D01ASF – 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

D01ASF calculates an approximation to the sine or the cosine transform of a function g over $[a, \infty)$:

$$I = \int_a^\infty g(x)\sin(\omega x)\,dx \quad \text{or} \quad I = \int_a^\infty g(x)\cos(\omega x)\,dx$$

(for a user-specified value of ω).

2 Specification

```
 SUBROUTINE D01ASF(G, A, OMEGA, KEY, EPSABS, RESULT, ABSERR,
1                  LIMLST, LST, ERLST, RSLST, IERLST, W, LW, IW,
2                  LIW, IFAIL)
 INTEGER           KEY, LIMLST, LST, IERLST(LIMLST), LW, IW(LIW),
1                  LIW, IFAIL
 real              G, A, OMEGA, EPSABS, RESULT, ABSERR,
1                  ERLST(LIMLST), RSLST(LIMLST), W(LW)
 EXTERNAL          G
```

3 Description

D01ASF is based upon the QUADPACK routine QAWFE (Piessens *et al.* [2]). It is an adaptive routine, designed to integrate a function of the form $g(x)w(x)$ over a semi-infinite interval, where $w(x)$ is either $\sin(\omega x)$ or $\cos(\omega x)$.

Over successive intervals

$$C_k = [a + (k-1)c, a + kc], \quad k = 1, 2, \ldots, \text{LST}$$

integration is performed by the same algorithm as is used by D01ANF. The intervals C_k are of constant length

$$c = \{2[|\omega|] + 1\}\pi/|\omega|, \;\; \omega \neq 0,$$

where $[|\omega|]$ represents the largest integer less than or equal to $|\omega|$. Since c equals an odd number of half periods, the integral contributions over succeeding intervals will alternate in sign when the function g is positive and monotonically decreasing over $[a, \infty)$. The algorithm, described by [2], incorporates a global acceptance criterion (as defined by Malcolm and Simpson [1]) together with the ϵ-algorithm (Wynn [3]) to perform extrapolation. The local error estimation is described by Piessens *et al.* [2].

If $\omega = 0$ and KEY $= 1$, the routine uses the same algorithm as D01AMF (with EPSREL $= 0.0$).

In contrast to the other routines in the Chapter Introduction, D01ASF works only with a user-specified **absolute** error tolerance (EPSABS). Over the interval C_k it attempts to satisfy the absolute accuracy requirement

$$\text{EPSA}_k = U_k \times \text{EPSABS}$$

where $U_k = (1-p)p^{k-1}$, for $k = 1, 2, \ldots$ and $p = 0.9$.

However, when difficulties occur during the integration over the kth sub-interval C_k such that the error flag IERLST(k) is non-zero, the accuracy requirement over subsequent intervals is relaxed. See Piessens *et al.* [2] for more details.

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

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

5 Parameters

1: G — ***real*** FUNCTION, supplied by the user. *External Procedure*

G must return the value of the function g at a given point.

Its specification is:

```
real FUNCTION G(X)
real           X
```

1: X — ***real*** *Input*

On entry: the point at which the function g must be evaluated.

G must be declared as EXTERNAL in the (sub)program from which D01ASF 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: OMEGA — ***real*** *Input*

On entry: the parameter ω in the weight function of the transform.

4: KEY — INTEGER *Input*

On entry: indicates which integral is to be computed:

if KEY = 1, $w(x) = \cos(\omega x)$;
if KEY = 2, $w(x) = \sin(\omega x)$.

Constraint: KEY = 1 or 2.

5: EPSABS — ***real*** *Input*

On entry: the absolute accuracy requirement. If EPSABS 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: LIMLST — INTEGER *Input*

On entry: an upper bound on the number of intervals C_k needed for the integration.

Suggested value: LIMLST = 50 is adequate for most problems.

Constraint: LIMLST $\geq$ 3.

9: LST — INTEGER *Output*

On exit: the number of intervals C_k actually used for the integration.

10: ERLST(LIMLST) — ***real*** array *Output*

On exit: ERLST(k) contains the error estimate corresponding to the integral contribution over the interval C_k, for $k = 1, 2, \ldots,$ LST.

11: RSLST(LIMLST) — ***real*** array *Output*

On exit: RSLST(k) contains the integral contribution over the interval C_k for $k = 1, 2, \ldots,$ LST.

12: IERLST(LIMLST) — INTEGER array *Output*

On exit: IERLST(k) contains the error flag corresponding to RSLST(k), for $k = 1, 2, \ldots,$ LST. See Section 6.

13: W(LW) — ***real*** array *Workspace*

14: LW — INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which D01ASF is called.. The value of LW (together with that of LIW below) imposes a bound on the number of sub-intervals into which each interval C_k 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 ≥ 4.

15: IW(LIW) — INTEGER array *Output*

On exit: IW(1) contains the maximum number of sub-intervals actually used for integrating over any of the intervals C_k. The rest of the array is used as workspace.

16: LIW — INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which D01ASF is called.. The number of sub-intervals into which each interval C_k may be divided cannot exceed LIW/2.

Suggested value: LIW = LW/2.

Constraint: LIW ≥ 2.

