NAG Library Function Document

nag_numdiff_1d_real_eval (d04bac)

1 Purpose

nag_numdiff_1d_real_eval (d04bac) 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_1d_real (d04aac), in that the abscissae and corresponding function values must be calculated before this function is called. The abscissae may be generated using nag_numdiff_1 d_real_absci (d04bbc).

2 Specification

```
#include <nag.h>
#include <nagd04.h>
void nag_numdiff_1d_real_eval (const double xval[], const double fval[],
     double der[], double erest[], NagError *fail)
```

3 Description

nag_numdiff_1d_real_eval (d04bac) provides a set of approximations:

$$\mathbf{der}[j-1], \quad j = 1, 2, \ldots, 14$$

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:

$$\mathbf{erest}[j-1], \quad j = 1, 2, \ldots, 14$$

which hopefully satisfy:

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

The results $\mathbf{der}[j-1]$ and $\mathbf{erest}[j-1]$ 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_1 d_real_eval (d04bac) 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_1d_real_eval (d04bac). 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_1 d_real_absci (d04bbc) if desired. For each derivative nag_numdiff_1d_real_eval (d04bac) 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:

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

and

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

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 Arguments

1: **xval**[**21**] – const double *Input*

On entry: the abscissae at which the function has been evaluated, as described in Section 3. These can be generated by calling nag_numdiff_1d_real_absci (d04bbc). 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**] – const double *Input*

On entry: **fval**$[j-1]$ must contain the function value at **xval**$[j-1]$, for $j = 1, 2, \ldots, 21$.

3: **der**[**14**] – double *Output*

On exit: the 14 derivative estimates.

4: **erest**[**14**] – double *Output*

On exit: the 14 error estimates for the derivatives.

5: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_SPACING_INCORRECT

On entry, the values of **xval** are not correctly spaced.
Derived $h = \langle value \rangle$.

NE_TOO_SMALL

The derived h is below tolerance.
Derived $h > \langle value \rangle$ is required. Derived $h = \langle value \rangle$.

7 Accuracy

The accuracy of the results is problem dependent. An estimate of the accuracy of each result **der**$[j-1]$ is returned in **erest**$[j-1]$ (see Sections 3, 5 and 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**$[13]$ will be unusable. As an aid to this process, the sign of **erest**$[j-1]$ 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_1d_real_eval (d04bac) indicates that the corresponding value of **der**$[j-1]$ 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**$[j-1]$ 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

nag_numdiff_1d_real_eval (d04bac) is not threaded in any implementation.

9 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_1d_real_absci (d04bbc), then they will be in ascending order. In particular, the target abscissa x_0 will be stored in **xval**[10].

10 Example

This example evaluates the derivatives of the polygamma function, calculated using nag_real_poly gamma (s14aec), and compares the first 3 derivatives calculated to those found using nag_real_polygamma (s14aec).

