

## nag\_1d\_spline\_intg (e02bdc)

### 1. Purpose

**nag\_1d\_spline\_intg (e02bdc)** computes the definite integral of a cubic spline from its B-spline representation.

### 2. Specification

```
#include <nag.h>
#include <nage02.h>

void nag_1d_spline_intg(Nag_Spline *spline, double *integral,
                        NagError *fail)
```

### 3. Description

This function computes the definite integral of the cubic spline  $s(x)$  between the limits  $x = a$  and  $x = b$ , where  $a$  and  $b$  are respectively the lower and upper limits of the range over which  $s(x)$  is defined. It is assumed that  $s(x)$  is represented in terms of its B-spline coefficients  $c_i$ , for  $i = 1, 2, \dots, \bar{n} + 3$  and (augmented) ordered knot set  $\lambda_i$ , for  $i = 1, 2, \dots, \bar{n} + 7$ , with  $\lambda_i = a$ , for  $i = 1, 2, 3, 4$  and  $\lambda_i = b$ , for  $i = \bar{n} + 4, \bar{n} + 5, \bar{n} + 6, \bar{n} + 7$ , (see **nag\_1d\_spline\_fit\_knots (e02bac)**), i.e.

$$s(x) = \sum_{i=1}^q c_i N_i(x)$$

Here  $q = \bar{n} + 3$ ,  $\bar{n}$  is the number of intervals of the spline and  $N_i(x)$  denotes the normalised B-spline of degree 3 (order 4) defined upon the knots  $\lambda_i, \lambda_{i+1}, \dots, \lambda_{i+4}$ .

The method employed uses the formula given in Section 3 of Cox (1975).

**nag\_1d\_spline\_intg** can be used to determine the definite integrals of cubic spline fits and interpolants produced by **nag\_1d\_spline\_fit\_knots (e02bac)**, **nag\_1d\_spline\_interpolant (e01bac)** and **nag\_1d\_spline\_fit (e02bec)**.

### 4. Parameters

#### spline

Input: pointer to structure of type **Nag\_Spline** with the following members:

#### n - Integer

Input:  $\bar{n} + 7$ , where  $\bar{n}$  is the number of intervals of the spline (which is one greater than the number of interior knots, i.e., the knots strictly within the range  $a$  to  $b$ ) over which the spline is defined.

Constraint: **spline.n**  $\geq 8$ .

#### lamda - double \*

Input: a pointer to which memory of size **spline.n** must be allocated. **spline.lamda[j - 1]** must be set to the value of the  $j$ th member of the complete set of knots,  $\lambda_j$  for  $j = 1, 2, \dots, \bar{n} + 7$ .

Constraint: the  $\lambda_j$  must be in non-decreasing order with **spline.lamda[spline.n - 4] > spline.lamda[3]** and satisfy

**spline.lamda[0] = spline.lamda[1] = spline.lamda[2] = spline.lamda[3]**

and

**spline.lamda[spline.n - 4] = spline.lamda[spline.n - 3] = spline.lamda[spline.n - 2] = spline.lamda[spline.n - 1]**

#### c - double \*

Input: a pointer to which memory of size **spline.n - 4** must be allocated. **spline.c** holds the coefficient  $c_i$  of the B-spline  $N_i(x)$ , for  $i = 1, 2, \dots, \bar{n} + 3$ .

**integral**

Output: the value of the definite integral of  $s(x)$  between the limits  $x = a$  and  $x = b$ , where  $a = \lambda_4$  and  $b = \lambda_{\bar{n}+4}$ .

Under normal usage, the call to nag\_1d\_spline\_intg will follow a call to nag\_1d\_spline\_fit\_knots (e02bac), nag\_1d\_spline\_interpolant (e01bac) or nag\_1d\_spline\_fit (e02bec). In that case, the structure **spline** will have been set up correctly for input to nag\_1d\_spline\_intg.

**fail**

The NAG error parameter, see the Essential Introduction to the NAG C Library.

**5. Error Indications and Warnings****NE\_INT\_ARG\_LT**

On entry, **spline.n** must not be less than 8: **spline.n** =  $\langle \text{value} \rangle$ .

