

SPLPAK

User's Guide to splfit and splval

1. Introduction

The MATLAB function `splfit` can be used to compute a cubic spline s that fits (or interpolates) a given set of data points, with or without end-point constraints. The function also returns estimates of the standard deviation of the data and the fit. `splval` can be used to evaluate either $s(t)$ at a vector of arguments, or s , s' , s'' and estimates of their standard deviation for a scalar argument, or the extrema of the spline. The functions can be downloaded from

<http://www.imm.dtu.dk/~hbn/Software>

Let $\{(t_i, y_i), w_i\}_{i=1}^m$ denote the given data points and weights. Further, we are given the spline knots

$$x_0 < x_1 \leq x_2 \leq \dots \leq x_{n-1} < x_n, \quad (1)$$

and it is assumed that they bracket the data points,

$$x_0 \leq t_i \leq x_n, \quad i = 1, \dots, m. \quad (2)$$

`splfit` finds the cubic spline over the given knots that minimizes

$$\varphi \equiv \sum_{i=1}^m w_i^2 (y_i - s(t_i))^2, \quad (3a)$$

possibly under the end-point constraints

$$\begin{aligned} \beta_{1,1}s(x_0) + \beta_{1,2}s'(x_0) + \beta_{1,3}s''(x_0) &= \beta_{1,4} \\ \beta_{2,1}s(x_n) + \beta_{2,2}s'(x_n) + \beta_{2,3}s''(x_n) &= \beta_{2,4}. \end{aligned} \quad (3b)$$

See Section 4 for examples.

The spline is determined in the form

$$s(t) = \sum_{j=1}^{n+3} c_j N_j(t), \quad (4)$$

where the N_j are normalized cubic B-spline, see [1, Section 2.2]. `splfit` also returns the pp (piecewise polynomial) representation of s , [1, (2.1)]: In each non vanishing knot interval the spline is a third order polynomial

$$\begin{aligned} \text{for } d_{1,k} \leq t \leq d_{1,k+1} : \\ s(t) = d_{2,k} + d_{3,k}(t - d_{1,k}) + d_{4,k}(t - d_{1,k})^2 \\ + d_{5,k}(t - d_{1,k})^3, \quad k = 1, \dots, p-1. \end{aligned} \quad (5a)$$

This is one of the representations used in `splval`, and arguments outside the knot range are permitted: values are obtained by \mathcal{C}^2 -continuation with second order polynomials,

$$\begin{aligned} \text{for } t < x_0 (= d_{1,1}) : \\ s(t) = d_{2,1} + d_{3,1}(t - d_{1,1}) + d_{4,1}(t - d_{1,1})^2, \end{aligned} \quad (5b)$$

$$\begin{aligned} \text{for } t > x_n (= d_{1,p}) : \\ s(t) = d_{2,p} + d_{3,p}(t - d_{1,p}) + d_{4,p}(t - d_{1,p})^2. \end{aligned} \quad (5c)$$

Finally, `splfit` also returns estimates of the standard deviation of the data and the spline coefficients $\{c_j\}$, see Section 5.2.

2. User's Guide to splfit

A typical call is

```
[fail, S] = splfit(tyw, x {, BC})
```

Input Parameters

tyw : Data points and weights. Array with abscissae $\{t_i\}$ in 1st column, ordinates $\{t_i\}$ in 2nd column, and – optionally – weights $\{w_i\}$ in 3rd column. If **tyw** has two columns only, then all weights are set to 1.

x : Knots with $x(i) = x_{i-1}$, $i = 1, \dots, n+1$. Must satisfy (1).

BC : (Optional). If present, then it should be a 2×4 array with the constraints (3b): $BC(k, j) = \beta_{k,j}$.

Output Parameters

fail : Performance indicator

- fail** = 0 : No problems (otherwise, the contents of **S** of no significance).
- 1 : **length(x)** < 2, i.e. the knot range is empty.
- 2 : Knots are not in increasing order (condition (1) is not satisfied).
- 3 : Some data abscissa is outside the knot range $[x_0, x_n]$.
- 4 : Too few active data points (points with strictly positive weights).
- 5 : The Schoenberg-Whitney condition [1, (5.9)] is not satisfied.

