NAG Library Function Document

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

nag_1d_spline_intg (e02bdc) 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 $i = 1, 2, \ldots, \bar{n} + 3$ and (augmented) ordered knot set λ_i, for $i = 1, 2, \ldots, \bar{n} + 7$, with $\lambda_i = a$, for $i = 1, 2, 3, 4$ and $\lambda_i = b$, for $i = \bar{n} + 4, \ldots, \bar{n} + 7$, (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 normalized B-spline of degree 3 (order 4) defined upon the knots $\lambda_i, \lambda_{i+1}, \ldots, \lambda_{i+4}$.

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

nag_1d_spline_intg (e02bdc) can be used to determine the definite integrals of cubic spline fits and interpolants produced by nag_1d_spline_interpolant (e01bac), nag_1d_spline_fit_knots (e02bac) and nag_1d_spline_fit (e02bec).

4 References

Cox M G (1975) An algorithm for spline interpolation *J. Inst. Math. Appl.* **15** 95–108

5 Arguments

1: **spline** – Nag_Spline *

Pointer to structure of type Nag_Spline with the following members:

n – Integer *Input*

On entry: $\bar{n} + 7$, 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** ≥ 8.

lamda – double * *Input*

On entry: a pointer to which memory of size **spline→n** must be allocated. **spline→lamda**$[j-1]$ must be set to the value of the jth member of the complete set of knots, λ_j for $j = 1, 2, \ldots, \bar{n} + 7$.

Constraint: the λ_j must be in nondecreasing 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*

On entry: 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 $i = 1, 2, \ldots, \bar{n} + 3$.

2: **integral** – double * *Output*

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

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_INT_ARG_LT

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

NE_KNOTS_CONS

On entry, the knots must satisfy the following constraints: **spline→lamda**[**spline→n** − 4] > **spline→lamda**[3], **spline→lamda**$[j] \geq$ **spline→lamda**$[j-1]$, for $j = 1, 2, \ldots,$ **spline→n** − 1, with equality in the cases $j = 1, 2, 3$, **spline→n** − 3, **spline→n** − 2 and **spline→n** − 1.

7 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 $2.2 \times (\bar{n} + 3) \times$ ***machine precision***.

8 Parallelism and Performance

Not applicable.

9 Further Comments

Under normal usage, the call to nag_1d_spline_intg (e02bdc) will follow a call to nag_1d_spline_interpolant (e01bac), nag_1d_spline_fit_knots (e02bac) 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 (e02bdc).

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

10 Example

This example determines 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 5) comprise the following values in the order indicated:

$\bar{n} + 7$

	spline.**lamda**$[j-1]$,
for	$j = 1, 2, \ldots,$ **spline**.**n**
spline.**c**$[j-1]$,	for $j = 1, 2, \ldots,$ **spline**.**n** $- 3$

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 datasets may be supplied. The only changes required to the program relate to the size of **spline**.**lamda** and the storage allocated to **spline**.**c** within the structure **spline**.

10.1 Program Text

```
/* nag_1d_spline_intg (e02bdc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 * Mark 8 revised, 2004.
 */

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


int main(void)
{
  Integer    exit_status = 0, j;
  NagError   fail;
  Nag_Spline spline;
  double     integral;

  INIT_FAIL(fail);

  /* Initialise spline */
  spline.lamda = 0;
  spline.c = 0;

  printf("nag_1d_spline_intg (e02bdc) 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
  while (scanf_s("%"NAG_IFMT"", &(spline.n)) != EOF)
#else
  while (scanf("%"NAG_IFMT"", &(spline.n)) != EOF)
#endif
    {
      if (spline.n > 0)
        {
          if (!(spline.c = NAG_ALLOC(spline.n, double)) ||
              !(spline.lamda = NAG_ALLOC(spline.n, double)))
            {
              printf("Storage allocation failed. Reduce the "
                     "size of spline.n\n");
```

```
              exit_status = 1;
              goto END;
            }
        }
      else
        {
          printf("spline.n is out of range : spline.n = %"NAG_IFMT"\n",
                  spline.n);
          exit_status = 1;
          goto END;
        }
      for (j = 0; j < spline.n; j++)
#ifdef _WIN32
        scanf_s("%lf", &(spline.lamda[j]));
#else
        scanf("%lf", &(spline.lamda[j]));
#endif
      for (j = 0; j < spline.n-3; j++)
#ifdef _WIN32
        scanf_s("%lf", &(spline.c[j]));
#else
        scanf("%lf", &(spline.c[j]));
#endif
      /* nag_1d_spline_intg (e02bdc).
       * Evaluation of fitted cubic spline, definite integral
       */
      nag_1d_spline_intg(&spline, &integral, &fail);
      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_1d_spline_intg (e02bdc).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }
      printf("Definite integral = %12.3e\n", integral);
      NAG_FREE(spline.c);
      NAG_FREE(spline.lamda);
    }
 END:
  return exit_status;
}
```

10.2 Program Data

```
nag_1d_spline_intg (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
```

10.3 Program Results

```
nag_1d_spline_intg (e02bdc) Example Program Results
Definite integral =   1.000e+02
```