NAG Toolbox for MATLAB

e02be

1 Purpose

e02be computes a cubic spline approximation to an arbitrary set of data points. The knots of the spline are located automatically, but a single parameter must be specified to control the trade-off between closeness of fit and smoothness of fit.

2 Syntax

```
[n, lamda, c, fp, wrk, iwrk, ifail] = e02be(start, x, y, w, s, n, lamda,
wrk, iwrk, 'm', m, 'nest', nest, 'lwrk', lwrk)
```

3 Description

e02be determines a smooth 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 spline is given in the B-spline representation

$$s(x) = \sum_{i=1}^{n-4} c_i N_i(x), \tag{1}$$

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

The total number n of these knots and their values $\lambda_1, \ldots, \lambda_n$ are chosen automatically by the function. The knots $\lambda_5, \ldots, \lambda_{n-4}$ are the interior knots; they divide the approximation interval $[x_1, x_m]$ into $n - 7$ sub-intervals. The coefficients $c_1, c_2, \ldots, c_{n-4}$ are then determined as the solution of the following constrained minimization problem:

minimize

$$\eta = \sum_{i=5}^{n-4} \delta_i^2 \tag{2}$$

subject to the constraint

$$\theta = \sum_{r=1}^{m} \epsilon_r^2 \leq S, \tag{3}$$

where δ_i stands for the discontinuity jump in the third order derivative of $s(x)$ at the interior knot λ_i,
ϵ_r denotes the weighted residual $w_r(y_r - s(x_r))$,
and S is a nonnegative number to be specified by you.

The quantity η can be seen as a measure of the (lack of) smoothness of $s(x)$, while closeness of fit is measured through θ. By means of the parameter S, 'the smoothing factor', you will then control the balance between these two (usually conflicting) properties. If S is too large, the spline will be too smooth and signal will be lost (underfit); if S is too small, the spline will pick up too much noise (overfit). In the extreme cases the function will return an interpolating spline ($\theta = 0$) if S is set to zero, and the weighted least-squares cubic polynomial ($\eta = 0$) if S is set very large. Experimenting with S values between these two extremes should result in a good compromise. (See Section 8.2 for advice on choice of S.)

The method employed is outlined in Section 8.3 and fully described in Dierckx 1975, Dierckx 1981a and Dierckx 1982. It involves an adaptive strategy for locating the knots of the cubic spline (depending on the function underlying the data and on the value of S), and an iterative method for solving the constrained minimization problem once the knots have been determined.

Values of the computed spline, or of its derivatives or definite integral, can subsequently be computed by calling e02bb, e02bc or e02bd, as described in Section 8.4.

4 References

Dierckx P 1975 An algorithm for smoothing, differentiating and integration of experimental data using spline functions *J. Comput. Appl. Math.* **1** 165–184

Dierckx P 1981a An improved algorithm for curve fitting with spline functions *Report TW54* Department of Computer Science, Katholieke Univerciteit Leuven

Dierckx P 1982 A fast algorithm for smoothing data on a rectangular grid while using spline functions *SIAM J. Numer. Anal.* **19** 1286–1304

Reinsch C H 1967 Smoothing by spline functions *Numer. Math.* **10** 177–183

5 Parameters

5.1 Compulsory Input Parameters

1: **start – string**

Must be set to 'C' or 'W'.

start = 'C' (Cold start)

The function will build up the knot set starting with no interior knots. No values need be assigned to the parameters **n**, **lamda**, **wrk** or **iwrk**.

start = 'W' (Warm start)

The function will restart the knot-placing strategy using the knots found in a previous call of the function. In this case, the parameters **n**, **lamda**, **wrk**, and **iwrk** must be unchanged from that previous call. This warm start can save much time in searching for a satisfactory value of **s**.

Constraint: **start** = 'C' or 'W'.

2: **x(m) – double array**

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

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

3: **y(m) – double array**

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

4: **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 Section 2.1.2 in the E02 Chapter Introduction.

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

5: **s – double scalar**

The smoothing factor, S.

If $\mathbf{s} = 0.0$, the function returns an interpolating spline.

If **s** is smaller than ***machine precision***, it is assumed equal to zero.

For advice on the choice of **s**, see Sections 3 and 8.2.

Constraint: $\mathbf{s} \geq 0.0$.

6: **n – int32 scalar**

If the warm start option is used, the value of **n** must be left unchanged from the previous call.

7: **lamda(nest) – double array**

If the warm start option is used, the values **lamda**(1), **lamda**$(2), \ldots,$ **lamda**(**n**) must be left unchanged from the previous call.

