NAG Library Routine Document

E04PCF

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

E04PCF solves a linear least squares problem with bounds on the variables.

2 Specification

```
SUBROUTINE E04PCF (ITYPE, M, N, A, LDA, B, BL, BU, TOL, X, RNORM, NFREE, W,     &
                   INDX, IFAIL)

INTEGER            ITYPE, M, N, LDA, NFREE, INDX(N), IFAIL
REAL (KIND=nag_wp) A(LDA,*), B(M), BL(N), BU(N), TOL, X(N), RNORM, W(N)
```

3 Description

Given an m by n matrix A, and an m-vector, b, E04PCF computes an n-vector, x, that solves the least-squares problem $Ax = b$ subject to $x(i)$ satisfying $l_i \leq x(i) \leq u_i$.

A facility is provided to return a 'regularized' solution, which will closely approximate a minimal length solution whenever A is not of full rank.

4 References

Lawson C L and Hanson R J (1974) *Solving Least-squares Problems* Prentice–Hall

5 Parameters

1: ITYPE – INTEGER *Input*

On entry:

ITYPE $= 0$
Specifies that a regularized solution will be computed.

ITYPE $= 1$
Specifies that no regularization is to take place.

2: M – INTEGER *Input*

On entry: m, the number of equations.

Constraint: M ≥ 0.

3: N – INTEGER *Input*

On entry: n, the number of variables.

Constraint: N ≥ 0.

4: A(LDA,$*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array A must be at least N.

On entry: the m by n matrix A.

On exit: if ITYPE = 1, A contains the product matrix, QA, where Q is an m by m orthogonal matrix generated by E04PCF; otherwise A is unchanged.

5: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which E04PCF is called.

Constraint: LDA $\geq$ M.

6: B(M) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the right-hand side vector b.

On exit: if ITYPE = 1, the product of Q times the original vector B; otherwise B is unchanged.

7: BL(N) – REAL (KIND=nag_wp) array *Input*
8: BU(N) – REAL (KIND=nag_wp) array *Input*

On entry: BL(i) and BU(i) must specify the lower and upper bounds, l_i and u_i respectively to be imposed on the solution vector $x(i)$.

Constraint: BL$(i) \leq$ BU(i), for $i = 1, 2, \ldots,$N.

9: TOL – REAL (KIND=nag_wp) *Input*

On entry: TOL specifies a parameter used to determine the relative linear dependence of a column vector for a variable moved from its initial value. It determines the computational rank of the matrix.

If on entry TOL < ***machine precision***, then ***machine precision*** is used.

10: X(N) – REAL (KIND=nag_wp) array *Output*

On exit: the solution vector x.

11: RNORM – REAL (KIND=nag_wp) *Output*

On exit: the Euclidean norm of the residual vector, $b - Ax$.

12: NFREE – INTEGER *Output*

On exit: indicates the number of components of the solution vector that are not at one of the constraints.

13: W(N) – REAL (KIND=nag_wp) array *Output*

On exit: contains the dual solution vector.

A value of W(i) equal to the special value -999 is indicative of the matrix A not having full rank. It is only likely to occur when ITYPE = 1. However a matrix may have less than full rank without W(i) being set to -999. Under these circumstances the value of W(i), and hence INDX(i), may be unreliable. If you have any doubts set ITYPE = 0. Otherwise values have the following meaning:

W$(i) = 0$
    If $x(i)$ is unconstrained.

W$(i) < 0$
    If $x(i)$ is constrained by its lower bound.

W$(i) > 0$
    If $x(i)$ is constrained by its upper bound.

W(i)
    May be any value if $l_i = u_i$.

14: INDX(N) – INTEGER array *Output*

On exit: the contents of this array describe the components of the solution vector as follows:

INDX(i), for $i = 1, 2, \ldots, \mathrm{NFREE}$
These elements have not hit a constraint i.e., W(i) = 0.

INDX(i), for $i = \mathrm{NFREE} + 1, \ldots, k$
These elements have been constrained by either the lower or upper bound constraint.

