NAG Toolbox for MATLAB

e02ba

1 Purpose

e02ba computes a weighted least-squares approximation to an arbitrary set of data points by a cubic spline with knots prescribed by you. Cubic spline interpolation can also be carried out.

2 Syntax

```
[lamda, c, ss, ifail] = e02ba(x, y, w, lamda, 'm', m, 'ncap7', ncap7)
```

3 Description

e02ba determines a least-squares cubic spline approximation $s(x)$ to the set of data points (x_r, y_r) with weights w_r, for $r = 1, 2, \ldots, m$. The value of **ncap7** $= \bar{n} + 7$, where $\bar{n}$ is the number of intervals of the spline (one greater than the number of interior knots), and the values of the knots $\lambda_5, \lambda_6, \ldots, \lambda_{\bar{n}+3}$, interior to the data interval, are prescribed by you.

$s(x)$ has the property that it minimizes θ, the sum of squares of the weighted residuals ϵ_r, for $r = 1, 2, \ldots, m$, where

$$\epsilon_r = w_r(y_r - s(x_r)).$$

The function produces this minimizing value of θ and the coefficients $c_1, c_2, \ldots, c_q$, where $q = \bar{n} + 3$, in the B-spline representation

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

Here $N_i(x)$ denotes the normalized B-spline of degree 3 defined upon the knots $\lambda_i, \lambda_{i+1}, \ldots, \lambda_{i+4}$.

In order to define the full set of B-splines required, eight additional knots $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ and $\lambda_{\bar{n}+4}, \lambda_{\bar{n}+5}, \lambda_{\bar{n}+6}, \lambda_{\bar{n}+7}$ are inserted automatically by the function. The first four of these are set equal to the smallest x_r and the last four to the largest x_r.

The representation of $s(x)$ in terms of B-splines is the most compact form possible in that only $\bar{n} + 3$ coefficients, in addition to the $\bar{n} + 7$ knots, fully define $s(x)$.

The method employed involves forming and then computing the least-squares solution of a set of m linear equations in the coefficients $c_i (i = 1, 2, \ldots, \bar{n} + 3)$. The equations are formed using a recurrence relation for B-splines that is unconditionally stable (see Cox 1972a and de Boor 1972), even for multiple (coincident) knots. The least-squares solution is also obtained in a stable manner by using orthogonal transformations, viz. a variant of Givens rotations (see Gentleman 1974 and Gentleman 1973). This requires only one equation to be stored at a time. Full advantage is taken of the structure of the equations, there being at most four nonzero values of $N_i(x)$ for any value of x and hence at most four coefficients in each equation.

For further details of the algorithm and its use see Cox 1974, Cox 1975b and Cox and Hayes 1973.

Subsequent evaluation of $s(x)$ from its B-spline representation may be carried out using e02bb. If derivatives of $s(x)$ are also required, e02bc may be used. e02bd can be used to compute the definite integral of $s(x)$.

4 References

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

Cox M G 1974 A data-fitting package for the non-specialist user *Software for Numerical Mathematics* (ed D J Evans) Academic Press

Cox M G 1975b Numerical methods for the interpolation and approximation of data by spline functions *PhD Thesis* City University, London

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

Gentleman W M 1973 Least-squares computations by Givens transformations without square roots *J. Inst. Math. Applic.* **12** 329–336

Gentleman W M 1974 Algorithm AS 75. Basic procedures for large sparse or weighted linear least-squares problems *Appl. Statist.* **23** 448–454

Schoenberg I J and Whitney A 1953 On Polya frequency functions III *Trans. Amer. Math. Soc.* **74** 246–259

5 Parameters

5.1 Compulsory Input Parameters

1: **x(m) – double array**

The values x_r of the independent variable (abscissa), for $r = 1, 2, \ldots, m$.

Constraint: $x_1 \leq x_2 \leq \cdots \leq x_m$.

2: **y(m) – double array**

The values y_r of the dependent variable (ordinate), for $r = 1, 2, \ldots, m$.

3: **w(m) – double array**

The values w_r of the weights, for $r = 1, 2, \ldots, m$. For advice on the choice of weights, see the E02 Chapter Introduction.

Constraint: $\mathbf{w}(r) > 0$, for $r = 1, 2, \ldots, m$.

4: **lamda(ncap7) – double array**

lamda(i) must be set to the $(i-4)$th (interior) knot, λ_i, for $i = 5, 6, \ldots, \bar{n}+3$.

Constraint: $\mathbf{x}(1) < \mathbf{lamda}(5) \leq \mathbf{lamda}(6) \leq \cdots \leq \mathbf{lamda}(\mathbf{ncap7}-4) < \mathbf{x}(\mathbf{m})$.

5.2 Optional Input Parameters

1: **m – int32 scalar**

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

the number m of data points.

Constraint: $\mathbf{m} \geq mdist \geq 4$, where $mdist$ is the number of distinct x values in the data.

2: **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 of the spline (which is one greater than the number of interior knots, i.e., the knots strictly within the range x_1 to x_m) over which the spline is defined.

Constraint: $8 \leq \mathbf{ncap7} \leq mdist+4$, where $mdist$ is the number of distinct x values in the data.

