NAG Library Routine Document

D01APF

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

D01APF is an adaptive integrator which calculates an approximation to the integral of a function $g(x)w(x)$ over a finite interval $[a, b]$:

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

where the weight function w has end point singularities of algebraico-logarithmic type.

2 Specification

```
SUBROUTINE D01APF (G, A, B, ALFA, BETA, KEY, EPSABS, EPSREL, RESULT,       &
                   ABSERR, W, LW, IW, LIW, IFAIL)

INTEGER             KEY, LW, IW(LIW), LIW, IFAIL
REAL (KIND=nag_wp) G, A, B, ALFA, BETA, EPSABS, EPSREL, RESULT, ABSERR,    &
                   W(LW)
EXTERNAL            G
```

3 Description

D01APF is based on the QUADPACK routine QAWSE (see Piessens *et al.* (1983)) and integrates a function of the form $g(x)w(x)$, where the weight function $w(x)$ may have algebraico-logarithmic singularities at the end points a and/or b. The strategy is a modification of that in D01AKF. We start by bisecting the original interval and applying modified Clenshaw–Curtis integration of orders 12 and 24 to both halves. Clenshaw–Curtis integration is then used on all sub-intervals which have a or b as one of their end points (see Piessens *et al.* (1974)). On the other sub-intervals Gauss–Kronrod (7–15 point) integration is carried out.

A 'global' acceptance criterion (as defined by Malcolm and Simpson (1976)) is used. The local error estimation control is described in Piessens *et al.* (1983).

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

Piessens R, Mertens I and Branders M (1974) Integration of functions having end-point singularities *Angew. Inf.* **16** 65–68

5 Parameters

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 D01APF 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.

Constraint: B $>$ A.

4: ALFA – REAL (KIND=nag_wp) *Input*

On entry: the parameter α in the weight function.

Constraint: ALFA > -1.0.

5: BETA – REAL (KIND=nag_wp) *Input*

On entry: the parameter β in the weight function.

Constraint: BETA > -1.0.

6: KEY – INTEGER *Input*

On entry: indicates which weight function is to be used.

KEY $= 1$

$$w(x) = (x-a)^{\alpha}(b-x)^{\beta}.$$

KEY $= 2$

$$w(x) = (x-a)^{\alpha}(b-x)^{\beta}\ln(x-a).$$

KEY $= 3$

$$w(x) = (x-a)^{\alpha}(b-x)^{\beta}\ln(b-x).$$

KEY $= 4$

$$w(x) = (x-a)^{\alpha}(b-x)^{\beta}\ln(x-a)\ln(b-x).$$

Constraint: KEY $= 1$, 2, 3 or 4.

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

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

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

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

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

On exit: the approximation to the integral I.

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

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

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

On exit: details of the computation see Section 9 for more information.

12: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which D01APF 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 $\geq$ 8.

13: 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.

14: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which D01APF 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 $\geq$ 2.

15: 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 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: D01APF 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 discontinuity or a singularity of algebraico-logarithmic type within the interval can be determined, the interval must be split up at this point and the

integrator called on the subranges. 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.

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, B $\leq$ A,
or ALFA ≤ -1.0,
or BETA ≤ -1.0,
or KEY $\neq 1, 2, 3$ or 4.

IFAIL $= 5$

On entry, LW < 8,
or LIW < 2.

IFAIL $= -99$

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

D01APF 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 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by D01APF 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 D01APF 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, $\int_{a_i}^{b_i} f(x)w(x)\,dx \simeq r_i$ and 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:

$a_i =$ W(i),

$b_i =$ W$(n+i)$,

$e_i =$ W$(2n+i)$ and

$r_i =$ W$(3n+i)$.

