D01FAFP

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

D01FAFP computes an approximation to an n-dimensional definite integral,

$$I = \int_{a_1}^{b_1} dx_1 ... \int_{a_n}^{b_n} dx_n \;\; f(x_1, x_2, ..., x_n)$$

in up to 10 dimensions over a hyper-rectangular region, using an adaptive subdivision strategy. The routine also returns an estimate of the absolute error. This routine is suitable for high accuracy work.

2 Specification

```
 SUBROUTINE D01FAFP(ICNTXT, NDIM, F, A, B, EPSABS, EPSREL, MAXFUN,
1                   SUBDIV, NDIVID, NSTEP, RESULT, ABSERR, NFUN,
2                   WORK, LW, IFAIL)
 INTEGER            ICNTXT, NDIM, MAXFUN, NDIVID(NDIM), NSTEP,
1                   NFUN, LW, IFAIL
 DOUBLE PRECISION   F, A(NDIM), B(NDIM), EPSABS, EPSREL, RESULT,
1                   ABSERR, WORK(LW)
 CHARACTER*1        SUBDIV
 EXTERNAL           F
```

3 Data Distribution

3.1 Definitions

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

m_p	–	the number of processor rows in the processor grid,
n_p	–	the number of processor columns in the processor grid,
p	–	$m_p \times n_p$, the total number of processors in the logical processor grid.

3.2 Global and Local Arguments

The input arguments NDIM, A, B, EPSABS, EPSREL, MAXFUN, SUBDIV, NDIVID, NSTEP, LW and IFAIL are global and so must have the same value on entry to the routine on each processor. The output arguments NDIVID, RESULT, ABSERR, NFUN and IFAIL are global, and so will have the same value on exit from the routine on each processor. The external procedure F is global. The input argument ICNTXT is local.

4 Arguments

1: ICNTXT — INTEGER *Local Input*

On entry: the BLACS context used by the communication mechanism, usually returned by a call to Z01AAFP.

2: NDIM — INTEGER *Global Input*

On entry: the number of dimensions of the integral, n.

Constraint: $2 \leq \text{NDIM} \leq 10$.

3: F — DOUBLE PRECISION FUNCTION, supplied by the user. *Global External Procedure*

F must return the value of the integrand f at a given point.

Its specification is:

```
DOUBLE PRECISION FUNCTION F(NDIM, X)
INTEGER                   NDIM
DOUBLE PRECISION          X(NDIM)
```

1: NDIM — INTEGER *Global Input*
On entry: the number of dimensions of the integral, n.

2: X(NDIM) — DOUBLE PRECISION array *Local Input*
On entry: the coordinates of the point at which the integrand must be evaluated.

F must be declared as EXTERNAL in the (sub)program from which D01FAFP is called. Arguments denoted as *Input* must **not** be changed by this procedure.

4: A(NDIM) — DOUBLE PRECISION array *Global Input*
On entry: the lower limits of integration, a_i, for $i = 1, 2, \ldots, n$.

5: B(NDIM) — DOUBLE PRECISION array *Global Input*
On entry: the upper limits of integration, b_i, for $i = 1, 2, \ldots, n$. It is not necessary that $a_i < b_i$.

6: EPSABS — DOUBLE PRECISION *Global Input*
On entry: the absolute accuracy required. If EPSABS is negative, its absolute value is used.

Constraint: EPSABS and EPSREL can not both be 0.0.

7: EPSREL — DOUBLE PRECISION *Global Input*
On entry: the relative accuracy required. If EPSREL is negative, its absolute value is used.

Constraint: EPSREL and EPSABS can not both be 0.0.

8: MAXFUN — INTEGER *Global Input*
On entry: the total maximum number of function evaluations to be allowed.

Constraint: MAXFUN $\geq n_f$, where $n_f = 3\,p\,[2^n + 4n(n-1)(n-2)/3 + 2n(3n+1) + 1]$.

9: SUBDIV — CHARACTER*1 *Global Input*
On entry: determines whether or not an initial, user-specified subdivision of the integration region is required:

if SUBDIV = 'U', the user must specify the number of initial subdivisions along each direction of integration region (see array NDIVID);
if SUBDIV = 'N', no user-specified initial subdivision is required. The initial subdivision is chosen internally.

Constraint: SUBDIV = 'U' or 'N'.

10: NDIVID(NDIM) — INTEGER array *Global Input/Global Output*
On entry: the number of initial subdivisions along each direction of the integration region. NDIVID(i) should contain the number of subdivisions along the ith direction. If SUBDIV = 'N', then this array need not be set.

On exit: the number of initial subdivisions used internally in the case SUBDIV = 'N'.

