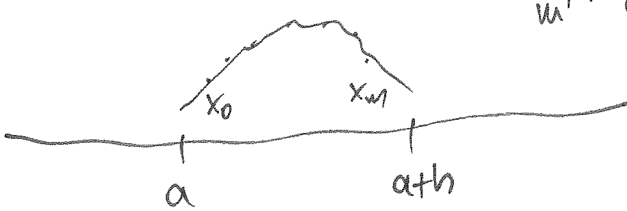
Splines

Recall

- polynomial interpolation works well for low degree polynomials
- high degrees not recommended.

What if we want more accuracy?

m^{th} degree polynomial

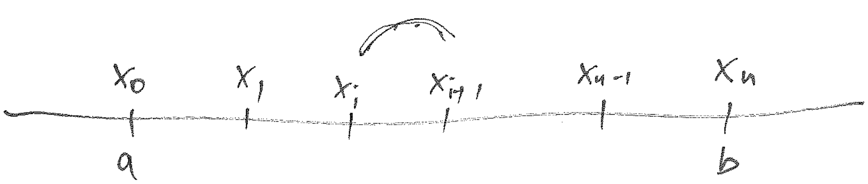


$$\text{error} = \frac{\omega(x)}{(m+1)!} f^{(m+1)}(\xi)$$

$$|\omega(x)| = |(x-x_0)\cdots(x-x_m)| < h^{m+1}$$

$$\boxed{\text{error} \le c \cdot h^{m+1}}$$

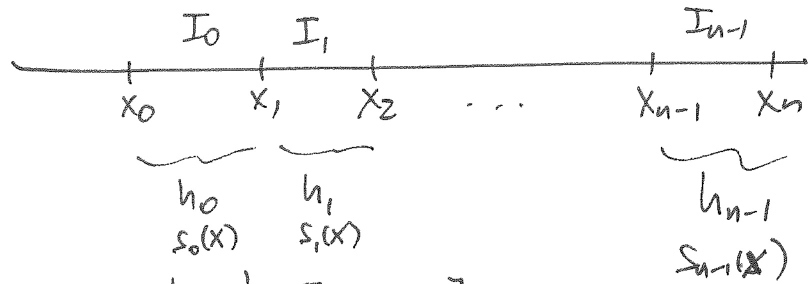
we hope



$$\text{total error} \le c \cdot h^{m+1}$$

<u>setup</u> $x_0 \ldots x_n$ <u>knots</u> $x_0 < x_1 < \ldots < x_n$

<u>Def:</u> $S_m^k = \{$ piecewise polynomials of degree m with k continuous derivatives $\}$



I_i = interval $[x_i, x_{i+1}]$

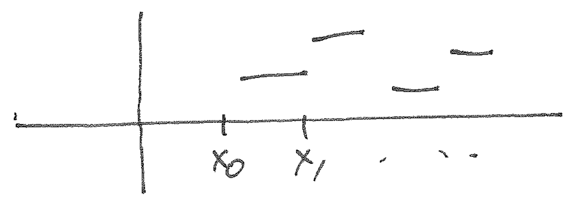
$h_i = x_{i+1} - x_i$

$s_i(x) = a_i + b_i (x - x_i) + c_i (x - x_i)^2 + \ldots$ on I_i

$s(x)$ = total function

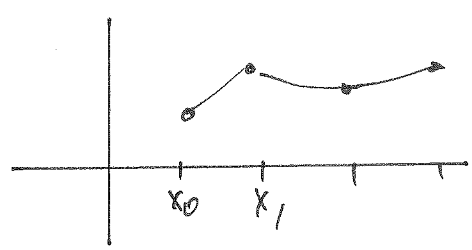
<u>Examples</u>

$\boxed{S_0^{-1}}$



piecewise constant
discontinuous

$\boxed{S_1^0}$



piecewise linear
continuous

$\boxed{S_3^1}$ cubic Hermite splines

$\boxed{S_3^2}$ cubic splines

How to describe a spline

$$\left\{\begin{array}{l} x_0 < x_1 < \dots < x_n \quad \text{knots} \\ k \\ m \\ a_0, \ b_0, \ c_0 \ \dots \\ a_1, \ b_1, \ c_1 \dots \\ \vdots \\ a_{n-1} \ b_{n-1} \ \dots \end{array}\right.$$

$$s_i(x) = a_i + b_i(x-x_i) + c_i\,(x-x_i)^3 + \dots$$

To evaluate $s(x)$

- determine interval I_i into which x falls
- evaluate appropriate $s_i(x)$

Dimension of S_m^k

$n+1$ points $x_0 \dots x_n$

n subintervals $I_0 \dots I_{n-1}$

$n-1$ interior points $x_1 \dots x_{n-1}$

number of coefficients: $n(m+1)$

constraints: $(n-1)(k+1)$

---

free parameters: $n(m-k) + k + 1$

examples:

S_0^{-1}	dim $= n$
S_1^0	$n+1$
S_3^1	$2n+2$
S_3^2	$n+3$

Note: in S^k_m we need $k < m$, otherwise this is a single polynomial

spline interpolation Given ξ_j interpolation points find spline which goes through these points.

we only consider $\xi_j = x_j$

Notation:

$$f_i = f(x_i)$$
$$f_i' = f'(x_i)$$
$$f_i'' = f''(x_i)$$

$\boxed{S^0_1}$

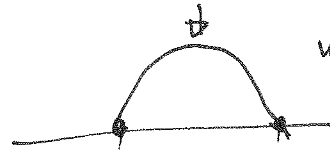
$$\left.\begin{array}{l} \dim = n+1 \\ f_0 \cdots f_n = (n+1) \text{ conditions} \end{array}\right\} \Rightarrow \text{unique solution}$$

