NAG Library Routine Document

E02GBF

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

E02GBF calculates an l_1 solution to an over-determined system of linear equations, possibly subject to linear inequality constraints.

2 Specification

```
SUBROUTINE E02GBF (M, N, MPL, E, LDE, F, X, MXS, MONIT, IPRINT, K, EL1N,    &
                   INDX, W, IW, IFAIL)

INTEGER             M, N, MPL, LDE, MXS, IPRINT, K, INDX(MPL), IW, IFAIL
REAL (KIND=nag_wp) E(LDE,MPL), F(MPL), X(N), EL1N, W(IW)
EXTERNAL            MONIT
```

3 Description

Given a matrix A with m rows and n columns $(m \geq n)$ and a vector b with m elements, the routine 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

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

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 routine 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 coordinates (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

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

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 routine, 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

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

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 non-negative 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 Arguments

1: M – INTEGER *Input*

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

Constraint: M $\geq$ N.

2: N – INTEGER *Input*

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

Constraint: N ≥ 2.

3: MPL – INTEGER *Input*

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

Constraint: MPL $\geq$ M.

4: E(LDE, MPL) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the equation and constraint matrices stored in the following manner.

The first m columns contain the m rows of the matrix A; element 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 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 $j = 1, 2, \ldots, l$.

On exit: unchanged, except possibly to the extent of a small multiple of the ***machine precision***. (See Section 9.)

5: LDE – INTEGER *Input*

On entry: the first dimension of the array E as declared in the (sub)program from which E02GBF is called.

Constraint: LDE $\geq$ N.

6: F(MPL) – REAL (KIND=nag_wp) array *Input*

On entry: 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 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.

7: X(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: X(i) must contain an estimate of the ith unknown, for $i = 1, 2, \ldots, n$. If no better initial estimate for X(i) is available, set X(i) = 0.0.

On exit: the latest estimate of the ith unknown, for $i = 1, 2, \ldots, n$. If IFAIL = 0 on exit, these are the solution values.

8: MXS – INTEGER *Input*

On entry: 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 X02BBF is used.

9: MONIT – SUBROUTINE, supplied by the user. *External Procedure*

MONIT can be used to print out the current values of any selection of its arguments. The frequency with which MONIT is called in E02GBF is controlled by IPRINT.

The specification of MONIT is:

```
SUBROUTINE MONIT (N, X, NITER, K, EL1N)
INTEGER             N, NITER, K
REAL (KIND=nag_wp) X(N), EL1N
```

1: N – INTEGER *Input*

On entry: the number n of unknowns (the number of columns of the matrix A).

2: X(N) – REAL (KIND=nag_wp) array *Input*

On entry: the latest estimate of the unknowns.

3: NITER – INTEGER *Input*

On entry: the number of iterations so far carried out.

4: K – INTEGER *Input*

On entry: 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 – REAL (KIND=nag_wp) *Input*

On entry: the l_1-norm of the current residuals of the over-determined system of equations.

MONIT must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which E02GBF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

10: IPRINT – INTEGER *Input*

On entry: the frequency of iteration print out.

IPRINT > 0
MONIT is called every IPRINT iterations and at the solution.

IPRINT $= 0$
Information is printed out at the solution only. Otherwise MONIT is not called (but a dummy routine must still be provided).

11: K – INTEGER *Output*

On exit: 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).

12: EL1N – REAL (KIND=nag_wp) *Output*

On exit: the l_1-norm (sum of absolute values) of the equation residuals.

13: INDX(MPL) – INTEGER array *Output*

On exit: specifies which columns of E relate to the inactive equations and constraints. INDX(1) up to INDX(K) number the active columns and INDX(K + 1) up to INDX(MPL) number the inactive columns.

14: W(IW) – REAL (KIND=nag_wp) array *Workspace*

15: IW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which E02GBF is called.

Constraint: IW $\geq 3 \times$ MPL $+ 5 \times$ N $+$ N$^2 + ($N $+ 1) \times ($N $+ 2)/2$.

16: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation 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 argument, 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 $= 1$

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

IFAIL $= 2$

The limit imposed by MXS has been reached without finding a solution. Consider restarting from the current point by simply calling E02GBF again without changing the arguments.

