

NAG Library Function Document

nag_monotonic_intg (e01bhc)

1 Purpose

nag_monotonic_intg (e01bhc) evaluates the definite integral of a piecewise cubic Hermite interpolant over the interval $[a, b]$.

2 Specification

```
#include <nag.h>
#include <nage01.h>

void nag_monotonic_intg (Integer n, const double x[], const double f[],
                        const double d[], double a, double b, double *integral, NagError *fail)
```

3 Description

nag_monotonic_intg (e01bhc) evaluates the definite integral of a piecewise cubic Hermite interpolant, as computed by nag_monotonic_interpolant (e01bec), over the interval $[a, b]$.

If either a or b lies outside the interval from $x[0]$ to $x[n-1]$, computation of the integral involves extrapolation and a warning is returned.

The function is derived from routine PCHIA in Fritsch (1982).

4 References

Fritsch F N (1982) PCHIP final specifications *Report UCID-30194* Lawrence Livermore National Laboratory

5 Arguments

- | | | |
|----|---|---------------------|
| 1: | n – Integer | <i>Input</i> |
| | <i>On entry:</i> n must be unchanged from the previous call of nag_monotonic_interpolant (e01bec). | |
| 2: | x[n] – const double | <i>Input</i> |
| 3: | f[n] – const double | <i>Input</i> |
| 4: | d[n] – const double | <i>Input</i> |
| | <i>On entry:</i> x , f and d must be unchanged from the previous call of nag_monotonic_interpolant (e01bec). | |
| 5: | a – double | <i>Input</i> |
| 6: | b – double | <i>Input</i> |
| | <i>On entry:</i> the interval $[a, b]$ over which integration is to be performed. | |
| 7: | integral – double * | <i>Output</i> |
| | <i>On exit:</i> the value of the definite integral of the interpolant over the interval $[a, b]$. | |
| 8: | fail – NagError * | <i>Input/Output</i> |
| | The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation). | |

6 Error Indicators and Warnings

NE_INT_ARG_LT

On entry, **n** = $\langle value \rangle$.
 Constraint: **n** ≥ 2 .

NE_NOT_MONOTONIC

On entry, $\mathbf{x}[r-1] \geq \mathbf{x}[r]$ for $r = \langle value \rangle : \mathbf{x}[r-1] = \langle value \rangle$, $\mathbf{x}[r] = \langle value \rangle$.
 The values of $\mathbf{x}[r]$, for $r = 0, 1, \dots, n-1$, are not in strictly increasing order.

NW_INTERVAL_EXTRAPOLATE

On entry, limits **a**, **b** must not be outside interval $[\mathbf{x}[0], \mathbf{x}[n-1]]$, **a** = $\langle value \rangle$, **b** = $\langle value \rangle$, $\mathbf{x}[0] = \langle value \rangle$, $\mathbf{x}[\langle value \rangle] = \langle value \rangle$. Extrapolation was performed to compute the integral. The value returned is therefore unreliable.

7 Accuracy

The computational error in the value returned for **integral** should be negligible in most practical situations.

8 Parallelism and Performance

nag_monotonic_intg (e01bhc) is not threaded in any implementation.

9 Further Comments

The time taken by nag_monotonic_intg (e01bhc) is approximately proportional to the number of data points included within the interval $[a, b]$.

10 Example

This example program reads in values of **n**, **x**, **f** and **d**. It then reads in pairs of values for **a** and **b**, and evaluates the definite integral of the interpolant over the interval **(a,b)** until end-of-file is reached.

10.1 Program Text

```
/* nag_monotonic_intg (e01bhc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nage01.h>

int main(void)
{
  Integer exit_status = 0, n, r;
  NagError fail;
  double a, b, *d = 0, *f = 0, integral, *x = 0;

  INIT_FAIL(fail);

  printf("nag_monotonic_intg (e01bhc) Example Program Results\n");
#ifdef _WIN32
```

```

    scanf_s("%*[^\\n]"); /* Skip heading in data file */
#else
    scanf("%*[^\\n]"); /* Skip heading in data file */
#endif
#ifdef _WIN32
    scanf_s("%" NAG_IFMT " ", &n);
#else
    scanf("%" NAG_IFMT " ", &n);
#endif
    if (n >= 2) {
        if (!(d = NAG_ALLOC(n, double)) ||
            !(f = NAG_ALLOC(n, double)) || !(x = NAG_ALLOC(n, double)))
        {
            printf("Allocation failure\\n");
            exit_status = -1;
            goto END;
        }
    }
    else {
        printf("Invalid n.\\n");
        exit_status = 1;
        return exit_status;
    }
    for (r = 0; r < n; r++)
#ifdef _WIN32
        scanf_s("%lf%lf%lf", &x[r], &f[r], &d[r]);
#else
        scanf("%lf%lf%lf", &x[r], &f[r], &d[r]);
#endif
    printf("
    printf("
            a
            b
            Integral\\n");
    printf("
    printf("
            a
            b
            over (a,b)\\n");
    /* Read a, b pairs until end of file and compute
     * definite integrals.
     */
#ifdef _WIN32
    while (scanf_s("%lf%lf", &a, &b) != EOF)
#else
    while (scanf("%lf%lf", &a, &b) != EOF)
#endif
    {
        /* nag_monotonic_intg (e01bhc).
         * Evaluation of interpolant computed by
         * nag_monotonic_interpolant (e01bec), definite integral
         */
        nag_monotonic_intg(n, x, f, d, a, b, &integral, &fail);
        if (fail.code != NE_NOERROR) {
            printf("Error from nag_monotonic_intg (e01bhc).\\n%s\\n", fail.message);
            exit_status = 1;
            goto END;
        }
        printf("%13.4f      %13.4f      %13.4f\\n", a, b, integral);
    }
END:
    NAG_FREE(d);
    NAG_FREE(f);
    NAG_FREE(x);
    return exit_status;
}

```

10.2 Program Data

nag_monotonic_intg (e01bhc) Example Program Data

9		
7.990	0.00000E+0	0.00000E+0
8.090	0.27643E-4	5.52510E-4
8.190	0.43749E-1	0.33587E+0
8.700	0.16918E+0	0.34944E+0
9.200	0.46943E+0	0.59696E+0
10.00	0.94374E+0	6.03260E-2
12.00	0.99864E+0	8.98335E-4

15.00	0.99992E+0	2.93954E-5
20.00	0.99999E+0	0.00000E+0
7.99	20.0	
10.0	12.0	
12.0	10.0	
15.0	15.0	

10.3 Program Results

nag_monotonic_intg (e01bhc) Example Program Results

a	b	Integral over (a,b)
7.9900	20.0000	10.7648
10.0000	12.0000	1.9622
12.0000	10.0000	-1.9622
15.0000	15.0000	0.0000
