NAG Toolbox for MATLAB

e02bb

1 Purpose

e02bb evaluates a cubic spline from its B-spline representation.

2 Syntax

```
[s, ifail] = e02bb(lamda, c, x, 'ncap7', ncap7)
```

3 Description

e02bb evaluates the cubic spline $s(x)$ at a prescribed argument x from its augmented knot set λ_i, for $i = 1, 2, \ldots, n + 7$, (see e02ba) and from the coefficients c_i, for $i = 1, 2, \ldots, q$ in its B-spline representation

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

Here $q = \bar{n} + 3$, where $\bar{n}$ is the number of intervals of the spline, and $N_i(x)$ denotes the normalized B-spline of degree 3 defined upon the knots $\lambda_i, \lambda_{i+1}, \ldots, \lambda_{i+4}$. The prescribed argument x must satisfy $\lambda_4 \leq x \leq \lambda_{\bar{n}+4}$.

It is assumed that $\lambda_j \geq \lambda_{j-1}$, for $j = 2, 3, \ldots, \bar{n} + 7$, and $\lambda_{\bar{n}+4} > \lambda_4$.

If x is a point at which 4 knots coincide, $s(x)$ is discontinuous at x; in this case, **s** contains the value defined as x is approached from the right.

The method employed is that of evaluation by taking convex combinations due to de Boor 1972. For further details of the algorithm and its use see Cox 1972a and Cox and Hayes 1973.

It is expected that a common use of e02bb will be the evaluation of the cubic spline approximations produced by e02ba. A generalization of e02bb which also forms the derivative of $s(x)$ is e02bc. e02bc takes about 50% longer than e02bb.

4 References

Cox M G 1972a The numerical evaluation of B-splines *J. Inst. Math. Appl.* **10** 134–149

Cox M G 1978 The numerical evaluation of a spline from its B-spline representation *J. Inst. Math. Appl.* **21** 135–143

Cox M G and Hayes J G 1973 Curve fitting: a guide and suite of algorithms for the non-specialist user *NPL Report NAC26* National Physical Laboratory

de Boor C 1972 On calculating with B-splines *J. Approx. Theory* **6** 50–62

5 Parameters

5.1 Compulsory Input Parameters

1: **lamda**(**ncap7**) – **double array**

lamda(j) 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 **lamda**(j) must be in nondecreasing order with **lamda**(**ncap7** $- 3$) $>$ **lamda**(4).

2: **c(ncap7) – double array**

The coefficient c_i of the B-spline $N_i(x)$, for $i = 1, 2, \ldots, \bar{n} + 3$. The remaining elements of the array are not used.

3: **x – double scalar**

The argument x at which the cubic spline is to be evaluated.

Constraint: $\mathbf{lamda}(4) \leq \mathbf{x} \leq \mathbf{lamda}(\mathbf{ncap7} - 3)$.

5.2 Optional Input Parameters

1: **ncap7 – int32 scalar**

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

$\bar{n} + 7$, where $\bar{n}$ is the number of intervals (one greater than the number of interior knots, i.e., the knots strictly within the range λ_4 to $\lambda_{\bar{n}+4}$) over which the spline is defined.

Constraint: $\mathbf{ncap7} \geq 8$.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **s – double scalar**

The value of the spline, $s(x)$.

2: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

The parameter **x** does not satisfy $\mathbf{lamda}(4) \leq \mathbf{x} \leq \mathbf{lamda}(\mathbf{ncap7} - 3)$.

In this case the value of **s** is set arbitrarily to zero.

ifail $= 2$

$\mathbf{ncap7} < 8$, i.e., the number of interior knots is negative.

7 Accuracy

The computed value of $s(x)$ has negligible error in most practical situations. Specifically, this value has an **absolute** error bounded in modulus by $18 \times c_{\max} \times$ ***machine precision***, where $c_{\max}$ is the largest in modulus of c_j, c_{j+1}, c_{j+2} and c_{j+3}, and j is an integer such that $\lambda_{j+3} \leq x \leq \lambda_{j+4}$. If c_j, c_{j+1}, c_{j+2} and c_{j+3} are all of the same sign, then the computed value of $s(x)$ has a **relative** error not exceeding $20 \times$ ***machine precision*** in modulus. For further details see Cox 1978.

8 Further Comments

The time taken is approximately $\mathbf{c} \times (1 + 0.1 \times \log(\bar{n} + 7))$ seconds, where **c** is a machine-dependent constant.

Note: the function does not test all the conditions on the knots given in the description of **lamda** in Section 5, since to do this would result in a computation time approximately linear in $\bar{n} + 7$ instead of $\log(\bar{n} + 7)$. All the conditions are tested in e02ba, however.

9 Example

```
lamda = [1;
     1;
     1;
     1;
     3;
     6;
     8;
     9;
     9;
     9;
     9];
c = [1;
     2;
     4;
     7;
     6;
     4;
     3;
     0;
     0;
     0;
     0];
x = 1;
[s, ifail] = e02bb(lamda, c, x)
```

```
s =
     1
ifail =
           0
```