NAG Fortran Library Routine Document

D01AJF

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

D01AJF is a general-purpose integrator which calculates an approximation to the integral of a function $f(x)$ over a finite interval $[a, b]$:

$$I = \int_a^b f(x)\,dx.$$

2 Specification

```
SUBROUTINE D01AJF(F, A, B, EPSABS, EPSREL, RESULT, ABSERR, W, LW, IW,
1                 LIW, IFAIL)
INTEGER           LW, IW(LIW), LIW, IFAIL
real              F, A, B, EPSABS, EPSREL, RESULT, ABSERR, W(LW)
EXTERNAL          F
```

3 Description

D01AJF is based upon the QUADPACK routine QAGS (Piessens *et al.* (1983)). It is an adaptive routine, using the Gauss 10-point and Kronrod 21-point rules. The algorithm, described by de Doncker (1978), incorporates a global acceptance criterion (as defined by Malcolm and Simpson (1976)) together with the ϵ-algorithm (Wynn (1956)) to perform extrapolation. The local error estimation is described by Piessens *et al.* (1983).

The routine is suitable as a general purpose integrator, and can be used when the integrand has singularities, especially when these are of algebraic or logarithmic type.

D01AJF requires the user to supply a function to evaluate the integrand at a single point.

The routine D01ATF uses an identical algorithm but requires the user to supply a subroutine to evaluate the integrand at an array of points. Therefore D01ATF will be more efficient if the evaluation can be performed in vector mode on a vector-processing machine.

4 References

de Doncker E (1978) An adaptive extrapolation algorithm for automatic integration *ACM SIGNUM Newsl.* **13 (2)** 12–18

Malcolm M A and Simpson R B (1976) Local versus global strategies for adaptive quadrature *ACM Trans. Math. Software* **1** 129–146

Piessens R, de Doncker-Kapenga E, Überhuber C and Kahaner D (1983) *QUADPACK, A Subroutine Package for Automatic Integration* Springer-Verlag

Wynn P (1956) On a device for computing the $e_m(S_n)$ transformation *Math. Tables Aids Comput.* **10** 91–96

5 Parameters

1: F – ***real*** FUNCTION, supplied by the user. *External Procedure*

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

Its specification is:

```
real FUNCTION F(X)
real             X
```

1: X – ***real*** *Input*

On entry: the point at which the integrand f must be evaluated.

F must be declared as EXTERNAL in the (sub)program from which D01AJF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: A – ***real*** *Input*

On entry: the lower limit of integration, a.

3: B – ***real*** *Input*

On entry: the upper limit of integration, b. It is not necessary that $a < b$.

4: EPSABS – ***real*** *Input*

On entry: the absolute accuracy required. If EPSABS is negative, the absolute value is used. See Section 7.

5: EPSREL – ***real*** *Input*

On entry: the relative accuracy required. If EPSREL is negative, the absolute value is used. See Section 7.

6: RESULT – ***real*** *Output*

On exit: the approximation to the integral I.

7: ABSERR – ***real*** *Output*

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

8: W(LW) – ***real*** array *Output*

On exit: details of the computation, as described in Section 8.

9: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which D01AJF is called. The value of LW (together with that of LIW below) imposes a bound on the number of sub-intervals into which the interval of integration may be divided by the routine. The number of sub-intervals cannot exceed LW/4. The more difficult the integrand, the larger LW should be.

Suggested value: a value in the range 800 to 2000 is adequate for most problems.

Constraint: LW $\geq$ 4.

10: IW(LIW) – INTEGER array *Output*

On exit: IW(1) contains the actual number of sub-intervals used. The rest of the array is used as workspace.

11: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which D01AJF is called. The number of sub-intervals into which the interval of integration may be divided cannot exceed LIW.

Suggested value: LIW = LW/4.

Constraint: LIW $\geq$ 1.

12: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

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.**

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

The maximum number of subdivisions allowed with the given workspace has been reached without the accuracy requirements being achieved. Look at the integrand in order to determine the integration difficulties. If the position of a local difficulty within the interval can be determined (e.g., a singularity of the integrand or its derivative, a peak, a discontinuity, etc.) you will probably gain from splitting up the interval at this point and calling the integrator on the subranges. If necessary, another integrator, which is designed for handling the type of difficulty involved, must be used. Alternatively, consider relaxing the accuracy requirements specified by EPSABS and EPSREL, or increasing the amount of workspace.

IFAIL = 2

Round-off error prevents the requested tolerance from being achieved. The error may be underestimated. Consider requesting less accuracy.

IFAIL = 3

Extremely bad local integrand behaviour causes a very strong subdivision around one (or more) points of the interval.

IFAIL = 4

The requested tolerance cannot be achieved, because the extrapolation does not increase the accuracy satisfactorily; the returned result is the best which can be obtained. The same advice applies as in the case of IFAIL = 1.

IFAIL = 5

The integral is probably divergent, or slowly convergent. Please note that divergence can occur with any non-zero value of IFAIL.

IFAIL $= 6$

On entry, LW < 4,
or LIW < 1.