S : Struct representing the spline. The fields are

- x** : Knots.
- c** : $(n+3)$ -vector of coefficients in B-spline representation (4) of s .
- pp** : Piecewise polynomial representation of s . $5 \times p$ array with $pp(k, j) = d_{k,j}$, cf. (5).
- sdv** : Estimated standard deviation of data.
- sdv** : Estimated standard deviation of c .

User's guide

3. User's Guide to splval

A typical call is

```
f = splval(S, t {, sp})
```

Input Parameters

S : Struct representing the spline as in **splinefit**.

t : If **sp** is present, then

sp = 1 : **t** should be a scalar value in the knot range.

2 : **t** is not used.

Otherwise, **t** is a vector of arguments for s .

sp : (Optional). If present, then

sp = 1 : Evaluate $s(t)$, $s'(t)$, $s''(t)$ and estimates of their standard deviation.

2 : Find local extrema of s , i.e. zeros of s' in $[x_0, x_n]$.

Output Parameter

f : If **sp** was present, then

sp = 1 : **f** is the matrix $\begin{bmatrix} s & s' & s'' \\ \sigma_s & \sigma_{s'} & \sigma_{s''} \end{bmatrix}$, with the elements evaluated at **t**.

2 : **f** is an array with the zeros of s' in the knot range.

Otherwise, **f** is a vector of the same type as **t** with

f(k) = $s(t_k)$, $k = 1, \dots, \text{length}(\mathbf{t})$.

4. Examples

We first consider interpolation of the function

$$Y(t) = 1 / (1 + 2t^2) \quad \text{for } 0 \leq t \leq 3.$$

We first take the knots $x_j = j$, $j = 0, 1, 2, 3$, and use $n+3 = 6$ equidistant interpolation points,

```
t = linspace(0,3,6); y = 1./(1 + 2*t.^2);
x = [0 1 2 3];
[fail S] = splfit([t(:) y(:)], x)
```

The result is

```
fail = 0
S = x: [0 1 2 3]
      c: [1.0000 0.7893 0.2135 0.1101 0.0558 0.0526]
      pp: [5x4 double]
      sdy: 0
      sdc: [0 0 0 0 0]
```

The contents of S.pp are

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 1.0000 & 0.3402 & 0.1138 & 0.0526 \\ -0.6322 & -0.4835 & -0.0924 & -0.0096 \\ -0.2315 & 0.3802 & 0.0109 & 0.0719 \\ 0.2039 & -0.1231 & 0.0203 & 0 \end{bmatrix},$$

and the commands

```
tt = linspace(-1,3,1,201); s = splval(S,tt);
plot(t,y,'o', tt,s,'-', x,0*x,'s', ...
      tt,1./(1+2*tt.^2),'-g')
```

give the following result

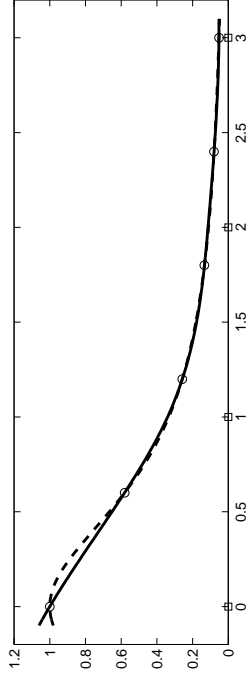


Figure 1. Knots $\{0, 1, 2, 3\}$

Next, we shift the knots, so that they are closer to the left and add the constraint $s'(0) = 0$. As data abscissae we take the knots plus one extra point, the midpoint of the last knot interval.

```
BC = [0 1 0 0; 0 0 0 0];
xb = [0 .5 1.4 3]; tb = [xb 2.2]'; yb = 1./(1 + 2*tb.^2);
[fail Sb] = splfit([tb yb],xb,BC);
```

Again we use `splval` to compute the spline, and get

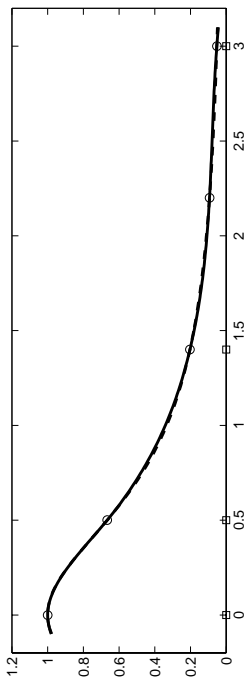


Figure 2. Knots $\{0, .5, 1.4, 3\}$. Constraint $s'(0) = 0$

If we use the constraints $s'(0) = s''(3) = 0$, we need no “extra” point, and use weights to suppress the “extra” point from before

```
BN = [0 1 0 0; 0 0 1 0]; wN = [ones(4,1); 0];
[fail SN] = splfit([tb yb wN],xb,BN)
```