"connect the dots"

$$\boxed{s_i(x) = f_i + \frac{f_{i+1} - f_i}{h_i}(x - x_i)}$$

Error on I_i, error $= \dfrac{\omega(x)}{2!} f''(\xi)$ max at midpoint value $\dfrac{h_i^2}{4}$

$$\omega(x) = (x - x_i)(x - x_{i+1})$$

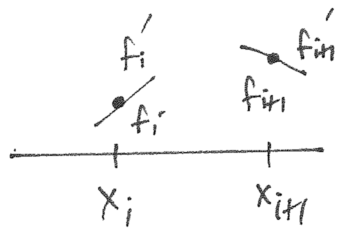


$$\boxed{\text{global error} \le \frac{h^2}{8} \max |f''|}$$

$\boxed{S_3^1}$

$\left.\begin{array}{l} \dim = 2n+2 \\ \left.\begin{array}{l} f_0 \cdots f_n \\ f_0' \cdots f_n' \end{array}\right\} \; 2n+2 \text{ conditions} \end{array}\right\} \Rightarrow$ unique solution

<u>algorithm</u> Hermite interpolation at x_i, x_{i+1}



$$
\begin{array}{c|ccccccc}
x_i & f_i & & & & & & \\
 & & f_i' & & & & & \\
x_i & f_i & & & \dfrac{f[x_i, x_{i+1}] - f_i'}{h_i} & & & \\
 & & f[x_i, x_{i+1}] & & & & \dfrac{f_{i+1}' + f_i' - 2f[x_i, x_{i+1}]}{h_i^2} & \\
x_{i+1} & f_{i+1} & & & \dfrac{f_{i+1}' - f[x_i, x_{i+1}]}{h_i} & & & \\
 & & f_{i+1}' & & & & & \\
x_{i+1} & f_{i+1} & & & & & &
\end{array}
$$

<u>Result</u>

$$s_i(x) = a_i + b_i (x - x_i) + c_i (x - x_i)^2 + d_i (x - x_i)^2 \underbrace{(x - x_{i+1})}_{= (x - x_i) - (x_{i+1} - x_i)}$$

$$= a_i + b_i (x - x_i) + [c_i - h_i d_i] (x - x_i)^2 + d_i (x - x_i)^3$$

where

$$\boxed{\begin{array}{l} a_i := f_i \\ b_i := f_i' \\ c_i = \dfrac{f[x_i, x_{i+1}] - f_i'}{h_i} \\ d_i := \dfrac{f_{i+1}' + f_i' - 2f[x_i, x_{i+1}]}{h_i^2} \end{array}}$$

error on I_i : $\frac{\omega(x)}{4!} f^{(4)}(\xi)$

$$|\omega(x)| = |(x-x_i)^2(x-x_{i+1})^2| \le \left(\frac{h^2}{4}\right)^2$$

global error $\le \frac{h^4}{384} f^{(4)}$

$\boxed{S_3^2}$ dim $= n+3$ } need 2 extra conditions

given $f_0 .. f_n$ $n+1$ conditions

possibilities

- prescribe f_0', f_n' clamped spline
- prescribe f_0'', f_n''
  (if $f_0'' = f_n'' = 0$ this is free or natural spline)
- estimate f_0' or f_0'' from nearby points
- not-a-knot spline
  3 continuous derivatives at x_1, x_{n-1}

$\boxed{\text{How To Compute } S_3^2}$

Method 1 set up coefficients $a_0 .. d_0, a_1 .. d_1 ..$
set up conditions

leads to $4n \times 4n$ system of equations
This works, but is not efficient.

Method 2 Use basis functions

B-splines ; we will look at these later

Method 3 Pretend we know $f_0' .. f_n'$

$$\boxed{\begin{aligned}
s_i(x) &= a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3 \\
s_i(x_i) &= a_i \\
s_i(x_{i+1}) &= a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 \\
s_i'(x) &= b_i + 2c_i(x-x_i) + 3d_i(x-x_i)^2 \\
s_i'(x_i) &= b_i \\
s_i'(x_{i+1}) &= b_i + 2c_i h_i + 3d_i h_i^2 \\
s_i''(x) &= 2c_i + 6d_i(x-x_i) \\
s_i''(x_i) &= 2c_i \\
s_i''(x_{i+1}) &= 2c_i + 6d_i h_i
\end{aligned}} \quad (*)$$

where

$$\boxed{\begin{aligned}
a_i &= f_i \\
b_i &= f_i' \\
c_i &= \frac{f[x_i, x_{i+1}] - f_i'}{h_i} - h_i d_i \\
d_i &= \frac{f_{i+1}' + f_i' - 2f[x_i, x_{i+1}]}{h_i^2}
\end{aligned}} \quad (**)$$

Continuity of s, s' is automatic.
Continuity of s'' must be enforced.

