

# NAG Library Routine Document

## E02GAF

**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

E02GAF calculates an  $l_1$  solution to an over-determined system of linear equations.

### 2 Specification

```
SUBROUTINE E02GAF (M, A, LDA, B, NPLUS2, TOLER, X, RESID, IRANK, ITER,      &
                  IWORK, IFAIL)
INTEGER           M, LDA, NPLUS2, IRANK, ITER, IWORK(M), IFAIL
REAL (KIND=nag_wp) A(LDA,NPLUS2), B(M), TOLER, X(NPLUS2), RESID
```

### 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

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

Here  $a_{ij}$  is the element in row  $i$  and column  $j$  of  $A$ ,  $b_i$  is the  $i$ th element of  $b$  and  $x_j$  the  $j$ th element of  $x$ . The matrix  $A$  need not be of full rank.

Typically in applications to data fitting, data consisting of  $m$  points with coordinates  $(t_i, y_i)$  are to be approximated in the  $l_1$  norm by a linear combination of known functions  $\phi_j(t)$ ,

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

This is equivalent to fitting an  $l_1$  solution to the over-determined system of equations

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

Thus if, for each value of  $i$  and  $j$ , the element  $a_{ij}$  of the matrix  $A$  in the previous paragraph 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  $\alpha_j$ . Note that the independent variable  $t$  above can, instead, be a vector of several independent variables (this includes the case where each  $\phi_i$  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 primal formulation of the  $l_1$  problem (see Barrodale and Roberts (1973) and Barrodale and Roberts (1974)). The modification allows several neighbouring simplex vertices to be passed through in a single iteration, providing a substantial improvement in efficiency.

## 4 References

Barrodale I and Roberts F D K (1973) An improved algorithm for discrete  $l_1$  linear approximation *SIAM J. Numer. Anal.* **10** 839–848

Barrodale I and Roberts F D K (1974) Solution of an overdetermined system of equations in the  $l_1$ -norm *Comm. ACM* **17(6)** 319–320

## 5 Parameters

- 1: M – INTEGER *Input*  
*On entry:* the number of equations,  $m$  (the number of rows of the matrix  $A$ ).  
*Constraint:*  $M \geq n \geq 1$ .
- 2: A(LDA, NPLUS2) – REAL (KIND=nag\_wp) array *Input/Output*  
*On entry:*  $A(i, j)$  must contain  $a_{ij}$ , the element in the  $i$ th row and  $j$ th column of the matrix  $A$ , for  $i = 1, 2, \dots, m$  and  $j = 1, 2, \dots, n$ . The remaining elements need not be set.  
*On exit:* contains the last simplex tableau generated by the simplex method.
- 3: LDA – INTEGER *Input*  
*On entry:* the first dimension of the array  $A$  as declared in the (sub)program from which E02GAF is called.  
*Constraint:*  $LDA \geq M + 2$ .
- 4: B(M) – REAL (KIND=nag\_wp) array *Input/Output*  
*On entry:*  $B(i)$  must contain  $b_i$ , the  $i$ th element of the vector  $b$ , for  $i = 1, 2, \dots, m$ .  
*On exit:* the  $i$ th residual  $r_i$  corresponding to the solution vector  $x$ , for  $i = 1, 2, \dots, m$ .
- 5: NPLUS2 – INTEGER *Input*  
*On entry:*  $n + 2$ , where  $n$  is the number of unknowns (the number of columns of the matrix  $A$ ).  
*Constraint:*  $3 \leq NPLUS2 \leq M + 2$ .
- 6: TOLER – REAL (KIND=nag\_wp) *Input*  
*On entry:* a non-negative value. In general TOLER specifies a threshold below which numbers are regarded as zero. The recommended threshold value is  $\epsilon^{2/3}$  where  $\epsilon$  is the *machine precision*. The recommended value can be computed within the routine by setting TOLER to zero. If premature termination occurs a larger value for TOLER may result in a valid solution.  
*Suggested value:* 0.0.
- 7: X(NPLUS2) – REAL (KIND=nag\_wp) array *Output*  
*On exit:*  $X(j)$  contains the  $j$ th element of the solution vector  $x$ , for  $j = 1, 2, \dots, n$ . The elements  $X(n + 1)$  and  $X(n + 2)$  are unused.
- 8: RESID – REAL (KIND=nag\_wp) *Output*  
*On exit:* the sum of the absolute values of the residuals for the solution vector  $x$ .
- 9: IRANK – INTEGER *Output*  
*On exit:* the computed rank of the matrix  $A$ .