The result is almost identical with the previous

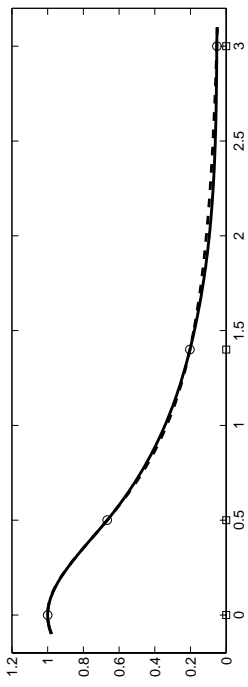


Figure 3. Knots $\{0, .5, 1.4, 3\}$. Constraints $s'(0) = s''(3) = 0$

Now, consider the ($m = 99$) data points shown by dots in Figure 4, and assume that they are “given” by the array `ty`. The data indicate that it is a good idea to use a *natural* spline (cf. [1, Chapter 5]), and the knots are chosen to be closest at the peaks.

```
x = [0 1.5 2.5 3 3.5 4.2 5 5.5 6 6.5 7.2 8.3 10];
B = [0 0 1 0;0 0 1 0];
[fail S] = splfit(ty,x,B);
```

The estimated standard deviation is $\sigma_y \simeq S.sdy = 0.0519$.

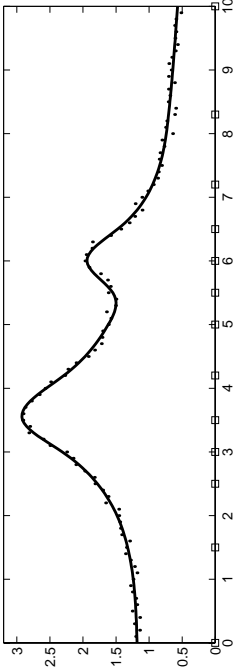


Figure 4. Data and fit with natural spline

The following command gives the positions of the local extrema,

```
z = splval(S,0,2)
```

The result is $z = [3.5607, 5.3332, 6.0062]$, and

```
ss = splval(S,z(k),1)
```

gives

```
ss = [ 2.9245e+00 -1.1102e-15 -5.3652e+00
       3.1620e-02  5.0620e-02  2.5700e-01 ] .
```

This verifies that $t = 3.5607$ is a root of s' , and its estimated standard deviation is $\sigma_{s'} \simeq 0.05062$.

Finally, to illustrate the use of multiple interior knots, consider the function

$$Y(t) = |t| \quad \text{for } -1 \leq t \leq 1,$$

as “given by”

$$(t_i, y_i, w_i) = (0.1(i - 10.5), |t_i|, 1), \quad i = 1, \dots, 20.$$

As before we fit with a natural spline. First we use 5 equidistant knots ($n=4$) and get

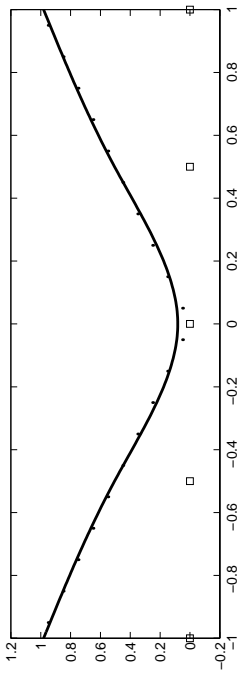


Figure 5. $Y(t) = |t|$, equidistant knots

Next, according to [1, Theorem 2.2] we can allow a jump in s' at $t=0$ by putting a triple knot there,

```
x = [-1 0 0 0 1]; B = [0 0 1 0;0 0 1 0];
[fail S] = splfit(ty,x,B);
```