10.1 Program Text

```
/* nag_numdiff_1d_real_eval (d04bac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd04.h>
#include <nags.h>
#include <nagx04.h>

int main(void)
{
/* DER_COMP is used to store results in column major format only. */
#define DER_COMP(I, J, K) der_comp[I + J*n_hbase + K*n_hbase*n_der_comp]

  Integer exit_status = 0;
  double x_0;
  Integer n_der_comp, n_display, n_hbase;
  double hbase;
  Integer i, j, k;
  char title[51];
  double der[14], erest[14], fval[21], xval[21];
  double *actder = 0, *der_comp = 0;
  char *clabs = 0, **clabsc = 0;
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_numdiff_1d_real_eval (d04bac) Example Program Results\n");

  printf("\n"
         "Find the derivatives of the polygamma (psi) function\n"
         "using function values generated by nag_real_polygamma (s14aec).\n\n"
         "Demonstrate the effect of successively reducing hbase.\n\n");
  fflush(stdout);

  n_der_comp = 3;
```

```
  n_display = 3;
  n_hbase = 4;

  if (!(clabs = NAG_ALLOC(n_der_comp * 10, char)) ||
      !(clabsc = NAG_ALLOC(n_der_comp, char *)) ||
      !(actder = NAG_ALLOC(n_display, double)) ||
      !(der_comp = NAG_ALLOC(n_hbase * n_der_comp * 14, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  x_0 = 0.05;

  /* Select an initial separation distance hbase. */
  hbase = 0.0025;

  /* Compute the actual derivatives at target location x_0
   * using nag_real_polygamma (s14aec), for comparison.
   */
  for (j = 0; j < n_display; j++) {
    /* nag_real_polygamma (s14aec).
     * Derivative of the psi function psi(x).
     */
    actder[j] = nag_real_polygamma(x_0, j + 1, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_real_polygamma (s14aec).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  }

  /* Attempt n_hbase approximations, reducing hbase by factor 0.1 each time. */
  for (j = 0; j < n_hbase; j++) {
    /* nag_numdiff_1d_real_absci (d04bbc).
     * Generates sample points for function evaluations
     * by nag_numdiff_1d_real_eval (d04bac).
     */
    nag_numdiff_1d_real_absci(x_0, hbase, xval);

    /* Calculate the corresponding objective function values. */
    for (k = 0; k < 21; k++) {
      fval[k] = nag_real_polygamma(xval[k], 0, &fail);
      if (fail.code != NE_NOERROR) {
        printf("Error from nag_real_polygamma (s14aec).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
      }
    }

    /* Calculate the derivative estimates. */

    /* nag_numdiff_1d_real_eval (d04bac).
     * Numerical differentiation, user-supplied function values,
     * derivatives up to order 14,
     * derivatives with respect to one real variable.
     */
    nag_numdiff_1d_real_eval(xval, fval, der, erest, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_numdiff_1d_real_eval (d04bac).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

    /* Store results in DER_COMP in column major format. */
    for (i = 0; i < 14; i++) {
      DER_COMP(j, 0, i) = hbase;
      DER_COMP(j, 1, i) = der[i];
      DER_COMP(j, 2, i) = erest[i];
```

```
    }

    /* Decrease hbase for next loop. */
    hbase = hbase * 0.1;
  }

  /* Display results for first n_display derivatives. */
  for (j = 0; j < n_display; j++) {
    printf(" Derivative (%" NAG_IFMT ") calculated using "
           "nag_real_polygamma (s14aec): %13.4e\n", j + 1, actder[j]);
    fflush(stdout);

#ifdef _WIN32
    sprintf_s(title, (unsigned)_countof(title),
              "Derivative and error estimates for derivative (%" NAG_IFMT ")",
              j + 1);
#else
    sprintf(title,
            "Derivative and error estimates for derivative (%" NAG_IFMT ")",
            j + 1);
#endif
#ifdef _WIN32
    sprintf_s(&clabs[0], 10, "hbase");
#else
    sprintf(&clabs[0], "hbase");
#endif
#ifdef _WIN32
    sprintf_s(&clabs[10], 10, "der[%" NAG_IFMT "]", j);
#else
    sprintf(&clabs[10], "der[%" NAG_IFMT "]", j);
#endif
#ifdef _WIN32
    sprintf_s(&clabs[20], 10, "erest[%" NAG_IFMT "]", j);
#else
    sprintf(&clabs[20], "erest[%" NAG_IFMT "]", j);
#endif
    for (k = 0; k < n_der_comp; k++)
      clabsc[k] = &clabs[k * 10];

    /* nag_gen_real_mat_print_comp (x04cbc).
     * Print real general matrix (comprehensive).
     */
    nag_gen_real_mat_print_comp(Nag_ColMajor, Nag_GeneralMatrix,
                                Nag_NonUnitDiag, n_hbase, n_der_comp,
                                &DER_COMP(0, 0, j), n_hbase, NULL, title,
                                Nag_NoLabels, NULL, Nag_CharacterLabels,
                                (const char **) clabsc, 0, 0, NULL, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_gen_real_mat_print_comp (x04cbc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
    printf("\n");
  }

END:
  NAG_FREE(actder);
  NAG_FREE(der_comp);
  NAG_FREE(clabs);
  NAG_FREE(clabsc);

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_numdiff_1d_real_eval (d04bac) Example Program Results

Find the derivatives of the polygamma (psi) function
using function values generated by nag_real_polygamma (s14aec).

Demonstrate the effect of successively reducing hbase.

 Derivative (1) calculated using nag_real_polygamma (s14aec):    4.0153e+02
 Derivative and error estimates for derivative (1)
         hbase       der[0]     erest[0]
     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 nag_real_polygamma (s14aec):   -1.6002e+04
 Derivative and error estimates for derivative (2)
         hbase       der[1]     erest[1]
     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 nag_real_polygamma (s14aec):    9.6001e+05
 Derivative and error estimates for derivative (3)
         hbase       der[2]     erest[2]
     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
```