Constraint: NDIVID(1) $\times$ NDIVID(2) $\times \cdots \times$ NDIVID(NDIM) $= p$.

11: NSTEP — INTEGER *Global Input*

On entry: the maximum number of adaptive subdivision stages performed between two dynamic load balancing steps (see Section 6.2).

Suggested value: NSTEP = 1.

Constraint: NSTEP ≥ 1.

12: RESULT — DOUBLE PRECISION *Global Output*

On exit: the approximation to the integral I.

13: ABSERR — DOUBLE PRECISION *Global Output*

On exit: an estimate of the modulus of the absolute error, which should be an upper bound for $|I-\text{RESULT}|$.

14: NFUN — INTEGER *Global Output*

On exit: the total number of function evaluations used in computing the integral.

15: WORK(LW) — DOUBLE PRECISION *Local Workspace*

16: LW — INTEGER *Global Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which D01FAFP is called. The value of LW imposes a bound on the number of subregions which can be stored on each processor. The number of of subregions on each processor cannot exceed LW/(2×NDIM+3).

Suggested value: LW $\geq$ (2×NDIM+3)(3×MAXFUN+n_f)/(2n_f).

Constraint: LW $\geq 2\times$ (2×NDIM+3).

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

On entry: IFAIL must be set to 0, −1 or 1. For users not familiar with this parameter (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 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).

Errors or warnings specified by the routine:

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 had an invalid value. This error occurred either because a global argument did not have the same value on all the logical processors (see Section 3.2), or because its value was incorrect. An explanatory message distinguishes between these two cases.

IFAIL = 1

MAXFUN was too small for D01FAFP to obtain the required accuracy. RESULT and ABSERR respectively contain current estimates for the integral and the absolute error.

IFAIL = 2

LW is too small for D01FAFP to continue. RESULT and ABSERR respectively contain current estimates for the integral and the absolute error.

6 Further Comments

6.1 Algorithmic Detail

This routine calculates an approximation to the integral

$$I = \int_{a_1}^{b_1} dx_1 ... \int_{a_n}^{b_n} dx_n \;\; f(x_1, x_2, ..., x_n) \tag{1}$$

using a global adaptive algorithm. It is based on a degree 9 integration rule developed by Genz and Malik [2] and applies the error estimate procedure proposed by Berntsen [1].

The subroutine divides the integration region into a number of subregions. Inside each subregion the integral and the integration error are estimated. The initial subdivision of the integration region is chosen according to the values in the array NDIVID, and each of the initial subregions is assigned to a different processor. The results of each processor are stored in a partially ordered list (a heap) in the processor's own local memory.

The routine then proceeds in stages. At each stage, each processor selects the subregion with the largest error estimate in its own heap. This subregion is halved along the coordinate axis where the integrand has the largest absolute fourth divided difference. The results from the two halves are used to update the integral and error estimates. The process continues until the required accuracy is attained or further subdivision would use more than MAXFUN function evaluations.

6.2 Parallelism Detail

The error estimates for the initial subregions may vary significantly, which means that the work initially assigned to the different processors is not of equal difficulty. In this case, the simple algorithmic scheme described in Section 6.1 performs poorly because processors assigned easy tasks quickly run out of subregions with large error estimates while at the same time such subregions are still available on processors assigned difficult tasks. D01FAFP overcomes this problem by dynamically redistributing difficult subregions between different processors. The dynamic load balancing scheme employed proceeds in stages, subsequent stages occuring at least every NSTEP subdivision steps. At each stage, each processor sends information about its local error estimates to a specific neighbouring processor. Upon receiving this information from a neighbouring processor, each processor decides whether or not it should offload some of its work and – if so – sends a number of subregions to this neighbouring processor.

The dynamic load balancing scheme described requires only communication between directly connected processors (according to a cyclic 2-d mesh topology) and can be shown to be highly scalable (see [3]). If the load balancing mechanism is successful, the total number of function evaluations NFUN used in computing the integral should not grow significantly when the number of processors p is increased. The parameter NSTEP can be used to increase the granularity of the algorithm if the parallel overhead associated with the dynamic load balancing scheme is too high. Larger values of NSTEP result in a reduced parallel overhead, but may potentially cause an increased load imbalance, which dimishes the performance of the algorithm.

6.3 Accuracy

An estimate of the absolute error is given, on exit, by the value of ABSERR.

7 References

[1] Bernsten, J (1989) Practical error estimation in adaptive multi-dimensional quadrature routines, *J. Comput. Appl. Math.* **25** 327–340.

[2] Genz, A C and Malik, A A (1983) An embedded family of fully symmetric numerical integration rules, *SIAM J. Numer. Anal.* **20** 580–588.