8: **wrk(lwrk) – double array**

If the warm start option is used on entry, the values **wrk**$(1), \ldots,$ **wrk**(n) must be left unchanged from the previous call.

9: **iwrk(nest) – int32 array**

If the warm start option is used, on entry, the values **iwrk**$(1), \ldots,$ **iwrk**(n) must be left unchanged from the previous call.

This array is used as workspace.

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.)

m, the number of data points.

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

2: **nest – int32 scalar**

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

an over-estimate for the number, n, of knots required.

Constraint: $\mathbf{nest} \geq 8$. In most practical situations, $\mathbf{nest} = \mathbf{m}/2$ is sufficient. **nest** never needs to be larger than $\mathbf{m} + 4$, the number of knots needed for interpolation ($\mathbf{s} = 0.0$).

3: **lwrk – int32 scalar**

Default: The dimension of the array **wrk**.

Constraint: $\mathbf{lwrk} \geq 4 \times \mathbf{m} + 16 \times \mathbf{nest} + 41$.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **n – int32 scalar**

The total number, n, of knots of the computed spline.

2: **lamda(nest) – double array**

The knots of the spline, i.e., the positions of the interior knots **lamda**(5), **lamda**$(6), \ldots,$ **lamda**$(\mathbf{n} - 4)$ as well as the positions of the additional knots

$$\mathbf{lamda}(1) = \mathbf{lamda}(2) = \mathbf{lamda}(3) = \mathbf{lamda}(4) = x_1$$

and

$$\mathbf{lamda}(\mathbf{n} - 3) = \mathbf{lamda}(\mathbf{n} - 2) = \mathbf{lamda}(\mathbf{n} - 1) = \mathbf{lamda}(\mathbf{n}) = x_m$$

needed for the B-spline representation.

3: **c**(**nest**) – **double array**

The coefficient c_i of the B-spline $N_i(x)$ in the spline approximation $s(x)$, for $i = 1, 2, \ldots, n - 4$.

4: **fp** – **double scalar**

The sum of the squared weighted residuals, θ, of the computed spline approximation. If **fp** $= 0.0$, this is an interpolating spline. **fp** should equal **s** within a relative tolerance of 0.001 unless $n = 8$ when the spline has no interior knots and so is simply a cubic polynomial. For knots to be inserted, **s** must be set to a value below the value of **fp** produced in this case.

5: **wrk**(**lwrk**) – **double array**

If the warm start option is used on entry, the values **wrk**$(1), \ldots,$ **wrk**(n) must be left unchanged from the previous call.

6: **iwrk**(**nest**) – **int32 array**

If the warm start option is used, on entry, the values **iwrk**$(1), \ldots,$ **iwrk**(n) must be left unchanged from the previous call.

This array is used as workspace.

7: **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 $= 1$

On entry, **start** $\neq$ 'C' or 'W',
or **m** < 4,
or **s** < 0.0,
or **s** $= 0.0$ and **nest** $<$ **m** $+ 4$,
or **nest** < 8,
or **lwrk** $< 4 \times$ **m** $+ 16 \times$ **nest** $+ 41$.

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 strictly increasing order.

ifail $= 4$

The number of knots required is greater than **nest**. Try increasing **nest** and, if necessary, supplying larger arrays for the parameters **lamda**, **c**, **wrk** and **iwrk**. However, if **nest** is already large, say **nest** $>$ **m**$/2$, then this error exit may indicate that **s** is too small.

ifail $= 5$

The iterative process used to compute the coefficients of the approximating spline has failed to converge. This error exit may occur if **s** has been set very small. If the error persists with increased **s**, consult NAG.

If **ifail** $= 4$ or 5, a spline approximation is returned, but it fails to satisfy the fitting criterion (see (2) and (3)) – perhaps by only a small amount, however.

7 Accuracy

On successful exit, the approximation returned is such that its weighted sum of squared residuals θ (as in (3)) is equal to the smoothing factor S, up to a specified relative tolerance of 0.001 – except that if $n = 8$, θ may be significantly less than S: in this case the computed spline is simply a weighted least-squares polynomial approximation of degree 3, i.e., a spline with no interior knots.

8 Further Comments

8.1 Timing

The time taken for a call of e02be depends on the complexity of the shape of the data, the value of the smoothing factor S, and the number of data points. If e02be is to be called for different values of S, much time can be saved by setting **start** = 'W' after the first call.

8.2 Choice of s

If the weights have been correctly chosen (see Section 2.1.2 in the E02 Chapter Introduction), the standard deviation of $w_r y_r$ would be the same for all r, equal to σ, say. In this case, choosing the smoothing factor S in the range $\sigma^2\left(m \pm \sqrt{2m}\right)$, as suggested by Reinsch 1967, is likely to give a good start in the search for a satisfactory value. Otherwise, experimenting with different values of S will be required from the start, taking account of the remarks in Section 3.

