X-Splines

Flexible and user-friendly

Jordi G. H. ⟨jordigh@octave.org⟩

Also @JordiGH@mathstodon.xyz on the Fediverse/Mastodon

August 24, 2018

Outline

Splines in general

X-splines

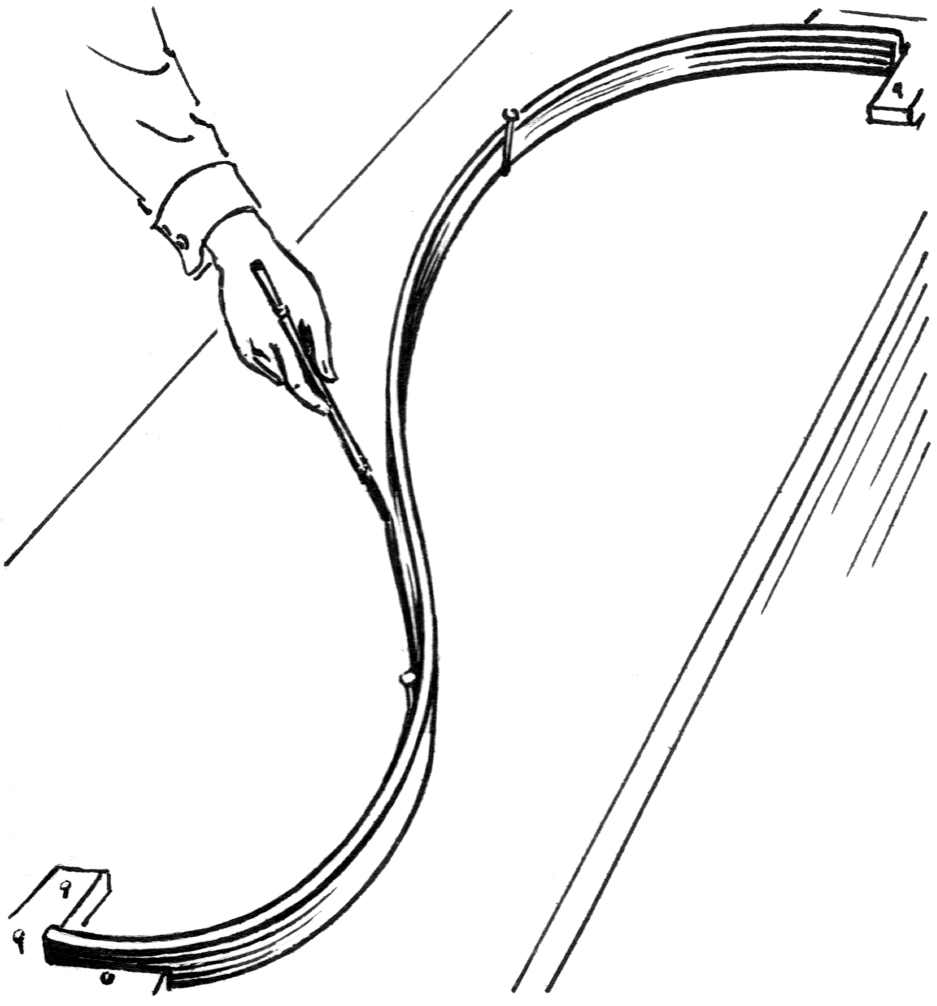
X-tended X-splines

General X-splines

Outline

What is a spline?

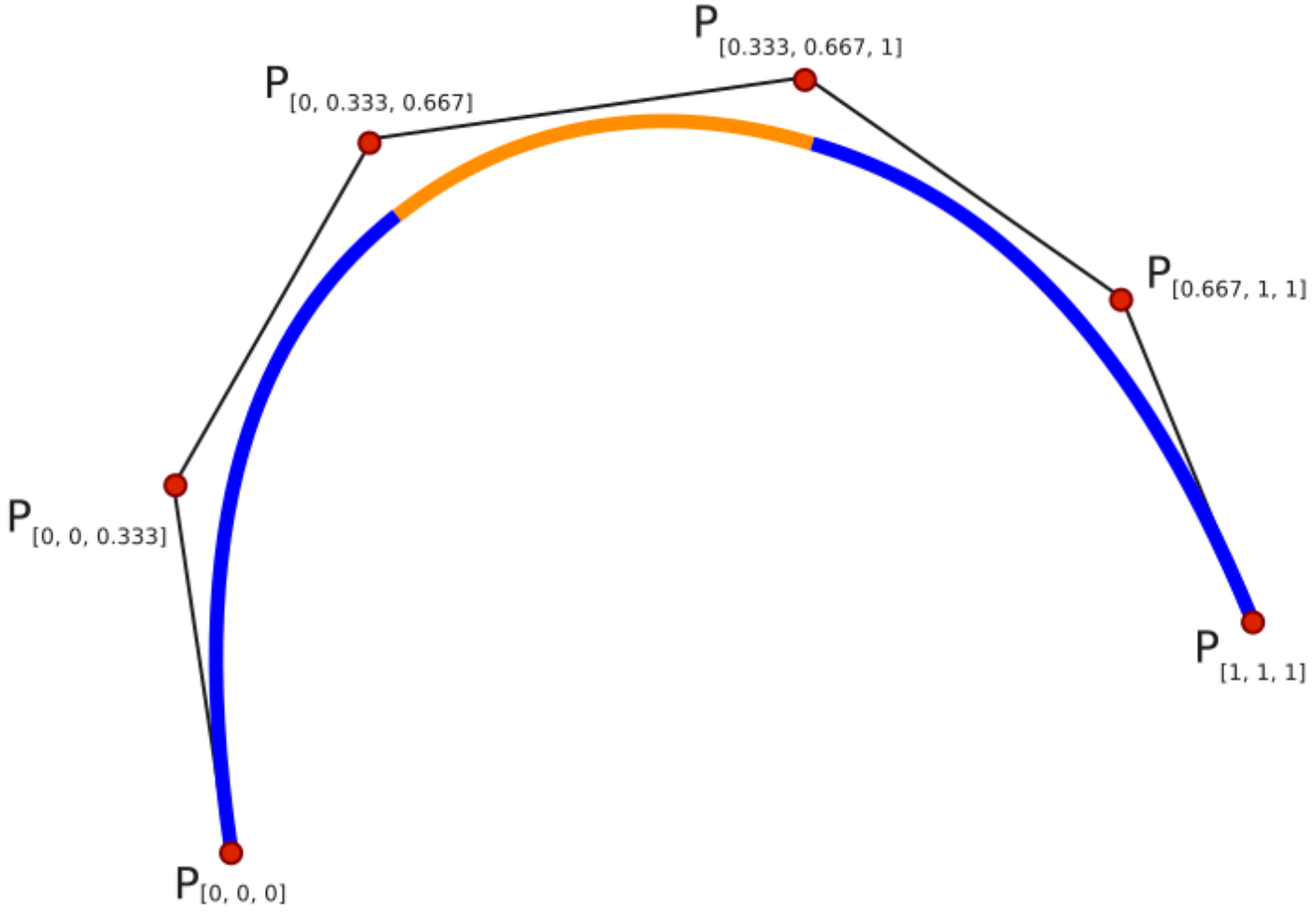
What is a spline?