10 Example

This example computes

$$\int_0^1 \ln x \cos(10\pi x)\,dx \quad \text{and} \quad \int_0^1 \frac{\sin(10x)}{\sqrt{x(1-x)}}\,dx.$$

10.1 Program Text

```
!   D01APF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module d01apfe_mod

!     D01APF 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                                :: g1, g2
!     .. Parameters ..
      Integer, Parameter, Public            :: lw = 800, nout = 6
      Integer, Parameter, Public            :: liw = lw/4
    Contains
      Function g1(x)

!       .. Use Statements ..
        Use nag_library, Only: x01aaf
!       .. Function Return Value ..
        Real (Kind=nag_wp)                    :: g1
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)       :: x
!       .. Local Scalars ..
        Real (Kind=nag_wp)                    :: a, pi
!       .. Intrinsic Procedures ..
        Intrinsic                             :: cos
!       .. Executable Statements ..
        pi = x01aaf(pi)
        a = 10.0E0_nag_wp*pi
        g1 = cos(a*x)
```

```
        Return

      End Function g1
      Function g2(x)

!       .. Function Return Value ..
        Real (Kind=nag_wp)                  :: g2
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)     :: x
!       .. Local Scalars ..
        Real (Kind=nag_wp)                  :: omega
!       .. Intrinsic Procedures ..
        Intrinsic                           :: sin
!       .. Executable Statements ..
        omega = 10.0E0_nag_wp
        g2 = sin(omega*x)

        Return

      End Function g2
    End Module d01apfe_mod
    Program d01apfe

!     D01APF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d01apf, nag_wp
      Use d01apfe_mod, Only: g1, g2, liw, lw, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                  :: a, abserr, alpha, b, beta,       &
                                             epsabs, epsrel, result
      Integer                             :: ifail, key, nof
!     .. Local Arrays ..
      Real (Kind=nag_wp)                  :: alpha_a(2), beta_a(2)
      Real (Kind=nag_wp), Allocatable     :: w(:)
      Integer, Allocatable                :: iw(:)
      Integer                             :: key_a(2)
!     .. Executable Statements ..
      Write (nout,*) 'D01APF Example Program Results'

      Allocate (w(lw),iw(liw))

      alpha_a = (/0.0_nag_wp,-0.5_nag_wp/)
      beta_a = (/0.0_nag_wp,-0.5_nag_wp/)
      key_a = (/2,1/)

      epsabs = 0.0_nag_wp
      epsrel = 1.0E-04_nag_wp
      a = 0.0_nag_wp
      b = 1.0_nag_wp

funs: Do nof = 1, 2

        alpha = alpha_a(nof)
        beta = beta_a(nof)
        key = key_a(nof)

        ifail = -1
        If (nof==1) Then
          Call d01apf(g1,a,b,alpha,beta,key,epsabs,epsrel,result,abserr,w,lw, &
            iw,liw,ifail)
        Else
          Call d01apf(g2,a,b,alpha,beta,key,epsabs,epsrel,result,abserr,w,lw, &
            iw,liw,ifail)
        End If

        If (ifail<0) Then
          Exit funs
        End If
```

```
        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,*)
        Write (nout,99998) 'ALPHA ', 'parameter in the weight function', alpha
        Write (nout,99998) 'BETA  ', 'parameter in the weight function', beta
        Write (nout,99997) 'KEY   ', 'which weight function is used', key

        If (ifail>3) Then
          Cycle funs
        End If
        Write (nout,*)
        Write (nout,99996) 'RESULT', 'approximation to the integral', result
        Write (nout,99998) 'ABSERR', 'estimate of the absolute error', abserr
        Write (nout,99997) 'IW(1)', 'number of subintervals used ', iw(1)

      End Do funs

99999 Format (1X,A6,' - ',A32,' = ',F10.4)
99998 Format (1X,A6,' - ',A32,' = ',E9.2)
99997 Format (1X,A6,' - ',A32,' = ',I4)
99996 Format (1X,A6,' - ',A32,' = ',F9.5)
    End Program d01apfe
```

10.2 Program Data

None.

10.3 Program Results

```
 D01APF 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-03

 ALPHA  - parameter in the weight function =  0.00E+00
 BETA   - parameter in the weight function =  0.00E+00
 KEY    -    which weight function is used =    2

 RESULT -    approximation to the integral =  -0.04899
 ABSERR -   estimate of the absolute error =  0.11E-06
  IW(1) -     number of subintervals used  =    4

 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-03

 ALPHA  - parameter in the weight function = -0.50E+00
 BETA   - parameter in the weight function = -0.50E+00
 KEY    -    which weight function is used =    1

 RESULT -    approximation to the integral =   0.53502
 ABSERR -   estimate of the absolute error =  0.19E-11
  IW(1) -     number of subintervals used  =    2
```