NAG Library Routine Document

D04AAF

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

D04AAF calculates a set of derivatives (up to order 14) of a function of one real variable at a point, together with a corresponding set of error estimates, using an extension of the Neville algorithm.

2 Specification

```
SUBROUTINE D04AAF (XVAL, NDER, HBASE, DER, EREST, FUN, IFAIL)
INTEGER             NDER, IFAIL
REAL (KIND=nag_wp) XVAL, HBASE, DER(14), EREST(14), FUN
EXTERNAL            FUN
```

3 Description

D04AAF provides a set of approximations:

$$\mathrm{DER}(j), \quad j = 1, 2, \ldots, n$$

to the derivatives:

$$f^{(j)}(x_0), \quad j = 1, 2, \ldots, n$$

of a real valued function $f(x)$ at a real abscissa x_0, together with a set of error estimates:

$$\mathrm{EREST}(j), \quad j = 1, 2, \ldots, n$$

which hopefully satisfy:

$$\left|\mathrm{DER}(j) - f^{(j)}(x_0)\right| < \mathrm{EREST}(j), \quad j = 1, 2, \ldots, n.$$

You must provide the value of x_0, a value of n (which is reduced to 14 should it exceed 14), a subroutine which evaluates $f(x)$ for all real x, and a step length h. The results $\mathrm{DER}(j)$ and $\mathrm{EREST}(j)$ are based on 21 function values:

$$f(x_0), f(x_0 \pm (2i-1)h), \quad i = 1, 2, \ldots, 10.$$

Internally D04AAF calculates the odd order derivatives and the even order derivatives separately. There is an option you can use for restricting the calculation to only odd (or even) order derivatives. For each derivative the routine employs an extension of the Neville Algorithm (see Lyness and Moler (1969)) to obtain a selection of approximations.

For example, for odd derivatives, based on 20 function values, D04AAF calculates a set of numbers:

$$T_{k,p,s}, \quad p = s, s+1, \ldots, 6, \quad k = 0, 1, \ldots, 9-p$$

each of which is an approximation to $f^{(2s+1)}(x_0)/(2s+1)!$. A specific approximation $T_{k,p,s}$ is of polynomial degree $2p+2$ and is based on polynomial interpolation using function values $f(x_0 \pm (2i-1)h)$, for $k = k, \ldots, k+p$. In the absence of round-off error, the better approximations would be associated with the larger values of p and of k. However, round-off error in function values has an increasingly contaminating effect for successively larger values of p. This routine proceeds to make a judicious choice between all the approximations in the following way.

For a specified value of s, let:

$$R_p = U_p - L_p, \quad p = s, s+1, \ldots, 6$$

where $U_p = \max_k\left(T_{k,p,s}\right)$ and $L_p = \min_k\left(T_{k,p,s}\right)$, for $k = 0, 1, \ldots, 9-p$, and let $\bar{p}$ be such that $R_{\bar{p}} = \min_p\left(R_p\right)$, for $p = s, \ldots, 6$.

The routine returns:

$$\mathrm{DER}(2s+1) = \frac{1}{8-\bar{p}} \times \left\{\sum_{k=0}^{9-\bar{p}} T_{k,\bar{p},s} - U_{\bar{p}} - L_{\bar{p}}\right\}(2s+1)!$$

and

$$\mathrm{EREST}(2s+1) = R_{\bar{p}} \times (2s+1)! \times K_{2s+1}$$

where K_j is a safety factor which has been assigned the values:

$$\begin{array}{ll} K_j = 1, & j \leq 9 \\ K_j = 1.5, & j = 10, 11 \\ K_j = 2 & j \geq 12, \end{array}$$

on the basis of performance statistics.

The even order derivatives are calculated in a precisely analogous manner.

4 References

Lyness J N and Moler C B (1966) van der Monde systems and numerical differentiation *Numer. Math.* **8** 458–464

Lyness J N and Moler C B (1969) Generalised Romberg methods for integrals of derivatives *Numer. Math.* **14** 1–14

5 Parameters

1: XVAL – REAL (KIND=nag_wp) *Input*

On entry: the point at which the derivatives are required, x_0.

2: NDER – INTEGER *Input*

On entry: must be set so that its absolute value is the highest order derivative required.

NDER > 0
: All derivatives up to order $\min(\mathrm{NDER}, 14)$ are calculated.

