D01AUF – NAG Fortran Library Routine Document

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

D01AUF is an adaptive integrator, especially suited to oscillating, non-singular integrands, 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 D01AUF(F, A, B, KEY, EPSABS, EPSREL, RESULT, ABSERR, W,
1                  LW, IW, LIW, IFAIL)
 INTEGER           KEY, LW, IW(LIW), LIW, IFAIL
 real              A, B, EPSABS, EPSREL, RESULT, ABSERR, W(LW)
 EXTERNAL          F
```

3 Description

D01AUF is based upon the QUADPACK routine QAG (Piessens *et al.* [3]). It is an adaptive routine, offering a choice of six Gauss–Kronrod rules. A global acceptance criterion (as defined by Malcolm and Simpson [1]) is used. The local error estimation is described by Piessens *et al.* [3].

Because this routine is based on integration rules of high order, it is especially suitable for non-singular oscillating integrands.

The routine requires a user-supplied subroutine to evaluate the integrand at an array of different points and is therefore particularly efficient when the evaluation can be performed in vector mode on a vector-processing machine. Otherwise this algorithm with KEY = 6 is identical to that used by D01AKF.

4 References

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

[2] Piessens R (1973) An algorithm for automatic integration *Angew. Inf.* **15** 399–401

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

5 Parameters

1: F — SUBROUTINE, supplied by the user. *External Procedure*

F must return the values of the integrand f at a set of points.

Its specification is:

```
SUBROUTINE F(X, FV, N)
INTEGER     N
real        X(N), FV(N)
```

1: X(N) — ***real*** array *Input*

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

> **2:** FV(N) — ***real*** array *Output*
>
> *On exit:* FV(j) must contain the value of f at the point X(j), for $j = 1, 2, \ldots,$ N.
>
> **3:** N — INTEGER *Input*
>
> *On entry:* the number of points at which the integrand is to be evaluated. The actual value of N is equal to the number of points in the Kronrod rule (see specification of KEY below).

F must be declared as EXTERNAL in the (sub)program from which D01AUF 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: KEY — INTEGER *Input*

On entry: which integration rule is to be used:

if KEY = 1 for the Gauss 7-point and Kronrod 15-point rule,
if KEY = 2 for the Gauss 10-point and Kronrod 21-point rule,
if KEY = 3 for the Gauss 15-point and Kronrod 31-point rule,
if KEY = 4 for the Gauss 20-point and Kronrod 41-point rule,
if KEY = 5 for the Gauss 25-point and Kronrod 51-point rule,
if KEY = 6 for the Gauss 30-point and Kronrod 61-point rule.

Suggested value: KEY = 6.

Constraint: KEY = 1, 2, 3, 4, 5 or 6.

5: EPSABS — ***real*** *Input*

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

6: EPSREL — ***real*** *Input*

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

7: RESULT — ***real*** *Output*

On exit: the approximation to the integral I.

8: ABSERR — ***real*** *Output*

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

9: W(LW) — ***real*** array *Output*

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

10: LW — INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which D01AUF 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.

11: IW(LIW) — INTEGER array *Output*

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

12: LIW — INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which D01AUF 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.

13: 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 or gives a warning (see Section 6).

For this routine, because the values of output parameters may be useful even if IFAIL $\neq$ 0 on exit, users are recommended to set IFAIL to -1 before entry. **It is then 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 specified 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. Probably 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. Consider requesting less accuracy.

IFAIL = 3

Extremely bad local integrand behaviour causes a very strong subdivision around one (or more) points of the interval. The same advice applies as in the case of IFAIL = 1.

IFAIL = 4

On entry, KEY < 1,
or KEY > 6.

IFAIL = 5

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

7 Accuracy

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

$$|I - \mathrm{RESULT}| \leq tol$$

where

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

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

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

8 Further Comments

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 D01AUF along with the integral contributions and error estimates over these 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 $\mathrm{RESULT} = \sum_{i=1}^{n} r_i$. 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:

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

9 Example

To compute

$$\int_0^{2\pi} x \sin(30x) \cos x \, dx.$$

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.

```
*     D01AUF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          LW, LIW
      PARAMETER        (LW=800,LIW=LW/4)
*     .. Scalars in Common ..
      INTEGER          KOUNT
*     .. Local Scalars ..
      real             A, ABSERR, B, EPSABS, EPSREL, PI, RESULT
      INTEGER          IFAIL, KEY
*     .. Local Arrays ..
      real             W(LW)
      INTEGER          IW(LIW)
```

```
*     .. External Functions ..
      real              X01AAF
      EXTERNAL          X01AAF
*     .. External Subroutines ..
      EXTERNAL          D01AUF, FST
*     .. Common blocks ..
      COMMON            /TELNUM/KOUNT
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D01AUF Example Program Results'
      PI = X01AAF(0.0e0)
      EPSABS = 0.0e0
      EPSREL = 1.0e-03
      A = 0.0e0
      B = 2.0e0*PI
      KEY = 6
      KOUNT = 0
      IFAIL = -1
*
      CALL D01AUF(FST,A,B,KEY,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.3) 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
*
      SUBROUTINE FST(X,FV,N)
*     .. Scalar Arguments ..
      INTEGER        N
*     .. Array Arguments ..
      real           FV(N), X(N)
*     .. Scalars in Common ..
      INTEGER        KOUNT
*     .. Local Scalars ..
      INTEGER        I
```

```
*     .. Intrinsic Functions ..
      INTRINSIC        COS, SIN
*     .. Common blocks ..
      COMMON           /TELNUM/KOUNT
*     .. Executable Statements ..
      KOUNT = KOUNT + N
      DO 20 I = 1, N
         FV(I) = X(I)*(SIN(30.0e0*X(I)))*COS(X(I))
   20 CONTINUE
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
D01AUF 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-02

RESULT - approximation to the integral =  -0.20967
ABSERR - estimate of the absolute error =  0.44E-13
KOUNT  - number of function evaluations =  427
IW(1)  - number of subintervals used =    4
```