17: 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. 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 D01ASF on the infinite subrange and an appropriate integrator on the finite subrange. Alternatively, consider relaxing the accuracy requirements specified by EPSABS 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. 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. 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 D01ASF on the infinite subrange and an appropriate integrator on the finite subrange. Alternatively, consider relaxing the accuracy requirements specified by EPSABS or increasing the amount of workspace.

Please note that divergence can occur with any non-zero value of IFAIL.

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, KEY < 1,
or KEY > 2,
or LIMLST < 3.

IFAIL = 7

Bad integration behaviour occurs within one or more of the intervals C_k. Location and type of the difficulty involved can be determined from the vector IERLST (see below).

IFAIL = 8

Maximum number of intervals C_k (= LIMLST) allowed has been achieved. Increase the value of LIMLST to allow more cycles.

IFAIL = 9

The extrapolation table constructed for convergence acceleration of the series formed by the integral contribution over the intervals C_k, does not converge to the required accuracy.

IFAIL = 10

On entry, LW < 4,
or LIW < 2.

In the cases IFAIL = 7, 8 or 9, additional information about the cause of the error can be obtained from the array IERLST, as follows:

IERLST(k) = 1
The maximum number of subdivisions = min(LW/4, LIW/2) has been achieved on the kth interval.

IERLST(k) = 2
Occurrence of round-off error is detected and prevents the tolerance imposed on the kth interval from being achieved.

IERLST(k) = 3
Extremely bad integrand behaviour occurs at some points of the kth interval.

IERLST(k) = 4
: The integration procedure over the kth interval does not converge (to within the required accuracy) due to round-off in the extrapolation procedure invoked on this interval. It is assumed that the result on this interval is the best which can be obtained.

IERLST(k) = 5
: The integral over the kth interval is probably divergent or slowly convergent. It must be noted that divergence can occur with any other value of IERLST(k).

7 Accuracy

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

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

where EPSABS is the user-specified absolute error tolerance. Moreover, it returns the quantity ABSERR, which, in normal circumstances, satisfies

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

8 Further Comments

None.

9 Example

To compute

$$\int_0^\infty \frac{1}{\sqrt{x}} \cos(\pi x/2)\, dx.$$

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.

```
*     D01ASF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NOUT
      PARAMETER        (NOUT=6)
      INTEGER          LW, LIW, LIMLST
      PARAMETER        (LW=800,LIW=LW/2,LIMLST=50)
*     .. Scalars in Common ..
      INTEGER          KOUNT
*     .. Local Scalars ..
      real             A, ABSERR, EPSABS, OMEGA, RESULT
      INTEGER          IFAIL, INTEGR, LST
*     .. Local Arrays ..
      real             ERLST(LIMLST), RSLST(LIMLST), W(LW)
      INTEGER          IERLST(LIMLST), IW(LIW)
*     .. External Functions ..
      real             FST, X01AAF
      EXTERNAL         FST, X01AAF
*     .. External Subroutines ..
      EXTERNAL         D01ASF
*     .. Common blocks ..
      COMMON           /TELNUM/KOUNT
```

```
*     .. Executable Statements ..
      WRITE (NOUT,*) 'D01ASF Example Program Results'
      EPSABS = 1.0e-03
      A = 0.0e0
      KOUNT = 0
      OMEGA = 0.5e0*X01AAF(0.0e0)
      INTEGR = 1
      IFAIL = -1
*
      CALL D01ASF(FST,A,OMEGA,INTEGR,EPSABS,RESULT,ABSERR,LIMLST,LST,
     +            ERLST,RSLST,IERLST,W,LW,IW,LIW,IFAIL)
*
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 'A      - lower limit of integration = ', A
      WRITE (NOUT,*) 'B      - upper limit of integration =  infinity'
      WRITE (NOUT,99998) 'EPSABS - absolute accuracy requested = ',
     +  EPSABS
      WRITE (NOUT,*)
      IF (IFAIL.NE.0) WRITE (NOUT,99996) 'IFAIL = ', IFAIL
      IF (IFAIL.NE.6 .AND. IFAIL.NE.10) 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) 'LST    - number of intervals used = ', LST
         WRITE (NOUT,99996)
     +   'IW(1)  - max. no. of subintervals used in any one interval = '
     +     , 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 ..
      INTEGER            KOUNT
*     .. Intrinsic Functions ..
      INTRINSIC          SQRT
*     .. Common blocks ..
      COMMON             /TELNUM/KOUNT
*     .. Executable Statements ..
      KOUNT = KOUNT + 1
      FST = 0.0e0
      IF (X.GT.0.0e0) FST = 1.0e0/SQRT(X)
      RETURN
      END
```

9.2 Program Data

None.

9.3 Program Results

```
D01ASF Example Program Results

A      - lower limit of integration =     0.0000
B      - upper limit of integration =  infinity
EPSABS - absolute accuracy requested =  0.10E-02

RESULT - approximation to the integral =   1.00000
ABSERR - estimate of the absolute error =  0.59E-03
KOUNT  - number of function evaluations =  380
LST    - number of intervals used =    6
IW(1)  - max. no. of subintervals used in any one interval =    8
```