NAG Toolbox for MATLAB

e02gb

1 Purpose

e02gb calculates an l_1 solution to an over-determined system of linear equations, possibly subject to linear inequality constraints.

2 Syntax

```
[e, x, k, el1n, indx, ifail] = e02gb(m, e, f, x, mxs, monit, iprint,
'n', n, 'mpl', mpl)
```

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_1 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_1-norm (the sum of the absolute values) of the residuals

$$r(x) = \sum_{i=1}^{m} |r_i|,$$

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.

If, in addition, a matrix C with l rows and n columns and a vector d with l elements, are given, the vector x computed by the function is such as to minimize the l_1-norm $r(x)$ subject to the set of inequality constraints $Cx \geq d$.

The matrices A and C need not be of full rank.

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

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

in the l_1-norm, possibly subject to linear inequality constraints on the coefficients α_j of the form $C\alpha \geq d$ where α is the vector of the α_j and C and d are as in the previous paragraph. This is equivalent to finding an l_1 solution to the over-determined system of equations

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

subject to $C\alpha \geq d$.

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 equal to y_i and C and d are also supplied to the function, 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 of ϕ_i is a function of a different variable, or set of variables).

The algorithm follows the Conn–Pietrzykowski approach (see Bartels *et al.* 1978 and Conn and Pietrzykowski 1977), which is via an exact penalty function

$$g(x) = \gamma r(x) - \sum_{i=1}^{l} \min\left(0, c_i^{\mathrm{T}} x - d_i\right),$$

where γ is a penalty parameter, c_i^{T} is the ith row of the matrix C, and d_i is the ith element of the vector d. It proceeds in a step-by-step manner much like the simplex method for linear programming but does not move from vertex to vertex and does not require the problem to be cast in a form containing only nonnegative unknowns. It uses stable procedures to update an orthogonal factorization of the current set of active equations and constraints.

4 References

Bartels R H, Conn A R and Charalambous C 1976 Minimisation techniques for piecewise Differentiable functions – the l_∞ solution to an overdetermined linear system *Technical Report No. 247, CORR 76/30* Mathematical Sciences Department, The John Hopkins University

Bartels R H, Conn A R and Sinclair J W 1976 A Fortran program for solving overdetermined systems of linear equations in the l_1 Sense *Technical Report No. 236, CORR 76/7* Mathematical Sciences Department, The John Hopkins University

Bartels R H, Conn A R and Sinclair J W 1978 Minimisation techniques for piecewise differentiable functions – the l_1 solution to an overdetermined linear system *SIAM J. Numer. Anal.* **15** 224–241

Conn A R and Pietrzykowski T 1977 A penalty-function method converging directly to a constrained optimum *SIAM J. Numer. Anal.* **14** 348–375

5 Parameters

5.1 Compulsory Input Parameters

1: **m – int32 scalar**

The number of equations in the over-determined system, m (i.e., the number of rows of the matrix A).

Constraint: $\mathbf{m} \geq 2$.

2: **e(lde,mpl) – double array**

lde, the first dimension of the array, must be at least **n**.

The equation and constraint matrices stored in the following manner:

The first m columns contain the m rows of the matrix A; element $\mathbf{e}(i,j)$ specifying the element a_{ji} in the jth row and ith column of A (the coefficient of the ith unknown in the jth equation), for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. The next l columns contain the l rows of the constraint matrix C; element $\mathbf{e}(i, j + m)$ containing the element c_{ji} in the jth row and ith column of C (the coefficient of the ith unknown in the jth constraint), for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, l$.

3: **f(mpl) – double array**

$\mathbf{f}(i)$, for $i = 1, 2, \ldots, m$ must contain b_i (the ith element of the right-hand side vector of the over-determined system of equations) and $\mathbf{f}(m + i)$, for $i = 1, 2, \ldots, l$ must contain d_i (the ith element of the right-hand side vector of the constraints), where l is the number of constraints.

4: **x(n) – double array**

$\mathbf{x}(i)$ must contain an estimate of the ith unknown, for $i = 1, 2, \ldots, n$. If no better initial estimate for $\mathbf{x}(i)$ is available, set $\mathbf{x}(i) = 0.0$.

5: **mxs – int32 scalar**

The maximum number of steps to be allowed for the solution of the unconstrained problem. Typically this may be a modest multiple of n. If, on entry, **mxs** is zero or negative, the value returned by x02bb is used.

6: **monit – string containing name of m-file**

monit can be used to print out the current values of any selection of its parameters. The frequency with which **monit** is called in e02gb is controlled by **iprint**.

Its specification is:

```
[] = monit(n, x, niter, k, el1n)
```

Input Parameters

1: **n – int32 scalar**

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

2: **x(n) – double array**

The latest estimate of the unknowns.

3: **niter – int32 scalar**

The number of iterations so far carried out.

4: **k – int32 scalar**

The total number of equations and constraints which are currently active (i.e., the number of equations with zero residuals plus the number of constraints which are satisfied as equations).

5: **el1n – double scalar**

The l_1-norm of the current residuals of the over-determined system of equations.

Output Parameters

7: **iprint – int32 scalar**

The frequency of iteration print out.

iprint > 0

user-supplied (sub)program **monit** is called every **iprint** iterations and at the solution.

iprint $= 0$

Information is printed out at the solution only. Otherwise user-supplied (sub)program **monit** is not called (but a dummy function must still be provided).