$$s_{i-1}''(x_i) = s_i''(x_i)$$

$$2c_{i-1} + 6\, d_{i-1} h_{i-1} = 2c_i$$

$$2\,\frac{f[x_{i-1},x_i] - f_{i-1}'}{h_{i-1}} \underbrace{- 2h_{i-1} d_{i-1} + 6 h_{i-1} d_{i-1}}_{= 4h_{i-1} \frac{f_i' + f_{i-1}' - 2f[x_{i-1},x_i]}{h_{i-1}^2}} = 2\,\frac{f[x_i,x_{i+1}] - f_i'}{h_i} - 2h_i d_i$$

times $\frac{h_{i-1} \cdot h_i}{2}$: $\quad h_i\left(f[x_{i-1},x_i] - f_{i-1}'\right) + 2h_i\left(f_i' + f_{i-1}' - 2f[x_{i-1},x_i]\right)$

$$= h_{i-1}\left(f[x_i,x_{i+1}] - f_i'\right) - h_{i-1}\left(f_{i+1}' + f_i' - 2f[x_i,x_{i+1}]\right)$$

$$f_{i-1}'(-h_i + 2h_i) + f_i'(2h_i + h_{i-1} + h_{i-1}) + f_{i+1}'(h_{i-1})$$

$$= -h_i f[x_{i-1},x_i] + 4h_i f[x_{i-1},x_i]$$
$$+ h_{i-1} f[x_i,x_{i+1}] + 2h_{i-1} f[x_i,x_{i+1}]$$

$$\boxed{\begin{aligned} &h_i f_{i-1}' + 2(h_i + h_{i-1}) f_i' + h_{i-1} f_{i+1}' \\ &\qquad = 3\left(h_i f[x_{i-1},x_i] + h_{i-1} f[x_i,x_{i+1}]\right)\end{aligned}}$$

$$\begin{pmatrix} h_1 & 2(h_0+h_1) & h_0 & & \\ & h_2 & 2(h_1+h_2) & h_1 & \\ & & \ddots & \ddots & \ddots \\ & & h_{n-1} & 2(h_{n-2}+h_{n-1}) & h_{n-2} \end{pmatrix} \begin{pmatrix} f_0' \\ \vdots \\ f_n' \end{pmatrix} = 3 \begin{pmatrix} h_0 f[x_1,x_2] + h_1 f[x_0,x_1] \\ \vdots \\ h_{n-2} f[x_{n-1},x_n] + h_{n-1} f[x_{n-2},x_{n-1}] \end{pmatrix}$$

$(n-1)$ eq. in $(n+1)$ unknowns

<u>clamped spline</u> f_0', f_n' given
reduce to $(n-1)\times(n-1)$

<u>free spline</u>

$$f_0'' = S_0''(x_0) = 2c_0 = 2\,\frac{f[x_0,x_1]-f_0'}{h_0} - 2h_0\,\frac{f_1'+f_0'-2f[x_0,x_1]}{h_0^2}$$

$$\boxed{2f_0' + f_1' = 3f[x_0,x_1] - \frac{h_0}{2} f_0''}$$

add this as first equation

$$\begin{pmatrix} 2 & 1 & & \\ h_1 & 2(h_0+h_1) & h_0 & \\ & & \ddots & \ddots \end{pmatrix} \begin{pmatrix} f_0' \\ f_1' \\ \vdots \end{pmatrix} = \begin{pmatrix} 3f[x_0,x_1] - \frac{h_0}{2} f_0'' \\ h_0 f[x_1,x_2] + h_1 f[x_0,x_1] \\ \vdots \end{pmatrix}$$

<u>not-a-knot</u> Homework

Method 4 — Pretend we know $f_0'' \dots f_n''$

First we need to find the interpolating polynomial for this data

$$s_i(x) = a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3$$

$$\left.\begin{array}{l} s_i(x_i) = f_i \\ s_i''(x_i) = f_i'' \\ s_i(x_{i+1}) = f_{i+1} \\ s_i''(x_{i+1}) = f_{i+1}'' \end{array}\right\} \Leftrightarrow \begin{array}{l} a_i = f_i \\ 2c_i = f_i'' \\ a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = f_{i+1} \\ 2c_i + 6h_i d_i = f_{i+1}'' \end{array}$$

$$\Rightarrow \boxed{\begin{array}{l} a_i = f_i \\ b_i = f[x_i, x_{i+1}] - \frac{h_i}{6}(2f_i'' + f_{i+1}'') \\ c_i = \frac{1}{2} f_i'' \\ d_i = \dfrac{f_{i+1}'' - f_i''}{6h_i} \end{array}} \quad (**)$$

this replaces (**) on page 7.

Continuity for s, s'' is automatic.
Continuity for s' means

$$s_{i-1}'(x_i) = s_i'(x_i)$$

$$b_{i-1} + 2c_{i-1}h_{i-1} + 3d_{i-1}h_{i-1}^2 = b_i$$

$$f[x_{i-1}, x_i] - \frac{h_{i-1}}{6}(2f_{i-1}'' + f_i'') + 2h_{i-1} \cdot \frac{1}{2} f_{i-1}'' + 3h_{i-1}^2 \frac{f_i'' - f_{i-1}''}{6h_{i-1}}$$

$$= f[x_i, x_{i+1}] - \frac{h_i}{6}(2f_i'' + f_{i+1}'')$$

<u>times 6</u>

$$6f[x_{i-1},x_i] - h_{i-1}(2f''_{i-1} + f''_i) + 6h_{i-1} f''_{i-1} + 3h_{i-1}(f''_i - f''_{i-1})$$
$$= 6f[x_i,x_{i+1}] - h_i(2f''_i + f''_{i+1})$$

$$f''_{i-1}(-2h_{i-1} + 6h_{i-1} - 3h_{i-1}) + f''_i(-h_{i-1} + 3h_{i-1} + 2h_i)$$
$$+ f''_{i+1}(h_i) = -6f[x_{i-1},x_i] + 6f[x_i,x_{i+1}]$$

$$\boxed{\begin{aligned} h_{i-1}f''_{i-1} + 2(h_{i-1}+h_i)f''_i + h_i f''_{i+1} &= 6\left(f[x_i,x_{i+1}] - f[x_{i-1},x_i]\right) \\ &= 6(h_{i-1}+h_i)\, f[x_{i-1},x_i,x_{i+1}] \end{aligned}}$$

$$\begin{pmatrix} h_0 & 2(h_0+h_1) & h_1 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \end{pmatrix} \begin{pmatrix} f''_0 \\ \vdots \\ \vdots \\ f''_n \end{pmatrix} = 6 \begin{pmatrix} f[x_1,x_2] - f[x_0,x_1] \\ \vdots \\ f[x_{n-1},x_n] - f[x_{n-2},x_{n-1}] \end{pmatrix}$$

