E04FDFP

NAG Parallel Library Routine Document

Note: before using this routine, please read the Users' Note for your implementation to check for implementation-dependent details. You are advised to enclose any calls to NAG Parallel Library routines between calls to Z01AAFP and Z01ABFP.

1 Description

E04FDFP is an easy-to-use algorithm for finding an unconstrained minimum of a sum of squares of m nonlinear functions in n variables:

$$\min_x F(x) = \sum_{i=1}^{m} f_i^2(x), \quad x \in \mathbb{R}^n,$$

where $x = (x_1, x_2, \ldots, x_n)^T$ and $m \geq n$. The user need not provide the Jacobian. It is intended for functions which are smooth although it may cope with occasional discontinuities.

2 Specification

```
   SUBROUTINE E04FDFP(ICNTXT, M, N, NB, ITERS, TAU, X, FVAL, FUNC, W,
  1                   LW, IW, LIW, IFAIL)
   DOUBLE PRECISION   TAU, X(N), FVAL, W(LW)
   INTEGER            ICNTXT, M, N, NB, ITERS, LW, IW(LIW), LIW, IFAIL
   EXTERNAL           FUNC
```

3 Usage

3.1 Definitions

The following definitions are used in describing the data distribution within this document:

m_p	–	the number of rows in the Library Grid.
n_p	–	the number of columns in the Library Grid.
p_r	–	the row grid coordinate of the calling processor.
p_c	–	the column grid coordinate of the calling processor.
N_b	–	the blocking factor for the distribution of the rows and columns of the Jacobian matrix.
numroc(α,b_ℓ,q,s,k)	–	a function which gives the **num**ber of **r**ows **o**r **c**olumns of a distributed matrix owned by the processor with the row or column coordinate q (p_r or p_c), where α is the total number of rows or columns of the matrix, b_ℓ is the blocking factor used (N_b), s is the row or column coordinate of the processor that possesses the first row or column of the distributed matrix and k is either n_p or m_p. The Library provides the function Z01CAFP (NUMROC) for the evaluation of this function.

3.2 Global and Local Arguments

The following global **input** arguments must have the same value on entry to the routine on each processor and the global **output** arguments will have the same value on exit from the routine on each processor:

Global input arguments: M, N, NB, ITERS, TAU, X, IFAIL

Global output arguments: ITERS, TAU, X, FVAL, IFAIL

The remaining arguments are local.

4 Arguments

1: ICNTXT — INTEGER *Local Input*

On entry: the Library context, usually returned by a call to the Library Grid initialisation routine Z01AAFP.

Note: the value of ICNTXT **must not** be changed.

2: M — INTEGER *Global Input*

On entry: m, the number of nonlinear terms in function $F(x)$.

Constraint: M ≥ 0

3: N — INTEGER *Global Input*

On entry: n, the number of independent variables.

Constraint: $0 \leq$ N $\leq$ M

4: NB — INTEGER *Global Input*

On entry: the matrix block size to be used for the solution of the least-squares problem at each iteration (see Section 2.6.2 of the E04 Chapter Introduction and Section 2.6.3 of the E04 Chapter Introduction).

Constraint: NB ≥ 1

5: ITERS — INTEGER *Global Input/Global Output*

On entry: the maximum number of iterations.

On exit: the actual number of iterations that the routine has performed.

Constraint: ITERS ≥ 0

6: TAU — DOUBLE PRECISION *Global Input/Global Output*

On entry: the relative accuracy to which the value of the objective function is required.

On exit: if TAU is smaller than the ***machine precision*** ϵ, E04FDFP will reset it to the default value ϵ.

7: X(N) — DOUBLE PRECISION array *Global Input/Global Output*

On entry: x, an initial approximation close to the (expected) local minimum.

On exit: the minimum found during the computation. On successful exit, X(j) is the jth component of the position of the minimum.

8: FVAL — DOUBLE PRECISION *Global Output*

On exit: the value of the sum of squares, $F(x)$, corresponding to the final point stored in X.

9: FUNC — SUBROUTINE, supplied by the user. *External Procedure*

This subroutine must be supplied by the user to calculate the vector of values $(f_1(x), \ldots, f_m(x))^T$ at any point x.