INDX(i), for $i = k + 1, \ldots, \mathrm{N}$
These elements are fixed by the bounds i.e., BL(i) = BU(i).

Here k is determined from NFREE and the number of fixed components. (Often the latter will be 0, so k will be N – NFREE.)

15: 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, because for this routine the values of the output parameters may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **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).

Note: E04PCF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL = 1

On entry, M = ⟨*value*⟩ and LDA = ⟨*value*⟩.
Constraint: LDA $\geq$ M.

On entry, M = ⟨*value*⟩ and N = ⟨*value*⟩.
Constraint: M ≥ 0 and N ≥ 0.

On entry, when i = ⟨*value*⟩, BL(i) = ⟨*value*⟩ and BU(i) = ⟨*value*⟩.
Constraint: BL(i) $\leq$ BU(i).

IFAIL = 2

Failed to converge in ⟨*value*⟩ iterations.

IFAIL = -999

Dynamic memory allocation failed.

7 Accuracy

Orthogonal rotations are used.

8 Further Comments

If either M or N is zero on entry then E04PCF sets IFAIL $= 0$ and simply returns without setting any other output variables.

9 Example

The example minimizes $\|Ax - b\|_2$ where

$$A = \begin{pmatrix} 0.05 & 0.05 & 0.25 & -0.25 \\ 0.25 & 0.25 & 0.05 & -0.05 \\ 0.35 & 0.35 & 1.75 & -1.75 \\ 1.75 & 1.75 & 0.35 & -0.35 \\ 0.30 & -0.30 & 0.30 & 0.30 \\ 0.40 & -0.40 & 0.40 & 0.40 \end{pmatrix}$$

and

$$b = \begin{pmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 5.0 & 6.0 \end{pmatrix}^{\mathrm{T}}$$

subject to $1 \leq x \leq 5$.

9.1 Program Text

```
    Program e04pcfe
!     e04pcf Example Program Text
!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: e04pcf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: tol = 0.0_nag_wp
      Integer, Parameter               :: itype = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: rnorm
      Integer                          :: i, ifail, lda, lwork, m, n, nfree
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), b(:), bl(:), bu(:), w(:), x(:)
      Integer, Allocatable             :: indx(:)
!     .. Executable Statements ..
      Write (nout,*) 'E04PCF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, n
      lda = m
      lwork = 4*n
      Allocate (a(lda,n),b(m),w(lwork),bl(n),bu(n),x(n))
      Allocate (indx(n))
      Read (nin,*)(a(i,1:n),i=1,m)
      Read (nin,*) b(1:m)
      Read (nin,*) bl(1:n)
      Read (nin,*) bu(1:n)
      ifail = 0
      Call e04pcf(itype,m,n,a,lda,b,bl,bu,tol,x,rnorm,nfree,w,indx,ifail)

      Write (nout,99999) 'Solution vector', x(1:n)
      Write (nout,*)
      Write (nout,99999) 'Dual Solution', w(1:n)
      Write (nout,*)
      Write (nout,99998) 'Residual', rnorm

99999 Format (1X,A/1X,8F9.4)
99998 Format (1X,A,1X,1F9.4)
    End Program e04pcfe
```

9.2 Program Data

```
E04PCF Example Program Data
  6  4                              : m, n
  0.05  0.05  0.25 -0.25
  0.25  0.25  0.05 -0.05
  0.35  0.35  1.75 -1.75
  1.75  1.75  0.35 -0.35
  0.30 -0.30  0.30  0.30
  0.40 -0.40  0.40  0.40            : matrix A
  1.0   2.0   3.0   4.0   5.0   6.0   : vector B
  1.0   1.0   1.0   1.0             : Lower bounds
  5.0   5.0   5.0   5.0             : Upper bounds
```

9.3 Program Results

```
 E04PCF Example Program Results

 Solution vector
    1.8133   1.0000   5.0000   4.3467

 Dual Solution
    0.0000  -2.7200   2.7200   0.0000

 Residual    3.4246
```