5.3 Input Parameters Omitted from the MATLAB Interface

work1, work2

5.4 Output Parameters

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

The input values are unchanged, and **lamda**(i), for $i = 1, 2, 3, 4$, **ncap7** $- 3$, **ncap7** $- 2$, **ncap7** $- 1$, **ncap7** contains the additional (exterior) knots introduced by the function. For advice on the choice of knots, see Section 3.3 in the E02 Chapter Introduction.

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: **ss – double scalar**

The residual sum of squares, θ.

4: **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 knots fail to satisfy the condition

x(1) < **lamda**(5) ≤ **lamda**(6) ≤ ⋯ ≤ **lamda**(**ncap7** − 4) < **x**(**m**).

Thus the knots are not in correct order or are not interior to the data interval.

ifail $= 2$

The weights are not all strictly positive.

ifail $= 3$

The values of **x**(r), for $r = 1, 2, \ldots,$ **m** are not in nondecreasing order.

ifail $= 4$

ncap7 < 8 (so the number of interior knots is negative) or **ncap7** $> mdist + 4$, where $mdist$ is the number of distinct x values in the data (so there cannot be a unique solution).

ifail $= 5$

The conditions specified by Schoenberg and Whitney 1953 fail to hold for at least one subset of the distinct data abscissae. That is, there is no subset of **ncap7** $- 4$ strictly increasing values, **x**($R(1)$), **x**($R(2)$), …, **x**(R(**ncap7** − 4)), among the abscissae such that

x($R(1)$) < **lamda**(1) < **x**($R(5)$),

x($R(2)$) < **lamda**(2) < **x**($R(6)$),

⋮

x(R(**ncap7** − 8)) < **lamda**(**ncap7** − 8) < **x**(R(**ncap7** − 4)).

This means that there is no unique solution: there are regions containing too many knots compared with the number of data points.

7 Accuracy

The rounding errors committed are such that the computed coefficients are exact for a slightly perturbed set of ordinates $y_r + \delta y_r$. The ratio of the root-mean-square value for the δy_r to the root-mean-square value of the y_r can be expected to be less than a small multiple of $\kappa \times m \times$ ***machine precision***, where κ is a condition number for the problem. Values of κ for $20 - -30$ practical data sets all proved to lie between 4.5 and 7.8 (see Cox 1975b). (Note that for these data sets, replacing the coincident end knots at the end points x_1 and x_m used in the function by various choices of non-coincident exterior knots gave values of κ between 16 and 180. Again see Cox 1975b for further details.) In general we would not expect κ to be large unless the choice of knots results in near-violation of the Schoenberg–Whitney conditions.

A cubic spline which adequately fits the data and is free from spurious oscillations is more likely to be obtained if the knots are chosen to be grouped more closely in regions where the function (underlying the data) or its derivatives change more rapidly than elsewhere.

8 Further Comments

The time taken is approximately $\mathbf{c} \times (2m + \bar{n} + 7)$ seconds, where $\mathbf{c}$ is a machine-dependent constant.

Multiple knots are permitted as long as their multiplicity does not exceed 4, i.e., the complete set of knots must satisfy $\lambda_i < \lambda_{i+4}$, for $i = 1, 2, \ldots, \bar{n} + 3$, (see Section 6). At a knot of multiplicity one (the usual case), $s(x)$ and its first two derivatives are continuous. At a knot of multiplicity two, $s(x)$ and its first derivative are continuous. At a knot of multiplicity three, $s(x)$ is continuous, and at a knot of multiplicity four, $s(x)$ is generally discontinuous.

The function can be used efficiently for cubic spline interpolation, i.e., if $m = \bar{n} + 3$. The abscissae must then of course satisfy $x_1 < x_2 < \cdots < x_m$. Recommended values for the knots in this case are $\lambda_i = x_{i-2}$, for $i = 5, 6, \ldots, \bar{n} + 3$.

9 Example

```
x = [0.2;
     0.47;
     0.74;
     1.09;
     1.6;
     1.9;
     2.6;
     3.1;
     4;
     5.15;
     6.17;
     8;
     10;
     12];
y = [0;
     2;
     4;
     6;
     8;
     8.619999999999999;
     9.1;
     8.9;
     8.15;
     7;
     6;
     4.54;
     3.39;
     2.56];
w = [0.2;
     0.2;
     0.3;
     0.7;
```

```
     0.9;
     1;
     1;
     1;
     0.8;
     0.5;
     0.7;
     1;
     1;
     1];
lamda = [0;
     0;
     0;
     0;
     1.5;
     2.6;
     4;
     8;
     0;
     0;
     0;
     0];
[lamdaOut, c, ss, ifail] = e02ba(x, y, w, lamda)
```

```
lamdaOut =
    0.2000
    0.2000
    0.2000
    0.2000
    1.5000
    2.6000
    4.0000
    8.0000
   12.0000
   12.0000
   12.0000
   12.0000
c =
   -0.0465
    3.6150
    8.5724
    9.4261
    7.2716
    4.1207
    3.0822
    2.5597
         0
         0
         0
         0
ss =
    0.0018
ifail =
           0
```