**NE\_KNOTS\_CONS**

On entry, the knots must satisfy the following constraints:

**spline.lamda**[**spline.n**−4] > **spline.lamda**[3], **spline.lamda**[*j*] ≥ **spline.lamda**[*j* − 1],  
for  $j = 1, 2, \dots, \text{spline.n} - 1$ , with equality in the cases  $j = 1, 2, 3, \text{spline.n} - 3, \text{spline.n} - 2$  and **spline.n** − 1.

**6. Further Comments**

The time taken by the function is approximately proportional to  $\bar{n} + 7$ .

**6.1. Accuracy**

The rounding errors are such that the computed value of the integral is exact for a slightly perturbed set of B-spline coefficients  $c_i$  differing in a relative sense from those supplied by no more than  $2.2 \times (\bar{n} + 3) \times \text{machine precision}$ .

**6.2. References**

Cox M G (1975) An Algorithm for Spline Interpolation *J. Inst. Math. Appl.* **15** 95–108.

**7. See Also**

nag\_1d\_spline\_interpolant (e01bac)  
nag\_1d\_spline\_fit\_knots (e02bac)  
nag\_1d\_spline\_fit (e02bec)

**8. Example**

Determine the definite integral over the interval  $0 \leq x \leq 6$  of a cubic spline having 6 interior knots at the positions  $\lambda = 1, 3, 3, 3, 4, 4$ , the 8 additional knots 0, 0, 0, 0, 6, 6, 6, 6, and the 10 B-spline coefficients 10, 12, 13, 15, 22, 26, 24, 18, 14, 12.

The input data items (using the notation of Section 4) comprise the following values in the order indicated:

$\bar{n} + 7$   
**spline.lamda**[*j*],                      for  $j = 0, 1, \dots, \text{spline.n} - 1$   
**spline.c**[*j*],                         for  $j = 0, 1, \dots, \text{spline.n} - 4$

The example program is written in a general form that will enable the definite integral of a cubic spline having an arbitrary number of knots to be computed. Any number of data sets may be supplied. The only changes required to the program relate to the size of the storage allocated to **spline.lamda** and **spline.c** within the structure **spline**.

**8.1. Program Text**

```
/* nag_1d_spline_intg(e02bdc) Example Program
*
* Copyright 1991 Numerical Algorithms Group.
*
* Mark 2, 1991.
```

```

*/

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nage02.h>

main()
{
    Integer j;
    double integral;
    Nag_Spline spline;

    Vprintf("e02bdc Example Program Results\n");
    Vscanf("%*[^\\n]"); /* Skip heading in data file */
    while(scanf("%ld",&(spline.n)) != EOF)
    {
        if (spline.n>0)
        {
            spline.c = NAG_ALLOC(spline.n, double);
            spline.lamda = NAG_ALLOC(spline.n, double);
            if (spline.c != (double *)0 && spline.lamda != (double *)0)
            {
                for (j=0; j<spline.n; j++)
                    Vscanf("%lf",&(spline.lamda[j]));
                for (j=0; j<spline.n-3; j++)
                    Vscanf("%lf",&(spline.c[j]));
                e02bdc(&spline, &integral, NAGERR_DEFAULT);
                Vprintf("Definite integral = %11.3e\\n",integral);
                NAG_FREE(spline.c);
                NAG_FREE(spline.lamda);
            }
            else
            {
                Vfprintf(stderr,"Storage allocation failed. Reduce the \
size of spline.n\\n");
                exit(EXIT_FAILURE);
            }
        }
        else
        {
            Vfprintf(stderr,"spline.n is out of range : spline.n = %ld\\n",
                spline.n);
            exit(EXIT_FAILURE);
        }
    }
    exit(EXIT_SUCCESS);
}

```

## 8.2. Program Data

e02bdc Example Program Data

14							
0.0	0.0	0.0	0.0	1.0	3.0	3.0	3.0
4.0	4.0	6.0	6.0	6.0	6.0		
10.0	12.0	13.0	15.0	22.0	26.0	24.0	18.0
14.0	12.0						

## 8.3. Program Results

e02bdc Example Program Results  
 Definite integral = 1.000e+02

---