NAG Library Routine Document

D01DAF

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

D01DAF attempts to evaluate a double integral to a specified absolute accuracy by repeated applications of the method described by Patterson (1968) and Patterson (1969).

2 Specification

```
SUBROUTINE D01DAF (YA, YB, PHI1, PHI2, F, ABSACC, ANS, NPTS, IFAIL)

INTEGER            NPTS, IFAIL
REAL (KIND=nag_wp) YA, YB, PHI1, PHI2, F, ABSACC, ANS
EXTERNAL           PHI1, PHI2, F
```

3 Description

D01DAF attempts to evaluate a definite integral of the form

$$I = \int_a^b \int_{\phi_1(y)}^{\phi_2(y)} f(x,y)\,dx\,dy$$

where a and b are constants and $\phi_1(y)$ and $\phi_2(y)$ are functions of the variable y.

The integral is evaluated by expressing it as

$$I = \int_a^b F(y)\,dy, \qquad \text{where} \qquad F(y) = \int_{\phi_1(y)}^{\phi_2(y)} f(x,y)\,dx.$$

Both the outer integral I and the inner integrals $F(y)$ are evaluated by the method, described by Patterson (1968) and Patterson (1969), of the optimum addition of points to Gauss quadrature formulae.

This method uses a family of interlacing common point formulae. Beginning with the 3-point Gauss rule, formulae using 7, 15, 31, 63, 127 and finally 255 points are derived. Each new formula contains all the points of the earlier formulae so that no function evaluations are wasted. Each integral is evaluated by applying these formulae successively until two results are obtained which differ by less than the specified absolute accuracy.

4 References

Patterson T N L (1968) On some Gauss and Lobatto based integration formulae *Math. Comput.* **22** 877–881

Patterson T N L (1969) The optimum addition of points to quadrature formulae, errata *Math. Comput.* **23** 892

5 Parameters

1: YA – REAL (KIND=nag_wp) *Input*

On entry: a, the lower limit of the integral.

2: YB – REAL (KIND=nag_wp) *Input*

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

3: PHI1 – REAL (KIND=nag_wp) FUNCTION, supplied by the user. *External Procedure*

PHI1 must return the lower limit of the inner integral for a given value of y.

The specification of PHI1 is:

```
FUNCTION PHI1 (Y)
REAL (KIND=nag_wp) PHI1
REAL (KIND=nag_wp) Y
```

1: Y – REAL (KIND=nag_wp) *Input*

On entry: the value of y for which the lower limit must be evaluated.

PHI1 must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D01DAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

4: PHI2 – REAL (KIND=nag_wp) FUNCTION, supplied by the user. *External Procedure*

PHI2 must return the upper limit of the inner integral for a given value of y.

The specification of PHI2 is:

```
FUNCTION PHI2 (Y)
REAL (KIND=nag_wp) PHI2
REAL (KIND=nag_wp) Y
```

1: Y – REAL (KIND=nag_wp) *Input*

On entry: the value of y for which the upper limit must be evaluated.

PHI2 must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D01DAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

