Splines in Endymion

José Grimm, Apics Team

January 29, 2009

Some of the code, definitions, and theorems are taken from Carl de Boor, "A Practical Guide to Splines", Springer Verlag, Vol 27 in 'Applied Mathematical Sciences', 27.

1 Piecewise polynomial functions, divided differences

A *polynomial of order* k is just a polynomial of degree less than k, it is defined by k coefficients. A *breakpoint sequence* is a sequence of real numbers $\xi_0 < \xi_1 < \ldots < \xi_l$. To each i, with $0 < i < l$, we associate an integer ν_i with $0 \leq \nu_i \leq k$. An element of $P_{k,\nu,\xi}$ will be a *piecewise polynomial function* (or pp function), which is a polynomial of order k on each interval $[\xi_i, \xi_{i+1}]$, subject to the condition that at ξ_i, polynomials to the left and the right agree with order ν_i. Thus the space P has dimension $n = kl - \sum \nu_i$.

1.1 Example. Assume $k = 4$, so that we consider polynomials of degree at most 3. Consider the case $\xi_i = 0$. A pp function near zero has the form $a + bx + cx^2 + dx^3$ for $x > 0$, and $a' + b'x + c'x^2 + d'x^3$ for $x < 0$. If $\nu_i = 0$, no condition is required. If $\nu_i = 1$, we want $a = a'$, if $\nu_i = 2$, we want $a = a'$ and $b = b'$, if $\nu_i = 3$, we want $a = a'$, $b = b'$ and $c = c'$, and if $\nu_i = 4$, we moreover want $d = d'$. If we know the function for $x < 0$, this leaves us $k - \nu_i$ degrees of freedom for the function on $[0, \xi_{i+1}]$. The case of interest is $\nu_i = k - 1$, this gives maximum regularity, and one degree of freedom for each interval. In this case $n = k + l - 1$. If we have some values f_i, and want $f(\xi_i) = f_i$, this removes $l + 1$ degrees of freedom, so that there are $k - 2$ remaining degree of freedom. This is zero if $k = 2$: there is a single continous piecewise linear function that passes through the points. For $k = 4$, we must add two additional constraints. This will be explained later.

Assume that we want periodic pp functions. If $f(x + \xi_l) = f(x + \xi_0)$ for small positive x, we can define the meaning of "agrees at order ν_l at ξ_l". If $\nu_l = k - 1$, this gives dimension l for the space P. However, the mapping that associates to f the values $f(\xi_i)$ is not necessarily injective. Example: assume $\xi_0 = -1$, $\xi_1 = 0$ and $\xi_2 = 1$. The function f with value $x + x^2$ for $x < 0$ and $x - x^2$ for $x > 0$ is a pp-function of order 3 with $\nu_i = 2$. The two functions f and the constant 1 form a basis of P. Thus, for every $g \in P$ we have $g(0) = g(1)$.

1.2 Given a function g and a sequence τ_i, we denote by $\mathcal{L}_\tau(g)$ the Lagrange interpolation of order $k+1$ to g at τ_i and by $[\tau_i, \ldots, \tau_{i+k}]g$ to be the leading coefficient of this polynomial. These are called the *divided differences*.

If all τ_i are different, then $h = \mathcal{L}_\tau(g)$ is defined by $h(\tau_j) = g(\tau_j)$. In this case an explicit formula is

$$\mathcal{L}_\tau(g)(t) = \sum_{j=i}^{i+k} g(\tau_j) \prod_{l\neq j} \frac{t-\tau_l}{\tau_j-\tau_l}. \qquad \text{(Lagrange)}$$

If we define $\tau_{i,i+k,j} = \prod_l 1/(\tau_j - \tau_l)$ where the product is over all l between i and $i+k$, excluding j, then the divided difference is just $\sum_j g(\tau_j)\tau_{i,i+k,j}$. Examples:

$$[\tau_1]g = g(\tau_1) \qquad [\tau_1,\tau_2]g = \frac{g(\tau_2)-g(\tau_1)}{\tau_2-\tau_1}.$$

Formula (1.9) below is an alternate representation. An example is

$$[a,b,c]g = \frac{g(a)-g(b)}{(a-b)(a-c)} - \frac{g(c)-g(b)}{(a-c)(b-c)}. \qquad (1.3)$$

1.4 The Lagrange interpolation polynomial (hence the divided differences) can be defined if some τ_j are repeated: both functions must agree at order k, where k is the number of repetitions. Taking limits in (1.3) gives

$$[a,b,b]g = \frac{g(a)-g(b)}{(a-b)^2} + \frac{g'(b)}{a-b}. \qquad (1.5)$$

A general formula is

$$\mathcal{L}_\tau(g)(t) = \sum_{j=i}^{i+k} (t-\tau_i)\cdots(t-\tau_{j-1})[\tau_i,\dots,\tau_j]g. \qquad \text{(Newton)}$$

Reverting the order of coefficients gives:

$$\mathcal{L}_\tau(g)(t) = \sum_{s=i}^{i+k} (t-\tau_{s+1})\cdots(t-\tau_{i+k})[\tau_s,\dots,\tau_{i+k}]g.$$

Proof. Let L_k be the Lagrange interpolation polynomial of g at $\tau_i,\dots,\tau_{i+k}$. We have by definition

$$L_k = L'_{k-1} + (t-\tau_i)\cdots(t-\tau_{i+k-1})[\tau_i,\dots,\tau_{i+k}]g$$

for some polynomial L'_{k-1} of order k. Then L'_{k-1} agrees with L_k, hence g for all τ_j with $i \le j \le i+k-1$. Is is hence L_{k-1}. This shows the Newton formula by induction.

