NAG Toolbox for MATLAB

e02gc

1 Purpose

e02gc calculates an l_∞ solution to an over-determined system of linear equations.

2 Syntax

```
[a, b, relerr, x, resmax, irank, iter, ifail] = e02gc(n, a, b, relerr, 'm', m, 'tol', tol)
```

3 Description

Given a matrix A with m rows and n columns $(m \geq n)$ and a vector b with m elements, the function calculates an l_∞ solution to the over-determined system of equations

$$Ax = b.$$

That is to say, it calculates a vector x, with n elements, which minimizes the l_∞ norm of the residuals (the absolutely largest residual)

$$r(x) = \max_{1 \leq i \leq m} |r_i|$$

where the residuals r_i are given by

$$r_i = b_i - \sum_{j=1}^{n} a_{ij} x_j, \qquad i = 1, 2, \ldots, m.$$

Here a_{ij} is the element in row i and column j of A, b_i is the ith element of b and x_j the jth element of x. The matrix A need not be of full rank. The solution is not unique in this case, and may not be unique even if A is of full rank.

Alternatively, in applications where a complete minimization of the l_∞ norm is not necessary, you may obtain an approximate solution, usually in shorter time, by giving an appropriate value to the parameter **relerr**.

Typically in applications to data fitting, data consisting of m points with co-ordinates (t_i, y_i) is to be approximated in the l_∞ norm by a linear combination of known functions $\phi_j(t)$,

$$\alpha_1 \phi_1(t) + \alpha_2 \phi_2(t) + \cdots + \alpha_n \phi_n(t).$$

This is equivalent to finding an l_∞ solution to the over-determined system of equations

$$\sum_{j=1}^{n} \phi_j(t_i) \alpha_j = y_i, \qquad i = 1, 2, \ldots, m.$$

Thus if, for each value of i and j the element a_{ij} of the matrix A above is set equal to the value of $\phi_j(t_i)$ and b_i is set equal to y_i, the solution vector x will contain the required values of the α_j. Note that the independent variable t above can, instead, be a vector of several independent variables (this includes the case where each ϕ_i is a function of a different variable, or set of variables).

The algorithm is a modification of the simplex method of linear programming applied to the dual formation of the l_∞ problem (see Barrodale and Phillips 1974 and Barrodale and Phillips 1975). The modifications are designed to improve the efficiency and stability of the simplex method for this particular application.

4 References

Barrodale I and Phillips C 1974 An improved algorithm for discrete Chebyshev linear approximation *Proc. 4th Manitoba Conf. Numerical Mathematics* 177–190 University of Manitoba, Canada

Barrodale I and Phillips C 1975 Solution of an overdetermined system of linear equations in the Chebyshev norm [F4] (Algorithm 495) *ACM Trans. Math. Software* **1 (3)** 264–270

5 Parameters

5.1 Compulsory Input Parameters

1: **n – int32 scalar**

the number of unknowns, n (the number of columns of the matrix A).

Constraint: $\mathbf{n} \geq 1$.

2: **a(lda,sda) – double array**

$\mathbf{a}(j,i)$ must contain a_{ij}, the element in the ith row and jth column of the matrix A for, $i = 1,2,\ldots,m$ and $j = 1,2,\ldots,n$ (that is, the **transpose** of the matrix). The remaining elements need not be set. Preferably, the columns of the matrix A (rows of the parameter **a**) should be scaled before entry: see Section 7.

3: **b(m) – double array**

$\mathbf{b}(i)$ must contain b_i, the ith element of the vector b, for $i = 1,2,\ldots,m$.

4: **relerr – double scalar**

Must be set to a bound on the relative error acceptable in the maximum residual at the solution.

If $\mathbf{relerr} \leq 0.0$, then the l_∞ solution is computed, and **relerr** is set to 0.0 on exit.

If $\mathbf{relerr} > 0.0$, then the function obtains instead an approximate solution for which the largest residual is less than $1.0 + \mathbf{relerr}$ times that of the l_∞ solution; on exit, **relerr** contains a smaller value such that the above bound still applies. (The usual result of this option, say with $\mathbf{relerr} = 0.1$, is a saving in the number of simplex iterations).

5.2 Optional Input Parameters

1: **m – int32 scalar**

Default: The dimension of the array **b**.

the number of equations, m (the number of rows of the matrix A).

Constraint: $\mathbf{m} \geq \mathbf{n}$.

