NAG Library Routine Document

D01RGF

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

D01RGF is a general purpose integrator which calculates an approximation to the integral of a function $f(x)$ over a finite interval $[a, b]$:

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

The routine is suitable as a general purpose integrator, and can be used when the integrand has singularities and infinities. In particular, the routine can continue if the subroutine F explicitly returns a quiet or signalling NaN or a signed infinity.

2 Specification

```
SUBROUTINE D01RGF (A, B, F, EPSABS, EPSREL, DINEST, ERREST, NEVALS,        &
                   IUSER, RUSER, IFAIL)

INTEGER             NEVALS, IUSER(*), IFAIL
REAL (KIND=nag_wp) A, B, EPSABS, EPSREL, DINEST, ERREST, RUSER(*)
EXTERNAL            F
```

3 Description

D01RGF uses the algorithm described in Gonnet (2010). It is an adaptive algorithm, similar to the QUADPACK routine QAGS (see Piessens *et al.* (1983), see also D01RAF) but includes significant differences regarding how the integrand is represented, how the integration error is estimated and how singularities and divergent integrals are treated. The local error estimation is described in Gonnet (2010).

D01RGF requires a subroutine to evaluate the integrand at an array of different points and is therefore amenable to parallel execution.

4 References

Gonnet P (2010) Increasing the reliability of adaptive quadrature using explicit interpolants *ACM Trans. Math. software* **37** 26

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

5 Parameters

1: A – REAL (KIND=nag_wp) *Input*

On entry: a, the lower limit of integration.

2: B – REAL (KIND=nag_wp) *Input*

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

Note: if A = B, the routine will immediately return DINEST = 0.0, ERREST = 0.0 and NEVALS = 0.

3: F – SUBROUTINE, supplied by the user. *External Procedure*

F must return the value of the integrand f at a set of points.

The specification of F is:

```
SUBROUTINE F (X, NX, FV, IFLAG, IUSER, RUSER)
INTEGER             NX, IFLAG, IUSER(*)
REAL (KIND=nag_wp) X(NX), FV(NX), RUSER(*)
```

1: X(NX) – REAL (KIND=nag_wp) array *Input*

On entry: the abscissae, x_i, for $i = 1, 2, \ldots, \mathrm{NX}$, at which function values are required.

2: NX – INTEGER *Input*

On entry: the number of abscissae at which a function value is required.

3: FV(NX) – REAL (KIND=nag_wp) array *Output*

On exit: FV must contain the values of the integrand f. $\mathrm{FV}(i) = f(x_i)$ for all $i = 1, 2, \ldots, \mathrm{NX}$.

4: IFLAG – INTEGER *Input/Output*

On entry: IFLAG $= 0$.

On exit: set IFLAG < 0 to force an immediate exit with IFAIL $= -1$.

5: IUSER($*$) – INTEGER array *User Workspace*

6: RUSER($*$) – REAL (KIND=nag_wp) array *User Workspace*

F is called with the parameters IUSER and RUSER as supplied to D01RGF. You are free to use the arrays IUSER and RUSER to supply information to F as an alternative to using COMMON global variables.

F must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D01RGF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

On entry: the absolute accuracy required.

If EPSABS is negative, |EPSABS| is used. See Section 7.

If EPSABS $= 0.0$, only the relative error will be used.

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

On entry: the relative accuracy required.

If EPSREL is negative, |EPSREL| is used. See Section 7.

If EPSREL $= 0.0$, only the absolute error will be used otherwise the actual value of EPSREL used by D01RGF is $\max(\textbf{\textit{machine precision}}, |\mathrm{EPSREL}|)$.

Constraint: at least one of EPSABS and EPSREL must be nonzero.

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

On exit: the estimate of the definite integral F.

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

On exit: the error estimate of the definite integral F.

8: NEVALS – INTEGER *Output*

On exit: the total number of points at which the integrand, f, has been evaluated.

9: IUSER(∗) – INTEGER array *User Workspace*
10: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by D01RGF, but are passed directly to F and may be used to pass information to this routine as an alternative to using COMMON global variables.

11: 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: D01RGF 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 requested accuracy was not achieved. Consider using larger values of EPSABS and EPSREL.

IFAIL $= 2$

The integral is probably divergent or slowly convergent.