1.6 The Leibnitz Formula for the product $f = gh$ is

$$[\tau_i,\dots,\tau_{i+k}]f = \sum_{r=i}^{i+k} [\tau_i,\dots,\tau_r]g \cdot [\tau_r,\dots,\tau_{i+k}]h. \qquad (1.7)$$

Proof. Define

$$F(t) = \sum_{r=i}^{i+k} (t-\tau_i)\cdots(t-\tau_{r-1})[\tau_i,\dots,\tau_r]g \sum_{s=i}^{i+k} (t-\tau_{s+1})\cdots(t-\tau_{i+k})[\tau_s,\dots,\tau_{i+k}]h.$$

By the Newton formulas, this agrees with gh at τ_j, being the product of the Lagrange interpolation polynomials. Expand the product as $\sum_{rs} \alpha_r\beta_s$, consider this as $A+B+C$ where A is the sum for $r < s$, B the sum for $r = s$ and C the sum for $r > s$. Now C vanishes at all τ_i, B is of degree k, and A is of degree less than k. Thus, $A+B$ is a polynomial of degree k that agrees with f, hence is the Lagrange interpolation of f. Its leading coefficient is that of B, and this is the RHS of (1.7).

1.8 Recurrence formula:

$$[\tau_i, \dots, \tau_{i+r}]g = \frac{[\tau_{i+1}, \dots, \tau_{i+r}]g - [\tau_i, \dots, \tau_{i+r-1}]g}{\tau_{i+r} - \tau_i}. \tag{1.9}$$

If the τ_i are distinct this is a trivial consequence of

$$\tau_{i,i+r,j} = \frac{\tau_{i+1,i+r,j} - \tau_{i,i+r-1,j}}{\tau_{i+r} - \tau_i}.$$

In general, we proceed as follows. Write (1.9) as $Ag = (Bg - Cg)/(\tau_{i+r} - \tau_i)$, where A, B and C are three linear operators. It suffices to consider the case where g is a polynomial. Write $g = h + P\prod(t - \tau_j)$, where h is the Lagrange interpolation and P is a polynomial. Obviously, A, B and C applied to $P\prod(t - \tau_j)$ give 0 as a result. Hence, we may assume that g is equal to the Lagrange interpolation polynomial at $\tau_i, \dots, \tau_{i+r}$. Note that A, B and C give a zero result when applied to a polynomial g of degree less than r. Using the Newton formula, it suffices to consider the case where

$$g = (t - \tau_i) \cdots (t - \tau_{i+r-1}).$$

Here $Ag = 1$, and $Cg = 0$. Decompose the first factor $t - \tau_i = (t - \tau_{i+r}) + (\tau_{i+r} - \tau_i)$. Then $g = (t - \tau_{i+1}) \cdots (t - \tau_{i+r}) + P$, where $P = (\tau_{i+r} - \tau_i)(x - \tau_{i+1}) \cdots (t - \tau_{i+r-1})$. We have $Bg = BP$ and $BP = \tau_{i+r} - \tau_i$.

Note the following application of the Leibnitz formula

$$[\tau_i, \dots, \tau_{i+k}]((t - x)g) = (\tau_i - x)[\tau_i, \dots, \tau_{i+k}]g + [\tau_{i+1}, \dots, \tau_{i+k}]g. \tag{1.10}$$

2 B-splines

Given a sequence $t_i \le t_{i+1} \le \ldots \le t_{i+k}$, we define

$$B_{i,k,t}(x) = (t_{i+k} - t_i)[t_i, \dots, t_{i+k}](\cdot - x)_+^{k-1}. \tag{2.1}$$

The notation $(\cdot - x)_+^{k-1}$ denotes the function $g(t)$, which depends on x, defined to be 0 if $t \le x$, and $(t - x)^{k-1}$ otherwise.

2.2 Basic properties. The Lagrange polynomial (and its leading coefficient) is a linear function of the value of g and its derivatives (it is of course a non-linear function of the interpolation points). As a function of x, g is a polynomial of order k, on each interval $[t_j, t_{j+1}]$. Thus, the same is true for B. Note that g is continuous (if $k > 1$), its derivative is continuous (if $k > 2$) etc. Hence the same is true for B_i, provided that the interpolation points are distinct. This implies that B is a pp-function, for the breakpoint sequence associated to t_i, where $k - \nu_i$ is the number of repetitions of t_i (this is obvious if there are no repetitions, the general case is explained later).

Note that $B_i(x)$ has support in $[t_i, t_{i+k}]$. In fact, if $x \ge t_{i+k}$, for all j in the interval $[i, i+k]$, we have $g(t_j) = 0$, hence the divided difference vanishes and $B(x)$ is zero. If $x \le t_i$, then $g(t_j) = (t_j - x)^{k-1}$, so that g takes the same values as $(t - x)^{k-1}$: this is the Lagrange interpolation polynomial, its coefficient in t^k is zero.

Note the following symmetry. Let $s_i = -t_{N-i}$ be the reflected sequence (we replace each knot by its opposite, then reorder them). For $t_i \le x \le t_{i+k}$, $B_{i,k,t}(t_i + x)$ is $t_{i+k} - t_i$

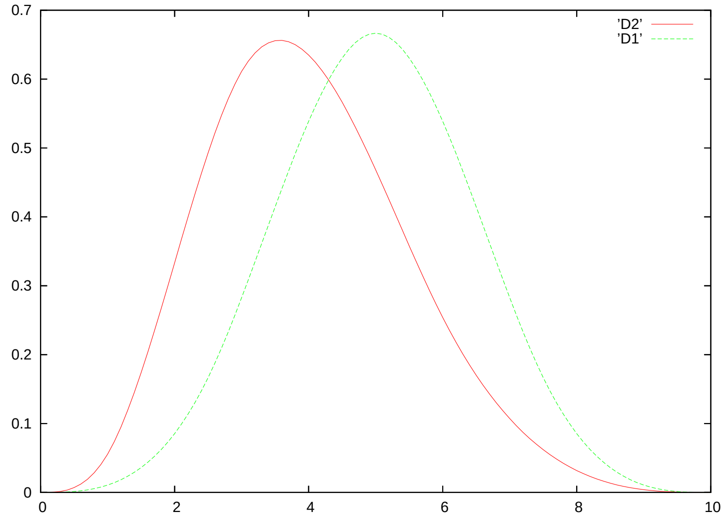

Figure 1: Two splines, corresponding to the knot sequence (0,2.5,5,7.5,10) and (0,1,3,6,10)

times the divided differences at t_i through t_{i+k} of a function G, such that $G(t_j) = 0$ if $t_j \le t_i + x$ and $G(t_j) = (t_j - t_i - x)^{k-1}$ otherwise. Let $B' = B_{N-i,k,s}$. Then $B'(s_{N-i-k} - x)$, for $s_{N-i-k} \le x \le s_{N-i}$, or equivalently $t_i \le -x \le t_{i+k}$ is $t_{i+k} - t_i$ times the divided differences at $-t_i$ through $-t_{i+k}$ of the function H such that $H(-t_j) = 0$ if $t_j \ge t_i + x$ and $H(-t_j) = (-t_j + t_i + x)^{k-1}$ otherwise. Now $G(t) \pm H(-t) = (t - t_i - x)^{k-1}$, where the sign depends on the parity of k. The divided differences of $G(t) \pm H(-t)$ at the $k+1$ points are zero. Hence $B'(s_{N-i-k} - x) = \pm B_{i,k,t}(t_i + x)$.

