

# NAG Library Function Document

## nag\_numdiff\_1d\_real (d04aac)

### 1 Purpose

nag\_numdiff\_1d\_real (d04aac) 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

```
#include <nag.h>
#include <nagd04.h>

void nag_numdiff_1d_real (double xval, Integer nder, double hbase,
    double der[], double ertest[],
    double (*fun)(double x, Nag_Comm *comm),
    Nag_Comm *comm, NagError *fail)
```

### 3 Description

nag\_numdiff\_1d\_real (d04aac) provides a set of approximations:

$$\mathbf{der}[j-1], \quad j = 1, 2, \dots, n$$

to the derivatives:

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

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

$$\mathbf{erest}[j-1], \quad j = 1, 2, \dots, n$$

which hopefully satisfy:

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

You must provide the value of  $x_0$ , a value of  $n$  (which is reduced to 14 should it exceed 14), a function which evaluates  $f(x)$  for all real  $x$ , and a step length  $h$ . The results  $\mathbf{der}[j-1]$  and  $\mathbf{erest}[j-1]$  are based on 21 function values:

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

Internally nag\_numdiff\_1d\_real (d04aac) 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 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, nag\_numdiff\_1d\_real (d04aac) calculates a set of numbers:

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

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  $k = k, \dots, k+p$ . 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:

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

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

The 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:

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

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 Arguments

- 1: **xval** – double *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(**nder**, 14) are calculated.  
**nder** < 0 and **nder** is even  
Only even order derivatives up to order min(–**nder**, 14) are calculated.  
**nder** < 0 and **nder** is odd  
Only odd order derivatives up to order min(–**nder**, 13) are calculated.
- 3: **hbase** – double *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** ≠ 0.0.

- 4: **der[14]** – double *Output*  
*On exit:* **der**[ $j - 1$ ] contains an approximation to the  $j$ th derivative of  $f(x)$  at  $x = \mathbf{xval}$ , so long as the  $j$ th derivative is one of those requested by you when specifying **nder**. For other values of  $j$ , **der**[ $j - 1$ ] is unused.
- 5: **erest[14]** – double *Output*  
*On exit:* an estimate of the absolute error in the corresponding result **der**[ $j - 1$ ] so long as the  $j$ th derivative is one of those requested by you when specifying **nder**. The sign of **erest**[ $j - 1$ ] is positive unless the result **der**[ $j - 1$ ] is questionable. It is set negative when  $|\mathbf{der}[j - 1]| < |\mathbf{erest}[j - 1]|$  or when for some other reason there is doubt about the validity of the result **der**[ $j - 1$ ] (see Section 6). For other values of  $j$ , **erest**[ $j - 1$ ] is unused.
- 6: **fun** – function, supplied by the user *External Function*  
**fun** must evaluate the function  $f(x)$  at a specified point.

The specification of **fun** is:

```
double fun (double x, Nag_Comm *comm)
```

1: **x** – double *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 `nag_numdiff_1d_real (d04aac)` the only values of  $x$  for which  $f(x)$  will be required are  $x = \mathbf{xval}$  and  $x = \mathbf{xval} \pm (2j - 1)\mathbf{hbase}$ , for  $j = 1, 2, \dots, 10$ ; and
- (ii)  $f(x_0)$  is always computed, but it is disregarded when only odd order derivatives are required.

2: **comm** – Nag\_Comm \*

Pointer to structure of type Nag\_Comm; the following members are relevant to **fun**.

**user** – double \*

**iuser** – Integer \*

**p** – Pointer

The type Pointer will be `void *`. Before calling `nag_numdiff_1d_real (d04aac)` you may allocate memory and initialize these pointers with various quantities for use by **fun** when called from `nag_numdiff_1d_real (d04aac)` (see Section 2.3.1.1 in How to Use the NAG Library and its Documentation).

- 7: **comm** – Nag\_Comm \*  
The NAG communication argument (see Section 2.3.1.1 in How to Use the NAG Library and its Documentation).
- 8: **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  $\langle value \rangle$  had an illegal value.

### NE\_INT

On entry, **nder** = 0.

Constraint: **nder**  $\neq$  0.

### 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\_REAL

On entry, **hbase** = 0.0.

Constraint: **hbase**  $\neq$  0.0.

## 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 the function 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 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**[ $j - 1$ ] 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 (d04aac) is not threaded in any implementation.

