

NAG Library Routine Document

D01AKF

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

D01AKF is an adaptive integrator, especially suited to oscillating, nonsingular integrands, which calculates an approximation to the integral of a function $f(x)$ over a finite interval $[a, b]$:

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

2 Specification

```

SUBROUTINE D01AKF (F, A, B, EPSABS, EPSREL, RESULT, ABSERR, W, LW, IW, LIW,      &
                  IFAIL)
INTEGER          LW, IW(LIW), LIW, IFAIL
REAL (KIND=nag_wp) F, A, B, EPSABS, EPSREL, RESULT, ABSERR, W(LW)
EXTERNAL        F

```

3 Description

D01AKF is based on the QUADPACK routine QAG (see Piessens *et al.* (1983)). It is an adaptive routine, using the Gauss 30-point and Kronrod 61-point rules. A 'global' acceptance criterion (as defined by Malcolm and Simpson (1976)) is used. The local error estimation is described in Piessens *et al.* (1983).

Because D01AKF is based on integration rules of high order, it is especially suitable for nonsingular oscillating integrands.

D01AKF requires you to supply a function to evaluate the integrand at a single point.

The routine D01AUF uses an identical algorithm but requires you to supply a subroutine to evaluate the integrand at an array of points. Therefore D01AUF will be more efficient if the evaluation can be performed in vector mode on a vector-processing machine.

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 (1973) An algorithm for automatic integration *Angew. Inf.* **15** 399–401

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

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 D01AKF 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: EPSABS – REAL (KIND=nag_wp)

Input

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

5: EPSREL – REAL (KIND=nag_wp)

Input

On entry: the relative accuracy required. If EPSREL 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: W(LW) – REAL (KIND=nag_wp) array

Output

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

9: LW – INTEGER

Input

On entry: the dimension of the array W as declared in the (sub)program from which D01AKF 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 4.

10: 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.

11: LIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which D01AKF 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: $LIW \geq 1$.

12: 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, if you are not familiar with this parameter, the recommended value is 0. **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).

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 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, $LW < 4$,
or $LIW < 1$.

7 Accuracy

D01AKF cannot guarantee, but in practice usually achieves, the following accuracy:

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

where

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

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

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

8 Further Comments

The time taken by D01AKF 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 D01AKF along with the integral contributions and error estimates over these sub-intervals.

Specifically, for $i = 1, 2, \dots, n$, 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 $\text{RESULT} = \sum_{i=1}^n r_i$. 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:

$$\begin{aligned} a_i &= \text{W}(i), \\ b_i &= \text{W}(n + i), \\ e_i &= \text{W}(2n + i) \text{ and} \\ r_i &= \text{W}(3n + i). \end{aligned}$$

9 Example

This example computes

$$\int_0^{2\pi} x \sin(30x) \cos x \, dx.$$

9.1 Program Text

```
! D01AKF Example Program Text
! Mark 24 Release. NAG Copyright 2012.

Module d01akfe_mod

! D01AKF 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
Integer, Parameter :: liw = lw/4
Contains
Function f(x)

! .. Function Return Value ..
Real (Kind=nag_wp) :: f
! .. Scalar Arguments ..
Real (Kind=nag_wp), Intent (In) :: x
! .. Intrinsic Procedures ..
Intrinsic :: cos, sin
! .. Executable Statements ..
f = x*(sin(30.OEO_nag_wp*x))*cos(x)

Return

End Function f
End Module d01akfe_mod
Program d01akfe
```

```

!      D01AKF Example Main Program

!      .. Use Statements ..
      Use nag_library, Only: d01akf, nag_wp, x01aaf
      Use d01akfe_mod, Only: f, liw, lw, nout
!      .. Implicit None Statement ..
      Implicit None
!      .. Local Scalars ..
      Real (Kind=nag_wp)                :: a, abserr, b, epsabs, epsrel,      &
                                         pi, result
      Integer                            :: ifail
!      .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable    :: w(:)
      Integer, Allocatable                :: iw(:)
!      .. Executable Statements ..
      Write (nout,*) 'D01AKF Example Program Results'

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

      pi = x01aaf(pi)
      epsabs = 0.0E0_nag_wp
      epsrel = 1.0E-03_nag_wp
      a = 0.0E0_nag_wp
      b = 2.0E0_nag_wp*pi

      ifail = -1
      Call d01akf(f,a,b,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
      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,' = ',F9.5)
99996 Format (1X,A6,' - ',A32,' = ',I4)
      End Program d01akfe

```

9.2 Program Data

None.

9.3 Program Results

D01AKF Example Program Results

A	-	lower limit of integration =	0.0000
B	-	upper limit of integration =	6.2832
EPSABS	-	absolute accuracy requested =	0.00E+00
EPSREL	-	relative accuracy requested =	0.10E-02
RESULT	-	approximation to the integral =	-0.20967
ABSERR	-	estimate of the absolute error =	0.45E-13
IW(1)	-	number of subintervals used =	4