Historically: a mechanical device for drawing curves

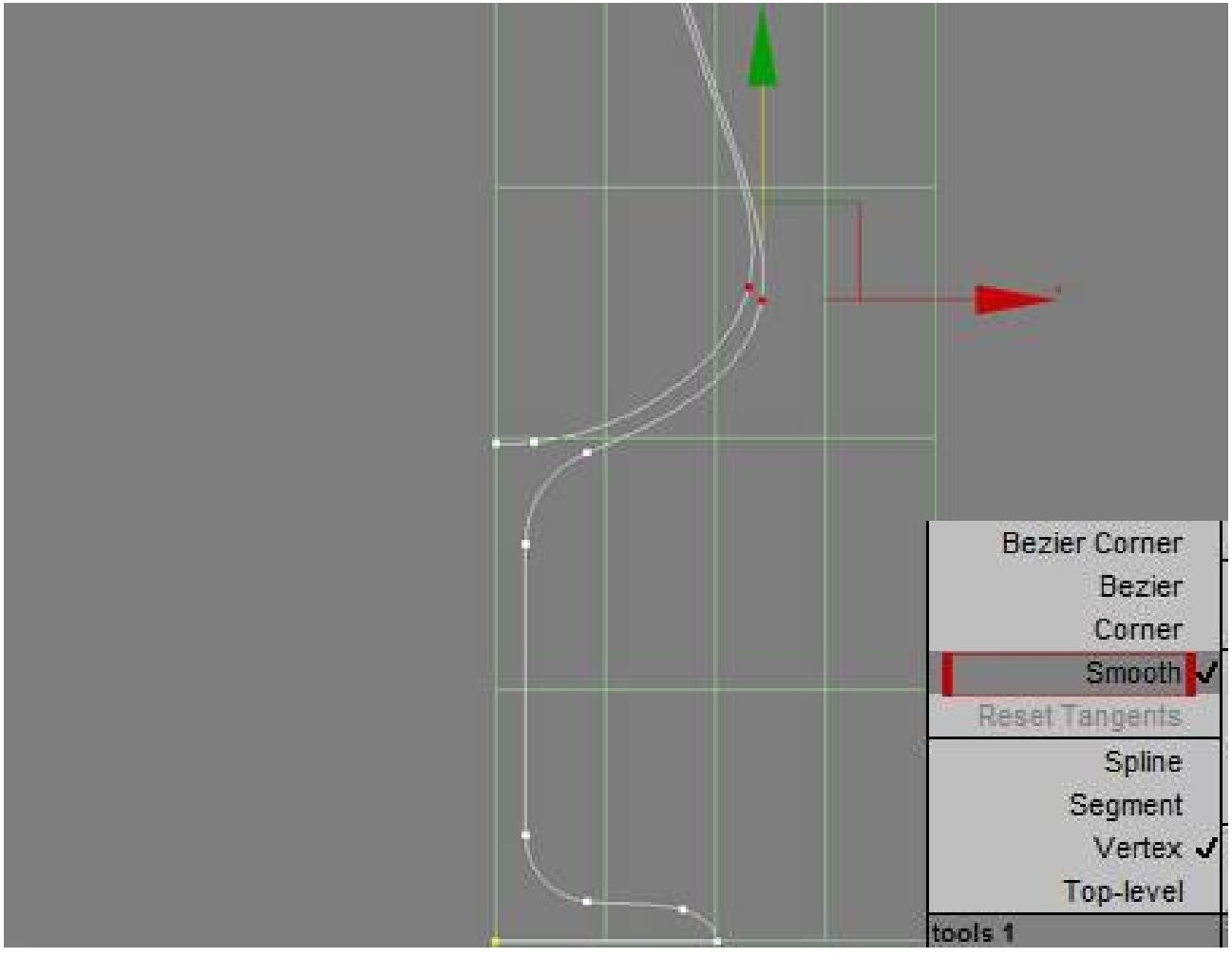
What is a spline?

