NAG Toolbox

nag_numdiff_rcomm (d04ba)

1 Purpose

nag_numdiff_rcomm (d04ba) calculates a set of derivatives (up to order 14) of a function at a point with respect to a single variable. A corresponding set of error estimates is also returned. Derivatives are calculated using an extension of the Neville algorithm. This function differs from nag_numdiff (d04aa), in that the abscissae and corresponding function values must be calculated before this function is called. The abscissae may be generated using nag_numdiff_sample (d04bb).

2 Syntax

```
[der, erest, ifail] = nag_numdiff_rcomm(xval, fval)
[der, erest, ifail] = d04ba(xval, fval)
```

3 Description

nag_numdiff_rcomm (d04ba) provides a set of approximations:

$$\mathbf{der}(j), \quad j = 1, 2, \ldots, 14$$

to the derivatives:

$$f^{(j)}(x_0), \quad j = 1, 2, \ldots, 14$$

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

$$\mathbf{erest}(j), \quad j = 1, 2, \ldots, 14$$

which hopefully satisfy:

$$\left|\mathbf{der}(j) - f^{(j)}(x_0)\right| < \mathbf{erest}(j), \quad j = 1, 2, \ldots, 14.$$

The results $\mathbf{der}(j)$ and $\mathbf{erest}(j)$ are based on 21 function values:

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

The abscissae x and the corresponding function values $f(x)$ should be passed into nag_numdiff_rcomm (d04ba) as the vectors **xval** and **fval** respectively. The step size h is derived from the abscissae in **xval**. See Section 9 for a discussion of how the derived value of h may affect the results of nag_numdiff_rcomm (d04ba). The order in which the abscissae and function values are stored in **xval** and **fval** is irrelevant, provided that the function value at any given index corresponds to the value of the abscissa at the same index. Abscissae may be automatically generated using nag_numdiff_sample (d04bb) if desired. For each derivative nag_numdiff_rcomm (d04ba) employs an extension of the Neville Algorithm (see Lyness and Moler (1969)) to obtain a selection of approximations.

For example, for odd derivatives, this function 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 function proceeds to make a judicious choice between all the approximations in the following way.

For a specified value of s, let:

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

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 $p = s, \ldots, 6$.

This 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

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

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 (1969) Generalised Romberg methods for integrals of derivatives *Numer. Math.* **14** 1–14

5 Parameters

5.1 Compulsory Input Parameters

1: **xval**(**21**) – REAL (KIND=nag_wp) array

The abscissae at which the function has been evaluated, as described in Section 3. These can be generated by calling nag_numdiff_sample (d04bb). The order of the abscissae is irrelevant.

Constraint: the values in **xval** must span the set $\{x_0, x_0 \pm (2j-1)h\}$, for $j = 1, 2, \ldots, 10$.

2: **fval**(**21**) – REAL (KIND=nag_wp) array

fval(j) must contain the function value at **xval**(j), for $j = 1, 2, \ldots, 21$.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **der**(**14**) – REAL (KIND=nag_wp) array

The 14 derivative estimates.

2: **erest**(**14**) – REAL (KIND=nag_wp) array

The 14 error estimates for the derivatives.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, the values of **xval** are not correctly spaced.

The derived h is below tolerance.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

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 Section 3, Section 5 and Section 9).

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 nag_numdiff_rcomm (d04ba) 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 step length h (see Section 9). The only hard and fast rule is that for a given objective function and h, 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 results depend very critically on the choice of the step length h. The overall accuracy is diminished as h becomes small (because of the effect of round-off error) and as h becomes large (because the discretization error also becomes large). If this function is used four or five times with different values of h one can find a reasonably good value. A process in which the value of h is successively halved (or doubled) is usually quite effective. Experience has shown that in cases in which the Taylor series for the objective function about x_0 has a finite radius of convergence R, the choices of $h > R/19$ are not likely to lead to good results. In this case some function values lie outside the circle of convergence.

As mentioned, the order of the abscissae in **xval** does not matter, provided the corresponding values of **fval** are ordered identically. If the abscissae are generated by nag_numdiff_sample (d04bb), then they will be in ascending order. In particular, the target abscissa x_0 will be stored in **xval**(11).

9 Example

This example evaluates the derivatives of the polygamma function, calculated using nag_specfun_psi_deriv_real (s14ae), and compares the first 3 derivatives calculated to those found using nag_specfun_psi_deriv_real (s14ae).

9.1 Program Text

```
     function d04ba_example

fprintf('d04ba example results\n\n');

fprintf('Find the derivatives of the polygamma (psi) function\n');
fprintf('using function values generated by s14ae.\n\n');
fprintf('Demonstrate the effect of successively reducing hbase.\n\n');

% Set the target location and calculate the objective value
x_0 = 0.05;
[f_0, ifail] = s14ae(x_0, nag_int(0));

% Compute the actual derivatives using s14ae for comparison
actder = zeros(3, 1);
for j=1:3
  [actder(j), ifail] = s14ae(x_0, nag_int(j));
end

der_comp = zeros(14, 3, 4);
% Attempt 4 applications, reducing hbase by factor 0.1 each time
for j=1:4
  % Generate the abscissa xval using d04bb
  hbase = 0.025*10^(-j);
  [xval] = d04bb(x_0, hbase);

  % Calculate the corresponding objective function values
  fval(11) = f_0;
  for k=[1:10,12:21]
    [fval(k), ifail] = s14ae(xval(k), nag_int(0));
  end

  % Call d04ba to calculate the derivative estimates
  [der, erest, ifail] = d04ba(xval, fval);

  % Store results in der_comp
  der_comp(:, 1, j) = hbase;
  der_comp(:, 2, j) = der;
  der_comp(:, 3, j) = erest;

end

% Display results for first 3 derivatives
for i=1:3
  fprintf('\nderivative %d calculated using s14ae: %11.4e\n', i, actder(i));
  fprintf('Derivative and error estimates for derivative %d\n', i);
  fprintf('      hbase        der(%d)      erest(%d)\n', i, i);
  for j=1:4
    fprintf('  %12.4e %12.4e %12.4e\n', der_comp(i,:,j));
  end
end
```

9.2 Program Results

```
     d04ba example results

Find the derivatives of the polygamma (psi) function
using function values generated by s14ae.

Demonstrate the effect of successively reducing hbase.

derivative 1 calculated using s14ae:  4.0153e+02
Derivative and error estimates for derivative 1
      hbase        der(1)      erest(1)
    2.5000e-03   4.0204e+02   1.3940e+02
    2.5000e-04   4.0153e+02   4.9170e-11
    2.5000e-05   4.0153e+02   2.1799e-10
    2.5000e-06   4.0153e+02   1.1826e-09
```

```
derivative 2 calculated using s14ae: -1.6002e+04
Derivative and error estimates for derivative 2
      hbase        der(2)      erest(2)
    2.5000e-03  -1.6022e+04   5.5760e+03
    2.5000e-04  -1.6002e+04   1.2831e-07
    2.5000e-05  -1.6002e+04   6.0543e-06
    2.5000e-06  -1.6002e+04   9.5762e-04

derivative 3 calculated using s14ae:  9.6001e+05
Derivative and error estimates for derivative 3
      hbase        der(3)      erest(3)
    2.5000e-03   9.1465e+05  -7.3750e+06
    2.5000e-04   9.6001e+05   2.3718e-04
    2.5000e-05   9.6001e+05   4.2253e-02
    2.5000e-06   9.6001e+05   5.9679e+01
```