5: 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, Y)
REAL (KIND=nag_wp) F
REAL (KIND=nag_wp) X, Y
```

1: X – REAL (KIND=nag_wp) *Input*
2: Y – REAL (KIND=nag_wp) *Input*

On entry: the coordinates of the point (x, y) 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 D01DAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

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

On entry: the absolute accuracy requested.

7: ANS – REAL (KIND=nag_wp) *Output*

On exit: the estimated value of the integral.

8: NPTS – INTEGER *Output*

On exit: the total number of function evaluations.

9: 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: D01DAF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL $= 1$

This indicates that 255 points have been used in the outer integral and convergence has not been obtained. All the inner integrals have, however, converged. In this case ANS may still contain an approximate estimate of the integral.

IFAIL $= 10 \times n$

This indicates that the outer integral has converged but n inner integrals have failed to converge with the use of 255 points. In this case ANS may still contain an approximate estimate of the integral, but its reliability will decrease as n increases.

IFAIL $= 10 \times n + 1$

This indicates that both the outer integral and n of the inner integrals have not converged. ANS may still contain an approximate estimate of the integral, but its reliability will decrease as n increases.

7 Accuracy

The absolute accuracy is specified by the variable ABSACC. If, on exit, IFAIL $= 0$ then the result is most likely correct to this accuracy. Even if IFAIL is nonzero on exit, it is still possible that the calculated result could differ from the true value by less than the given accuracy.

8 Further Comments

The time taken by D01DAF depends upon the complexity of the integrand and the accuracy requested.

With Patterson's method accidental convergence may occasionally occur, when two estimates of an integral agree to within the requested accuracy, but both estimates differ considerably from the true result. This could occur in either the outer integral or in one or more of the inner integrals.

If it occurs in the outer integral then apparent convergence is likely to be obtained with considerably fewer integrand evaluations than may be expected. If it occurs in an inner integral, the incorrect value could make the function $F(y)$ appear to be badly behaved, in which case a very large number of pivots may be needed for the overall evaluation of the integral. Thus both unexpectedly small and unexpectedly large numbers of integrand evaluations should be considered as indicating possible trouble. If accidental convergence is suspected, the integral may be recomputed, requesting better accuracy; if the new request is more stringent than the degree of accidental agreement (which is of course unknown), improved results should be obtained. This is only possible when the accidental agreement is not better than machine accuracy. It should be noted that the routine requests the same accuracy for the inner integrals as for the outer integral. In practice it has been found that in the vast majority of cases this has proved to be adequate for the overall result of the double integral to be accurate to within the specified value.

The routine is not well-suited to non-smooth integrands, i.e., integrands having some kind of analytic discontinuity (such as a discontinuous or infinite partial derivative of some low-order) in, on the boundary of, or near, the region of integration. **Warning**: such singularities may be induced by incautiously presenting an apparently smooth interval over the positive quadrant of the unit circle, R

$$I = \int_R (x+y)\,dx\,dy.$$

This may be presented to D01DAF as

$$I = \int_0^1 dy \int_0^{\sqrt{1-y^2}} (x+y)\,dx = \int_0^1 \left(\tfrac{1}{2}\left(1-y^2\right) + y\sqrt{1-y^2}\right) dy$$

but here the outer integral has an induced square-root singularity stemming from the way the region has been presented to D01DAF. This situation should be avoided by re-casting the problem. For the example given, the use of polar coordinates would avoid the difficulty:

$$I = \int_0^1 dr \int_0^{\frac{\pi}{2}} r^2(\cos\upsilon + \sin\upsilon)\,d\upsilon.$$

9 Example

This example evaluates the integral discussed in Section 8, presenting it to D01DAF first as

$$\int_0^1 \int_0^{\sqrt{1-y^2}} (x+y)\,dx\,dy$$

and then as

$$\int_0^1 \int_0^{\frac{\pi}{2}} r^2(\cos\upsilon + \sin\upsilon)\,d\upsilon\,dr.$$

Note the difference in the number of function evaluations.

9.1 Program Text

```
!   D01DAF Example Program Text
!   Mark 24 Release. NAG Copyright 2012.

    Module d01dafe_mod

!     D01DAF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                    :: nout = 6
    Contains
      Function phi1(y)
```

```
!       .. Function Return Value ..
        Real (Kind=nag_wp)                    :: phi1
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)       :: y
!       .. Executable Statements ..
        phi1 = 0.0E0_nag_wp

        Return

      End Function phi1
      Function phi2a(y)

!       .. Function Return Value ..
        Real (Kind=nag_wp)                    :: phi2a
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)       :: y
!       .. Intrinsic Procedures ..
        Intrinsic                             :: sqrt
!       .. Executable Statements ..
        phi2a = sqrt(1.0E0_nag_wp-y*y)

        Return

      End Function phi2a
      Function fa(x,y)

!       .. Function Return Value ..
        Real (Kind=nag_wp)                    :: fa
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)       :: x, y
!       .. Executable Statements ..
        fa = x + y

        Return

      End Function fa
      Function phi2b(y)

!       .. Use Statements ..
        Use nag_library, Only: x01aaf
!       .. Function Return Value ..
        Real (Kind=nag_wp)                    :: phi2b
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)       :: y
!       .. Executable Statements ..
        phi2b = 0.5E0_nag_wp*x01aaf(y)

        Return

      End Function phi2b
      Function fb(x,y)

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

        Return

      End Function fb
    End Module d01dafe_mod
    Program d01dafe

!     D01DAF Example Main Program

!     .. Use Statements ..
```

```
      Use nag_library, Only: d01daf, nag_wp
      Use d01dafe_mod, Only: fa, fb, nout, phi1, phi2a, phi2b
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                 :: absacc, ans, ya, yb
      Integer                            :: ifail, npts
!     .. Executable Statements ..
      Write (nout,*) 'D01DAF Example Program Results'

      ya = 0.0E0_nag_wp
      yb = 1.0E0_nag_wp
      absacc = 1.0E-6_nag_wp

      ifail = 0
      Call d01daf(ya,yb,phi1,phi2a,fa,absacc,ans,npts,ifail)

      Write (nout,*)
      Write (nout,*) 'First formulation'
      Write (nout,99999) 'Integral =', ans
      Write (nout,99998) 'Number of function evaluations =', npts

      ifail = 0
      Call d01daf(ya,yb,phi1,phi2b,fb,absacc,ans,npts,ifail)

      Write (nout,*)
      Write (nout,*) 'Second formulation'
      Write (nout,99999) 'Integral =', ans
      Write (nout,99998) 'Number of function evaluations =', npts

99999 Format (1X,A,F9.4)
99998 Format (1X,A,I5)
    End Program d01dafe
```

9.2 Program Data

None.

9.3 Program Results

```
 D01DAF Example Program Results

 First formulation
 Integral =   0.6667
 Number of function evaluations =  189

 Second formulation
 Integral =   0.6667
 Number of function evaluations =   89
```