<u>free spline</u> if f_0'', f_n'' are known, eliminate them

$\Rightarrow$ $(n-1)\times(n-1)$ system

<u>clamped spline</u> $f_0' = S_0'(x_0) = b_0 = f[x_0,x_1] - \frac{h_0}{6}(2f_0''+f_1'')$

<u>times 6</u>: $2h_0 f_0'' + h_0 f_1'' = 6f[x_0,x_1] - 6f_0'$

$$\begin{pmatrix} 2h_0 & h_0 & & \\ h_0 & 2(h_0+h_1) & h_1 & \\ & \ddots & \ddots & \end{pmatrix} \begin{pmatrix} f_0'' \\ \vdots \\ \vdots \end{pmatrix} = 6 \begin{pmatrix} f[x_0,x_1]-f_0' \\ f[x_1,x_2]-f[x_0,x_1] \\ \vdots \\ \vdots \end{pmatrix}$$

<u>not-a-knot</u> exercise

<u>Facts</u> ① Solving an $n\times n$ system of equations takes $O(n^3)$ ops

Solving a triangular system : $O(n^2)$

solving a banded system : $O(n)$

These equations are easy to solve, even for large n

② Solution is guaranteed by <u>diagonal dominance</u>

Error Estimates For Cubic Splines

clamped spline $\quad$ error $\leq \frac{5}{384} h^4 \max |f^{(4)}|$ $\quad$ (sharp)

natural spline $\quad$ error $= O(h^2)$ near ends

not-a-knot $\quad$? $\quad$ probably $O(h^4)$

Bessel splines: $\quad$ estimate f' by quadratic interpolation, then do Hermite splines

error $= O(h^3)$ $\quad$ because of error in derivatives

Problem with Splines

monotone data does not necessarily produce monotone splines

Curves

do splines in x, y separately

Thm Given partition Δ, function values f_j

(a) Assume g interpolates f_j at Δ, s = free spline interpolant.
Then
$$\int_a^b |s''(x)|^2 dx \le \int_a^b |g''(x)|^2 dx$$
with equality only if $g=s$.

(b) Assume g interpolates f_j, f_0', f_n', s = clamped spline
Then
$$\int_a^b |s''(x)| dx \le \int_a^b |g''(x)|^2 dx$$
with equality only if $g=s$.

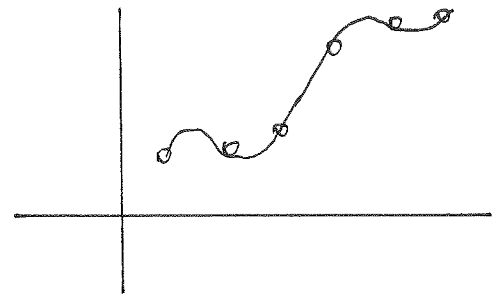
also true for periodic

proof: First we show that
$$\int_a^b s''(x)\left[g''(x) - s''(x)\right]dx = 0$$
integrate by parts:
$$\int_a^b s''(x)\left[g''(x)-s''(x)\right]dx = \underbrace{\cancel{s''(x)\left[g'(x) - s'(x)\right]\Big|_a^b}}_{=0 \text{ for (a) and (b)}} - \int_a^b s'''(x)\left[g'(x)-s'(x)\right]dx$$

s''' = piecewise constant
$$= -\sum_{i=1}^n c_i \int_{x_{i-1}}^{x_i} \left[g'(x) - s'(x)\right]dx$$
$$= -\sum_{i=1}^n c_i \left\{ g(x_i) - s(x_i) - g(x_{i-1}) + s(x_{i-1}) \right\} = 0$$
So, s'', $g''-s''$ are orthogonal
$$\Rightarrow \int |g''|^2 = \int |s''|^2 + \int |g''-s''|^2 \ge \int |s''|^2$$
equality only if $g''-s''=0 \Rightarrow g=s$

Math 561
Extra Topics in Splines

Tension Splines



Regular Splines are often not monotone for monotone data.

Think of a spline as a flexible rod, and imagine pulling at the ends. That will straighten out the wiggles.

Numerical Implementation

τ = tension parameter

A tension spline is a C^2 function which interpolates at (x_i, f_i), and on each subinterval satisfies

$$s^{(4)} - \tau^2 s'' = 0$$

$\tau = 0$ equation is $s^{(4)} = 0 \Rightarrow s$ = cubic polynomial

This leads to standard splines.