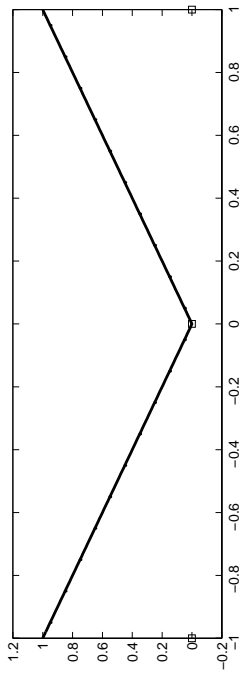


Figure 6. $Y(t) = |t|$, triple knot at $t=0$

The figure illustrates a great improvement. Also, the estimated standard deviation is reduced from $2.1110\text{e-}02$ to $2.6341\text{e-}16$, which is of the same order of magnitude as the machine accuracy. The piecewise polynomial representation is returned as

$$S.pp = \begin{bmatrix} -1 & 0 & 1 \\ 1 & \epsilon & 1 \\ -1 & 1 & 1 \\ 0 & \epsilon & 0 \\ \epsilon & \epsilon & 0 \end{bmatrix},$$

where ϵ indicates a number that is zero except for rounding errors.

This shows that there are two intervals in the knot range, and according to (5)

$$s(t) = \begin{cases} 1 - (t + 1) & \text{for } -1 \leq t \leq 0 \\ 0 + (t - 0) & \text{for } 0 < t \leq 1 \end{cases} .$$

5. Algorithms

5.1. Computation of B-spline coefficients. With the representation (4) the function φ in (3) can be expressed as

$$\varphi = \|\mathbf{W}\mathbf{r}\|_2^2 \quad \text{with} \quad \mathbf{r} = \mathbf{y} - \mathbf{A}\mathbf{c} , \quad (6)$$

where $\mathbf{W} = \text{diag}(w_1, \dots, w_m)$ and \mathbf{A} is the $m \times (n+3)$ matrix with $a_{ij} = N_j(t_i)$. The minimizing \mathbf{c} is found by orthogonal transformation. In the computation it is exploited that in each row of \mathbf{A} there is only 4 nonzero elements: $a_{i,j}; j+3$ for $t_i \in [x_{j-1}, x_j]$, and the orthogonal transformation is performed as a series of subtransformations, each involving all data points in a knot interval. See [1, Section 5.1.1] for a discussion of this.

Now consider end-point conditions (3b): $N_k(x_0) = N'(x_0) = N''(x_0) = 0$, and the first condition takes the form

$$\gamma_1 c_1 + \gamma_2 c_2 + \gamma_3 c_3 = \beta_{1,4} , \quad (7a)$$

where

$$\gamma_k = \beta_{1,1} N_k(x_0) + \beta_{1,2} N'_k(x_0) + \beta_{1,2} N''_k(x_0), \quad k = 1, 2, 3 . \quad (7b)$$

If $\gamma_1 \neq 0$ we say that the constraint is active, and (7a) can be reformulated to

$$c_1 = \alpha_1 c_2 + \alpha_2 c_3 + \alpha_3 . \quad (7c)$$

Thus, the ‘‘efficient’’ number of unknown coefficients is reduced from $\tilde{n} = n+3$ to $\tilde{n} = n+2$, and the residual vector

$$\mathbf{r} = \mathbf{y} - (c_1 \mathbf{A};_{,1} + c_2 \mathbf{A};_{,2} + c_3 \mathbf{A};_{,3} + c_4 \mathbf{A};_{,4} + \dots + c_{n+3} \mathbf{A};_{,n+3}) \quad (8a)$$

is modified to

$$\tilde{\mathbf{r}} = \tilde{\mathbf{y}} - (c_2 \tilde{\mathbf{A}};_{,2} + c_3 \tilde{\mathbf{A}};_{,3} + c_4 \mathbf{A};_{,4} + \dots + c_{n+3} \mathbf{A};_{,n+3}) , \quad (8b)$$

where

$$\tilde{\mathbf{y}} = \mathbf{y} - \alpha_3 \mathbf{A};_{,1}; \quad \tilde{\mathbf{A}};_{,j+1} = \mathbf{A};_{,j+1} + \alpha_j \mathbf{A};_{,1}, \quad j = 1, 2 . \quad (8c)$$

The only nonzero elements in $\mathbf{A};_{,1}$ correspond to data points in the first knot interval $[x_0, x_1]$, and this modification of \mathbf{A} and \mathbf{y} is made before the first orthogonal subtransformation. An active boundary condition at the other end similarly results in

$$c_{n+3} = \alpha_4 c_{n+1} + \alpha_5 c_{n+2} + \alpha_6 , \quad (9)$$

and modifications of \mathbf{A} , \mathbf{y} and $\tilde{\mathbf{n}}$.