5.2 Optional Input Parameters

1: **n – int32 scalar**

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

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

Constraint: $\mathbf{m} \geq \mathbf{n} \geq 2$.

2: **mpl – int32 scalar**

Default: The dimension of the arrays **e**, **f**, **indx**. (An error is raised if these dimensions are not equal.)

$m + l$, where l is the number of constraints (which may be zero).

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

5.3 Input Parameters Omitted from the MATLAB Interface

lde, w, iw

5.4 Output Parameters

1: **e(lde,mpl) – double array**

Unchanged, except possibly to the extent of a small multiple of the ***machine precision***. (See Section 8.)

2: **x(n) – double array**

The latest estimate of the ith unknown, for $i = 1, 2, \ldots, n$. If $\mathbf{ifail} = 0$ on exit, these are the solution values.

3: **k – int32 scalar**

The total number of equations and constraints which are then active (i.e., the number of equations with zero residuals plus the number of constraints which are satisfied as equalities).

4: **el1n – double scalar**

The l_1-norm (sum of absolute values) of the equation residuals.

5: **indx(mpl) – int32 array**

Specifies which columns of **e** relate to the inactive equations and constraints. $\mathbf{indx}(1)$ up to $\mathbf{indx}(\mathbf{k})$ number the active columns and $\mathbf{indx}(\mathbf{k} + 1)$ up to $\mathbf{indx}(\mathbf{mpl})$ number the inactive columns.

6: **ifail – int32 scalar**

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

$\mathbf{ifail} = 1$

The constraints cannot all be satisfied simultaneously: they are not compatible with one another. Hence no solution is possible.

$\mathbf{ifail} = 2$

The limit imposed by **mxs** has been reached without finding a solution. Consider restarting from the current point by simply calling e02gb again without changing the parameters.

$\mathbf{ifail} = 3$

The function has failed because of numerical difficulties; the problem is too ill-conditioned. Consider rescaling the unknowns.

ifail = 4

On entry, one or more of the following conditions are violated:

$\mathbf{m} \geq \mathbf{n} \geq 2$,

or $\mathbf{mpl} \geq \mathbf{m}$,

or $\mathbf{iw} \geq 3 \times \mathbf{mpl} + 5 \times \mathbf{n} + \mathbf{n}^2 + (\mathbf{n}+1) \times (\mathbf{n}+2)/2$,

or $\mathbf{lde} \geq \mathbf{n}$.

Alternatively elements 1 to **m** of one of the first **mpl** columns of the array **e** are all zero – this corresponds to a zero row in either of the matrices A or C.

7 Accuracy

The method is stable.

8 Further Comments

The effect of m and n on the time and on the number of iterations varies from problem to problem, but typically the number of iterations is a small multiple of n and the total time taken is approximately proportional to mn^2.

Linear dependencies among the rows or columns of A and C are not necessarily a problem to the algorithm. Solutions can be obtained from rank-deficient A and C. However, the algorithm requires that at every step the currently active columns of **e** form a linearly independent set. If this is not the case at any step, small, random perturbations of the order of rounding error are added to the appropriate columns of **e**. Normally this perturbation process will not affect the solution significantly. It does mean, however, that results may not be exactly reproducible.

9 Example

e02gb_monit.m

```
function [] = monit(n, x, niter, k, elin)

  fprintf('\n Results at iteration %d\n', niter);
  fprintf('X-Values\n');
  disp(transpose(x));
  fprintf('Norm of residuals = %12.5f\n', elin);
```

```
m = int32(6);
e = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0;
     0, 0.2, 0.4, 0.6, 0.8, 1, 1, 1, 1, 1, 1, 1;
      0, 0.04000000000000001, 0.16, 0.36, 0.6400000000000001, 1, 0, 0.4,
0.8, 1.2, 1.6, 2;
                0,   0.008000000000000002,   0.06400000000000002,   0.216,
0.5120000000000001, 1, 0, 0.12, 0.48, 1.08, 1.92, 3];
f = [0;
     0.07000000000000001;
     0.07000000000000001;
     0.11;
     0.27;
     0.68;
     0;
     0;
     0;
     0;
     0;
     0];
x = [0;
```

```
     0;
     0;
     0];
mxs = int32(50);
iprint = int32(0);
[eOut, xOut, k, el1n, indx, ifail] = e02gb(m, e, f, x, mxs,
'e02gb_monit', iprint)
```

```
Results at iteration 10
X-Values
         0    0.6943   -2.1482    2.1339
Norm of residuals =      0.00957
eOut =
  Columns 1 through 7
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000         0
         0    0.2000    0.4000    0.6000    0.8000    1.0000    1.0000
         0    0.0400    0.1600    0.3600    0.6400    1.0000         0
         0    0.0080    0.0640    0.2160    0.5120    1.0000         0
  Columns 8 through 12
         0         0   -0.0000    0.0000    0.0000
    1.0000    1.0000    1.0000    1.0000    1.0000
    0.4000    0.8000    1.2000    1.6000    2.0000
    0.1200    0.4800    1.0800    1.9200    3.0000
xOut =
         0
    0.6943
   -2.1482
    2.1339
k =
           4
el1n =
    0.0096
indx =
           6
           2
           9
           1
           5
          10
           4
          11
           3
           7
          12
           8
ifail =
           0
```