E04JBFP

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

E04JBFP is an easy-to-use Newton algorithm for finding a minimum of a function $F(x)$, where $x = (x_1, x_2, \ldots, x_n)^T$, using function values only. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

2 Specification

```
SUBROUTINE E04JBFP(ICNTXT, N, NB, X, TAU, F, FUNC, IW, LIW, W, LW,
1                  IFAIL)
INTEGER            ICNTXT, N, NB, IW(LIW), LIW, LW, IFAIL
DOUBLE PRECISION   X(N), TAU, F, W(LW)
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: N, NB, X, TAU, IFAIL

Global output arguments: X, TAU, F, 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: N — INTEGER *Global Input*

On entry: n, the number of independent variables x_j.

Constraint: N ≥ 1

3: NB — INTEGER *Global Input*

On entry: the matrix block size to be used for the cyclic two-dimensional block distribution (see the the F07 Chapter Introduction for details) of the Hessian and gradient of the objective function used in the solution of the system of equations generated on 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

4: 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.

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

On entry: TAU should be set to the relative accuracy required in the function value at the minimum (but not less than the accuracy to which the function values can be computed).

On exit: if TAU is set to less than ***machine precision*** it will be reset to ***machine precision***.

6: F — DOUBLE PRECISION *Global Output*

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

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

This subroutine must calculate the value $F(x)$ at any point x.

Its specification is:

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

1: N — INTEGER *Global Input*

On entry: the number of variables n.

2: X(N) — DOUBLE PRECISION array *Local Input*

On entry: the point x at which the value of the f is required.

3: F — DOUBLE PRECISION *Local Output*

On exit: the function value at the point x.

FUNC must be declared as EXTERNAL in the (sub)program from which E04JBFP is called. Arguments denoted as *Input* must **not** be changed by this procedure.

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

9: LIW — INTEGER *Local Input*

On entry: the dimension of the array IW as declared in the (sub)program from which E04JBFP is called.

Constraint: LIW $\geq$ N.

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

11: LW — INTEGER *Local Input*

On entry: the dimension of the array W as declared in the (sub)program from which E04JBFP is called.

Constraint: LW $\geq 2 \times$ N + numroc(N,NB,p_r,0,m_p) $\times$ (1+numroc(N,NB,p_c,0,n_p)).

12: 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

E04JBFP was unable to solve the system of equations, $Hp = -\nabla F$ generated (see Section 6.1). This may be due to the Hessian of the function being singular at some points.

IFAIL = 2

E04JBFP has failed in the one-dimensional line search. This may be due to requesting too much accuracy or due to discontinuities in the function.

IFAIL = 3

There have been $64 \times n$ iterations, yet the algorithm does not seem to be converging. The calculation can be restarted from the final point held in X. The error may also indicate that $F(x)$ has no minimum.

overflow

It is possible for an arithmetic overflow to occur during the execution of E04JBFP. Since the routine is unconstrained, there is no certain way of preventing this. The user should ensure that the problem is well-scaled, as described in the the E04 Chapter Introduction.

6 Further Comments

The number of iterations required depends on the number of variables, the behaviour of $F(x)$ and the distance of the starting point from the solution. Unless $F(x)$ can be evaluated very efficiently, the run time will be dominated by the time spent in FUNC. Ideally, the problem should be scaled so that at the solution the value of $F(x)$ and the corresponding values of $x_1, x_2, \ldots, x_n$ are each in the range $(-1, +1)$, and so that at points a unit distance away from the solution, F is approximately a unit value greater than at 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 a sensible scaling will reduce the difficulty of the minimization problem so that E04JBFP will take less computer time.

6.1 Algorithmic Detail

On each iteration, the Hessian, H, and the gradient of the function $\nabla F(x)$ are computed using a forward or central differences method and the system of equations:

$$Hp = -\nabla F$$

(see Gill *et al.* [1] for details) is solved to find the Newton direction p in which to search for a minimum. Then a one-dimensional line search is applied to minimise the function with respect to a step length along the Newton direction:

$$\min_\alpha F(x = \alpha p).$$

This process is repeated until the the solution appears to have converged or until a point is reached where a large number of iterations have been performed without sufficient progress.

6.2 Parallelism Detail

The elements of the Hessian can all be evaluated completely in parallel and this is the principal source of parallelism. The solutionof the system of equations also involves some parallelism, the LU factorization and solve are described in Chapter F04.

6.3 Accuracy

When a successful exit is made then, for a computer with a mantissa of t decimal digits, one would expect to get about $t/2 - 1$ decimal digits accuracy in x and about $t - 1$ decimal digits accuracy in F, provided the problem is reasonably well scaled.

7 References

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

8 Example

A program to minimize

$$F = \sin^2(x_1 - 1)x_2^2 x_3^2$$

starting from the initial guess $(0.5, 0.5, 0.5)^T$.

8.1 Example Text

```
*     E04JBFP Example Program Text
*     NAG Parallel Library Release 2. NAG Copyright 1996.
*     .. Parameters ..
      INTEGER          LIW, LW, NMAX, MPMAX, NPMAX, NOUT
      PARAMETER        (LIW=1000,LW=1000,NMAX=10,MPMAX=2,NPMAX=1,NOUT=6)
*     .. Local Scalars ..
      DOUBLE PRECISION F, TAU
      INTEGER          ICNTXT, IFAIL, K, MP, N, NB, NP
      LOGICAL          ROOT
*     .. Local Arrays ..
      DOUBLE PRECISION W(LW), X(NMAX)
      INTEGER          IW(LIW)
*     .. External Functions ..
      LOGICAL          Z01ACFP
      EXTERNAL         Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         E04JBFP, FUNC1, Z01AAFP, Z01ABFP
*     .. Executable Statements ..

      ROOT = Z01ACFP()

      IF (ROOT) WRITE (NOUT,*) 'E04JBFP Example Program Results'

      MP = MPMAX
      NP = NPMAX

      IFAIL = 0
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)

      N = 3
      X(1) = 0.5D0
      X(2) = 0.5D0
      X(3) = 0.5D0
      NB = 1
      TAU = 1.0D-6

      IFAIL = 1
      CALL E04JBFP(ICNTXT,N,NB,X,TAU,F,FUNC1,IW,LIW,W,LW,IFAIL)
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,99999) F
         WRITE (NOUT,99998) (X(K),K=1,N)
      END IF

      IFAIL = 0
      CALL Z01ABFP(ICNTXT,'n',IFAIL)

      STOP

99999 FORMAT (1X,'Function value on exit is = ',F6.4)
99998 FORMAT (1X,'at the point x = ( ',3F6.2,' )')
      END

      SUBROUTINE FUNC1(N,X,F)

*     .. Scalar Arguments ..
```

```
      DOUBLE PRECISION F
      INTEGER          N
*     .. Array Arguments ..
      DOUBLE PRECISION X(N)
*     .. Intrinsic Functions ..
      INTRINSIC        SIN
*     .. Executable Statements ..
      F = SIN(X(1)-1.0D0)**2*X(2)**2*X(3)**2

      RETURN

      END
```

8.2 Example Data

None.

8.3 Example Results

```
E04JBFP Example Program Results

Function value on exit is = 0.0000
at the point x = (   1.00  0.01  0.01 )
```