2.3 Basic recurrence. Applying (1.10) to

$$(t-x)_+^{k-1} = (t-x)(t-x)_+^{k-2}.$$

gives

$$[t_i, \dots, t_{i+k}](\cdot - x)_+^{k-1} = (t_i - x)[t_i, \dots, t_{i+k}](\cdot - x)_+^{k-2} + [t_{i+1}, \dots, t_{i+k}](\cdot - x)_+^{k-2}.$$

Apply (1.9) to the first term, and simplify. We get the following recurrence formula

$$B_{i,k}(x) = \frac{x - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(x). \tag{2.4}$$

2.5 In the case $k = 1$, $B_{i,1}(x) = 1$ if $t_i < x < t_{i+1}$, and the function is zero otherwise. Its value is not well-defined at t_i or t_{i+1}. In the case $k = 2$, the previous formula gives

$$B_{i,2}(x) = \frac{x - t_i}{t_{i+1} - t_i} \quad (t_i < x < t_{i+1}), \qquad B_{i,2}(x) = \frac{t_{i+2} - x}{t_{i+2} - t_{i+1}} \quad (t_{i+1} < x < t_{i+2}).$$

Hence $B_{i,2}$ is a piecewise linear continuous function taking the value 0 at $-\infty$, t_i, t_{i+2}, ∞, and the value 1 at t_{i+1}. We allow the degenerate case $t_i = t_{i+1}$ or $t_{i+1} = t_{i+2}$. In this case the function can be discontinuous. In the special case $t_i = t_{i+2}$, the function is zero.

2.6 In the case $k = 3$, we get the following formulas, valid respectively for $t_i < x < t_{i+1}$, $t_{i+2} < x < t_{i+3}$ and $t_{i+1} < x < t_{i+2}$:

$$B_{i,3}(x) = \frac{(x-t_i)^2}{(t_{i+1}-t_i)(t_{i+2}-t_i)}, \qquad B_{i,3}(x) = \frac{(t_{i+3}-x)^2}{(t_{i+3}-t_{i+1})(t_{i+3}-t_{i+2})}$$

$$B_{i,3}(x) = \frac{(x-t_i)(t_{i+2}-x)}{(t_{i+2}-t_i)(t_{i+2}-t_{i+1})} + \frac{(x-t_{i+1})(t_{i+3}-x)}{(t_{i+3}-t_{i+1})(t_{i+2}-t_{i+1})}$$

2.7 Consider the case $k = 4$. For simplicity, we consider here the case $t_i = a+ib$. Fix some i, define $x' = (x-a)/b - i$. The values of the splines on the intervals $[a+ib, a+(i+1)b]$, $[a+(i+1)b, a+(i+2)b]$ $[a+(i+2)b, a+(i+3)b]$ and $[a+(i+3)b, a+(i+4)b]$ are

$$6B(x) = (x')^3, \text{ or } 4-3(x'-2)^2x', \text{ or } 4+3(x'-2)^2(x'-4), \text{ or } (4-x')^3.$$

2.8 The B-splines form a partition of unity, namely, they have compact support, are non-negative, and satisfy

$$\sum_{j=i}^{i+k-1} B_{i,k} = 1 \qquad t_{i+k-1} < x < t_{i+k}. \tag{2.9}$$

The fact that B is non-negative is obvious by (2.4). In fact, if $x < t_i$, then $B_{i,k-1}(x) = 0$, otherwise $x - t_i \geq 0$, and $B_{i,k-1}(x) \geq 0$, etc.

Let $S_{i,k} = \sum B_{j,k}$, where the sum is for $i \leq j \leq i+k-1$. If $t_{i+k-1} < x < t_{i+k}$ we have $B_{i+k,k-1}(x) = B_{i,k-1}(x) = 0$. Hence

$$S_{j,k} = \sum_{j=i}^{i+k-1} B_{i,k} = \sum_{j=i}^{i+k-1} \frac{x-t_{j+1}}{t_{j+k}-t_{j+1}} B_{j+1,k-1}(x) + \sum_{j=i}^{i+k-2} \frac{t_{j+k}-x}{t_{j+k}-t_{j+1}} B_{j+1,k-1}(x).$$

This is $\sum_{j=i}^{i+k-2} B_{j+1,k-1}(x) = S_{i-1,k-1}$. This hows equation (2.9) by induction.

2.10 Evaluation of splines. The `eval_spline` procedure of Endymion uses equation (2.4). It takes three arguments: a number x, an index L, and a knot sequence t_i. It is assumed that $t_L \leq x \leq t_{L+1}$. The only non-vanishing B-splines of order k have index i with $t_i < x < t_{i+k}$, hence $L-k+1 \leq i \leq L$. We consider only the case $k = 4$, so that $i = L-3$, $i = L-2$, $i = L-1$ and $i = L$. In order to simplify notations we write a, b, c, etc., instead of t_{L+1}, t_{L+2}, etc.

Thus we assume that the knots are $\ldots \leq D \leq C \leq B \leq A < a \leq b \leq c \leq d \leq \ldots$, etc, and $A \leq x \leq a$. There are four splines that do not vanish at x, they have support in $[D,a]$, $[C,b]$, $[B,c]$, $[A,d]$. The values at x are computed in h. Values of D and d are not required.