2: **tol – double scalar**

A threshold below which numbers are regarded as zero. The recommended threshold value is $10.0 \times \epsilon$, where ϵ is the ***machine precision***. If $\mathbf{tol} \leq 0.0$ on entry, the recommended value is used within the function. If premature termination occurs, a larger value for **tol** may result in a valid solution.

Suggested value: 0.0.

Default: 0.0

5.3 Input Parameters Omitted from the MATLAB Interface

sda, lda

5.4 Output Parameters

1: **a(lda,sda) – double array**

Contains the last simplex tableau.

2: **b(m) – double array**

The ith residual r_i corresponding to the solution vector x, for $i = 1, 2, \ldots, m$. Note however that these residuals may contain few significant figures, especially when **resmax** is within one or two orders of magnitude of **tol**. Indeed if **resmax** $\leq$ **tol**, the elements **b**(i) may all be set to zero. It is therefore often advisable to compute the residuals directly.

3: **relerr – double scalar**

Is altered as described above.

4: **x(n) – double array**

If **ifail** $= 0$ or 1 , **x**(j) contains the jth element of the solution vector x, for $j = 1, 2, \ldots, n$. Whether this is an l_∞ solution or an approximation to one, depends on the value of **relerr** on entry.

5: **resmax – double scalar**

If **ifail** $= 0$ or 1 , **resmax** contains the absolute value of the largest residual(s) for the solution vector x. (See **b**.)

6: **irank – int32 scalar**

If **ifail** $= 0$ or 1 , **irank** contains the computed rank of the matrix A.

7: **iter – int32 scalar**

If **ifail** $= 0$ or 1 , **iter** contains the number of iterations taken by the simplex method.

8: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Note: e02gc may return useful information for one or more of the following detected errors or warnings.

ifail $= 1$

An optimal solution has been obtained but this may not be unique (perhaps simply because the matrix A is not of full rank, i.e., **irank** $<$ **n**).

ifail $= 2$

The calculations have terminated prematurely due to rounding errors. Experiment with larger values of **tol** or try rescaling the columns of the matrix (see Section 8).

ifail $= 3$

On entry, **lda** $<$ **n** $+ 3$,
or **sda** $<$ **m** $+ 1$,
or **m** $<$ **n**,
or **n** < 1.

7 Accuracy

Experience suggests that the computational accuracy of the solution x is comparable with the accuracy that could be obtained by applying Gaussian elimination with partial pivoting to the $n + 1$ equations which have residuals of largest absolute value. The accuracy therefore varies with the conditioning of the problem, but has been found generally very satisfactory in practice.

8 Further Comments

The effects of m and n on the time and on the number of iterations in the simplex method vary from problem to problem, but typically the number of iterations is a small multiple of n and the total time is approximately proportional to mn^2.

It is recommended that, before the function is entered, the columns of the matrix A are scaled so that the largest element in each column is of the order of unity. This should improve the conditioning of the matrix, and also enable the parameter **tol** to perform its correct function. The solution x obtained will then, of course, relate to the scaled form of the matrix. Thus if the scaling is such that, for each $j = 1, 2, \ldots, n$, the elements of the jth column are multiplied by the constant k_j, the element x_j of the solution vector x must be multiplied by k_j if it is desired to recover the solution corresponding to the original matrix A.

9 Example

```
n = int32(3);
a = zeros(6, 6);
for i = 1:5
  a(1, i) = exp((i-1)/5);
  a(2, i) = exp(-(i-1)/5);
  a(3, i) = 1;
end
b = [4.501;
     4.36;
     4.333;
     4.418;
     4.625];
relerr = 0;
[aOut, bOut, relerrOut, x, resmax, irank, iter, ifail] = e02gc(n, a, b,
relerr)
```

```
aOut =
   -3.0207   -5.5042    8.8604    1.0000    0.3355    6.0000
   -1.4796   -4.9123    5.4459   -0.3289    0.0541   -4.0000
    3.0207    5.5042   -8.3604   -0.0000    0.1645    8.0000
    1.4796    4.9123   -5.9459    0.3289    0.4459   -7.0000
    1.0049    2.0149    1.4822    0.0003    0.0010         0
    1.0000    2.0000    3.0000    5.0000         0         0
bOut =
   -0.0010
    0.0007
    0.0010
   -0.0010
    0.0010
relerrOut =
     0
x =
    1.0049
    2.0149
    1.4822
resmax =
    0.0010
irank =
           3
iter =
           4
ifail =
           0
```