You probably know them from computer graphics or CAD.



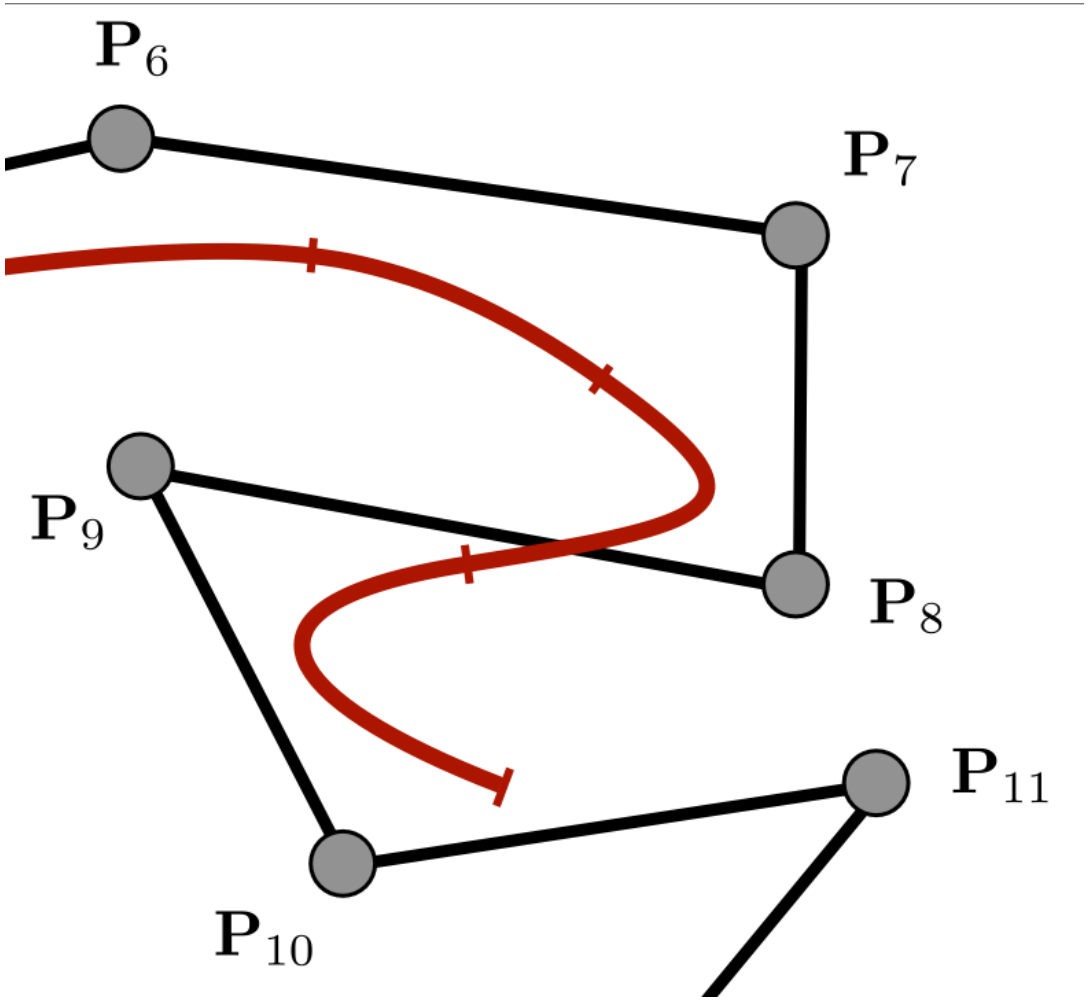
What is a spline?

You probably know them from computer graphics or CAD.