This algorithm has an interesting interpretation: The $\{N_j\}_{j=1}^{n+3}$ form a basis for cubic splines over the given knots, and in the unconstrained case we find the best (in the weighted least squares sense) linear combination of these. In the constrained case we construct a new basis $\{\tilde{N}_j\}_{j=1}^n$ of cubic splines that satisfy the constraints; this basis has fewer elements than the original, is constructed by linear combinations of the $\{N_j\}$, and involves a ‘‘shift function’’ $\alpha_3 N_1 + \alpha_6 N_{n+3}$. Next, the $\{\tilde{c}_j\}$ are found by weighted least squares:

$$\text{Minimize } \varphi \equiv \|\mathbf{W}\tilde{\mathbf{r}}\|_2^2 \quad \text{with} \quad \tilde{\mathbf{r}} = \tilde{\mathbf{y}} - \tilde{\mathbf{A}}\tilde{\mathbf{c}} , \quad (10)$$

and finally (7c) and (9) are used to find the coefficients with respect to the original basis.

As hinted, the data is processed according to the knot intervals, and the first step consists in reordering the input so that $t_i \geq t_{i-1}$, $i = 2, \dots, m$.

B-splines involve an extended knot set $\tilde{\mathbf{x}}$ with two extra knots at either end. We use multiple end knots,

$$\tilde{\mathbf{x}} = [x_0 \ x_0 \ x_1 \ \cdots \ x_n \ x_n \ x_n]. \quad (11)$$

5.2. Estimation of standard deviation. The orthogonal transformation can be expressed in the form

$$\mathbf{Q}^\top (\mathbf{W}[\tilde{\mathbf{A}} \ \tilde{\mathbf{y}}]) = \left[\begin{array}{c|c|c} \mathbf{R} & \mathbf{b} & \\ \hline \mathbf{0}^\top & \rho & \\ \hline \mathbf{0} & \mathbf{0} & \end{array} \right], \quad (12)$$

(where $\tilde{\mathbf{A}} = \mathbf{A}$, $\tilde{\mathbf{y}} = \mathbf{y}$ if there is no active constraint). \mathbf{Q} is an orthogonal matrix and \mathbf{R} is upper triangular (with bandwidth 4). The coefficients are found by back substitution in

$$\mathbf{R}\tilde{\mathbf{c}} = \mathbf{b} \quad (13)$$

(followed by use of (7c) and (9) in case of boundary conditions), and the corresponding value of φ as defined by (10) is $\varphi = \rho^2$.

Each orthogonal subtransformation works on the expanded matrix corresponding to $[\tilde{\mathbf{A}} \ \tilde{\mathbf{y}}]$ (with 5 “active” columns), and after finishing the transformation, the estimated standard deviation is found by

$$\sigma_y \simeq \text{sd}y \equiv |\rho|/\sqrt{\tilde{m} - \tilde{n}}, \quad (14)$$

where \tilde{m} is the number of “active” data points, i.e. points with $w_i > 0$. $\text{sd}y$ is also an estimate of the standard deviation of each element in \mathbf{b} , and from (13) it follows that $\tilde{\mathbf{c}} = \mathbf{R}^{-1}\mathbf{b}$, and therefore (exploiting that also \mathbf{R}^{-1} is upper triangular)

$$\sigma_{c_k}^2 = \sum_{j=k}^{\tilde{n}} [\mathbf{R}^{-1}]_{k,j}^2 \sigma_{b_j}^2. \quad (15)$$

By inserting $\sigma_{b_j} \simeq \text{sd}y$ we get estimates of σ_{c_k} , and in case of constraints we use (7c) and (9) to get estimates of σ_{c_1} and $\sigma_{c_{n+3}}$.

