NAG Library Routine Document

D01ALF

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

D01ALF is a general purpose integrator which calculates an approximation to the integral of a function $f(x)$ over a finite interval $[a, b]$:

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

where the integrand may have local singular behaviour at a finite number of points within the integration interval.

2 Specification

```
SUBROUTINE D01ALF (F, A, B, NPTS, POINTS, EPSABS, EPSREL, RESULT, ABSERR,      &
                   W, LW, IW, LIW, IFAIL)

INTEGER            NPTS, LW, IW(LIW), LIW, IFAIL
REAL (KIND=nag_wp) F, A, B, POINTS(*), EPSABS, EPSREL, RESULT, ABSERR,         &
                   W(LW)
EXTERNAL           F
```

3 Description

D01ALF is based on the QUADPACK routine QAGP (see Piessens *et al.* (1983)). It is very similar to D01AJF, but allows you to supply 'break points', points at which the integrand is known to be difficult. It employs an adaptive algorithm, using the Gauss 10-point and Kronrod 21-point rules. 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 user-supplied 'break points' always occur as the end points of some sub-interval during the adaptive process. 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 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 (KIND=nag_wp) 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:

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

1: X – REAL (KIND=nag_wp) *Input*

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

F must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D01ALF is called. Parameters 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: B – REAL (KIND=nag_wp) *Input*

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

4: NPTS – INTEGER *Input*

On entry: the number of user-supplied break points within the integration interval.

Constraint: $\mathrm{NPTS} \geq 0$ and $\mathrm{NPTS} < \min((\mathrm{LW} - 2 \times \mathrm{NPTS} - 4)/4, (\mathrm{LIW} - \mathrm{NPTS} - 2)/2)$.

5: POINTS($*$) – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array POINTS must be at least $\max(1, \mathrm{NPTS})$.

On entry: the user-specified break points.

Constraint: the break points must all lie within the interval of integration (but may be supplied in any order).

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

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

7: EPSREL – REAL (KIND=nag_wp) *Input*

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

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

On exit: the approximation to the integral I.

9: ABSERR – REAL (KIND=nag_wp) *Output*

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

10: W(LW) – REAL (KIND=nag_wp) array *Output*

On exit: details of the computation see Section 8 for more information.

11: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which D01ALF 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 $(\text{LW} - 2 \times \text{NPTS} - 4)/4$. 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: $\text{LW} \geq 2 \times \text{NPTS} + 8$.

12: 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.

13: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which D01ALF is called. The number of sub-intervals into which the interval of integration may be divided cannot exceed $(\text{LIW} - \text{NPTS} - 2)/2$.

Suggested value: $\text{LIW} = \text{LW}/2$.

Constraint: $\text{LIW} \geq \text{NPTS} + 4$.

14: 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.

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 $\text{IFAIL} \neq 0$ 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: $\text{IFAIL} = 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $\text{IFAIL} = 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Note: D01ALF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

$\text{IFAIL} = 1$

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.) it should be supplied to the routine as an element of the vector POINTS. 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.

$\text{IFAIL} = 2$

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 $= 6$

The input is invalid: break points are specified outside the integration range, $\text{NPTS} > \min((\text{LW} - 2 \times \text{NPTS} - 4)/4, (\text{LIW} - \text{NPTS} - 2)/2)$ or $\text{NPTS} < 0$. RESULT and ABSERR are set to zero.

IFAIL $= 7$

On entry, $\text{LW} < 2 \times \text{NPTS} + 8$,
or $\text{LIW} < \text{NPTS} + 4$.

7 Accuracy

D01ALF 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 tolerances. 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 D01ALF 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 D01ALF 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 $\text{RESULT} = \sum_{i=1}^{n} r_i$ unless D01ALF 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:

$a_i = \text{W}(i)$,

$b_i = \text{W}(n + i)$,

$e_i = \text{W}(2n + i)$ and

$r_i = \text{W}(3n + i)$.

9 Example

This example computes

$$\int_0^1 \frac{1}{\sqrt{|x-1/7|}}\,dx.$$

A break point is specified at $x = 1/7$, at which point the integrand is infinite. (For definiteness the function FST returns the value 0.0 at this point.)

9.1 Program Text

```
!   D01ALF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module d01alfe_mod

!     D01ALF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                   :: lw = 800, nout = 6, npts = 1
      Integer, Parameter                   :: liw = lw/2
    Contains
      Function f(x)

!       .. Function Return Value ..
        Real (Kind=nag_wp)                   :: f
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)      :: x
!       .. Local Scalars ..
        Real (Kind=nag_wp)                   :: a
!       .. Intrinsic Procedures ..
        Intrinsic                            :: abs
!       .. Executable Statements ..
        a = abs(x-1.0E0_nag_wp/7.0E0_nag_wp)

        If (a/=0.0E0_nag_wp) Then
          f = a**(-0.5E0_nag_wp)
        Else
          f = 0.0E0_nag_wp
        End If

        Return

      End Function f
    End Module d01alfe_mod
    Program d01alfe

!     D01ALF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d01alf, nag_wp
      Use d01alfe_mod, Only: f, liw, lw, nout, npts
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: a, abserr, b, epsabs, epsrel,    &
                                              result
      Integer                              :: ifail
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable      :: points(:), w(:)
      Integer, Allocatable                 :: iw(:)
!     .. Executable Statements ..
      Write (nout,*) 'D01ALF Example Program Results'
```

```
      Allocate (points(npts),w(lw),iw(liw))

      epsabs = 0.0E0_nag_wp
      epsrel = 1.0E-03_nag_wp
      a = 0.0E0_nag_wp
      b = 1.0E0_nag_wp
      points(1) = 1.0E0_nag_wp/7.0E0_nag_wp

      ifail = -1
      Call d01alf(f,a,b,npts,points,epsabs,epsrel,result,abserr,w,lw,iw,liw, &
        ifail)

      If (ifail>=0) Then
        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,99995) 'POINTS(1)', 'given break-point', points(1)
      End If

      If (ifail>=0 .And. ifail<=5) 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) 'IW(1) ', 'number of subintervals used', iw(1)
      End If

99999 Format (1X,A6,'    - ',A32,' = ',F10.4)
99998 Format (1X,A6,'    - ',A32,' = ',E9.2)
99997 Format (1X,A6,'    - ',A32,' = ',F9.5)
99996 Format (1X,A6,'    - ',A32,' = ',I4)
99995 Format (1X,A9,' - ',A32,' = ',F10.4)

    End Program d01alfe
```

9.2 Program Data

None.

9.3 Program Results

```
 D01ALF Example Program Results

 A         -       lower limit of integration =     0.0000
 B         -       upper limit of integration =     1.0000
 EPSABS    -      absolute accuracy requested =  0.00E+00
 EPSREL    -      relative accuracy requested =  0.10E-02
 POINTS(1) -                given break-point =     0.1429

 RESULT    -    approximation to the integral =   2.60757
 ABSERR    -   estimate of the absolute error =  0.62E-13
 IW(1)     -      number of subintervals used =   12
```