Let $u = (a-x)/(a-A)$, $v = (x-A)/(a-A)$. These are the values of the two B-splines of order 2 that do not vanish at x. Write $d_2 = u/(a-B)$, $d_3 = v/(b-A)$, $u_1 = d_2(a-x)$, $u_2 = d_2(x-B) + d_3(b-x)$, and $u_3 = d_3(x-A)$. The quantities u_i are the values of the three B-splines of order 3 that do not vanish at x. Write $e_1 = u_1/(a-C)$, $e_2 = u_2/(b-B)$, $e_3 = u_3/(c-A)$. The values of the B-splines of order 4 that do not vanish at x are

$$h_0 = e_1(a-x), h_1 = e_1(x-C) + e_2(b-x), h_2 = e_2(x-B) + e_3(c-x), h_3 = e_3(x-A).$$

An alternate presentation of the formulas is

$$u_1 = \frac{(a-x)(a-x)}{(a-B)(a-A)}, \quad u_2 = \frac{(a-x)(x-B)}{(a-A)(a-B)} + \frac{(b-x)(x-A)}{(b-A)(a-A)}, \quad u_3 = \frac{(x-A)(x-A)}{(b-A)(a-A)},$$

then

$$h_0 = \frac{(a-x)^3}{(a-A)(a-B)(a-C)}, \qquad h_3 = \frac{(x-A)^3}{(c-A)(b-A)(a-A)},$$

$$h_1 = \frac{x-C}{a-C}u_1 + \frac{b-x}{b-B}u_2, \qquad h_2 = \frac{x-B}{b-B}u_2 + \frac{c-x}{c-A}u_3.$$

2.11 Assume that the knots are (..., 0, 2, 3, 4, 5, 6, ..., $n-3$, $n-2$, n, ...). The first value 0 and the last value n is repeated as much as needed. There is a hole between 0 and 2, a hole between $n-2$ and n. If we evaluate at $i+1/2$, where i is an integer, we get in general the four values 1/48, 23/48, 23/48 and 1/48. We write this as (1,23,23,1)/48. According to (2.9), the sum of the four coefficients is always 1. If x is an integer, we chose $A = x$ (except for $x = n$). The coefficients are generally (1,4,1,0)/6. Exceptions: For $x = 0$ we get (1,0,0,0)/1, For $x = 1$ we get (9,37,23,3)/72, For $x = 2$ we get (1,5,3,0)/9, For $x = 3$ we get (3,17,4,0)/24, For $x = n-3$ we get (4,17,3,0)/24, For $x = n-2$ we get (3,5,1)/9, For $x = n-1$ we get (3,23,37,1)/72, For $x = n$ we get (0,0,0,1)/1.

2.12 More relations. Assume $t_{i+k-1} \le x \le t_{i+k}$. The following is obvious from the definition.

$$B_i(x) = \prod_{j=i+1}^{i+k-1} \frac{t_{i+k}-x}{t_{i+k}-t_j}. \tag{2.13}$$

Assume now $t_i \le x \le t_{i+1}$. Then

$$B_i(x) = \prod_{j=i+1}^{i+k-1} \frac{x-t_i}{t_j-t_i}. \tag{2.14}$$

Assume again $t_{i+k-1} \le x \le t_{i+k}$. We have then by definition

$$B_{i+1}(x) = \frac{t_{i+k+1}-t_{i+1}}{t_{i+k}-t_{i+k+1}} \prod_{j=i+1}^{i+k-1} \frac{t_{i+k}-x}{t_{i+k}-t_j} + \prod_{j=i+2}^{i+k} \frac{t_{i+k+1}-x}{t_{i+k+1}-t_j}.$$

Define $d = \dfrac{t_{i+k+1}-t_{i+1}}{t_{i+k}-t_{i+k+1}}$. The first term is dB_i, thus

$$B_i(x) + B_{i+1}(x) = \frac{1}{t_{i+k}-t_{i+k+1}} \left[\frac{(t_{i+k}-x)^{k-1}}{\prod_j (t_{i+k}-t_j)} - \frac{(t_{i+k+1}-x)^{k-1}}{\prod_j (t_{i+k+1}-t_j)} \right] \tag{2.15}$$

where the products on j are for $i+2$ to $i+k-1$.

We have now the (perhaps surprising) formula

$$(k-1)B_{i+1}(x) = (t_{i+1}-x)B_i'(x) + (x-t_{i+k+1})(B_i'(x) + B_{i+1}'(x)), \tag{2.16}$$

where B_i' is the derivative of B_i. This can be shown as follows: Define $u = t_{i+k}$, $v = t_{i+k+1}$, let d be as above. We have shown that for some constants a and b, we have

$$B_i(x) = a(x-u)^{k-1} \qquad B_{i+1} = ad(x-u)^{k-1} + b(x-v)^{k-1}.$$

The RHS of (2.16) is then $d(v-u)B'_i + (x-v)B'_{i+1}$, hence

$$(k-1)[ad(v-u)(x-u)^{k-2} + ad(x-v)(x-u)^{k-2} + b(x-v)^{k-1}]$$

and this is clearly the LHS of the equation.

Assume now $t_i \le x \le t_{i+1}$. The analog formula is

$$(k-1)B_{i-1}(x) = (t_{i+k-1} - x)B'_i(x) + (x - t_{i-1})(B'_i(x) + B'_{i-1}(x)). \tag{2.17}$$

2.18 Evaluation. There is a function in Endymion that converts splines to pp functions. More precisely, we shall assume $k = 4$, and consider a pp function $B = \sum \alpha_i B_{i,k,t}$. Let x be a point, j an index such that $t_{j+3} < x < t_{j+4}$. Then B is a polynomial of order 4 in the interval, so that we can write

$$B(y) = s_0 + s_1(y-x) + s_2(y-x)^2 + s_3(y-x)^3 \qquad (t_{j+3} < y < t_{j+4})$$

The program computes some T_i such that $B(x) = T_0$, $B'(x) = 3T_1$, $B''(x) = 6T_2$ and $B'''(x) = 6T_3$. From this we get $s_0 = T_0$, $s_1 = 3T_1$, $s_2 = 3T_2$ and $s_3 = T_3$.