- 10: ITER – INTEGER *Output*  
*On exit:* the number of iterations taken by the simplex method.
- 11: IWORK(M) – INTEGER array *Workspace*
- 12: 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

An optimal solution has been obtained but this may not be unique.

IFAIL = 2

The calculations have terminated prematurely due to rounding errors. Experiment with larger values of TOLER or try scaling the columns of the matrix (see Section 9).

IFAIL = 3

On entry, NPLUS2 < 3,  
 or NPLUS2 > M + 2,  
 or LDA < M + 2.

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

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  $n$  equations satisfied by this algorithm (i.e., those equations with zero residuals). The accuracy therefore varies with the conditioning of the problem, but has been found generally very satisfactory in practice.

## 8 Parallelism and Performance

Not applicable.

## 9 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 taken is approximately proportional to  $mn^2$ .

It is recommended that, before the routine 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 TOLER 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  $j = 1, 2, \dots, n$ , the elements of the  $j$ th 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  $A$ .

## 10 Example

Suppose we wish to approximate a set of data by a curve of the form

$$y = Ke^t + Le^{-t} + M$$

where  $K$ ,  $L$  and  $M$  are unknown. Given values  $y_i$  at 5 points  $t_i$  we may form the over-determined set of equations for  $K$ ,  $L$  and  $M$

$$e^{x_i}K + e^{-x_i}L + M = y_i, \quad i = 1, 2, \dots, 5.$$

E02GAF is used to solve these in the  $l_1$  sense.

### 10.1 Program Text

```

Program e02gafe
!      E02GAF Example Program Text
!
!      Mark 25 Release. NAG Copyright 2014.
!
!      .. Use Statements ..
Use nag_library, Only: e02gaf, nag_wp
!      .. Implicit None Statement ..
Implicit None
!      .. Parameters ..
Integer, Parameter          :: nin = 5, nout = 6, nplus2 = 5
!      .. Local Scalars ..
Real (Kind=nag_wp)         :: resid, t, toler
Integer                    :: i, ifail, irank, iter, lda, m
!      .. Local Arrays ..
Real (Kind=nag_wp), Allocatable :: a(:,,:), b(:), x(:)
Integer, Allocatable       :: iwork(:)
!      .. Intrinsic Procedures ..
Intrinsic                  :: exp
!      .. Executable Statements ..
Write (nout,*) 'E02GAF Example Program Results'
!
!      Skip heading in data file

```

```

Read (nin,*)

Read (nin,*) m
lda = m + 2
Allocate (a(lda,nplus2),iwork(m),b(m),x(nplus2))

Do i = 1, m
  Read (nin,*) t, b(i)
  a(i,1) = exp(t)
  a(i,2) = exp(-t)
End Do

a(1:m,3) = 1.0E0_nag_wp
toler = 0.0E0_nag_wp

ifail = -1
Call e02gaf(m,a,lda,b,nplus2,toler,x,resid,irank,iter,iwork,ifail)

Select Case (ifail)
Case (0,1)
  Write (nout,*)
  Write (nout,99999) 'Resid = ', resid, ' Rank = ', irank, &
    ' Iterations = ', iter, ' IFAIL =', ifail
  Write (nout,*)
  Write (nout,*) 'Solution'
  Write (nout,99998) x(1:(nplus2-2))
End Select

99999 Format (1X,A,E10.2,A,I5,A,I5,A,I5)
99998 Format (1X,6F10.4)
End Program e02gafe

```

## 10.2 Program Data

```

E02GAF Example Program Data
5
0.0 4.501
0.2 4.360
0.4 4.333
0.6 4.418
0.8 4.625

```

## 10.3 Program Results

E02GAF Example Program Results

Resid = 0.28E-02 Rank = 3 Iterations = 5 IFAIL = 0

Solution  
1.0014 2.0035 1.4960

---