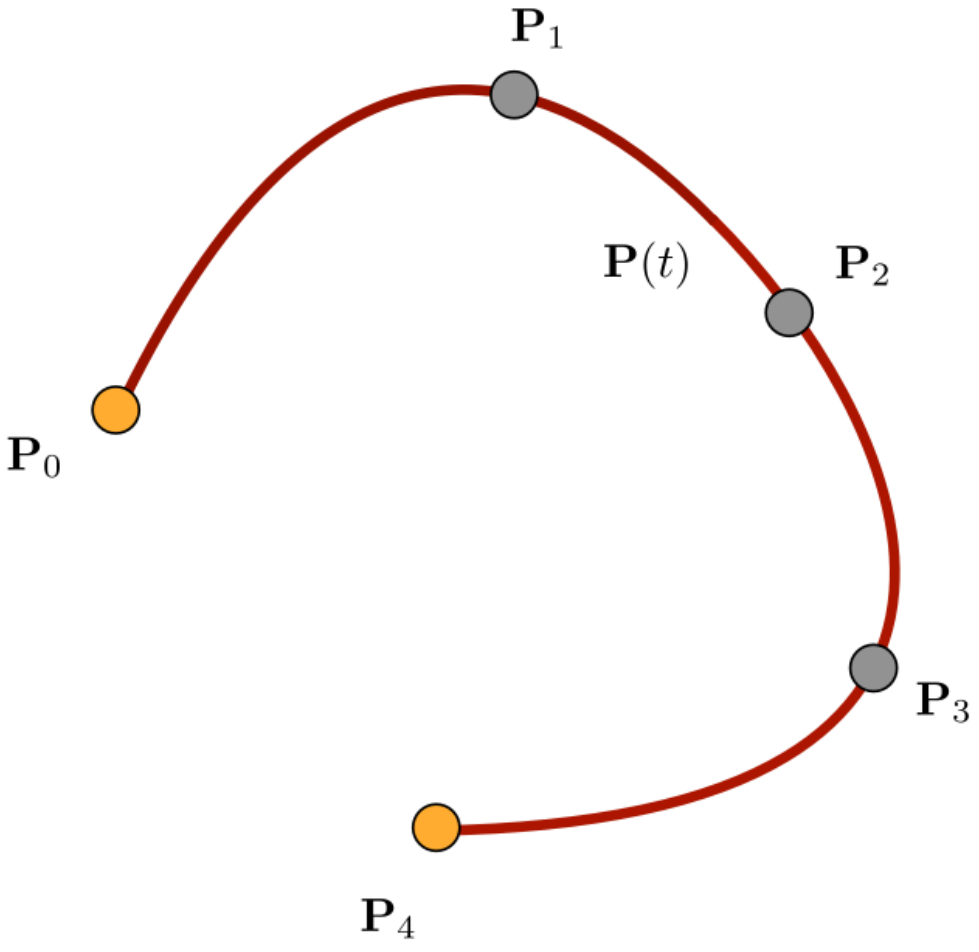
Guiding example: b-splines

These splines *approximate*.



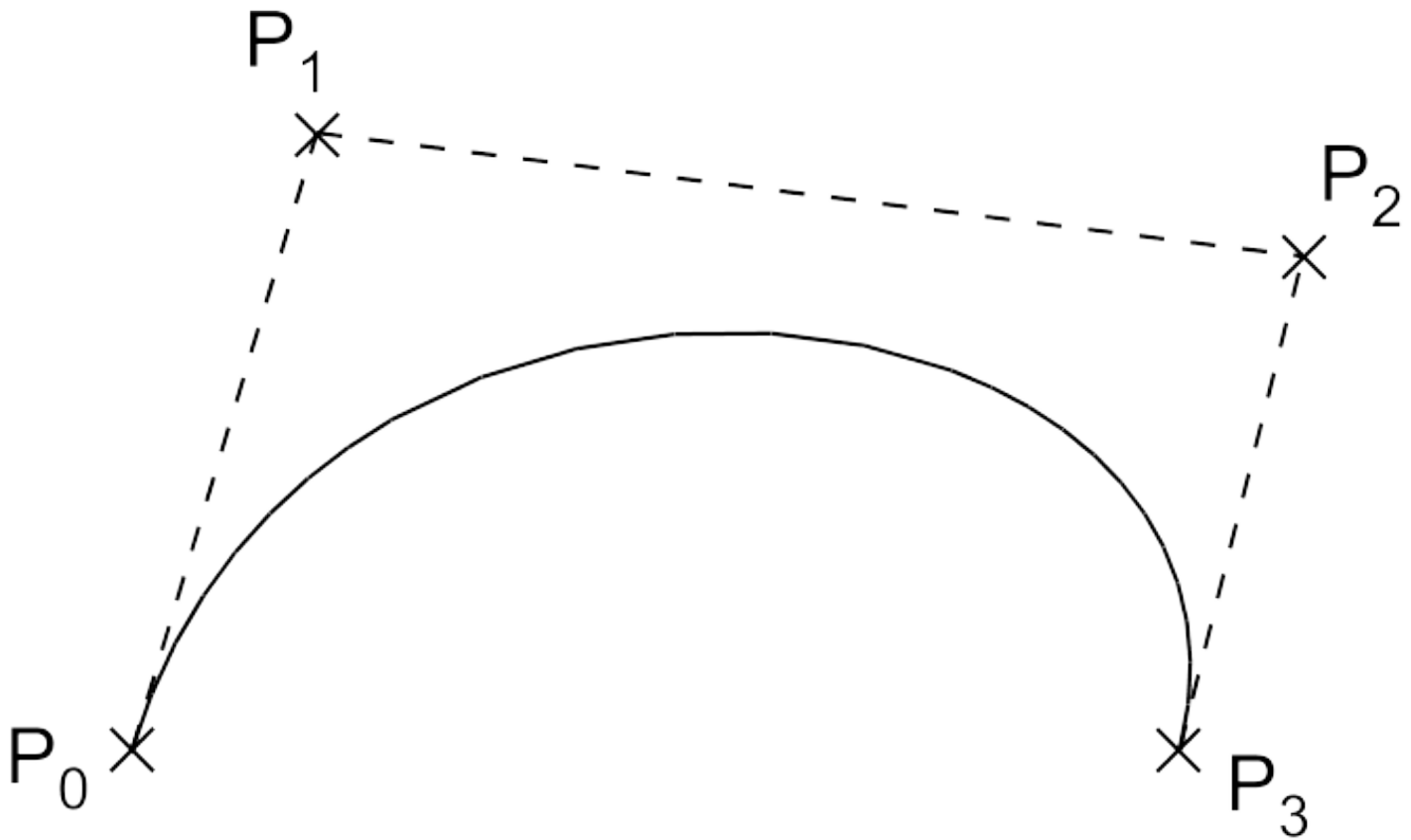
Guiding example: Catmull-Rom splines

These splines *interpolate*.