In fact, on each interval there are four non-vanishing splines, so that the program computes T_{ij}, then multiplies by the coefficients shown above, (depending on whether the quantities s_i or the derivatives of B is wanted), multiplies by α_i and sums. For simplicity, we shall assume $j = 0$, so that the splines are B_0, B_1, B_2 and B_3. The program uses the following quantities

$$m_{21} = \frac{-1}{(t_4-t_3)(t_4-t_2)(t_5-t_2)} \qquad m_{22} = \frac{1}{(t_4-t_3)(t_5-t_2)(t_5-t_3)}$$
$$m_{31} = \frac{t_4-x}{(t_4-t_3)(t_4-t_2)(t_5-t_2)} \qquad m_{32} = \frac{x-t_3}{(t_4-t_3)(t_5-t_2)(t_5-t_3)}$$
$$m_{41} = \frac{(x-t_2)(t_4-x)}{(t_4-t_3)(t_4-t_2)(t_5-t_2)} \qquad m_{42} = \frac{(x-t_3)(t_5-x)}{(t_4-t_3)(t_5-t_2)(t_5-t_3)}$$

as well as

$$A_0 = m_{41} + m_{42}, \qquad A_1 = 2(m_{31} - m_{32}), \qquad A_2 = 2(m_{21} - m_{22}).$$

It is clear that the second derivative of A_0 is A_2. It is also easy to see that the first derivative of A_0 is A_1.

Since $t_3 \le x \le t_4$ we have, by relation (2.13):

$$B_0(x) = \frac{(x-t_4)^3}{(t_1-t_4)(t_2-t_4)(t_3-t_4)},$$

and (2.14) gives

$$B_3(x) = \frac{(x-t_3)^3}{(t_4-t_3)(t_5-t_3)(t_6-t_3)}.$$

If c is inverse of the denonimator of B_0 we get $T_{03} = c$, $T_{02} = (x-t_4)T_{03}$, $T_{01} = (x-t_4)T_{02}$, $T_{00} = (x-t_4)T_{01}$, and similar formulas for T_{3i}. Now, by (2.15) we have

$$B_0(x) + B_1(x) = \frac{1}{t_4-t_5}\left[\frac{(t_4-x)^3}{(t_4-t_2)(t_4-t_3)} - \frac{(t_5-x)^3}{(t_5-t_2)(t_5-t_3)}\right].$$

We pretend that the derivative of this expression is $-3A_0$. Hence $T_{11} = -T_{01} - m_{41} - m_{42}$, $T_{12} = -T_{02} - m_{31} + m_{32}$, $T_{13} = -T_{03} - m_{21} + m_{22}$.

The relation $\sum B_i = 1$ gives $\sum B_i' = 0$ after differentation. Hence $T_{21} = -T_{31} + m_{41} + m_{42}$, $T_{22} = -T_{32} + m_{31} - m_{32}$, $T_{23} = -T_{33} + m_{21} - m_{22}$.

Equations (2.16) and (2.17) read

$$3B_1(x) = (t_1 - x)B_1'(x) + (x - t_5)(B_0'(x) + B_1'(x))$$

$$3B_2(x) = (t_6 - x)B_3'(x) + (x - t_2)(B_2'(x) + B_3'(x))$$

This gives $T_{10} = (t_1-x)T_{01}+(t_5-x)(m_{41}+m_{42})$ and $T_{20} = (t_6-x)T_{31}+(x-t_2)(m_{41}+m_{42})$

2.19 Derivatives We assume here that the knots have no repetition. Let Df be the derivative of f with respect to x. Using relation (1.9) gives

$$B_{i,k,t}(x) = [t_{i+1}, \dots, t_{i+k}](\cdot - x)_+^{k-1} - [t_i, \dots, t_{i+k-1}](\cdot - x)_+^{k-1}.$$

Now

$$D(t-x)_+^{k-1} = -(k-1)(t-x)_+^{k-2}$$

so that

$$DB_{i,k}(x) = (k-1)\left[-\frac{B_{i+1,k-1,t}}{t_{i+k} - t_{i+1}}(x) + \frac{B_{i,k-1,t}}{t_{i+k-1} - t_i}(x)\right] \tag{2.20}$$

hence

$$D(\sum \alpha_i B_{i,k,t}) = (k-1)\sum \frac{\alpha_i - \alpha_{i-1}}{t_{i+k-1} - t_i} B_{i,k-1,t}. \tag{2.21}$$

Hence

$$D^j(\sum_i \alpha_i B_{ik}) = \sum_i \alpha_i^{(j)} B_{i,k-j}. \tag{2.22}$$

where

$$\alpha_r^{(j+1)} = \frac{\alpha_r^{(j)} - \alpha_{r-1}^{(j)}}{t_{r+k-j-1} - t_r}(k-j-1). \tag{2.23}$$

2.24 There is a function `disc_jump` in Endymion that computes the jump of the third derivatives at a knot. It does not use (2.23), but direct computation. Define

$$g(t) = (t-x)_+^{k-1} \qquad f_i = t_{i+k} - t_i \qquad t_{i,i+k,j} = \prod_{l=i}^{k} \frac{1}{t_j - t_l}.$$

Then $B_i = f_i \sum_j g(t_j) t_{i,i+k,j}$ and

$$B_i^{(k-1)} = (k-1)! f_i \sum_j (t_j - x)_+^0 t_{i,i+k,j}.$$

The objective is to compute the jump J_{ij} of this derivative at t_j. This is obviously $(k-1)! f_i t_{i,i+k,j}$. The program omits the factorial, but normalises it by multiplication by $((n-7)/(t_{n-4}-t_3))^3$. In the case where $t_i = a + ib$, this is $1/b^3$. From section 2.7, we know that the third derivatives, after normalisation, are 1, -3, 3 and -1, so that the jumps are 1, -4, 6, -4 and 1.

3 Splines and pp functions

3.1 Main result The set of $B_{i,k,t}$ is a basis of $P_{k,\nu,\xi}$ provided that the breakpoint sequence $\xi_0 < \xi_1 < \ldots < \xi_l$ is related to the knot sequence $t_0 \le t_1 \le \ldots \le t_{n+k-2} \le t_{n+k-1}$ by the condition: Each ξ_i must appear exactly $k - \nu_i$ in the sequence t_j, where $\nu_0 = 0$ and $\nu_l = 0$.

This gives $n = k + \sum_{i=1}^{l-1}(k - \nu_i)$. This is the dimension of P. Hence we must prove that B-splines are in P, and linearly independent. The condition $t_i = \xi_0$ for $i < k$ can be replaced by $t_i \le \xi_0$, likewise for the case $i \ge n - 1$.