$\tau \to \infty$ $s'' \approx 0 \Rightarrow s \approx$ linear

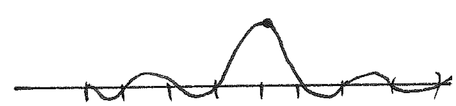
This produces something close to "connect the dots"

Algorithm is similar to standard cubic splines:
Assume f_i, m_i are known, then we can write down the solution in each subinterval (involves sinh, cosh)
Leads to a tridiagonal system for m_i

Basis Functions For Spline Spaces

Recall: for polynomials, we had
- regular basis $1, x, x^2, \ldots$
- Lagrange polynomial basis
  has property $p(x) = \sum f_i \, l_{n,i}(x)$
- Newton basis

For Splines, we have
- regular basis (de Boor calls this pp-form)
  on $[x_i, x_{i+1}]$, $s(x) = c_{i0} + c_{i1}(x-x_i) + c_{i2}(x-x_i)^2 + c_{i3}(x-x_i)^3$
- we could construct a counterpart to Lagrange basis,
  with property $s(x) = \sum f_i \, b_{ni}(x)$
  but basis functions b_{ni} would be global



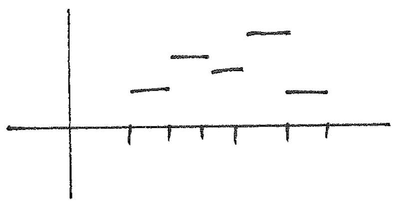
we want basis functions with short support.

Def: support of f = $\operatorname{supp} f = \overline{\{x : f(x) \neq 0\}}$
(closure of the set where f is nonzero)

We will consider some examples first before defining these B-splines in general.

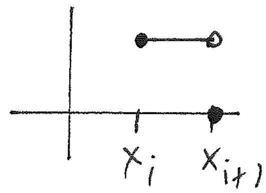
Initially we consider knots $\cdots x_{-2} < x_{-1} < x_0 < x_1 < x_2 < \cdots$
(infinite in both directions, so we don't have to deal with boundary)

S_0^{-1}



piecewise constant

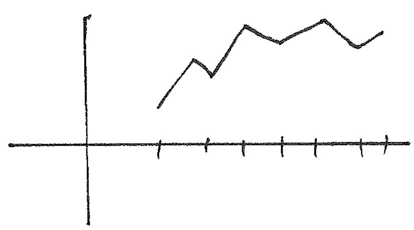
obvious basis is



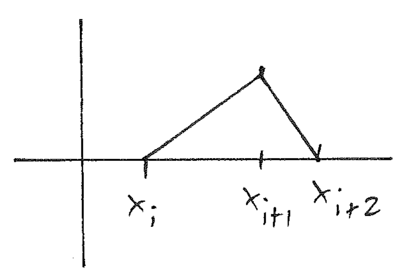
$$B_i^0 = \begin{cases} 1 & \text{if } x \in [x_i, x_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

<u>properties</u>:
- piecewise polynomial of degree 0
- right continuous, but not continuous
- support $[x_i, x_{i+1}]$
- $B_i^0(x) \geq 0 \quad \forall x, \qquad B_i^0(x) > 0$ on (x_i, x_{i+1})
- $\sum_i B_i^0(x) \equiv 1.$

S_1^0



piecewise linear



$$B_i^1 = \begin{cases} \dfrac{x - x_i}{x_{i+1} - x_i} & \text{on } [x_i, x_{i+1}) \\[2ex] \dfrac{x_{i+1} - x}{x_{i+1} - x_i} & \text{on } [x_{i+1}, x_{i+2}) \\[2ex] 0 & \text{otherwise} \end{cases}$$

properties:
- piecewise linear
- continuous
- support $[x_i, x_{i+2}]$
- $B_i^1(x) \geq 0 \ \forall x, \quad B_i^1(x) > 0$ on (x_i, x_{i+2})
- $\sum_i B_i^1(x) \equiv 1$

General Approach

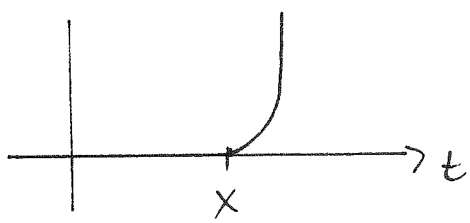
Definition 1 (recursive) $B_i^0(x) = \begin{cases} 1 & \text{if } x \in [x_i, x_{i+1}) \\ 0 & \text{otherwise} \end{cases}$

$$\boxed{B_i^k(x) = \frac{x - x_i}{x_{i+k} - x_i} B_i^{k-1}(x) + \frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}} B_{i+1}^{k-1}(x)}$$

Definition 2

$$\boxed{B_i^k(x) = (x_{i+k+1} - x_i)\,(\cdot - x)_+^k [x_i \ldots x_{i+k+1}]}$$

Explanation:



$$(t-x)_+^k = \begin{cases} (t-x)^k & \text{if } t \geq x \\ 0 & \text{otherwise} \end{cases}$$

To find the value of $B_i^k(x)$ for some fixed x, we consider $(t-x)_+^k$ and take the $(k+1)^{st}$ finite difference at $[x_i \ldots x_{i+k+1}]$.
The variable t disappears in the process, just as $\int f(x)dx$ or $\int f(t)dt$ are the same thing, so we write this as

$$(\cdot - x)_+^k$$

The $\cdot$ represents a variable whose name is unimportant.

The second definition is really not very intuitive, but it is useful for some proofs. We will show that both definitions are the same.

<u>Recall:</u> Leibniz formula (generalized product rule)

$$(f\cdot g)^{(n)} = \sum_{i=0}^{n} \binom{n}{i} f^{(i)} \cdot g^{(n-i)}$$

There is a corresponding formula for divided differences:

<u>Lemma</u> If $h = f\cdot g$, then

$$h[x_i \ldots x_{i+k}] = \sum_{j=0}^{k} f[x_i \ldots x_{i+j}] \cdot g[x_{i+j} \ldots x_{i+k}]$$

<u>Theorem</u> Definitions 1, 2 of $B_i^k(x)$ are the same

<u>Proof:</u> This is easy to verify directly for $k=0$
If we can show that B_i^k from def. 2 (divided difference) satisfies the recursion relation from def. 1, we are done.