[3] D'Apuzzo, M, Lapegna, M and Murli, A Scalability and Load Balancing in Adaptive Algorithms for multi-dimensional integration *Technical Report n. 2/95* , Centro di Ricerche per il Calcolo Parallelo e i Supercalcolatori. December 1995.

8 Example

This example calculates the integral

$$\int_0^1 \int_0^1 \int_0^1 \int_0^1 \sum_{k=0}^{5} \cos(0.5 + k(x_1 + x_2 + x_3 + x_4) - 4) dx_4 \; dx_3 \; dx_2 \; dx_1,$$

8.1 Example Text

```
*     D01FAFP Example Program Text
*     NAG Parallel Library Release 2. NAG Copyright 1996.
*     .. Parameters ..
      INTEGER          NOUT, NDIM, LW, NFUNRL
      PARAMETER        (NOUT=6,NDIM=4,LW=1000*(2*NDIM+3),
     +                 NFUNRL=2*NDIM*(3*NDIM+1)+4*NDIM*(NDIM-1)*(NDIM-2)
     +                 /3+2**NDIM+1)
*     .. Local Scalars ..
      DOUBLE PRECISION ABSERR, EPSABS, EPSREL, RESULT
      INTEGER          I, ICNTXT, IFAIL, MAXFUN, MP, NFUN, NP, NSTEP
      LOGICAL          ROOT
      CHARACTER        SUBDIV
*     .. Local Arrays ..
      DOUBLE PRECISION A(NDIM), B(NDIM), WORK(LW)
      INTEGER          NDIVID(NDIM)
*     .. External Functions ..
      DOUBLE PRECISION F
      LOGICAL          Z01ACFP
      EXTERNAL         F, Z01ACFP
*     .. External Subroutines ..
      EXTERNAL         D01FAFP, Z01AAFP, Z01ABFP
*     .. Executable Statements ..

      ROOT = Z01ACFP()
      IF (ROOT) WRITE (NOUT,*) 'D01FAFP Example Program Results'

      MP = 2
      NP = 2
      IFAIL = 0
*
      CALL Z01AAFP(ICNTXT,MP,NP,IFAIL)
*
*     .. Initialise input arguments ..
*
      DO 20 I = 1, NDIM
         A(I) = 0.D0
         B(I) = 1.D0
   20 CONTINUE
      EPSABS = 0.0D0
      EPSREL = 1.0D-4
      SUBDIV = 'U'
      NDIVID(1) = 2
      NDIVID(2) = 2
      NDIVID(3) = 1
      NDIVID(4) = 1
      MAXFUN = (2*(LW-1))/(2*NDIM+3)*NFUNRL
      NSTEP = 1
*
      CALL D01FAFP(ICNTXT,NDIM,F,A,B,EPSABS,EPSREL,MAXFUN,SUBDIV,NDIVID,
```

```
     +             NSTEP,RESULT,ABSERR,NFUN,WORK,LW,IFAIL)
*
      IF (ROOT) THEN
         WRITE (NOUT,*)
         WRITE (NOUT,99999) 'Computed result                  = ',
     +     RESULT
         WRITE (NOUT,99998) 'Computed absolute error          = ',
     +     ABSERR
         WRITE (NOUT,99997) 'No. of function evaluations      = ',
     +     NFUN
      END IF

      CALL Z01ABFP(ICNTXT,'No',IFAIL)

      STOP
*
99999 FORMAT (1x,A,F12.4)
99998 FORMAT (1x,A,E12.2)
99997 FORMAT (1x,A,I12)
      END
*
*
      DOUBLE PRECISION FUNCTION F(NDIM,X)
*     .. Scalar Arguments ..
      INTEGER                  NDIM
*     .. Array Arguments ..
      DOUBLE PRECISION         X(NDIM)
*     .. Local Scalars ..
      DOUBLE PRECISION         SUM1, SUM2
      INTEGER                  K
*     .. Intrinsic Functions ..
      INTRINSIC                COS, DBLE
*     .. Executable Statements ..
      SUM1 = 0.0D0
      DO 20 K = 1, NDIM
         SUM1 = SUM1 + X(K)
   20 CONTINUE
      SUM2 = 0.0D0
      DO 40 K = 0, 5
         SUM2 = SUM2 + COS(0.5D0+DBLE(K)*SUM1-DBLE(NDIM))
   40 CONTINUE
      F = SUM2
      RETURN
      END
```

8.2 Example Data

None.

8.3 Example Results

```
D01FAFP Example Program Results

Computed result                  =      -0.5991
Computed absolute error          =     0.56E-04
No. of function evaluations      =        31518
```