IFAIL $= 3$

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

IFAIL $= 4$

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.

On entry, IW is too small. IW $= \langle value \rangle$. Minimum possible dimension: $\langle value \rangle$.

On entry, LDE $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDE $\geq$ N.

On entry, M $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: M $\geq$ N.

On entry, MPL $= \langle value \rangle$ and M $= \langle value \rangle$.
Constraint: MPL $\geq$ M.

On entry, N $= \langle value \rangle$.
Constraint: N ≥ 2.

IFAIL $= -99$

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

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The method is stable.

8 Parallelism and Performance

E02GBF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 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.

10 Example

Suppose we wish to approximate in $[0,1]$ a set of data by a curve of the form

$$y = ax^3 + bx^2 + cx + d$$

which has non-negative slope at the data points. Given points (t_i, y_i) we may form the equations

$$y_i = at_i^3 + bt_i^2 + ct_i + d$$

for $i = 1, 2, \ldots, 6$, for the 6 data points. The requirement of a non-negative slope at the data points demands

$$3at_i^2 + 2bt_i + c \geq 0$$

for each t_i and these form the constraints.

(Note that, for fitting with polynomials, it would usually be advisable to work with the polynomial expressed in Chebyshev series form (see the E02 Chapter Introduction). The power series form is used here for simplicity of exposition.)

10.1 Program Text

```
!   E02GBF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.
    Module e02gbfe_mod

!     E02GBF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: monit
!     .. Parameters ..
      Integer, Parameter, Public       :: n = 4, nin = 5, nout = 6
    Contains
      Subroutine monit(n,x,niter,k,el1n)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: el1n
        Integer, Intent (In)            :: k, n, niter
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: x(n)
!       .. Executable Statements ..
        Write (nout,*)
        Write (nout,99999) 'Results at iteration ', niter
        Write (nout,*) 'X-values'
        Write (nout,99998) x
        Write (nout,99997) 'Norm of residuals =', el1n

        Return

99999   Format (1X,A,I5)
99998   Format (1X,4F15.4)
99997   Format (1X,A,E12.5)
      End Subroutine monit
```

```
    End Module e02gbfe_mod
    Program e02gbfe

!     E02GBF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: e02gbf, nag_wp
      Use e02gbfe_mod, Only: monit, n, nin, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: el1n, t
      Integer                         :: i, ifail, iprint, iw, k, l, lde, m,  &
                                         mpl, mxs
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: e(:,:), f(:), w(:), x(:)
      Integer, Allocatable             :: indx(:)
!     .. Executable Statements ..
      Write (nout,*) 'E02GBF Example Program Results'

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

      Read (nin,*) m
      lde = n
      l = m
      mpl = m + l
      iw = 3*mpl + 5*n + n**2 + (n+1)*(n+2)/2
      Allocate (e(lde,mpl),f(mpl),x(n),indx(mpl),w(iw))

      Do i = 1, m
        Read (nin,*) t, f(i)
        e(1:4,i) = (/1.0_nag_wp,t,t*t,t*t*t/)
        e(1:4,m+i) = (/0.0_nag_wp,1.0_nag_wp,2.0_nag_wp*t,3.0_nag_wp*t*t/)
        f(m+i) = 0.0_nag_wp
      End Do

      x(1:n) = 0.0_nag_wp
      mxs = 50

!     * Set IPRINT=1 to obtain output from MONIT at each iteration *
      iprint = 0

      ifail = -1
      Call e02gbf(m,n,m+l,e,lde,f,x,mxs,monit,iprint,k,el1n,indx,w,iw,ifail)

    End Program e02gbfe
```

10.2 Program Data

```
E02GBF Example Program Data
  6
  0.00  0.00
  0.20  0.07
  0.40  0.07
  0.60  0.11
  0.80  0.27
  1.00  0.68
```

10.3 Program Results

```
 E02GBF Example Program Results

 Results at iteration     8
 X-values
          0.0000         0.6943        -2.1482         2.1339
 Norm of residuals = 0.95714E-02
```