Observe that for $k \geq 1$,

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

so

$$\begin{aligned}
\frac{1}{x_{i+k+1}-x_i} B_i^k(x) &= (\cdot - x)_+^k [x_i \ldots x_{i+k+1}] \\
&= \underbrace{\boxed{(\cdot - x)[x_i]}}_{x_i - x} (\cdot - x)_+^{k-1} [x_i \ldots x_{i+k+1}] \\
&+ \underbrace{\boxed{(\cdot - x)[x_i, x_{i+1}]}}_{1} (\cdot - x)_+^{k-1} [x_{i+1} \ldots x_{i+k+1}] \\
&+ \underbrace{\boxed{(\cdot - x)[x_i, x_{i+1}, x_{i+2}]}}_{0} (\cdot - x)_+^{k-1} [x_{i+2} \ldots x_{i+k+1}] \\
&+ \ldots
\end{aligned}$$

$$\frac{1}{x_{i+k+1}-x_i} B_i^k(x) = (x_i - x)(\cdot - x)_+^{k-1}[x_i \ldots x_{i+k+1}]$$
$$+ (\cdot - x)_+^{k-1}[x_{i+1} \ldots x_{i+k+1}]$$

$$= (x_i - x)\, \frac{(\cdot - x)_+^{k-1}[x_{i+1} \ldots x_{i+k+1}] - (\cdot - x)_+^{k-1}[x_i \ldots x_{i+k}]}{x_{i+k+1} - x_i}$$
$$+ (\cdot - x)_+^{k-1}[x_{i+1} \ldots x_{i+k+1}]$$

<u>multiply through by $(x_{i+k+1} - x_i)$</u>

$$B_i^k(x) = \Big[(x_i - x) + (x_{i+k+1} - x_i) \Big] (\cdot - x)_+^{k-1}[x_{i+1} \ldots x_{i+k+1}]$$
$$+ (x - x_i)(\cdot - x)_+^{k-1}[x_i \ldots x_{i+k}]$$

$$= \frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}} \underbrace{(x_{i+k+1} - x_{i+1})(\cdot - x)_+^{k-1}[x_{i+1} \ldots x_{i+k+1}]}_{B_{i+1}^{k-1}(x)}$$

$$+ \frac{x - x_i}{x_{i+k} - x_i} \underbrace{(x_{i+k} - x_i)(\cdot - x)_+^{k-1}[x_i \ldots x_{i+k}]}_{B_i^{k-1}(x)} \quad ∎$$

Properties of B_i^k

ⓐ piecewise polynomial of degree k

pf: induction based on recursion formula

ⓑ B_i^k form a basis

pf: requires some more lemmas; we will skip the proof

ⓒ $(k-1)$ times continuously differentiable

pf: B_i^k is C^∞ except at knots.

Use induction based on recursion formula for the knots

ⓓ support $[x_i, x_{i+k+1}]$

pf: directly from divided difference formula,

or by induction based on recursion formula

ⓔ $\sum_i B_i^k(x) \equiv 1$

proof: by induction based on recursion formula

$$\sum_i B_i^k(x) = \sum_i \frac{x-x_i}{x_{i+k}-x_i} B_i^{k-1}(x) + \boxed{\sum_i \frac{x_{i+k+1}-x}{x_{i+k+1}-x_{i+1}} B_{i+1}^{k-1}(x)} \qquad j=i+1$$

$$= \sum_j \frac{x_{j+k}-x}{x_{j+k}-x_j} B_j^{k-1}(x)$$

$$= \sum_i \underbrace{\left[\frac{x-x_i}{x_{i+k}-x_i} + \frac{x_{i+k}-x}{x_{i+k}-x_i}\right]}_{=1} B_i^{k-1}(x) = \sum_i B_i^{k-1}(x)$$

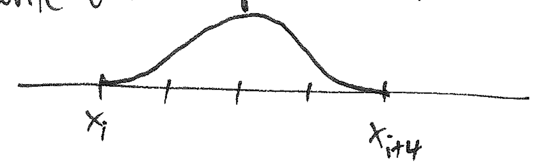
ⓕ $B_i^k(x) \geq 0 \;\; \forall x, \quad B_i^k(x) > 0$ on (x_i, x_{i+k+1})

pf: obvious from recursion formula

Doing Spline Interpolation With B_i^3

set up $\quad s(x) = \sum_i c_i^3 B_i^3(x)$

write out a system of equations $\quad \sum c_i^3 B_i^3(x_j) = f(x_j)$



$B_i^3(x_j) \neq 0$ only for $j = i+1, i+2, i+3$

leads to tridiagonal system

Advantage over other algorithms

1. we can use interpolation points ξ_j different from knots x_j

   Thm if we choose ξ_j so that $\xi_j \in \operatorname{supp} B_j^k$, ξ_j distinct, there is a unique solution.

2. we can do least squares problems this way
3. we can accommodate more general splines (see below)

Evaluating B-splines

Suppose we have found $s(x) = \sum c_i^k B_i^k(x)$

How do we evaluate that for given x?