We may assume $0 < \nu_i < k$. In fact, if $\nu_i = k$, every element of P is a polynomial in a neighborhood of ξ_i. Thus, if ξ' and ν' are the sequence without ξ_i and ν_i, we have $P_{k,\nu,\xi} = P_{k,\nu',\xi'}$. On the other hand, since ξ_i does not appear in the knot sequence, the set of B-splines is not affected by this change. Assume $\nu_i = 0$. There are no constraint at the point ξ_i. Hence, if ξ' and ξ'' are the sequences obtained for $j \le i$ or $j \ge i$, likewise for ν' and ν'', we have that $P_{k,\nu,\xi}$ is the direct sum of $P_{k,\nu',\xi'}$ and $P_{k,\nu'',\xi''}$. Assume that $\xi_i = t_{j+1} = t_{j+k}$. The set of B-splines associated to the knots up to t_{j+k} is a basis of the first space, the set of B-splines associated to the knots starting with t_j is a basis of the second set.

Let's show that the B-splines are in P. We know that they are polynomials on each $[t_j, t_{j+1}]$. Let δ_{ij} be the regularity of B_i at t_j. This is 0 if the function is discontinuous, 1 if the function is continuous, but the first derivative is not, etc. Let δ'_{ij} be regularity of the splines of order $k-1$ with the same knot sequence. Let r_i be the number of repetitions of $t)i$. We must show that $\delta_{ij} \ge \nu'_j$, where ν'_j is the ν_l at the t_j, hence $k - \nu'_j = r_j$, so that we must show $\delta_{ij} \ge k - r_j$. We shall use (2.20). It says that the δ_{ij} is at least one more that the minimum of $delta'_{ij}$ and $delta'_{i+1,j}$. These two quantities are at least $k - 1 - r_j$ by induction.

Assume $\sum \alpha_i B_{ik} = 0$. Let's differentiate, and consider relation (2.21). By induction, the B-splines of order $k-1$ are linearly independent, and this gives $\alpha + i = \alpha_{i+1}$ (we use here the fact that ν_i is not zero, i.e. that the denominator $t_{i+k-1} - t_i$ does not vanish. We get $\alpha_i(\sum B_{ik}) = 1$. Since the sum of the b_i is one, this implies $\alpha_i = 0$.

3.2 Interpolator The purpose of the `interpolator` class is to help finding the zero of a function f. The assumption is that f is defined for $x \ge 0$, is decreasing, and vahishes somewhere. A simple algorithm is: evaluate the function at two points, find a function r that fits at these points, get the zero of r, discard one of the two points, and iterate until precision is met. Our algorithm uses three points. Initially we know the value at zero, infinity and a third point.

We assume that the function looks like $r(p) = (up + v)/(zp - w)$. So that the problem becomes Given three points (p_1, f_1), (p_2, f_2) and (p_3, f_3), find the rational function such that $r(p_i) = f_i$, then p such that $r(p) = 0$.

We want to find p such that

$$\begin{pmatrix} p_1 & 1 & f_1 & f_1 p_1 \\ p_2 & 1 & f_2 & f_2 p_2 \\ p_3 & 1 & f_3 & f_3 p_3 \\ p & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ -z \end{pmatrix} = 0$$

has a non-zero solution (u, v, w, z). This implies that the matrix is singular, hence

$$p = \frac{\begin{vmatrix} p_1 & f_1 & f_1 p_1 \\ p_2 & f_2 & f_2 p_2 \\ p_3 & f_3 & f_3 p_3 \end{vmatrix}}{\begin{vmatrix} 1 & f_1 & f_1 p_1 \\ 1 & f_2 & f_2 p_2 \\ 1 & f_3 & f_3 p_3 \end{vmatrix}}$$

Consider $h_1 = f_1(f_2 - f_3)$, $h_2 = f_2(f_3 - f_1)$, $h_3 = f_3(f_1 - f_2)$. We have then

$$p = -\frac{h_1 p_2 p_3 + h_2 p_1 p_3 + h_3 p_1 p_2}{h_1 p_1 + h_2 p_2 + h_3 p_3}.$$

In the special case where $p_3 = \infty$, this formula simplifies to

$$p = -h_1 p_2 + h_2 p_1 / h_3.$$

3.3 Assume now $p_1 < p_2 < p_3$ and $f_1 > f_2 > f_3$, and $f_1 f_3 < 0$. It is obvious that the function r is uniquely defined: in fact, if z is non-zero, we can normalise it as $z = 1$, then r exists if and only if the points (p_i, f_i) are not aligned, and if z is zero, then r is linear and is defined if and anly if the points are aligned. Assume that r has a pole in the interval $[p_1, p_3]$. The condition $r(p_1) > r(p_3)$ implies that r is locally increasing; if p_2 is in the interval, we have either $r(p_2) > r(p_1)$ or $r(p_2) < r(p_3)$. This contradicts assumptions $f_1 > f_2 > f_3$. Thus r is continuous on $[p_1, p_3]$. Because of $f_1 f_3 < 0$, r has a zero between p_1 and p_3. Ifn fact, if $f_2 > 0$, we have $p_2 < p < p_3$ and we can discard p_1, if $f_2 < 0$ we have $p_1 < p < p_2$ and we can discard p_3.