Its specification is:

```
SUBROUTINE          FUNC(M, N, F, X)
INTEGER             M, N
DOUBLE PRECISION    F(M), X(N)
```

1: M — INTEGER *Global Input*

On entry: the number of residuals m.

2: N — INTEGER *Global Input*
On entry: the number of variables n.

3: F(M) — DOUBLE PRECISION array *Local Output*
On exit: F(i) must contain the value of f_i at the point x, for $i = 1, 2, \ldots, m$.

4: X(N) — DOUBLE PRECISION array *Local Input*
On entry: the point x at which the values of the $f_i(x)$ are required.

FUNC must be declared as EXTERNAL in the (sub)program from which E04FDFP is called. Arguments denoted as *Input* must **not** be changed by this procedure.

10: W(LW) — DOUBLE PRECISION array *Local Workspace*

11: LW — INTEGER *Local Input*

On entry: the size of the array W as declared in the (sub)program from which E04FDFP is called.

Constraint: let M_l = numroc(M,NB,p_r,0,m_p) and N_l = numroc(N,NB,p_c,0,n_p), then

$$\text{LW} \geq M_l \times N_l + M_l + N_l + \text{M} + \text{N} + \max \begin{cases} 2 \times \text{M}, \\ \text{NB} \times (M_l + N_l + \text{NB}), \\ \text{N} + N_l, \\ \frac{3}{2}\text{NB}^2. \end{cases}$$

12: IW(LIW) — INTEGER array *Local Workspace*

13: LIW — INTEGER *Local Input*

On entry: the size of the array IW as declared in the (sub)program from which E04FDFP is called.

Constraint: LIW $\geq$ numroc(N,NB,p_c,0,n_p).

14: IFAIL — INTEGER *Global Input/Global Output*

The NAG Parallel Library provides a mechanism, via the routine Z02EAFP, to reduce the amount of parameter validation performed by this routine. For a full description refer to the Z02 Chapter Introduction.

On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this argument (described in the Essential Introduction) the recommended values are:

IFAIL = 0, if multigridding is **not** employed;
IFAIL = -1, if multigridding is employed.

On exit: IFAIL = 0 (or -9999 if reduced error checking is enabled) unless the routine detects an error (see Section 5).

5 Errors and Warnings

If on entry IFAIL = 0 or -1, explanatory error messages are output from the root processor (or processor {0,0} when the root processor is not available) on the current error message unit (as defined by X04AAF).

5.1 Full Error Checking Mode Only

IFAIL = -2000

The routine has been called with an invalid value of ICNTXT on one or more processors.

IFAIL = −1000

The logical processor grid and library mechanism (Library Grid) have not been correctly defined, see Z01AAFP.

IFAIL = $-i$

On entry, the ith argument was invalid. This error occured either because a global argument did not have the same value on all logical processors, or because its value on one or more processors was incorrect. An explanatory message distinguishes between these two cases.

5.2 Any Error Checking Mode

IFAIL = 1

The routine failed to find a solution to the required accuracy within the maximum number of iterations specified. This may be due to the user specifying too great an accuracy requirement.

IFAIL =2

The routine has calculated a Jacobian for the objective function which is nearly singular. If this error occurs it is worthwhile retrying the problem with a different starting point.

IFAIL = 3

The routine has failed in the line search. This may be caused by the user requesting too much accuracy or the routine may have encountered a discontinuity in the function.

6 Further Comments

The routine is intended for problems which do not generate rank-deficient Jacobians. The routine may also fail to find a solution when the residual is not small.

The number of iterations required depends on the number of variables, the number of residuals and their behaviour, and the distance of the starting point from the solution.

Ideally, the problem should be scaled so that the minimum value of the sum of squares is in the range (0,1), and so that at points a unit distance away from the solution the sum of squares is approximately a unit value greater than the minimum. It is unlikely that the user will be able to follow these recommendations very closely, but it is worth trying (by guesswork), as sensible scaling will reduce the difficulty of the minimization problem, consequently E04FDFP will take less computer time.