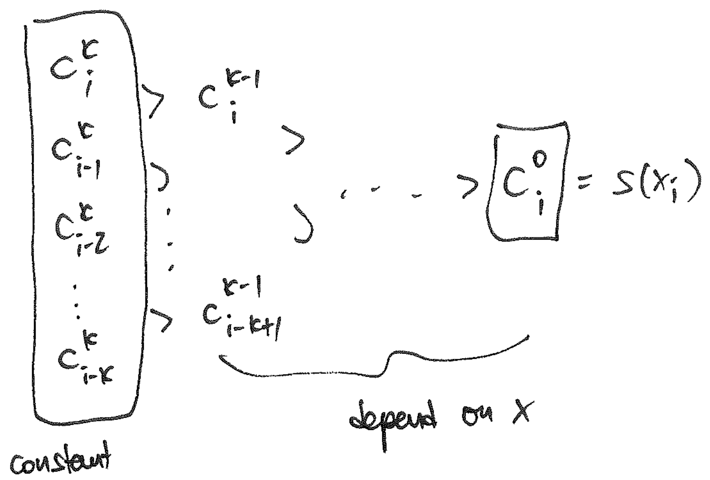
Trick: Consider $\quad s(x) = \sum_i c_i^k(x) B_i^k(x)$

verify
$$\boxed{\sum_i c_i^k(x) B_i^k(x) = \sum_i c_i^{k-1}(x) B_i^{k-1}(x)}$$

if we define
$$c_i^{k-1}(x) = \frac{x - x_i}{x_{i+k} - x_i} c_i^k(x) + \frac{x_{i+k} - x}{x_{i+k} - x_i} c_{i-1}^k(x)$$

proof: substitute recursion formula.

we can build a triangular table



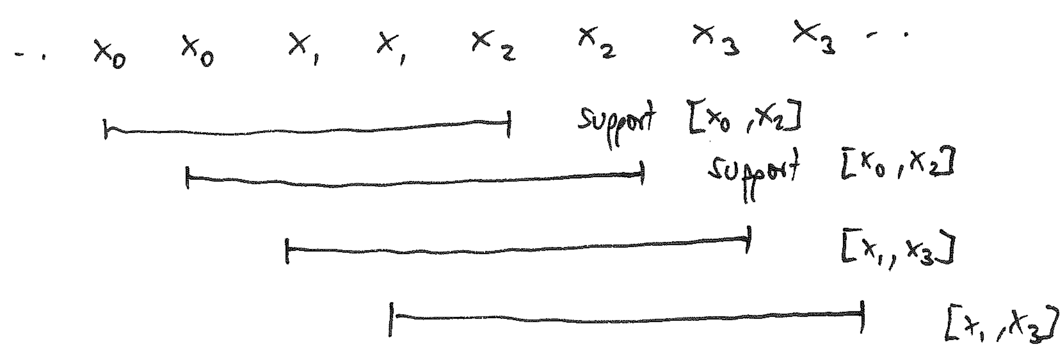
Generalization

Knots can be repeated. Every repetition lowers smoothness by one.

Example: $x_0 \quad x_1 \quad x_2\, x_2 \quad x_3 \quad x_3 \quad x_3 \quad x_4 \quad x_5$

$\ldots \quad C^2 \quad C^1 \quad C^0 \quad C^2 \quad \ldots$

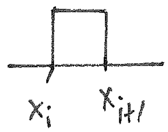
Hermite splines



Start with $\quad x_i \;\; x_i \;\; x_{i+1} \;\; x_{i+1} \;\; x_{i+2} \; \cdots$

renumber $\quad \xi_i \;\; \xi_{i+1} \;\; \xi_{i+2} \;\; \xi_{i+3} \;\; \xi_{i+4} \; \cdots$ ← used to number B_i^k

$B_i^0 = 0 \qquad$ interval $[x_i, x_i]$

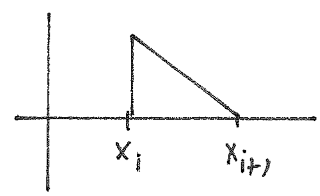
$B_{i+1}^0 =$  $\qquad [x_i, x_{i+1}]$

$B_{i+2}^0 = 0 \qquad [x_{i+1}, x_{i+1}]$

⋮

$$B_i^1(x) = \cancel{\frac{x-\xi_i}{\xi_{i+1}-\xi_i}}_{=0} B_i^0(x) + \frac{\xi_{i+2}-x}{\xi_{i+2}-\xi_{i+1}} B_{i+1}^0(x) = \frac{x_{i+1}-x}{x_{i+1}-x_i} B_{i+1}^0(x)$$

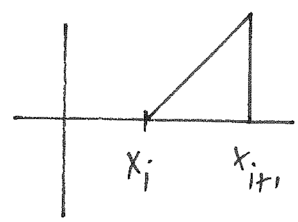
on $[x_i, x_i, x_{i+1}]$



<u>note</u>: x_i repeated ⟹ less smoothness at x_i

likewise $\quad B_{i+1}^1(x) = \dfrac{x-x_i}{x_{i+1}-x_i} B_{i+1}^0$

on $[x_i, x_{i+1}, x_{i+1}]$

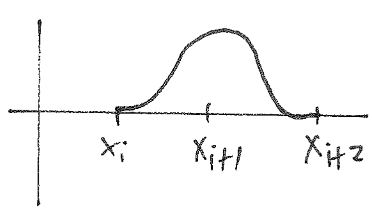


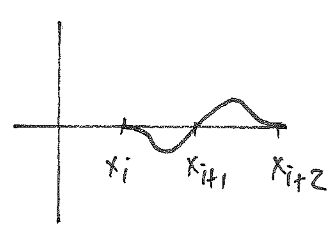
I gave up at this point.