7 Accuracy

The routine cannot guarantee, but in practice usually achieves, the following accuracy:

$$|I - \text{RESULT}| \leq tol,$$

where

$$tol = \max\{|\text{EPSABS}|, |\text{EPSREL}| \times |I|\},$$

and EPSABS and EPSREL are user-specified absolute and relative error tolerance. Moreover, it returns the quantity ABSERR which, in normal circumstances, satisfies

$$|I - \text{RESULT}| \leq \text{ABSERR} \leq tol.$$

8 Further Comments

The time taken by the routine depends on the integrand and the accuracy required.

If IFAIL $\neq 0$ on exit, then the user may wish to examine the contents of the array W, which contains the end-points of the sub-intervals used by D01AJF along with the integral contributions and error estimates over the sub-intervals.

Specifically, for $i = 1, 2, \ldots, n$, let r_i denote the approximation to the value of the integral over the sub-interval $[a_i, b_i]$ in the partition of $[a, b]$ and e_i be the corresponding absolute error estimate.

Then, $\int_{a_i}^{b_i} f(x)\,dx \simeq r_i$ and RESULT $= \sum_{i=1}^{n} r_i$, unless D01AJF terminates while testing for divergence of the integral (see Section 3.4.3 of Piessens *et al.* (1983)). In this case, RESULT (and ABSERR) are taken to be the values returned from the extrapolation process. The value of n is returned in IW(1), and the values a_i, b_i, e_i and r_i are stored consecutively in the array W, that is:

$$\begin{aligned} a_i &= \text{W}(i), \\ b_i &= \text{W}(n+i), \\ e_i &= \text{W}(2n+i) \text{ and} \\ r_i &= \text{W}(3n+i). \end{aligned}$$

9 Example

To compute

$$\int_0^{2\pi} \frac{x \sin(30x)}{\sqrt{\left(1 - \left(\frac{x}{2\pi}\right)^2\right)}}\,dx.$$

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     D01AJF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          LW, LIW
      PARAMETER        (LW=800,LIW=LW/4)
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
*     .. Scalars in Common ..
      real             PI
      INTEGER          KOUNT
```

```
*     .. Local Scalars ..
      real              A, ABSERR, B, EPSABS, EPSREL, RESULT
      INTEGER           IFAIL
*     .. Local Arrays ..
      real              W(LW)
      INTEGER           IW(LIW)
*     .. External Functions ..
      real              FST, X01AAF
      EXTERNAL          FST, X01AAF
*     .. External Subroutines ..
      EXTERNAL          D01AJF
*     .. Common blocks ..
      COMMON            /TELNUM/PI, KOUNT
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D01AJF Example Program Results'
      PI = X01AAF(PI)
      EPSABS = 0.0e0
      EPSREL = 1.0e-04
      A = 0.0e0
      B = 2.0e0*PI
      KOUNT = 0
      IFAIL = -1
*
      CALL D01AJF(FST,A,B,EPSABS,EPSREL,RESULT,ABSERR,W,LW,IW,LIW,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 'A      - lower limit of integration = ', A
      WRITE (NOUT,99999) 'B      - upper limit of integration = ', B
      WRITE (NOUT,99998) 'EPSABS - absolute accuracy requested = ',
     +  EPSABS
      WRITE (NOUT,99998) 'EPSREL - relative accuracy requested = ',
     +  EPSREL
      WRITE (NOUT,*)
      IF (IFAIL.NE.0) WRITE (NOUT,99996) 'IFAIL = ', IFAIL
      IF (IFAIL.LE.5) THEN
         WRITE (NOUT,99997) 'RESULT - approximation to the integral = ',
     +     RESULT
         WRITE (NOUT,99998) 'ABSERR - estimate of the absolute error = '
     +     , ABSERR
         WRITE (NOUT,99996) 'KOUNT  - number of function evaluations = '
     +     , KOUNT
         WRITE (NOUT,99996) 'IW(1)  - number of subintervals used = ',
     +     IW(1)
      END IF
      STOP
*
99999 FORMAT (1X,A,F10.4)
99998 FORMAT (1X,A,e9.2)
99997 FORMAT (1X,A,F9.5)
99996 FORMAT (1X,A,I4)
      END
*
      real  FUNCTION FST(X)
*     .. Scalar Arguments ..
      real              X
*     .. Scalars in Common ..
      real              PI
      INTEGER           KOUNT
*     .. Intrinsic Functions ..
      INTRINSIC         SIN, SQRT
*     .. Common blocks ..
      COMMON            /TELNUM/PI, KOUNT
*     .. Executable Statements ..
      KOUNT = KOUNT + 1
      FST = X*SIN(30.0e0*X)/SQRT(1.0e0-X**2/(4.0e0*PI**2))
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
D01AJF Example Program Results

A      - lower limit of integration =     0.0000
B      - upper limit of integration =     6.2832
EPSABS - absolute accuracy requested =  0.00E+00
EPSREL - relative accuracy requested =  0.10E-03

RESULT - approximation to the integral =  -2.54326
ABSERR - estimate of the absolute error =  0.13E-04
KOUNT  - number of function evaluations =  777
IW(1)  - number of subintervals used =   19
```