NAG C Library Function Document

nag_1d_spline_interpolant (e01bac)

1 Purpose

nag_1d_spline_interpolant (e01bac) determines a cubic spline interpolant to a given set of data.

2 Specification

```c
#include <nag.h>
#include <nag01.h>

void nag_1d_spline_interpolant(Integer m, double x[], double y[],
    Nag_Spline *spline, NagError *fail)
```

3 Description

This function determines a cubic spline \( s(x) \), defined in the range \( x_1 \leq x \leq x_m \), which interpolates (passes exactly through) the set of data points \( (x_i, y_i) \), for \( i = 1, 2, \ldots, m \), where \( m \geq 4 \) and \( x_1 < x_2 < \ldots < x_m \). Unlike some other spline interpolation algorithms, derivative end conditions are not imposed. The spline interpolant chosen has \( m - 4 \) interior knots \( \lambda_5, \lambda_6, \ldots, \lambda_m \), which are set to the values of \( x_3, x_4, \ldots, x_{m-2} \) respectively. This spline is represented in its B-spline form (see Cox (1975a)):

\[
s(x) = \sum_{i=1}^{m} c_i N_i(x)
\]

where \( N_i(x) \) denotes the normalised B-Spline of degree 3, defined upon the knots \( \lambda_1, \lambda_{i+1}, \ldots, \lambda_{i+4} \), and \( c_i \) denotes its coefficient, whose value is to be determined by the routine.

The use of B-splines requires eight additional knots \( \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_{m+1}, \lambda_{m+2}, \lambda_{m+3} \) and \( \lambda_{m+4} \) to be specified; the function sets the first four of these to \( x_1 \) and the last four to \( x_m \).

The algorithm for determining the coefficients is as described in Cox (1975a) except that QR factorization is used instead of LU decomposition. The implementation of the algorithm involves setting up appropriate information for the related function nag_1d_spline_fit_knots (e02bac) followed by a call of that function. (For further details of nag_1d_spline_fit_knots (e02bac), see the function document.)

Values of the spline interpolant, or of its derivatives or definite integral, can subsequently be computed as detailed in Section 6.

4 Parameters

1: \( m \) – Integer \hspace{1cm} Input

\( On \ entry: m \), the number of data points.

\( Constraint: m \geq 4. \)

2: \( x[m] \) – double \hspace{1cm} Input

\( On \ entry: x[i - 1] \) must be set to \( x_i \), the \( i \)th data value of the independent variable \( x \), for \( i = 1, 2, \ldots, m \).

\( Constraint: x[i] < x[i + 1], \) for \( i = 0, 1, \ldots, m - 2. \)

3: \( y[m] \) – double \hspace{1cm} Input

\( On \ entry: y[i - 1] \) must be set to \( y_i \), the \( i \)th data value of the dependent variable \( y \), for \( i = 1, 2, \ldots, m \).
4:  **spline** – Nag_Spline *

   *On exit:* Pointer to structure of type Nag_Spline with the following members:

   - **n** – Integer *Output*
     *On exit:* the size of the storage internally allocated to spline.lambda and spline.c. This is set to \(m + 4\).

   - **lambda** – double *
     *On exit:* pointer to which storage of size spline.n is internally allocated. spline.lambda[i − 1] contains the \(i\)th knot, for \(i = 1, 2, \ldots, m + 4\).

   - **c** – double *
     *On exit:* pointer to which storage of size spline.n−4 is internally allocated. spline.c[i − 1] contains the coefficient \(c_i\) of the B-spline \(N_i(x)\), for \(i = 1, 2, \ldots, m\).

   Note that when the information contained in the pointers spline.lambda and spline.c is no longer of use, or before a new call to nag_1d_spline_interpolat with the same spline, the user should free this storage using the NAG macro **NAG_FREE**. This storage will not have been allocated if this function returns with fail.code \(\neq\) **NE_NOERROR**.

5:  **fail** – NagError *

   *Input/Output*

   The NAG error parameter (see the Essential Introduction).

5  **Error Indicators and Warnings**

   **NE_INT_ARG_LT**

   On entry, \(m\) must not be less than 4: \(m = <value>\).

   **NE_NOT STRICTLY_INCREASING**

   The sequence \(x\) is not strictly increasing: \(x[<value>] = <value>\), \(x[<value>] = <value>\).

   **NE_ALLOC_FAIL**

   Memory allocation failed.

6  **Further Comments**

   The time taken by the function is approximately proportional to \(m\).

   All the \(x_i\) are used as knot positions except \(x_2\) and \(x_{m−1}\). This choice of knots (see Cox (1977)) means that \(s(x)\) is composed of \(m − 3\) cubic arcs as follows. If \(m = 4\), there is just a single arc space spanning the whole interval \(x_1\) to \(x_4\). If \(m \geq 5\), the first and last arcs span the intervals \(x_1\) to \(x_3\) and \(x_{m−2}\) to \(x_m\) respectively. Additionally if \(m \geq 6\), the \(i\)th arc, for \(i = 2, 3, \ldots, m − 4\) spans the interval \(x_{i+1}\) to \(x_{i+2}\).

   After the call

   ```c
   e01bac(m, x, y, &spline, &fail)
   ```

   the following operations may be carried out on the interpolant \(s(x)\).

