NAG Toolbox

nag_interp_1d_spline (e01ba)

1 Purpose

nag_interp_1d_spline (e01ba) determines a cubic spline interpolant to a given set of data.

2 Syntax

```
[lamda, c, ifail] = nag_interp_1d_spline(x, y, 'm', m)
[lamda, c, ifail] = e01ba(x, y, 'm', m)
```

3 Description

nag_interp_1d_spline (e01ba) 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 $x_1 < x_2 < \cdots < x_m$. 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 (1975)):

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

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

The use of B-splines requires eight additional knots λ_1, λ_2, λ_3, λ_4, λ_{m+1}, λ_{m+2}, λ_{m+3} and λ_{m+4} to be specified; nag_interp_1d_spline (e01ba) 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 (1975) 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_fit_1dspline_knots (e02ba) followed by a call of that function. (See nag_fit_1dspline_knots (e02ba) for further details.)

Values of the spline interpolant, or of its derivatives or definite integral, can subsequently be computed as detailed in Section 9.

4 References

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

Cox M G (1977) A survey of numerical methods for data and function approximation *The State of the Art in Numerical Analysis* (ed D A H Jacobs) 627–668 Academic Press

5 Parameters

5.1 Compulsory Input Parameters

1: **x**(**m**) – REAL (KIND=nag_wp) array

x(i) must be set to x_i, the ith data value of the independent variable x, for $i = 1, 2, \ldots, m$.

Constraint: **x**$(i) <$ **x**$(i + 1)$, for $i = 1, 2, \ldots,$ **m** $- 1$.

2: **y**(**m**) – REAL (KIND=nag_wp) array

y(i) must be set to y_i, the ith data value of the dependent variable y, for $i = 1, 2, \ldots, m$.

5.2 Optional Input Parameters

1: **m** – INTEGER

Default: the dimension of the arrays **x**, **y**. (An error is raised if these dimensions are not equal.)

m, the number of data points.

Constraint: $\mathbf{m} \geq 4$.

5.3 Output Parameters

1: **lamda**(lck) – REAL (KIND=nag_wp) array

$lck = \mathbf{m} + 4$.

The value of λ_i, the ith knot, for $i = 1, 2, \ldots, m + 4$.

2: **c**(lck) – REAL (KIND=nag_wp) array

$lck = \mathbf{m} + 4$.

The coefficient c_i of the B-spline $N_i(x)$, for $i = 1, 2, \ldots, m$. The remaining elements of the array are not used.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, $\mathbf{m} < 4$,
or $lck < \mathbf{m} + 4$,
or $lwrk < 6 \times \mathbf{m} + 16$.

ifail $= 2$

The **x**-values fail to satisfy the condition

$\mathbf{x}(1) < \mathbf{x}(2) < \mathbf{x}(3) < \cdots < \mathbf{x}(\mathbf{m})$.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 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 δy_i to that of the y_i is no greater than a small multiple of the relative ***machine precision***.

8 Further Comments

The time taken by nag_interp_1d_spline (e01ba) 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 ith arc, for $i = 2, 3, \ldots, m-4$, spans the interval x_{i+1} to x_{i+2}.

After the call

```
[lamda, c, ifail] = e01ba(x, y, lck);
```

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

The value of $s(x)$ at $x = \mathbf{x}$ can be provided in the double variable **s** by the call

```
[s, ifail] = e02bb(lamda, c, x);
```

(see nag_fit_1dspline_eval (e02bb)).

The values of $s(x)$ and its first three derivatives at $x = \mathbf{x}$ can be provided in the double array **s** of dimension 4, by the call

```
[s, ifail] = e02bc(lamda, c, x, left);
```

(see nag_fit_1dspline_deriv (e02bc)).

Here **left** must specify whether the left- or right-hand value of the third derivative is required (see nag_fit_1dspline_deriv (e02bc) for details).

The value of the integral of $s(x)$ over the range x_1 to x_m can be provided in the double variable **dint** by

```
[dint, ifail] = e02bd(lamda, c);
```

(see nag_fit_1dspline_integ (e02bd)).

9 Example

This example sets up data from 7 values of the exponential function in the interval 0 to 1. nag_interp_1d_spline (e01ba) is then called to compute a spline interpolant to these data.

The spline is evaluated by nag_fit_1dspline_eval (e02bb), 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.

9.1 Program Text

```
     function e01ba_example

fprintf('e01ba example results\n\n');

x = [0     0.2     0.4     0.6     0.75     0.9     1];
y = exp(x);
[lamda, c, ifail] = e01ba(x, y);

fprintf('\n    j    knot lamda(j+2)   b-spline coeff c(j)\n\n');
j = 1;
fprintf('%4d%35.4f\n', j, c(1));
m = size(x,2);
for j = 2:m - 1;
  fprintf('%4d%15.4f%20.4f\n', j, lamda(j+2), c(j));
end
fprintf('%4d%35.4f\n', m, c(m));
fprintf('\n    R        Abscissa            Ordinate             Spline\n\n');
for r = 1:m;
  [fit, ifail] = e02bb( ...
                      lamda, c, x(r));

  fprintf('%4d%15.4f%20.4f%20.4f\n', r, x(r), y(r), fit);
  if r<m;
    xarg = (x(r)+x(r+1))/2;
```

```
    [fit, ifail] = e02bb( ...
                          lamda, c, xarg);
    fprintf('%19.4f%40.4f\n', xarg, fit);
  end
end
```

9.2 Program Results

```
     e01ba example results

   j    knot lamda(j+2)   b-spline coeff c(j)

   1                             1.0000
   2         0.0000              1.1336
   3         0.4000              1.3726
   4         0.6000              1.7827
   5         0.7500              2.1744
   6         1.0000              2.4918
   7                             2.7183

   R        Abscissa            Ordinate             Spline

   1         0.0000              1.0000              1.0000
             0.1000                                  1.1052
   2         0.2000              1.2214              1.2214
             0.3000                                  1.3498
   3         0.4000              1.4918              1.4918
             0.5000                                  1.6487
   4         0.6000              1.8221              1.8221
             0.6750                                  1.9640
   5         0.7500              2.1170              2.1170
             0.8250                                  2.2819
   6         0.9000              2.4596              2.4596
             0.9500                                  2.5857
   7         1.0000              2.7183              2.7183
```