3.4 Computing Fourier coefficients The Fourier coefficient of index j of a function f s given by

$$c_j(f) = \frac{1}{2\pi} \int_0^{2\pi} f(\theta) e^{-ij\theta} d\theta.$$

We shall consider the case where f is defined by splines. This means that there are some intervals I_k, and on each I_k we have $f(x) = f_k(x)$. Define f_k to be zero outside I_k. Then $f = \sum_k f_k$, so that $c_j(f) = \sum_k c_j(f_k)$. For each k we can restrict the integration interval

$$c_j(f_k) = \frac{1}{2\pi} \int_{\theta_{1k}}^{\theta_{2k}} f_k(\theta) e^{-ij\theta} d\theta.$$

Lets's assume that f_k is a polynomial on the interval, say $f_k(\theta + t) = \sum_l a_{kl} \theta^l$. We chose $t = (\theta_1 + \theta_2)/2$, this is numerically much better then $t = 0$. We have

$$c_j(f_k) = \frac{1}{2\pi} e^{-ijt} \int_{\theta_{1k} - t}^{\theta_{2k} - t} f_k(\theta + t) e^{-ij\theta} d\theta$$

Note that the integral is between $-\delta_k$ and $+\delta_k$, where $2\delta_k = t_{2k} - t_{1k}$, hence if

$$m_{lj}(\delta) = \frac{1}{2\pi} \int_{-\delta}^{\delta} \theta^l e^{-ij\theta} d\theta$$

we get

$$c_j(f) = \sum_k \sum_j e^{-ijt_k} a_{kl} m_{lj}(\delta_k).$$

Let's compute m_{lj} for $0 \le l \le 3$. For $j = 0$, the integrand is a polynomial.

$$m_{lj}(\delta) = \frac{\delta^L - (-\delta)^L}{2\pi L}, \qquad L = l + 1$$

This is 0 if $l = 1$ or $l = 3$. This is δ/π if $l = 1$ and $\delta^3/3\pi$ if $l = 2$. In the case $l = 0$, we have

$$\frac{1}{2\pi} \int_{-\delta}^{\delta} e^{-ij\theta} d\theta = \frac{\sin j\delta}{j\pi}.$$

In the general case, we integrate by parts: we integrate the exponential, and differentiate θ^l. The result is

$$m_{lj} = \frac{1}{j} \left[\frac{\delta^l + (-\delta)^l}{2\pi} \sin j\delta + i \frac{\delta^l - (-\delta)^l}{2\pi} \cos j\delta - ilm_{l-1,j} \right].$$

$$m_{lj} = \frac{1}{j}[\frac{\delta^l}{\pi} \sin j\delta - ilm_{l-1,j}] \quad (l \text{ even}) \qquad = \frac{1}{j}[i\frac{\delta^l}{\pi} \cos j\delta - ilm_{l-1,j}] \quad (l \text{ odd}).$$

4 Computing splines

The user program takes three arguments, a vector of points τ_i, a vector of values g_i, that defines a function g such that $g(\tau_i) = g_i$ and a vector of weights w_i. Let δ_i be $f(\tau_i) - g(\tau_i)$, $d_1(f, g)$ be the sum of the $\delta_i^2 w_i$. Denote by $d_p(f, g)$ the sum of $d_1(f, g)$ and p times the disc-norm of f (this is a quantity that measures how irregular the function f is). The user gives a number s, the target for $d_1(f, g)$. The idea of the program is the following:

- Given a knot sequence, a subsequence of the τ_i, we find f that such that $d_1 = d_1(f, g)$ is minimal. If $d_1 > s$, more knots must be used. If $s = 0$ all τ_i will be used.

- Given p, we find f that such that $d = d(f, g)$ is minimal. Let $d_p = d_1(f, g)$ for this p. This is an increasing function of p, and $d_p > d_1$. We chose p such that $d_p = s>$

  Different values of p are tried. In each case, we find f such that $\|f - g\|^2$ is minimal, and chose p such that $\|f - g\|_1^2 = s$.

The function f is defined to be $\sum c_i B_i$. Then c_i is a solution of a linear system.

We consider the following matrices. For each i, x_i is τ_i, y_i is $g(\tau_i)$, and w_i is the weight associated to this point. Let t_j be the knot sequence, and l be such that $t_{l+3} \le x_i \le t_{l+4}$. Let h_0, h_1,h_2 and h_3 be the values of the splines that do not vanish at x_i, as computed by the eval-spline procedure. We have $Q_{i,l+j} = h_{ij}$ and $R_{ij} = Q_{ij} w_i$.

We want to find coefficients c_j such that $\sum_j c_j B_i(\tau_j) = y_i$. This can be written as $\sum_j Q_{ij} c_j = y_i$. This is $Qc = y$. If we multiply by the transposer of $Q*$ and introduce weights we get

$$\sum_{jl} Q_{li} Q_{lj} w_l^2 c_j = \sum_l Q_{li} w_l^2 y_l$$

This is $Q^* Qc = Q^* y$.

This function is called with the following arguments: Find a knot sequence t, the splines for it, and the coefficients c, such that, if $f = \sum_{i=0}^{n-1} c_i B_i$, and g the function that takes the value `gtau(i)` at `tau(i)`, then the weighted L^2 norm of $f - g$ is s. Return values are t, c, n, and the norm |fp|. Input parameters are $k = 3$, s, the maximum number of knots `nest`, and the structure `data`.

Main idea: Define

$$\|f\| = \sum_i f(\tau_i)^2 w_i + p \sum_j f_j^2.$$

In case $f = g$, the data, we have $g_j = 0$, and $g(\tau_i) = |gtau(i)|$. Otherwise, given the knot sequence ξ_i, and the B-splines B_i associated to it, we have $f = \sum c_j B_j$, and f_j is the discontinuity of the second derivative of f at ξ_j.

Minimising the norm of $f - g$ is then obviously to solve in c_i the following system:

$$\sum_{jl} B_j(\tau_l) B_i(\tau_l) w_l c_j + p \sum_{jl} \Delta B_j(\xi_l) \Delta B_i(\xi_l) c_j = \sum_l B_i(\tau_l) g(\tau_l) w_l.$$

The quantity p is defined in such a way that the norm of $f - g$ is exactly s. Since $p \geq 0$, in a first pass we chose ξ_j such that for $p = 0$, the norm is less than s. After that, p is chosen in a second pass.

exterior_knots: We assume here $t_0 = t_1 = t_2 = t_3 = \xi_0 = \tau_0$, and similarly $t_i = \tau - m - 1$ for the last data points. Moreover $\nu_i = k - 1$. In otherwords, our splines will be c^2. We add the $2k_1$ knots that are exterior, making them equal to the left and right boundary of the interval.

If $s = 0$, the result is an interpolating spline. Use the maximum number of knots. Note that the default is to use the minimal number of knots.

init_knots1: We assume that `nrdata(i)` is zero if τ_i is not a knot. Otherwise, it is the location of the next knot. In other words, if $t_j = \tau_i$ and $t_{j+1} = \tau_{i+l}$ then l will be in |nrdata(i)|. This piece of code assume $k = 3$. In that case, τ_1 and τ_{m-2} are missing.