NDER < 0 and NDER is even
: Only even order derivatives up to order $\min(-\mathrm{NDER}, 14)$ are calculated.

NDER < 0 and NDER is odd
: Only odd order derivatives up to order $\min(-\mathrm{NDER}, 13)$ are calculated.

3: HBASE – REAL (KIND=nag_wp) *Input*

On entry: the initial step length which may be positive or negative. For advice on the choice of HBASE see Section 9.

Constraint: HBASE $\neq 0.0$.

4: DER(14) – REAL (KIND=nag_wp) array *Output*

On exit: DER(j) contains an approximation to the jth derivative of $f(x)$ at $x =$ XVAL, so long as the jth derivative is one of those requested by you when specifying NDER. For other values of j, DER(j) is unused.

5: EREST(14) – REAL (KIND=nag_wp) array *Output*

On exit: an estimate of the absolute error in the corresponding result DER(j) so long as the jth derivative is one of those requested by you when specifying NDER. The sign of EREST(j) is positive unless the result DER(j) is questionable. It is set negative when $|\text{DER}(j)| < |\text{EREST}(j)|$ or when for some other reason there is doubt about the validity of the result DER(j) (see Section 6). For other values of j, EREST(j) is unused.

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

FUN must evaluate the function $f(x)$ at a specified point.

The specification of FUN is:

```
FUNCTION FUN (X)
REAL (KIND=nag_wp) FUN
REAL (KIND=nag_wp) X
```

1: X – REAL (KIND=nag_wp) *Input*

On entry: the value of the argument x.

If you have equally spaced tabular data, the following information may be useful:

(i) in any call of D04AAF the only values of x for which $f(x)$ will be required are $x = \text{XVAL}$ and $x = \text{XVAL} \pm (2j-1)\text{HBASE}$, for $j = 1, 2, \ldots, 10$; and

(ii) $f(x_0)$ is always computed, but it is disregarded when only odd order derivatives are required.

FUN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which D04AAF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

7: 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 $= 1$

On entry, NDER $= 0$,
or HBASE $= 0.0$.

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.

If IFAIL has a value zero on exit then D04AAF has terminated successfully, but before any use is made of a derivative DER(j) the value of EREST(j) must be checked.

7 Accuracy

The accuracy of the results is problem dependent. An estimate of the accuracy of each result DER(j) is returned in EREST(j) (see Sections 3, 5 and 9).

A basic feature of any floating-point routine for numerical differentiation based on real function values on the real axis is that successively higher order derivative approximations are successively less accurate. It is expected that in most cases DER(14) will be unusable. As an aid to this process, the sign of EREST(j) is set negative when the estimated absolute error is greater than the approximate derivative itself, i.e., when the approximate derivative may be so inaccurate that it may even have the wrong sign. It is also set negative in some other cases when information available to the routine indicates that the corresponding value of DER(j) is questionable.

The actual values in EREST depend on the accuracy of the function values, the properties of the machine arithmetic, the analytic properties of the function being differentiated and the user-supplied step length HBASE (see Section 9). The only hard and fast rule is that for a given FUN(XVAL) and HBASE, the values of EREST(j) increase with increasing j. The limit of 14 is dictated by experience. Only very rarely can one obtain meaningful approximations for higher order derivatives on conventional machines.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by D04AAF depends on the time spent for function evaluations. Otherwise the time is roughly equivalent to that required to evaluate the function 21 times and calculate a finite difference table having about 200 entries in total.

The results depend very critically on the choice of the user-supplied step length HBASE. The overall accuracy is diminished as HBASE becomes small (because of the effect of round-off error) and as HBASE becomes large (because the discretization error also becomes large). If the routine is used four or five times with different values of HBASE one can find a reasonably good value. A process in which the value of HBASE is successively halved (or doubled) is usually quite effective. Experience has shown that in cases in which the Taylor series for FUN(X) about XVAL has a finite radius of convergence R, the choices of HBASE $> R/19$ are not likely to lead to good results. In this case some function values lie outside the circle of convergence.

10 Example

This example evaluates the odd-order derivatives of the function:

$$f(x) = \tfrac{1}{2}e^{2x-1}$$

up to order 7 at the point $x = \frac{1}{2}$. Several different values of HBASE are used, to illustrate that:

(i) extreme choices of HBASE, either too large or too small, yield poor results;

(ii) the quality of these results is adequately indicated by the values of EREST.