5.3. Computation of pp representation. By comparison with Taylor expansion from $d_{1,k}$ we see that the representation (5) corresponds to

$$\begin{aligned} d_{2,k} &= s(d_{1,k}), & d_{3,k} &= s'(d_{1,k}) \\ d_{4,k} &= \frac{1}{2}s''(d_{1,k}) & d_{5,k} &= \frac{1}{6}s'''(d_{1,k}), \end{aligned}$$

where $d_{1,k} = x_j$ is a knot. As in (7b) we see that

$$s^{(q)}(x_j) = \sum_{r=j+1}^{j+3} c_r N_r^{(q)}(x_j), \quad q = 0, 1, 2, \quad (16)$$

which is used to get $d_{2:4,k}$. The last coefficient is found from the condition

$$s''(x_{j+1}) = 2d_{4,k} + 6d_{5,k}(x_{j+1} - x_j)$$

with the left-hand side computed by (16).

This approach is used to get the first $p-1$ columns of \mathbf{pp} . The last column is just

$$\mathbf{pp} :_p = [x_n \ s(x_n) \ s'(x_n) \ \frac{1}{2}s''(x_n) \ 0]^\top.$$

5.4. Finding the extrema of the spline. In each non vanishing knot interval s' is a 2nd order polynomial, so the task specified by `sp = 2` in `splval` can be stated as follows: For each interval given by the first row in `S.pp` solve the quadratic equation

$$3d_{5,k}\xi^2 + 2d_{4,k}\xi + d_{3,k} = 0. \quad (17a)$$

For each of the two roots we check that it is real and lies in the range $0 \leq \xi \leq d_{1,k+1} - d_{1,k}$. In that case

$$f_r = d_{1,k} + \xi \quad (17b)$$

is a root of s' . If a knot is also an extremum point, it may be found in both adjoining intervals. Therefore we “compress” \mathbf{f} so that the

returned values have a minimum distance of

$$\tau = \max\{10\epsilon_m(d_{1,p} - d_{1,1}), 10^{-3} \cdot \min_k\{d_{1,k+1} - d_{1,k}\}\}.$$

5.5. Auxiliary functions. The file `splfit.m` includes some auxiliary functions:

```
N = splbsd(t,z)
```

For a scalar $t \in [x_{j-1}, x_j]$ and z holding the subvector $\hat{x}_{j-3:j+2}$ defined in (11), the function returns the values of the four nonzero B-splines and their first two derivatives,

$$N = \begin{bmatrix} N_j(t) & N_{j+1}(t) & N_{j+2}(t) & N_{j+3}(t) \\ N'_j(t) & N'_{j+1}(t) & N'_{j+2}(t) & N'_{j+3}(t) \\ N''_j(t) & N''_{j+1}(t) & N''_{j+2}(t) & N''_{j+3}(t) \end{bmatrix}.$$

The computation is performed as described in [1, Appendix A.1].

```
N = splbsv(t,z)
```

For a vector t with all elements in $[x_{j-1}, x_j]$ and z holding the subvector $\hat{x}_{j-3:j+2}$, the function returns the values of the four nonzero B-splines, computed by a vectorized version of the algorithm in [1, Appendix A.1].

```
[c,sdy,sdc] = splfit1(tyw,x,alfa,con)
```

Handles the special case $n = 1$. `con` is a vector with two elements. If there is an active constraint at x_0 , then `con(1) = 1` and `alfa(1:3) = $\alpha_1:3$` ; (7c); otherwise `con(1) = 0`. Similarly `con(2)` flags an active constraint at x_n and `alfa(4:6) = $\alpha_4:6$` ; (9).

```
pp = splmpp(x,c)
```

Performs the computation described in section 5.3.

```
sdc = splvbc(R,sdy)
```

Performs the computations defined by (15). R is an $\tilde{n} \times 4$ array representing the banded upper triangular matrix \mathbf{R} , and the banded structure is exploited in computing the elements of \mathbf{R}^{-1} .

The file `splval.m` includes two auxiliary functions: `splbsd` described above and

```
vdv = splvdv(S,t)
```

This is the function used when `splval` is called with `sp = 1`. From (4) we find

$$s^{(q)}(t) = \sum_{j=1}^{n+3} c_j N_j^{(q)}(t).$$

The $\{c_j\}$ are considered as stochastic, and we find

$$\sigma_{s^{(q)}}^2(t) = \sum_{j=1}^{n+3} \left(N_j^{(q)}(t) \right)^2 \sigma_{c_j}^2.$$

The values $\{N_j^{(q)}(t)\}$ are computed by `splbsd` and the $\{\sigma_{c_j}\}$ are replaced by `S.sdc`.

References

- [1] H.B. Nielsen: *Cubic Splines*. IMM, DTU. 1998.