B_i^3 would be on $[x_i, x_i, x_{i+1}, x_{i+1}, x_{i+2}]$ / C^2 at x_{i+2}, C^1 at x_i, x_{i+2}

$B_{i+1}^3 \qquad [x_i, x_{i+1}, x_{i+1}, x_{i+2}, x_{i+2}]$ — C^2 at x_i, C^1 at x_{i+1}, x_{i+2}

Different from the usual basis (probably)





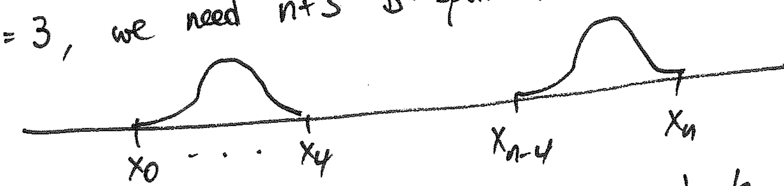
Endpoints

The usual way to handle boundaries is by putting extra knots at the ends.

Example: $k=3$ cubic splines

Thm with knots $x_0 \ldots x_n$, the dimension of the spline space on $[x_0, x_n]$ is $n+k$.

For $k=3$, we need $n+3$ B-splines.

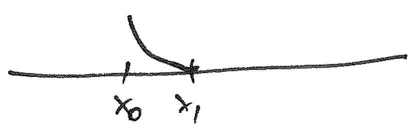


interior B-splines

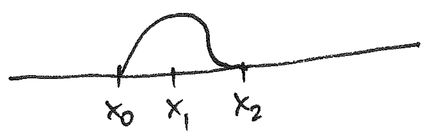
There are $n-3$ interior B-splines. we need 6 extra functions.

Put 3 extra knots at left

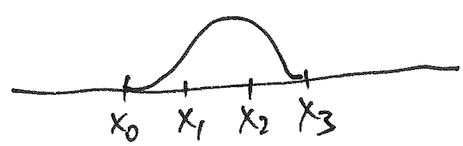
3 extra functions



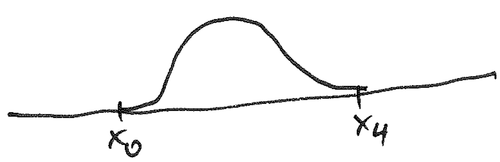
$[x_0, x_0, x_0, x_0, x_1]$
discontinuous at x_0



$[x_0, x_0, x_0, x_1, x_2]$
continuous but not differentiable at x_0



$[x_0, x_0, x_1, x_2, x_3]$
C^1 but not C^2 at x_0

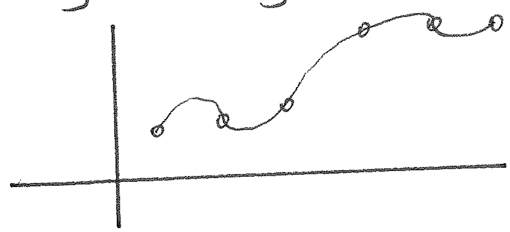


$[x_0, x_1, x_2, x_3, x_4]$
C^2; this is first interior spline

Same at right end.

Application in CAD/CAM

It is tempting to design a computer graphics system where user can move around the interpolation points (x_i, f_i). Unfortunately that may result in extra wiggles in the curve:



Instead, design it so that the control points are (x_i, c_i^k).

Theorem Cubic B-splines are variation diminishing.

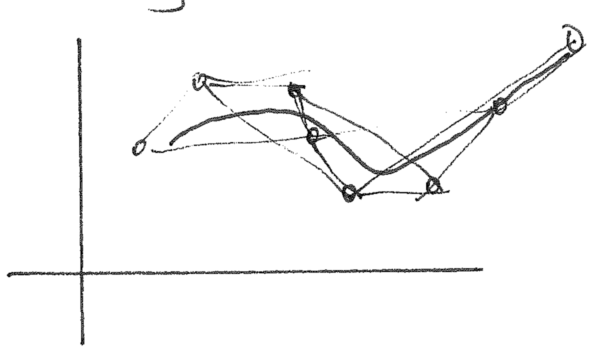
That is, the curve $\sum c_i^k B_i^k(x)$ has no more monotonicity changes than the sequence $\{c_i^k\}$.

$\{c_i^k\}$ increasing / decreasing $\Rightarrow$ curve increasing / decreasing

$\{c_i^k\}$ convex / concave $\Rightarrow$ curve convex / concave

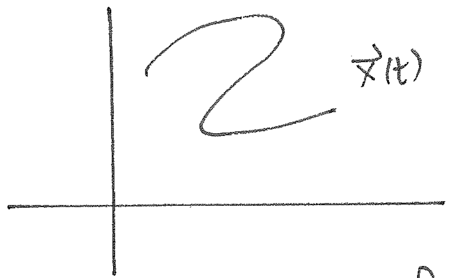
Theorem (Bézier)

Put a quadrilateral around every group of four successive control points. The curve will run inside the resulting quadrilaterals.



More general curves

Parametric curves $\vec{x}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$



are more general than $y = f(x)$.

To implement them, divide t-axis $t_0 < t_1 < \dots < t_n$

and use B-splines for both $x(t), y(t)$.

Overall effect: $\vec{s}(t) = \sum \vec{c}_i^{\,k} B_i^k(t)$

Bernstein Polynomials

Yet another basis for n^{th} degree polynomials on $[a, b]$ is

$$B_i^n(x) = \binom{n}{i} \frac{(b-x)^i (x-a)^{n-i}}{(b-a)^n}$$

On $[0,1]$: $$B_i^n(x) = \binom{n}{i} (1-x)^i x^{n-i}$$

Obvious properties:

$$\sum_i B_i^n(x) = 1$$

$$0 \leq B_i^n(x) \leq \binom{n}{i}$$

$$0 < B_i^n(x) \quad \text{on } (0,1)$$

we can represent a polynomial as

$$p(x) = \sum c_i^n B_i^n(x)$$

Fact: This representation is also variation diminishing.

This is a basis of polynomials that has very similar properties to B-splines.