In the main loop, we increase the number of knots, until the good number if found. These knots are found by solving a linear equation. We start by emptying the matrix and the RHS.

5 Part two

We have determined the number of knots and their position.

We now compute the B-Spline coefficients of the smoothing spline `SP`. The observation matrix A is extended by the rows of matrix B expressing that the k-th derivative discontinuities of |SP| at the interior knots must be zero. The corresponding weights of these additional rows are set to $1/p$. Iteratively we then have to determine the value of p such that

$$F(p) = \sum w_i [g(\tau_i) - |SP|(\tau_i)] = s.$$

We already know that the least-squares k-th degree polynomial corresponds to $p = 0$, and that the least-squares spline corresponds to $p = \infty$. The iteration process which is proposed here, makes use of rational interpolation. Since $F(p)$ is a convex and strictly decreasing function of p, it can be approximated by a rational function $R(p) = (Up + V)/(p + W)$. Three values of p (p_1, p_2, p_3) with corresponding values of $F(p)$ are used to calculate the new value of p such that $R(p) = s$. Convergence is guaranteed by taking $F_1 > 0$ and $F_3 < 0$.

code of fill-gsave: This piece of code does the following: add hh' to the matrix. In other words, add $h_i h_j$ at positions $(i+L, j+L)$ where the offset L is `it`. Now, since the matrix is symmetric, only half of the matrix needs to be stored. We store it at locations $(j+L, i-j)$, i.e. $g[i-j+5(j+L)]$.

6 The Completion algorithm

6.1 The Moebius Transform Let $z = \phi(s) = (s-1)/(s+1)$. Then $s = (1+z)/(1-z)$. The transformation of the function is the following:

$$g(z) = (\frac{1+s}{2}) \cdot f(s) \qquad f(s) = (1-z) \cdot g(z)$$

It is obvious that $|z| \leq 1$ if and only if $\Re s \geq 0$. Assume equality, namely $\exp(i\theta) = \phi(i\omega)$ where ω is real. This gives

$$\frac{\omega^2 - 1}{\omega^2 + 1} + i \frac{2\omega}{\omega^2 + 1} = \cos\theta + i \sin\theta.$$

This implies $\tan(\theta/2) = \frac{1}{\omega}$. We can also write $\tan\theta = \frac{2\omega}{\omega^2 - 1}$. The function from ω to θ maps the interval $[-\infty, \infty]$ to the interval $[0, 2\pi]$, it is monotone decreasing. The Endymion code has a *Moebius* function. It computes m such that $\tan m = 2\omega/(\omega^2 - 1)$ and $-\pi/2 \leq m \leq \pi/2$. Then $\theta = m + 2\pi$ if $\omega < -1$, $\theta = m + \pi$ if $-1 < \omega < 1$ and $\theta = m$ if $\omega > 1$.

6.2 From Line to Circle We describe here the method convert-to-circle from CvCircle. Given a PW function $f : x_i \to f_i$ it computes $g : y_i \to g_i$ as follows. If the boolean data-on-circle is true, data are already on the circle. We take $f_i = g_i$ and $y_i = x_i - \omega_0 + \pi$. Said otherwise: if x_i is symmetric around ω_0 then y_i is symmetric around π.

Otherwise, for each i, we let $\omega_a = x_i$. We define $\omega = (\omega_a - \omega_0)R$ where R is in the variable real-coef, this should be $1/d\omega$. If real-transform is true, then we use $\omega = (\omega_a/\omega_0 - \omega_0/\omega_a)R$ instead. Let θ be the Moebius transform of ω. Then y_i is θ (since θ is decreasing, we must re-order the sequence later on). The quantity f_i is multiplied by c^n where $c = (1 + i\omega)/2$. Here n is the value at inf-order. Normally this is one. The quantity f_i is also multiplied by $\exp(it)$ where $t = \alpha\omega_1 + \beta$ where $\omega_1 = (\omega_a - \omega_{n0})/d\omega_n$ (this is some kind of normalisation of ω). The two quantities α and β are normally zero.

If x_i is symmetric around ω_0 then y_i is symmetric around π, unless the real transformation is used; in that case, we call make-symmetric (this can either truncate the interval, or shift it).

7 Fourier series

7.1 The Toeplitz Matrix Assume $0 \leq \theta_1 \leq \theta_2 \leq \pi$, and that I is the set formed by $[\theta_1, \theta_2]$ and its symmetric part $[-\theta_2, -\theta_1]$. The set J is $[-\pi, \pi] - I$. Define

$$\langle f | g \rangle_J = \frac{1}{2\pi} \int_J \overline{f(e^{i\theta})} g(e^{i\theta}) d\theta.$$

We consider the case where $f = z^k$ and $g = z^j$. Then

$$U_{jk} = \frac{1}{2\pi} \int_J e^{i(j-k)\theta} d\theta = b_{j-k}.$$

The U matrix is called the Toeplitz matrix, because U_{ij} depends only on $i-j$. If V is the matrix corresponding to the set I, then $U+V$ is the identity matrix.

Easy computations show that

$$b_0 = 1 - \frac{\theta_2 - \theta_1}{\pi} \qquad b_j = b_{-j} = \frac{\sin(j\theta_1) - \sin(j\theta_2)}{j\pi}.$$

The `Toeplitz` data structure contains the parameters θ_1, θ_2, the b_j table, and the size of the table. There are methods that return U_{ij} or V_{ij} as a function of i and j.

If c_k are the coefficients of f then we have

$$\|f\|_I^2 + \|f\|_J^2 = \|f\| = \sum \|c_k\|^2$$

and

$$\|f\|_J^2 = \sum_{jk} \overline{c_k} \cdot c_j \cdot \langle e^{ik\theta} | e^{ij\theta} \rangle = \sum_{jk} \overline{c}_k \cdot c_j \cdot Ukj.$$

If $c_k = x_k + iy_k$, $U_{kj} = U_{jk} = T_{j-k}$ this is

$$\|f\|_J^2 = T_0 \|f\|^2 + 2 \sum_{j<k} (x_k x_j + y_k y_j) T_{k-j}.$$