

NAG Library Routine Document

D01AMF

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

D01AMF calculates an approximation to the integral of a function $f(x)$ over an infinite or semi-infinite interval $[a, b]$:

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

2 Specification

```

SUBROUTINE D01AMF(F, BOUND, INF, EPSABS, EPSREL, RESULT, ABSERR, W, LW,
1                IW, LIW, IFAIL)
    INTEGER          INF, LW, IW(LIW), LIW, IFAIL
    double precision F, BOUND, EPSABS, EPSREL, RESULT, ABSERR, W(LW)
    EXTERNAL         F

```

3 Description

D01AMF is based on the QUADPACK routine QAGI (see Piessens *et al.* (1983)). The entire infinite integration range is first transformed to $[0, 1]$ using one of the identities:

$$\int_{-\infty}^a f(x) dx = \int_0^1 f\left(a - \frac{1-t}{t}\right) \frac{1}{t^2} dt$$

$$\int_a^{\infty} f(x) dx = \int_0^1 f\left(a + \frac{1-t}{t}\right) \frac{1}{t^2} dt$$

$$\int_{-\infty}^{\infty} f(x) dx = \int_0^{\infty} (f(x) + f(-x)) dx = \int_0^1 \left[f\left(\frac{1-t}{t}\right) + f\left(\frac{-1+t}{t}\right) \right] \frac{1}{t^2} dt$$

where a represents a finite integration limit. An adaptive procedure, based on the Gauss 7-point and Kronrod 15-point rules, is then employed on the transformed integral. The algorithm, described in de Doncker (1978), incorporates a global acceptance criterion (as defined by Malcolm and Simpson (1976)) together with the ϵ -algorithm (see Wynn (1956)) to perform extrapolation. The local error estimation is described in Piessens *et al.* (1983).

4 References

de Doncker E (1978) An adaptive extrapolation algorithm for automatic integration *ACM SIGNUM News*. **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 – **double precision** FUNCTION, supplied by the user. *External Procedure*
 F must return the value of the integrand f at a given point.

The specification of F is:

```
double precision FUNCTION F(X)
double precision          X
```

- 1: X – **double precision** *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 D01AMF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

- 2: BOUND – **double precision** *Input*
On entry: the finite limit of the integration range (if present). BOUND is not used if the interval is doubly infinite.
- 3: INF – INTEGER *Input*
On entry: indicates the kind of integration range.
 INF = 1
 The range is $[BOUND, +\infty)$.
 INF = -1
 The range is $(-\infty, BOUND]$.
 INF = 2
 The range is $(-\infty, +\infty)$.
Constraint: INF = -1, 1 or 2.
- 4: EPSABS – **double precision** *Input*
On entry: the absolute accuracy required. If EPSABS is negative, the absolute value is used. See Section 7.
- 5: EPSREL – **double precision** *Input*
On entry: the relative accuracy required. If EPSREL is negative, the absolute value is used. See Section 7.
- 6: RESULT – **double precision** *Output*
On exit: the approximation to the integral I .
- 7: ABSERR – **double precision** *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) – **double precision** 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 D01AMF is called. The value of LW (together with that of LIW) 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: $LW = 800$ 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 D01AMF 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 . If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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).

Note: D01AMF may return useful information for one or more of the following detected errors or warnings.

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 D01AMF on the infinite subrange and an appropriate integrator on the finite subrange. 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

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 nonzero value of IFAIL.

IFAIL = 6

On entry, LW < 4,
or LIW < 1,
or INF $\neq$ -1, 1 or 2.

7 Accuracy

D01AMF cannot guarantee, but in practice usually achieves, the following accuracy:

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

where

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

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

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

8 Further Comments

The time taken by D01AMF depends on the integrand and the accuracy required.

If IFAIL $\neq$ 0 on exit, then you may wish to examine the contents of the array W, which contains the end points of the sub-intervals used by D01AMF along with the integral contributions and error estimates over these sub-intervals.

Specifically, for $i = 1, 2, \dots, n$, 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 $\text{RESULT} = \sum_{i=1}^n r_i$, unless D01AMF 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:

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

Note: this information applies to the integral transformed to $(0, 1)$ as described in Section 3, not to the original integral.

9 Example

This example computes

$$\int_0^{\infty} \frac{1}{(x+1)\sqrt{x}} dx.$$

The exact answer is π .

9.1 Program Text

```
*      D01AMF 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 ..
      INTEGER          KOUNT
*      .. Local Scalars ..
      DOUBLE PRECISION ABSERR, BOUND, EPSABS, EPSREL, RESULT
      INTEGER          IFAIL, INF
*      .. Local Arrays ..
      DOUBLE PRECISION W(LW)
      INTEGER          IW(LIW)
*      .. External Functions ..
      DOUBLE PRECISION F
      EXTERNAL          F
*      .. External Subroutines ..
      EXTERNAL          D01AMF
*      .. Common blocks ..
      COMMON            /TELNUM/KOUNT
*      .. Executable Statements ..
      WRITE (NOUT,*) 'D01AMF Example Program Results'
      EPSABS = 0.0D0
      EPSREL = 1.0D-04
      BOUND = 0.0D0
      INF = 1
      KOUNT = 0
      IFAIL = 1

*      CALL D01AMF(F,BOUND,INF,EPSABS,EPSREL,RESULT,ABSERR,W,LW,IW,LIW,
+              IFAIL)

*      IF (IFAIL.GE.0) THEN
          WRITE (NOUT,*)
          WRITE (NOUT,99999) 'A      - lower limit of integration = ',
+              BOUND
          WRITE (NOUT,*) 'B      - upper limit of integration = infinity'
          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
          ELSE
              WRITE (NOUT,*)
              WRITE (NOUT,99995) ' ** D01AMF returned with IFAIL = ', IFAIL
          END IF
```

```

*
99999 FORMAT (1X,A,F10.4)
99998 FORMAT (1X,A,E9.2)
99997 FORMAT (1X,A,F9.5)
99996 FORMAT (1X,A,I4)
99995 FORMAT (1X,A,I5)
      END
*
      DOUBLE PRECISION FUNCTION F(X)
*      .. Scalar Arguments ..
      DOUBLE PRECISION X
*      .. Scalars in Common ..
      INTEGER          KOUNT
*      .. Intrinsic Functions ..
      INTRINSIC        SQR
*      .. Common blocks ..
      COMMON           /TELNUM/KOUNT
*      .. Executable Statements ..
      KOUNT = KOUNT + 1
      F = 1.0D0/((X+1.0D0)*SQR(X))
      RETURN
      END

```

9.2 Program Data

None.

9.3 Program Results

D01AMF Example Program Results

```

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

RESULT - approximation to the integral =   3.14159
ABSERR - estimate of the absolute error =  0.27E-04
KOUNT  - number of function evaluations =   285
IW(1)  - number of subintervals used =    10

```
