NAG Library Routine Document

D01AQF

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

D01AQF calculates an approximation to the Hilbert transform of a function $g(x)$ over $[a,b]$:

$$I = \int_a^b \frac{g(x)}{x-c} dx$$

for user-specified values of a, b and c.

2 Specification

```
SUBROUTINE D01AQF (G, A, B, C, EPSABS, EPSREL, RESULT, ABSERR, W, LW,      &
                   IW, LIW, IFAIL)
INTEGER            LW, IW(LIW), LIW, IFAIL
REAL (KIND=nag_wp) G, A, B, C, EPSABS, EPSREL, RESULT, ABSERR, W(LW)
EXTERNAL           G
```

3 Description

D01AQF is based on the QUADPACK routine QAWC (see Piessens *et al.* (1983)) and integrates a function of the form $g(x)w(x)$, where the weight function

$$w(x) = \frac{1}{x-c}$$

is that of the Hilbert transform. (If $a < c < b$ the integral has to be interpreted in the sense of a Cauchy principal value.) It is an adaptive routine which employs a 'global' acceptance criterion (as defined by Malcolm and Simpson (1976)). Special care is taken to ensure that c is never the end point of a sub-interval (see Piessens *et al.* (1976)). On each sub-interval (c_1, c_2) modified Clenshaw–Curtis integration of orders 12 and 24 is performed if $c_1 - d \le c \le c_2 + d$ where $d = (c_2 - c_1)/20$. Otherwise the Gauss 7-point and Kronrod 15-point rules are used. The local error estimation is described by 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, Ûberhuber C and Kahaner D (1983) *QUADPACK, A Subroutine Package for Automatic Integration* Springer–Verlag

Piessens R, van Roy–Branders M and Mertens I (1976) The automatic evaluation of Cauchy principal value integrals *Angew. Inf.* **18** 31–35

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 D01AQF 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: B – REAL (KIND=nag_wp) *Input*

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

4: C – REAL (KIND=nag_wp) *Input*

On entry: the argument c in the weight function.

Constraint: C must not equal A or B.

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: EPSREL – REAL (KIND=nag_wp) *Input*

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

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

On exit: the approximation to the integral I.

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

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

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

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

10: LW – INTEGER *Input*

On entry: the dimension of the array W as declared in the (sub)program from which D01AQF 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 $\mathrm{LW}/4$. The more difficult the integrand, the larger LW should be.

Suggested value: $\mathrm{LW} = 800$ to 2000 is adequate for most problems.

Constraint: $\mathrm{LW} \geq 4$.

11: 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.

12: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which D01AQF is called. The number of sub-intervals into which the interval of integration may be divided cannot exceed LIW.

Suggested value: $\text{LIW} = \text{LW}/4$.

Constraint: $\text{LIW} \geq 1$.

13: 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 $\text{IFAIL} \neq 0$ 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: D01AQF 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 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.

$\text{IFAIL} = 3$

Extremely bad local behaviour of $g(x)$ causes a very strong subdivision around one (or more) points of the interval. The same advice applies as in the case of $\text{IFAIL} = 1$.

$\text{IFAIL} = 4$

On entry, $\text{C} = \text{A}$ or $\text{C} = \text{B}$.

$\text{IFAIL} = 5$

On entry, $\text{LW} < 4$,
or $\text{LIW} < 1$.

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.

7 Accuracy

D01AQF cannot guarantee, but in practice usually achieves, the following accuracy:

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

where

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

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

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

8 Parallelism and Performance

D01AQF is not threaded in any implementation.

9 Further Comments

The time taken by D01AQF 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 D01AQF 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, $\displaystyle\int_{a_i}^{b_i} g(x)w(x)\,dx \simeq r_i$ and $\mathrm{RESULT} = \displaystyle\sum_{i=1}^{n} r_i$. The value of n is returned in $\mathrm{IW}(1)$, and the values a_i, b_i, e_i and r_i are stored consecutively in the array W, that is:

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

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

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

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

10 Example

This example computes the Cauchy principal value of

$$\int_{-1}^{1} \frac{dx}{\left(x^2 + 0.01^2\right)\left(x - \frac{1}{2}\right)}.$$

10.1 Program Text

```
!   D01AQF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d01aqfe_mod

!     D01AQF 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       :: lw = 800, nout = 6
      Integer, Parameter, Public       :: liw = lw/4
    Contains
      Function g(x)

!       .. Function Return Value ..
        Real (Kind=nag_wp)             :: g
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: x
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: aa
!       .. Executable Statements ..
        aa = 0.01E0_nag_wp
        g = 1.0E0_nag_wp/(x**2+aa**2)

        Return

      End Function g
    End Module d01aqfe_mod
    Program d01aqfe

!     D01AQF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d01aqf, nag_wp
      Use d01aqfe_mod, Only: g, liw, lw, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: a, abserr, b, c, epsabs, epsrel,     &
                                          result
      Integer                          :: ifail
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: w(:)
      Integer, Allocatable             :: iw(:)
!     .. Executable Statements ..
      Write (nout,*) 'D01AQF Example Program Results'

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

      epsabs = 0.0E0_nag_wp
      epsrel = 1.0E-04_nag_wp
      a = -1.0E0_nag_wp
      b = 1.0E0_nag_wp
      c = 0.5E0_nag_wp

      ifail = -1
      Call d01aqf(g,a,b,c,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,99998) 'C     ', 'weight function parameter', c
      End If

      If (ifail>=0 .And. ifail<=3) 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,' = ',F9.2)
99996 Format (1X,A6,' - ',A32,' = ',I4)
    End Program d01aqfe
```

10.2 Program Data

None.

10.3 Program Results

```
 D01AQF Example Program Results

 A      -       lower limit of integration =    -1.0000
 B      -       upper limit of integration =     1.0000
 EPSABS -      absolute accuracy requested =  0.00E+00
 EPSREL -      relative accuracy requested =  0.10E-03
 C      -        weight function parameter =  0.50E+00

 RESULT -    approximation to the integral =   -628.46
 ABSERR -   estimate of the absolute error =  0.13E-01
 IW(1)  -      number of subintervals used =    8
```