   The value of \(s(x)\) at \(x = xval\) can be provided in the variable \(xval\) by calling the function

   ```c
   e02bbc(xval, &xval, &spline, &fail)
   ```

   The values of \(s(x)\) and its first three derivatives at \(x = xval\) can be provided in the array \(sdif\) of dimension 4, by the call

   ```c
   e02bcc(derivs, xval, sdif, &spline, &fail)
   ```
Here \texttt{derivs} must specify whether the left- or right-hand value of the third derivative is required (see \texttt{nag_l1d_spline_deriv} for details). The value of the integral of $s(x)$ over the range $x_1$ to $x_m$ can be provided in the variable \texttt{sint} by

\begin{verbatim}
\texttt{e02bdc(}&\texttt{spline, }\texttt{sint, }\texttt{&fail)}
\end{verbatim}

### 6.1 Accuracy
The rounding errors incurred are such that the computed spline is an exact interpolant for a slightly perturbed set of ordinates $y_i + \delta y_i$. The ratio of the root-mean-square value of the $\delta y_i$ to that of the $y_i$ is no greater than a small multiple of the relative \textit{machine precision}.

### 6.2 References


### 7 See Also
\begin{verbatim}
nag_l1d_spline_fit_knots (e02bac)
nag_l1d_spline_evaluate (e02bbc)
nag_l1d_spline_deriv (e02bcc)
nag_l1d_spline_intg (e02bdc)
\end{verbatim}

### 8 Example
The following example program sets up data from 7 values of the exponential function in the interval 0 to 1. \texttt{nag_l1d_spline_interpolant} is then called to compute a spline interpolant to these data.

The spline is evaluated by \texttt{nag_l1d_spline_evaluate} (e02bbc), at the data points and at points halfway between each adjacent pair of data points, and the spline values and the values of $e^x$ are printed out.

### 8.1 Program Text
\begin{verbatim}
/* nag_l1d_spline_interpolant(e01bac) Example Program */
/* Copyright 1991 Numerical Algorithms Group. */
/* Mark 2, 1991. */
/* Mark 6 revised, 2000. */

#include <nag.h>
#include <stdio.h>
#include <math.h>
#include <nag_stdb.h>
#include <nag.e01.h>
#include <nag.e02.h>

#define MMAX 7

main()
{
  static double x[MMAX] = {0.0, 0.2, 0.4, 0.6, 0.75, 0.9, 1.0};
  Integer i, j;
  double y[MMAX], fit, xarg;
  Nag_Spline spline;
  Integer m = MMAX;

  ...
Vprintf("e01bac Example Program Results\n");
for (i=0; i<m; ++i)
    y[i] = exp(x[i]);
e01bac(m, x, y, &spline, NAGERR_DEFAULT);
Vprintf("\nNumber of distinct knots = %ld\n\n", m-2);
Vprintf("Distinct knots located at \n\n");
for (j=3; j<m+1; j++)
    Vprintf("%8.4f\%s", spline.lamda[j], (j-3)*5==4 || j==m ? "\n" : "");
Vprintf("\n\nJ   B-spline coeff c\n\n");
for (j=0; j<m; ++j)
    Vprintf("%ld %13.4f\n",j+1,spline.c[j]);
Vprintf("\nJ   Abscissa   Ordinate   Spline\n\n");
for (j=0; j<m; ++j)
{
    e02bbc(x[j], &fit, &spline, NAGERR_DEFAULT);
    Vprintf("%13.4f %13.4f %13.4f\n",j+1,x[j],y[j],fit);
    if (j < m-1)
    {
        xarg = (x[j] + x[j+1]) * 0.5;
        e02bbc(xarg, &fit, &spline, NAGERR_DEFAULT);
        Vprintf("%13.4f %13.4f\n",xarg,fit);
    }
}
/* Free memory allocated by e01bac */
NAG_FREE(spline.lamda);
NAG_FREE(spline.c);
exit(EXIT_SUCCESS);
}

8.2 Program Data
None.

8.3 Program Results
e01bac Example Program Results

Number of distinct knots = 5

Distinct knots located at

0.0000  0.4000  0.6000  0.7500  1.0000

J   B-spline coeff c

1    1.0000
2    1.1336
3    1.3726
4    1.7827
5    2.1744
6    2.4918
7    2.7183

J   Abscissa   Ordinate   Spline

1    0.0000   1.0000    1.0000
  0.1000   1.1052
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.2000</td>
<td>1.2214</td>
<td>1.2214</td>
</tr>
<tr>
<td></td>
<td>0.3000</td>
<td></td>
<td>1.3498</td>
</tr>
<tr>
<td>3</td>
<td>0.4000</td>
<td>1.4918</td>
<td>1.4918</td>
</tr>
<tr>
<td></td>
<td>0.5000</td>
<td></td>
<td>1.6487</td>
</tr>
<tr>
<td>4</td>
<td>0.6000</td>
<td>1.8221</td>
<td>1.8221</td>
</tr>
<tr>
<td></td>
<td>0.6750</td>
<td></td>
<td>1.9640</td>
</tr>
<tr>
<td>5</td>
<td>0.7500</td>
<td>2.1170</td>
<td>2.1170</td>
</tr>
<tr>
<td></td>
<td>0.8250</td>
<td></td>
<td>2.2819</td>
</tr>
<tr>
<td>6</td>
<td>0.9000</td>
<td>2.4596</td>
<td>2.4596</td>
</tr>
<tr>
<td></td>
<td>0.9500</td>
<td></td>
<td>2.5857</td>
</tr>
<tr>
<td>7</td>
<td>1.0000</td>
<td>2.7183</td>
<td>2.7183</td>
</tr>
</tbody>
</table>