Another famous example: Bézier curves

Closely related to B-splines



What makes a spline a spline?

General formula:

What makes a spline a spline?

General formula:
For each *node* in 2d (or 3d) space there is a corresponding *blending function*:

$$
\begin{aligned}
(x_0, y_0) = P_0 &\longleftrightarrow F_0(t) \\
(x_1, y_1) = P_1 &\longleftrightarrow F_1(t) \\
&\vdots \\
(x_n, y_n) = P_n &\longleftrightarrow F_n(t)
\end{aligned}
$$

What makes a spline a spline?

General formula:
For each *node* in 2d (or 3d) space there is a corresponding *blending function*:

$$\begin{aligned}(x_0, y_0) = P_0 &\longleftrightarrow F_0(t)\\ (x_1, y_1) = P_1 &\longleftrightarrow F_1(t)\\ &\vdots\\ (x_n, y_n) = P_n &\longleftrightarrow F_n(t)\end{aligned}$$

and we form a *linear combination*

$$\begin{aligned}C(t) &= \sum_{k=0}^{n} F_k(t) P_k\\ &= F_0(t)P_0 + F_1(t)P_1 + \cdots + F_n(t)P_n.\end{aligned}$$

What makes a spline a spline?

General formula:
For each *node* in 2d (or 3d) space there is a corresponding *blending function*:

$$\begin{aligned}(x_0, y_0) = P_0 &\longleftrightarrow F_0(t)\\ (x_1, y_1) = P_1 &\longleftrightarrow F_1(t)\\ &\vdots\\ (x_n, y_n) = P_n &\longleftrightarrow F_n(t)\end{aligned}$$

and we form a *linear combination*

$$\begin{aligned}C(t) &= \sum_{k=0}^{n} F_k(t) P_k\\ &= F_0(t)P_0 + F_1(t)P_1 + \cdots + F_n(t)P_n.\end{aligned}$$

The parameter t runs continuously from 0 to n.

Convexity (optional)

Convexity (optional)

The blending functions $F_k(t)$ must satisfy

Convexity (optional)

The blending functions $F_k(t)$ must satisfy

- $1 = \sum_{k=0}^{n} F_k(t)$

Convexity (optional)

The blending functions $F_k(t)$ must satisfy

- $1 = \sum_{k=0}^{n} F_k(t)$ (technical jargon: *partition of unity*)

Convexity (optional)

The blending functions $F_k(t)$ must satisfy

- $1 = \sum_{k=0}^{n} F_k(t)$ (technical jargon: *partition of unity*)
- $0 \leq F_k(t)$

Convexity (optional)

The blending functions $F_k(t)$ must satisfy

- $1 = \sum_{k=0}^{n} F_k(t)$ (technical jargon: *partition of unity*)
- $0 \leq F_k(t)$

This implies every point on the spline curve

$$C(t) = \sum_{k=0}^{n} F_k(t) P_k$$

Convexity (optional)

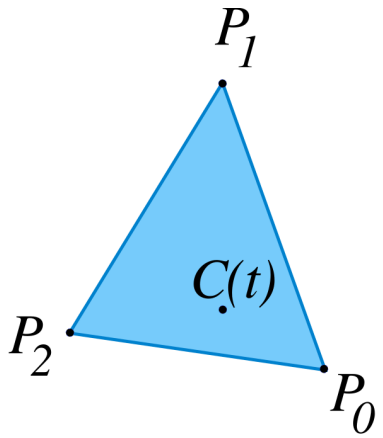
The blending functions $F_k(t)$ must satisfy

- $1 = \sum_{k=0}^{n} F_k(t)$ (technical jargon: *partition of unity*)
- $0 \leq F_k(t)$