IFAIL $= 14$

Both EPSABS $= 0.0$ and EPSREL $= 0.0$.

IFAIL $= -1$

Exit requested from F with IFLAG $= \langle value \rangle$.

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

D01RGF cannot guarantee, but in practice usually achieves, the following accuracy:

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

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 ERREST which, in normal circumstances, satisfies

$$|I - \text{DINEST}| \leq \text{ERREST} \leq tol.$$

8 Parallelism and Performance

D01RGF is currently neither directly nor indirectly threaded. In particular, the user-supplied subroutine F is not called from within a parallel region initialized inside D01RGF.

The user-supplied subroutine F uses a vectorized interface, allowing for the required vector of function values to be evaluated in parallel; for example by placing appropriate OpenMP directives in the code for the user-supplied subroutine F.

9 Further Comments

The time taken by D01RGF depends on the integrand and the accuracy required.

D01RGF is suitable for evaluating integrals that have singularities within the requested interval.

In particular, D01RGF accepts non-finite values on return from the user-supplied subroutine F, and will adapt the integration rule accordingly to eliminate such points. Non-finite values include NaNs and infinities.

10 Example

This example computes

$$\int_{-1}^{1} \frac{\sin(x)}{x} \ln(10(1-x)).$$

10.1 Program Text

```
!   D01RGF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module d01rgfe_mod

!     D01ATF 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                                :: f
!     .. Parameters ..
      Integer, Parameter, Public            :: nout = 6
    Contains
      Subroutine f(x,nx,fv,iflag,iuser,ruser)

!       .. Scalar Arguments ..
        Integer, Intent (Inout)               :: iflag
        Integer, Intent (In)                  :: nx
```

```
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Out)     :: fv(nx)
        Real (Kind=nag_wp), Intent (Inout)   :: ruser(*)
        Real (Kind=nag_wp), Intent (In)      :: x(nx)
        Integer, Intent (Inout)              :: iuser(*)
!       .. Intrinsic Procedures ..
        Intrinsic                            :: log, sin
!       .. Executable Statements ..
        fv = sin(x)/x*log(10.0_nag_wp*(1.0_nag_wp-x))
        Return
      End Subroutine f
    End Module d01rgfe_mod
    Program d01rgfe

!     D01RGF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d01rgf, nag_wp, x07caf, x07cbf
      Use d01rgfe_mod, Only: f, nout
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                   :: a, b, dinest, epsabs, epsrel,    &
                                              errest
      Integer                              :: ifail, nevals
!     .. Local Arrays ..
      Real (Kind=nag_wp)                   :: ruser(1)
      Integer                              :: exmode(3), exmode_old(3), iuser(1)
!     .. Executable Statements ..
      Write (nout,*) 'D01RGF Example Program Results'

!     The example function can raise various exceptions - it contains
!     a division by zero and a log singularity - although its integral
!     is well behaved.

!     Save the original halting mode
      Call x07caf(exmode_old)

!     Turn exception halting mode off for the three common exceptions
!     overflow, division-by-zero, and invalid operation.
      exmode = (/0,0,0/)
      Call x07cbf(exmode)

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

!     Evaluate the integral
      ifail = -1
      Call d01rgf(a,b,f,epsabs,epsrel,dinest,errest,nevals,iuser,ruser,ifail)

      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,*)
      If (ifail>=0) Then
        Write (nout,99997) 'DINEST', 'approximation to the integral', dinest
        Write (nout,99998) 'ERREST', 'estimate of the absolute error', errest
        Write (nout,99996) 'NEVALS', 'number of function evaluations', nevals
      End If

!     Restore the original halting mode
      Call x07cbf(exmode_old)
```

```
99999 Format (1X,A6,' - ',A30,' = ',F10.4)
99998 Format (1X,A6,' - ',A30,' = ',E10.2)
99997 Format (1X,A6,' - ',A30,' = ',F10.5)
99996 Format (1X,A6,' - ',A30,' = ',I10)
    End Program d01rgfe
```

10.2 Program Data

None.

10.3 Program Results

```
 D01RGF 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

 DINEST -  approximation to the integral =    3.81155
 ERREST - estimate of the absolute error =   0.34E-03
 NEVALS - number of function evaluations =        593
```