In that case, in view of computation time and memory requirements, it is recommended to start with a very large value for S and so determine the least-squares cubic polynomial; the value returned in **fp**, call it θ_0, gives an upper bound for S. Then progressively decrease the value of S to obtain closer fits – say by a factor of 10 in the beginning, i.e., $S = \theta_0/10$, $S = \theta_0/100$, and so on, and more carefully as the approximation shows more details.

The number of knots of the spline returned, and their location, generally depend on the value of S and on the behaviour of the function underlying the data. However, if e02be is called with **start** = 'W', the knots returned may also depend on the smoothing factors of the previous calls. Therefore if, after a number of trials with different values of S and **start** = 'W', a fit can finally be accepted as satisfactory, it may be worthwhile to call e02be once more with the selected value for S but now using **start** = 'C'. Often, e02be then returns an approximation with the same quality of fit but with fewer knots, which is therefore better if data reduction is also important.

8.3 Outline of Method Used

If $S = 0$, the requisite number of knots is known in advance, i.e., $n = m + 4$; the interior knots are located immediately as $\lambda_i = x_{i-2}$, for $i = 5, 6, \ldots, n-4$. The corresponding least-squares spline (see e02ba) is then an interpolating spline and therefore a solution of the problem.

If $S > 0$, a suitable knot set is built up in stages (starting with no interior knots in the case of a cold start but with the knot set found in a previous call if a warm start is chosen). At each stage, a spline is fitted to the data by least-squares (see e02ba) and θ, the weighted sum of squares of residuals, is computed. If $\theta > S$, new knots are added to the knot set to reduce θ at the next stage. The new knots are located in intervals where the fit is particularly poor, their number depending on the value of S and on the progress made so far in reducing θ. Sooner or later, we find that $\theta \leq S$ and at that point the knot set is accepted. The function then goes on to compute the (unique) spline which has this knot set and which satisfies the full fitting criterion specified by (2) and (3). The theoretical solution has $\theta = S$. The function computes the spline by an iterative scheme which is ended when $\theta = S$ within a relative tolerance of 0.001. The main part of each iteration consists of a linear least-squares computation of special form, done in a similarly stable and efficient manner as in e02ba.

An exception occurs when the function finds at the start that, even with no interior knots ($n = 8$), the least-squares spline already has its weighted sum of squares of residuals $\leq S$. In this case, since this spline (which is simply a cubic polynomial) also has an optimal value for the smoothness measure η, namely zero, it is returned at once as the (trivial) solution. It will usually mean that S has been chosen too large.

For further details of the algorithm and its use, see Dierckx 1981a.

8.4 Evaluation of Computed Spline

The value of the computed spline at a given value **x** may be obtained in the double variable **s** by the call:

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

where **n**, **lamda** and **c** are the output parameters of e02be.

The values of the spline and its first three derivatives at a given value **x** may be obtained in the double array **s** of dimension at least 4 by the call:

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

where if **left** $= 1$, left-hand derivatives are computed and if **left** $\neq 1$, right-hand derivatives are calculated. The value of **left** is only relevant if **x** is an interior knot (see e02bc).

The value of the definite integral of the spline over the interval $\mathbf{x}(1)$ to $\mathbf{x}(\mathbf{m})$ can be obtained in the double variable **dint** by the call:

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

(see e02bd).

9 Example

```
start = 'C';
x = [0;
     0.5;
     1;
     1.5;
     2;
     2.5;
     3;
     4;
     4.5;
     5;
     5.5;
     6;
     7;
     7.5;
     8];
y = [-1.1;
     -0.372;
     0.4309999999999999;
     1.69;
     2.11;
     3.1;
     4.23;
     4.35;
     4.81;
     4.61;
     4.79;
     5.23;
     6.35;
     7.19;
     7.97];
w = [1;
     2;
     1.5;
     1;
     3;
     1;
     0.5;
     1;
     2;
     2.5;
     1;
     3;
     1;
     2;
     1];
s = 1;
n = int32(0);
lamda = zeros(54,1);
wrk = zeros(1105, 1);
iwrk = zeros(54, 1, 'int32');
[nOut, lamdaOut, c, fp, wrkOut, iwrkOut, ifail] = ...
    e02be(start, x, y, w, s, n, lamda, wrk, iwrk)
```

```
nOut =
           9
lamdaOut =
     array elided
c =
     array elided
fp =
    1.0003
wrkOut =
     array elided
iwrkOut =
     array elided
ifail =
           0
```