**m**) – INTEGER array

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.

6: **k12**(**m**) – INTEGER array

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).

7: **z** – REAL (KIND=nag_wp)

The value of the minimized total cost.

8: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

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.

ifail $= -399$

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

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

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

8 Further Comments

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

9 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.

9.1 Program Text

```
function h03ab_example

fprintf('h03ab example results\n\n');

m = 3 + 3;
kost = [nag_int(8), 8, 11;
                5,  8, 14;
                4,  3, 10];
k15  = [nag_int(1); 5; 6;
                4;  4; 4];
maxit = nag_int(200);

[k15, numit, k6, k8, k11, k12, z, ifail] = ...
h03ab( ...
       kost, k15, maxit);

fprintf('Total cost = %5.1f\n\n', z);
fprintf('Goods   from   to\n');
for j = 1:m-1
  fprintf('%4d%7d%6d\n',k11(j),k6(j),k8(j));
end
```

9.2 Program Results

```
h03ab example results

Total cost =  77.0

Goods   from   to
   4      3     2
   2      3     3
   1      2     3
   1      1     3
   4      2     1
```