10.1 Program Text

```
!   D04AAF Example Program Text
!   Mark 25 Release. NAG Copyright 2014.

    Module d04aafe_mod

!     D04AAF 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                                :: fun
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter, Public :: h_init = 0.5_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: h_reduce = 0.1_nag_wp
      Real (Kind=nag_wp), Parameter, Public :: xval = 0.5_nag_wp
      Integer, Parameter, Public            :: nder = -7, nout = 6
!   nder: abs(nder) is largest order derivative required;
!         nder < 0 means only odd or even derivatives.
!   h_init:   initial step size.
!   h_reduce: reduction factor applied to successive step sizes.
!   xval:     derivatives evaluated at x=xval.
    Contains
      Function fun(x)

!       .. Function Return Value ..
        Real (Kind=nag_wp)                  :: fun
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In)     :: x
!       .. Intrinsic Procedures ..
        Intrinsic                           :: exp
!       .. Executable Statements ..
        fun = 0.5_nag_wp*exp(2.0_nag_wp*x-1.0_nag_wp)

        Return

      End Function fun
    End Module d04aafe_mod
    Program d04aafe

!     D04AAF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d04aaf, nag_wp
      Use d04aafe_mod, Only: fun, h_init, h_reduce, nder, nout, xval
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)                  :: hbase
      Integer                             :: i, ifail, j, k, l
!     .. Local Arrays ..
      Real (Kind=nag_wp)                  :: der(14), erest(14)
!     .. Intrinsic Procedures ..
      Intrinsic                           :: abs
!     .. Executable Statements ..
      Write (nout,*) 'D04AAF Example Program Results'

      Write (nout,*)
      Write (nout,*) 'Four separate runs to calculate the first &
        &four odd order derivatives of'
      Write (nout,*) '   FUN(X) = 0.5*exp(2.0*X-1.0) at X = 0.5.'
```

```
      Write (nout,*) 'The exact results are 1, 4, 16 and 64'
      Write (nout,*)
      Write (nout,*) 'Input parameters common to all four runs'
      Write (nout,99999) '  XVAL = ', xval, '    NDER = ', nder, &
        '    IFAIL = 0'
      Write (nout,*)

      hbase = h_init
      l = abs(nder)

      If (nder>=0) Then
        j = 1
      Else
        j = 2
      End If

      Do k = 1, 4

        ifail = 0
        Call d04aaf(xval,nder,hbase,der,erest,fun,ifail)

        Write (nout,*)
        Write (nout,99998) 'with step length', hbase, '  the results are'
        Write (nout,*) 'Order        Derivative       Error estimate'

        Do i = 1, l, j
          Write (nout,99997) i, der(i), erest(i)
        End Do

        hbase = hbase*h_reduce
      End Do

99999 Format (1X,A,F4.1,A,I2,A)
99998 Format (1X,A,F9.4,A)
99997 Format (1X,I2,2E21.4)
    End Program d04aafe
```

10.2 Program Data

None.

10.3 Program Results

```
 D04AAF Example Program Results

 Four separate runs to calculate the first four odd order derivatives of
    FUN(X) = 0.5*exp(2.0*X-1.0) at X = 0.5.
 The exact results are 1, 4, 16 and 64

 Input parameters common to all four runs
   XVAL =  0.5    NDER = -7    IFAIL = 0


 with step length   0.5000  the results are
 Order        Derivative       Error estimate
  1           0.1392E+04          -0.1073E+06
  3          -0.3139E+04          -0.1438E+06
  5           0.8762E+04          -0.2479E+06
  7          -0.2475E+05          -0.4484E+06

 with step length   0.0500  the results are
 Order        Derivative       Error estimate
  1           0.1000E+01           0.1529E-10
  3           0.4000E+01           0.2112E-08
  5           0.1600E+02           0.3815E-06
  7           0.6400E+02           0.7384E-04

 with step length   0.0050  the results are
 Order        Derivative       Error estimate
  1           0.1000E+01           0.1277E-13
```

```
  3           0.4000E+01           0.4190E-09
  5           0.1600E+02           0.1463E-04
  7           0.6404E+02           0.2973E+00

 with step length   0.0005  the results are
 Order        Derivative       Error estimate
  1           0.1000E+01           0.1427E-12
  3           0.4000E+01           0.3087E-06
  5           0.1599E+02           0.6331E+00
  7           0.3825E+05          -0.1964E+07
```