NAG Toolbox for MATLAB

d04aa

1 Purpose

d04aa 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 Syntax

```
[der, erest, ifail] = d04aa(xval, nder, hbase, fun)
```

3 Description

d04aa provides a set of approximations:

$$\mathbf{der}(j), \qquad j = 1, 2, \ldots, n$$

to the derivatives:

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

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

$$\mathbf{erest}(j), \qquad j = 1, 2, \ldots, n$$

which hopefully satisfy:

$$\left|\mathbf{der}(j) - f^{(j)}(x_0)\right| < \mathbf{erest}(j), \qquad j = 1, 2, \ldots, n.$$

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

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

Internally the function calculates the odd order derivatives and the even order derivatives separately. There is a user option for restricting the calculation to only odd (or even) order derivatives. For each derivative the function 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, the function calculates a set of numbers:

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

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 $i = k, k+1, \ldots, k+p$. 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 function proceeds to make a judicious choice between all the approximations in the following way.

For a specified value of s, let:

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

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

The function returns:

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

and

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

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

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

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

5.1 Compulsory Input Parameters

1: **xval – double scalar**

The point at which the derivatives are required, x_0.

2: **nder – int32 scalar**

Must be set so that its absolute value is the highest order derivative required.

nder > 0

All derivatives up to order $\min(\mathbf{nder}, 14)$ are calculated.

nder < 0 and **nder** is even

Only even order derivatives up to order $\min(-\mathbf{nder}, 14)$ are calculated.

nder < 0 and **nder** is odd

Only odd order derivatives up to order $\min(-\mathbf{nder}, 13)$ are calculated.

3: **hbase – double scalar**

The initial step length which may be positive or negative.

(If set to zero the function does not proceed with any calculation, but sets the error flag **ifail** and returns to the (sub)program from which d04aa is called.) For advice on the choice of **hbase** see Section 8.

4: **fun – string containing name of m-file**

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

Its specification is:

```
[result] = fun(x)
```

Input Parameters

1: **x – double scalar**

The value of the argument x.

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

(i) in any call of d04aa the only values of **x** that will be required are $\mathbf{x} = \mathbf{xval}$ and $\mathbf{x} = \mathbf{xval} \pm (2j-1)\mathbf{hbase}$, for $j = 1, 2, \ldots, 10$; and

(ii) **fun**(**xval**) is always called, but it is disregarded when only odd order derivatives are required.

Output Parameters

1: **result – double scalar**

The result of the function.

5.2 Optional Input Parameters

None.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **der**(**14**) **– double array**

An approximation to the jth derivative of $f(x)$ at $x = \mathbf{xval}$, 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.

2: **erest**(**14**) **– double array**

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 $|\mathbf{der}(j)| < |\mathbf{erest}(j)|$ 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.

3: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, **nder** = 0,
or **hbase** = 0.

If **ifail** has a value zero on exit then d04aa 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 8).

A basic feature of any floating-point function 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 function 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 8). The only hard and fast rule is that for a given **fun**$(\mathbf{x})$ 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 Further Comments

The time taken by d04aa 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 function 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**$(\mathbf{x})$ about **xval** has a finite radius of convergence R, the choices of **hbase** $> R/21$ are not likely to lead to good results. In this case some function values lie outside the circle of convergence.

9 Example

d04aa_fun.m

```
function [result] = fun(x)
  result = 0.5d0*exp(2.0d0*x-1.0d0);
```

```
xval = 0.5;
nder = int32(-7);
hbase = 0.5;
[der, erest, ifail] = d04aa(xval, nder, hbase, 'd04aa_fun')
```

```
der =
   1.0e+04 *
    0.1392
         0
   -0.3139
         0
    0.8762
         0
   -2.4753
         0
         0
         0
         0
         0
         0
```

```
         0
erest =
   1.0e+05 *
   -1.0734
         0
   -1.4378
         0
   -2.4790
         0
   -4.4838
         0
         0
         0
         0
         0
         0
         0
ifail =
           0
```