This implies every point on the spline curve

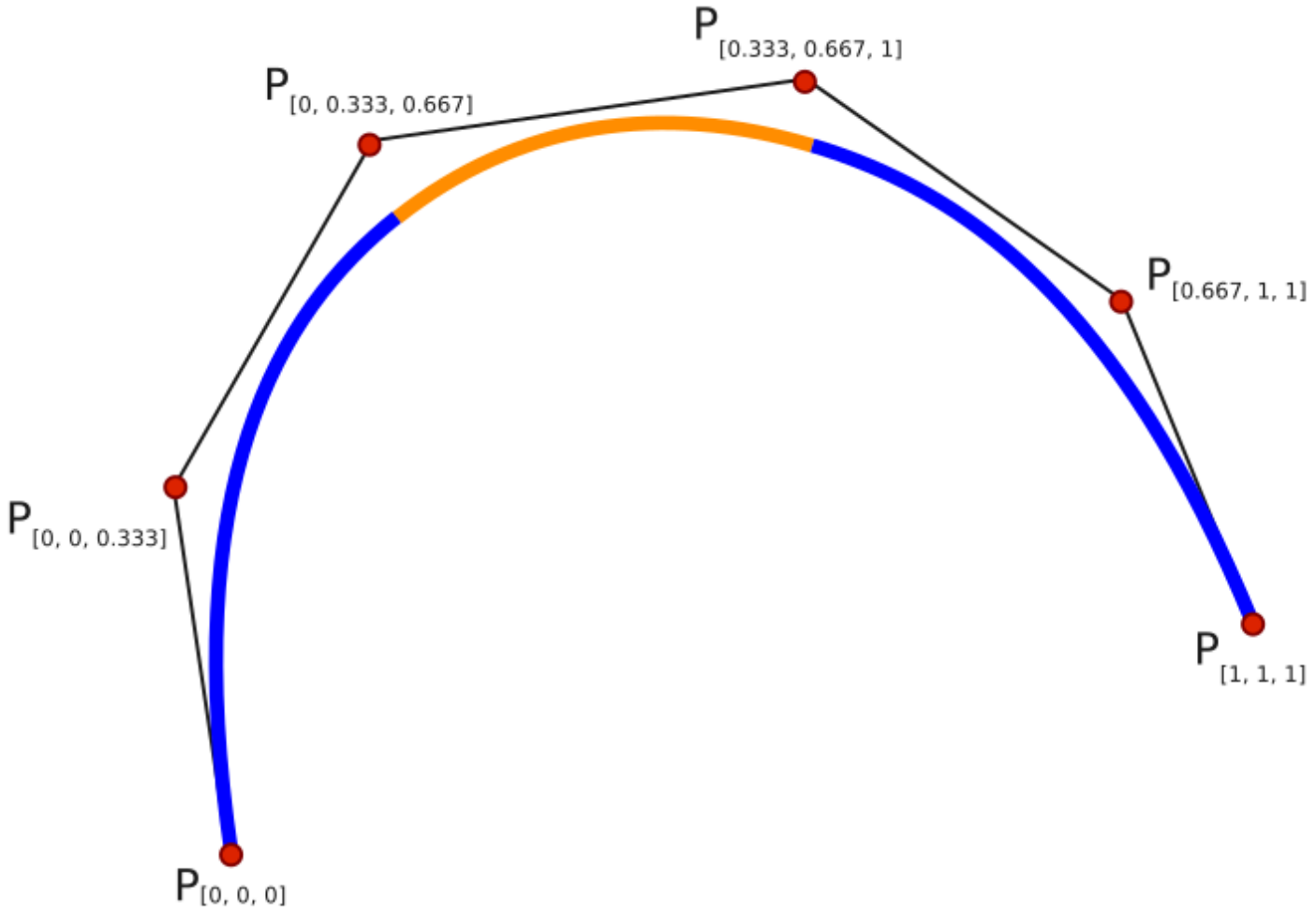
$$C(t) = \sum_{k=0}^{n} F_k(t) P_k$$

is a *convex combination* of the spline nodes.



Convexity

Remember?

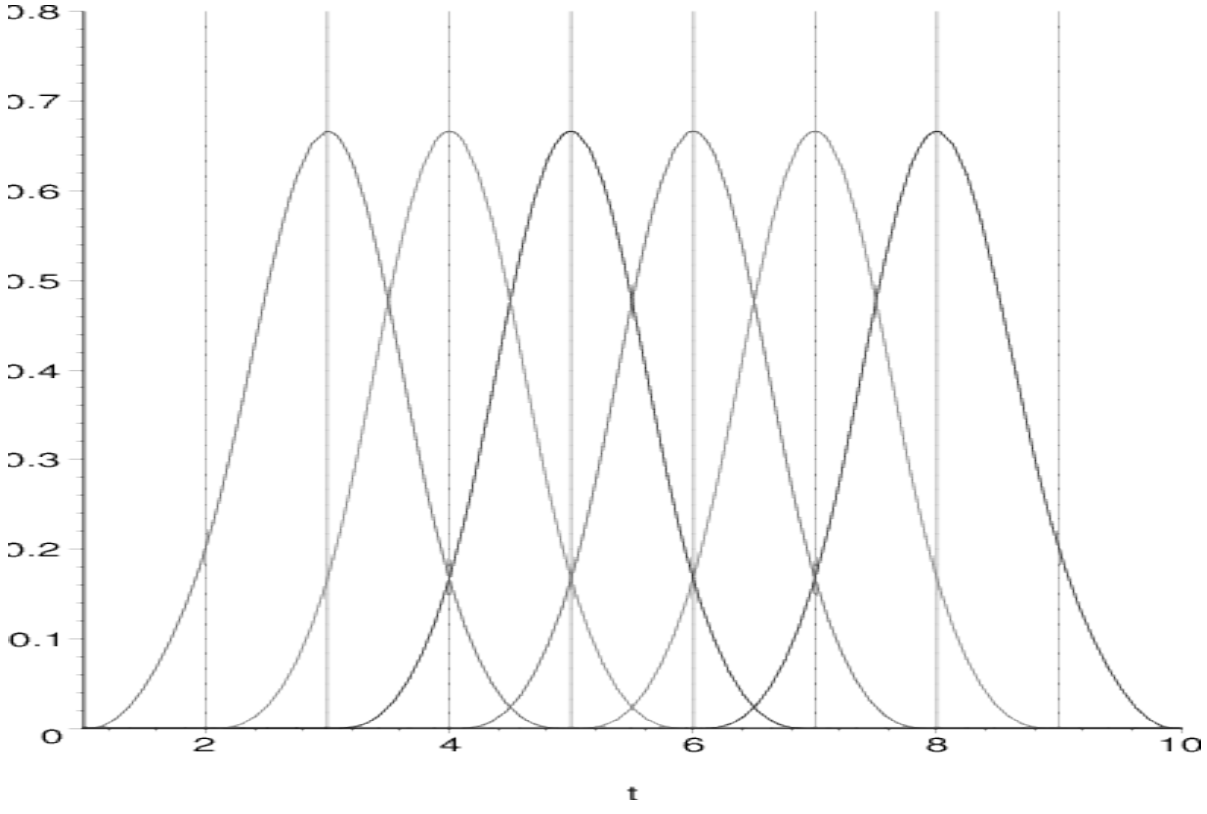


Smooth and steady

Splines must be smooth (not spiky) and steady (not too wobbly)