## 9 Further Comments

The time taken by `nag_numdiff_1d_real` (d04aac) 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(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:

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

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

```
/* nag_numdiff_1d_real (d04aac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd04.h>

#ifdef __cplusplus
extern "C"
{
#endif
    static double NAG_CALL fun(double x, Nag_Comm *comm);
#ifdef __cplusplus
}
#endif

int main(void)
{
    static double ruser[1] = { -1.0 };
    Integer exit_status = 0;
    double hbase;
    Integer i, k, l, start, step;
    double h_init;
    double h_reduce;
    double xval;
    Integer nder;
    double der[14], erest[14];
    Nag_Comm comm;
    NagError fail;
```

```

INIT_FAIL(fail);

printf("nag_numdiff_ld_real (d04aac) Example Program Results\n");

/* For communication with user-supplied functions: */
comm.user = ruser;

/* abs(nder) is largest order derivative required. */
nder = -7;
l = abs(nder);
/* nder < 0 and nder is even means only even derivatives,
 * and nder < 0 and nder is odd, only odd derivatives.
 */
if (nder < 0) {
    start = (l % 2 ? 0 : 1);
    step = 2;
}
else {
    start = 0;
    step = 1;
}
/* Initial step size. */
h_init = 0.5;
hbase = h_init;
/* Reduction factor applied to successive step sizes. */
h_reduce = 0.1;
/* Derivatives will be evaluated at x = xval. */
xval = 0.5;

printf("\n"
       "Four separate runs to calculate the first four odd order derivatives "
       "of\n"
       "    fun(x) = 0.5*exp(2.0*x-1.0) at x = 0.5.\n"
       "The exact results are 1, 4, 16 and 64\n\n"
       "Input parameters common to all four runs\n"
       "    xval = %f      nder = %" NAG_IFMT "\n", xval, nder);

for (k = 0; k < 4; k++) {
    /* nag_numdiff_ld_real (d04aac).
     * Numerical differentiation, derivatives up to order 14,
     * function of one real variable.
     */
    nag_numdiff_ld_real(xval, nder, hbase, der, erest, fun, &comm, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_numdiff_ld_real (d04aac).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    printf("\n"
           "with step length %f    the results are\n"
           "Order      Derivative      Error estimate\n", hbase);

    for (i = start; i < MIN(l, 14); i += step)
        printf("%2" NAG_IFMT " %21.4e %21.4e\n", i + 1, der[i], erest[i]);

    hbase = hbase * h_reduce;
}

END:
return exit_status;
}

static double NAG_CALL fun(double x, Nag_Comm *comm)
{
    if (comm->user[0] == -1.0) {
        printf("(User-supplied callback fun, first invocation.)\n");
        comm->user[0] = 0.0;
    }
    return 0.5 * exp(2.0 * x - 1.0);
}

```

## 10.2 Program Data

None.

## 10.3 Program Results

nag\_numdiff\_ld\_real (d04aac) Example Program Results

Four separate runs to calculate the first four odd order derivatives of  
 $\text{fun}(x) = 0.5 \cdot \exp(2.0 \cdot x - 1.0)$  at  $x = 0.5$ .  
 The exact results are 1, 4, 16 and 64

Input parameters common to all four runs  
 $x_{\text{val}} = 0.500000$        $\text{nder} = -7$   
 (User-supplied callback fun, first invocation.)

with step length 0.500000      the results are

Order	Derivative	Error estimate
1	1.3919e+03	-1.0734e+05
3	-3.1386e+03	-1.4378e+05
5	8.7619e+03	-2.4790e+05
7	-2.4753e+04	-4.4838e+05

with step length 0.050000      the results are

Order	Derivative	Error estimate
1	1.0000e+00	1.5294e-11
3	4.0000e+00	2.1125e-09
5	1.6000e+01	3.8149e-07
7	6.4000e+01	7.3845e-05

with step length 0.005000      the results are

Order	Derivative	Error estimate
1	1.0000e+00	1.2768e-14
3	4.0000e+00	4.1903e-10
5	1.6000e+01	1.4629e-05
7	6.4039e+01	2.9729e-01

with step length 0.000500      the results are

Order	Derivative	Error estimate
1	1.0000e+00	1.4266e-13
3	4.0000e+00	3.0869e-07
5	1.5988e+01	6.3314e-01
7	3.8249e+04	-1.9644e+06

---