6.1 Algorithmic Detail

The routine uses a Gauss–Newton method to generate improved estimates of the minimising position. On each iteration the Jacobian matrix, J, of the objective function is estimated numerically and a search direction, p, is calculated by solving the linear least-squares problem:

$$\min_p \|Jp + f\|_2,$$

where f is the vector of function values, $f_i = f_i(x)$. A QR factorization is used to solve this system of equations. Once the search direction has been generated, a one-dimensional minimization with respect to the search direction p is used to calculate a step length α and the solution vector is updated:

$$x^{(k+1)} = x^{(k)} + \alpha p^{(k)},$$

where $x^{(k)}$ denotes the kth iterate. The QR factorization is performed using F08AEFP (PDGEQRF). See Chapter F08 for further details.

6.2 Parallelism Detail

The elements of the Jacobian can all be evaluated completely in parallel and this is the principal source of parallelism. The solution of the system of equations also involves some parallelism, the QR factorization and solve are described in Chapter F08.

6.3 Accuracy

If the routine completes successfully (on exit IFAIL = 0) then the value of the objective function $F(x)$ at the minimum point is given to an approximate accuracy of $\tau(1 + F_{\min})$ and the position of the minimum is given to an accuracy of $||x_k - x_{\min}|| \leq \sqrt{\tau}(1 + ||x_{\min}||)$, where $x_{\min}$ is the position of the exact minimum of $F(x)$ and x_k is the computed position of the minimum returned by E04FDFP.

7 References

[1] Gill P E, Murray W and Wright M H (1981) *Practical Optimization* Academic Press

8 Example

This example calculates the minimum point of Rosenbrock's function,

$$F(x) = f_1^2(x) + f_2^2(x),$$

where $f_1(x) = 10(x_2 - x_1^2)$ and $f_2(x) = 1 - x_1$ starting from the point $x = (-1.2, 1.0)^T$. The minimum of the function occurs at $x = (1.0, 1.0)^T$ and has a value 0.0.

8.1 Example Text

```
*     E04FDFP Example Program Text
*     NAG Parallel Library Release 2. NAG Copyright 1996.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          NB
      PARAMETER        (NB=1)
      INTEGER          M, N, RCONST, CCONST, LIW
      PARAMETER        (M=2,N=2,RCONST=2,CCONST=2,LIW=1000)
      INTEGER          LW
      PARAMETER        (LW=1000)
*     .. Local Scalars ..
      DOUBLE PRECISION FVAL, TAU
      INTEGER          ICNTXT, IFAIL, ITERS, MP, NP
      LOGICAL          ROOT
*     .. Local Arrays ..
      DOUBLE PRECISION W(LW), X(N)
      INTEGER          IW(LIW)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         E04FDFP, FUNC, Z01AAFP, Z01ABFP
*     .. Executable Statements ..
      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'E04FDFP Example Program Results'
*
      MP = RCONST
      NP = CCONST
*
      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
      ITERS = 20
      TAU = 1.0D-4
      X(1) = -1.2D0
```

```
      X(2) = 1.0D0
*
*     Compute the minimum
*
      IFAIL = 0
      CALL E04FDFP(ICNTXT,M,N,NB,ITERS,TAU,X,FVAL,FUNC,W,LW,IW,LIW,
     +             IFAIL)
*
*     Print the lowest point
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,99999) FVAL
         WRITE (NOUT,99998) X(1), X(2), ITERS
      END IF
*
      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'N',IFAIL)
*
      STOP
*
99999 FORMAT (1X,'On exit, the sum of squares = ',F6.4)
99998 FORMAT (1X,'at the point x = (',2F6.2,' )  in',I3,' iterations')
      END
*
      SUBROUTINE FUNC(M,N,F,X)
*     .. Scalar Arguments ..
      INTEGER          M, N
*     .. Array Arguments ..
      DOUBLE PRECISION F(M), X(N)
*     .. Executable Statements ..
      F(1) = 10.0D0*(X(2)-X(1)*X(1))
      F(2) = 1.0D0 - X(1)
      RETURN
      END
```

8.2 Example Data

None.

8.3 Example Results

```
E04FDFP Example Program Results

On exit, the sum of squares = 0.0000
at the point x = (  1.00  1.00 )  in 15 iterations
```