NAG Library Routine Document

D01ASF

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, LIMLST,       &
                   LST, ERLST, RSLST, IERLST, W, LW, IW, LIW, IFAIL)
INTEGER            KEY, LIMLST, LST, IERLST(LIMLST), LW, IW(LIW), LIW,     &
                   IFAIL
REAL (KIND=nag_wp) G, A, OMEGA, EPSABS, RESULT, ABSERR, ERLST(LIMLST),     &
                   RSLST(LIMLST), W(LW)
EXTERNAL           G
```

3 Description

D01ASF is based on the QUADPACK routine QAWFE (see Piessens *et al.* (1983)). 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

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

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

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

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 in Piessens *et al.* (1983), 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 by Piessens *et al.* (1983).

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

In contrast to the other routines in Chapter D01, D01ASF works only with an **absolute** error tolerance (EPSABS). Over the interval C_k it attempts to satisfy the absolute accuracy requirement

$$EPSA_k = U_k \times \mathrm{EPSABS},$$

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 $\mathrm{IERLST}(k)$ is nonzero, the accuracy requirement over subsequent intervals is relaxed. See Piessens *et al.* (1983) for more details.

4 References

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, Ûberhuber 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 Arguments

1: G – REAL (KIND=nag_wp) FUNCTION, supplied by the user. *External Procedure*

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

The specification of G is:

```
FUNCTION G (X)
REAL (KIND=nag_wp) G
REAL (KIND=nag_wp) X
```

1: X – REAL (KIND=nag_wp) *Input*

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

G must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which D01ASF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

2: A – REAL (KIND=nag_wp) *Input*

On entry: a, the lower limit of integration.

3: OMEGA – REAL (KIND=nag_wp) *Input*

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

4: KEY – INTEGER *Input*

On entry: indicates which integral is to be computed.

KEY $= 1$
$w(x) = \cos(\omega x)$.

KEY $= 2$
$w(x) = \sin(\omega x)$.

Constraint: KEY $= 1$ or 2.

5: EPSABS – REAL (KIND=nag_wp) *Input*

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

6: RESULT – REAL (KIND=nag_wp) *Output*

On exit: the approximation to the integral I.

7: ABSERR – REAL (KIND=nag_wp) *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 ≥ 3.

9: LST – INTEGER *Output*

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

10: ERLST(LIMLST) – REAL (KIND=nag_wp) 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 (KIND=nag_wp) 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 (KIND=nag_wp) 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) 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. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

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 arguments 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.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

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: D01ASF 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 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. 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 nonzero value of IFAIL.

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, KEY $\neq 1$ or 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.

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.

IFAIL = −99

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL = −399

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL = −999

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

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 $= \min(\mathrm{LW}/4, \mathrm{LIW}/2)$ 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

D01ASF cannot guarantee, but in practice usually achieves, the following accuracy:

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

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

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

8 Parallelism and Performance

D01ASF is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example computes

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

10.1 Program Text

```
!   D01ASF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d01asfe_mod

!     D01ASF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: g
!     .. Parameters ..
      Integer, Parameter, Public       :: limlst = 50, lw = 800, nout = 6
      Integer, Parameter, Public       :: liw = lw/2
    Contains
      Function g(x)

!       .. Function Return Value ..
        Real (Kind=nag_wp)             :: g
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: x
!       .. Intrinsic Procedures ..
        Intrinsic                      :: sqrt
!       .. Executable Statements ..
        If (x>0.0E0_nag_wp) Then
          g = 1.0E0_nag_wp/sqrt(x)
        Else
          g = 0.0E0_nag_wp
        End If

        Return

      End Function g
    End Module d01asfe_mod
    Program d01asfe

!     D01ASF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d01asf, nag_wp, x01aaf
      Use d01asfe_mod, Only: g, limlst, liw, lw, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: a, abserr, epsabs, omega, result
      Integer                          :: ifail, key, lst
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: erlst(:), rslst(:), w(:)
      Integer, Allocatable             :: ierlst(:), iw(:)
!     .. Executable Statements ..
      Write (nout,*) 'D01ASF Example Program Results'
```

```
      Allocate (erlst(limlst),rslst(limlst),w(lw),ierlst(limlst),iw(liw))

      epsabs = 1.0E-03_nag_wp
      a = 0.0E0_nag_wp
      omega = 0.5E0_nag_wp*x01aaf(omega)
      key = 1

      ifail = -1
      Call d01asf(g,a,omega,key,epsabs,result,abserr,limlst,lst,erlst,rslst,   &
        ierlst,w,lw,iw,liw,ifail)

      If (ifail>=0) Then
        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

        If (ifail/=6 .And. ifail/=10) Then
          Write (nout,*)
          Write (nout,99997) 'RESULT - approximation to the integral = ',      &
            result
          Write (nout,99998) 'ABSERR - estimate of the absolute error = ',     &
            abserr
          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

      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)
    End Program d01asfe
```

10.2 Program Data

None.

10.3 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
 LST    - number of intervals used =    6
 IW(1)  - max. no. of subintervals used in any one interval =    8
```