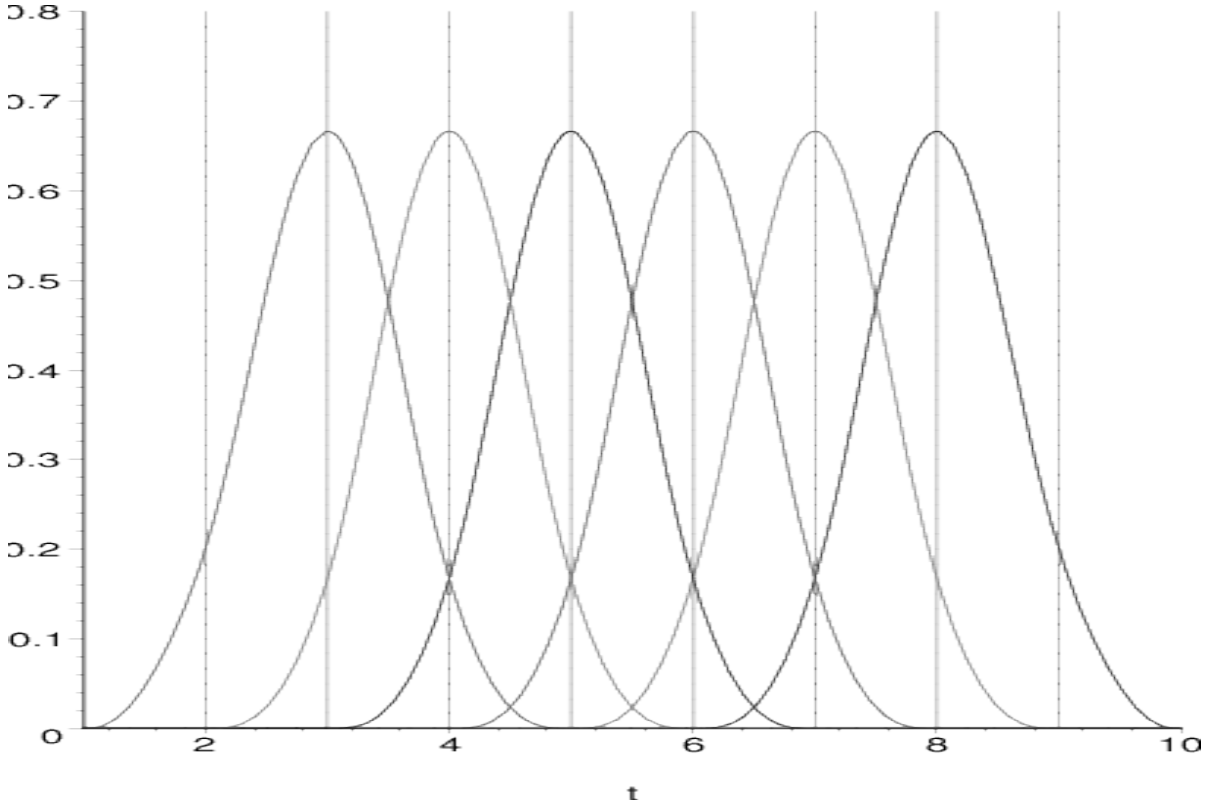
Smooth and steady

Splines must be smooth (not spiky) and steady (not too wobbly)
Therefore, blending functions $F_k(t)$ must look like this:



Smooth and steady

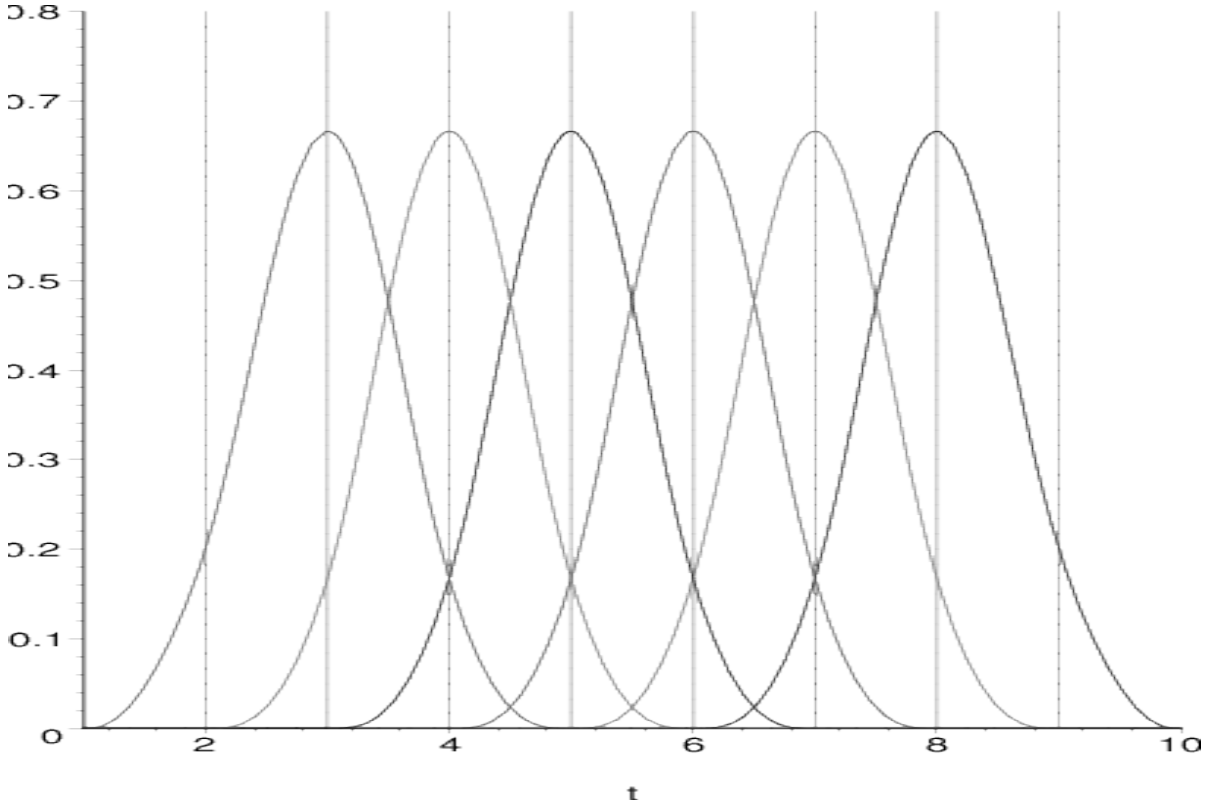
Splines must be smooth (not spiky) and steady (not too wobbly)
Therefore, blending functions $F_k(t)$ must look like this:



- Bump-shaped (smoothness)

Smooth and steady

Splines must be smooth (not spiky) and steady (not too wobbly)
Therefore, blending functions $F_k(t)$ must look like this:



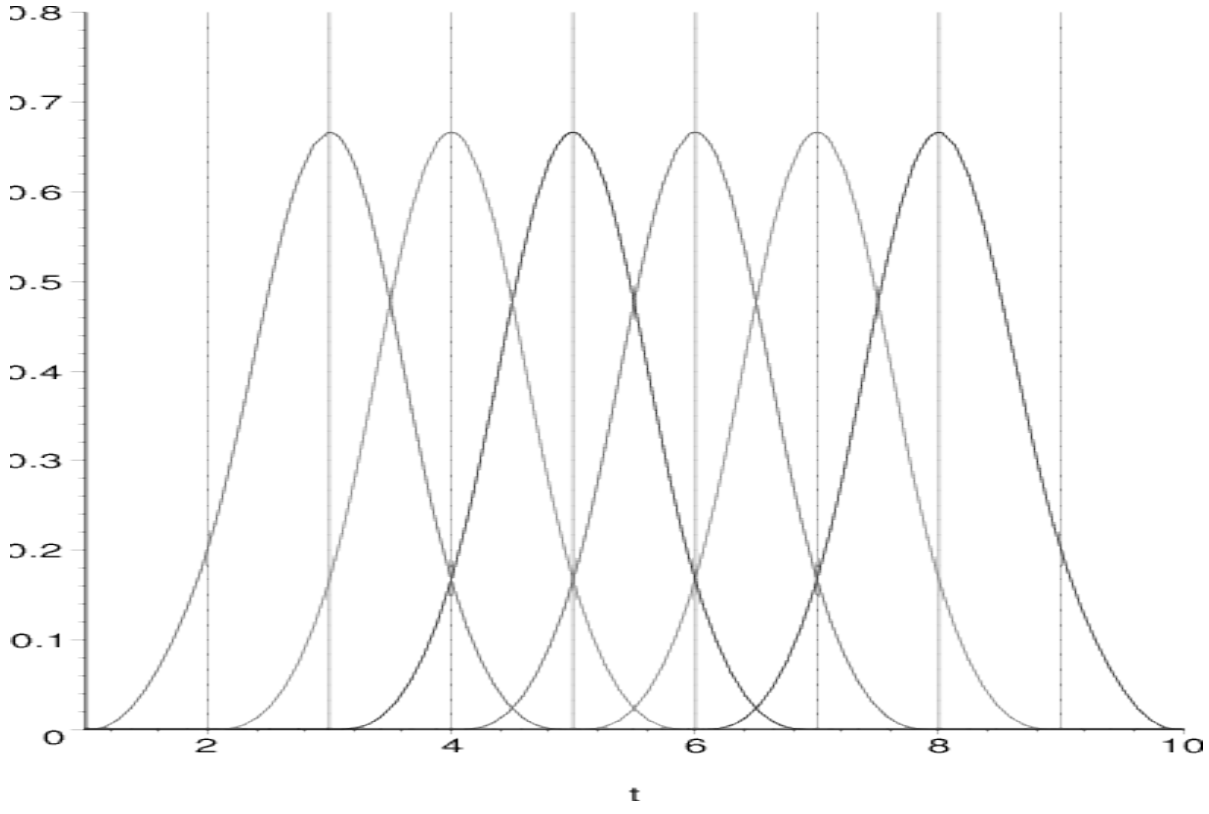
- Bump-shaped (smoothness)
- Always go in opposite direction when they cross (steadiness)

Local influence

Each blending function $F_k(t)$ must not contribute to nodes P_k that are too far away.

Local influence

Each blending function $F_k(t)$ must not contribute to nodes P_k that are too far away.
Therefore, the blending functions are zero outside of a small zone (in technical jargon: *compact support*).



Outline

Additional requirements

To summarize, we require at first:

Additional requirements

To summarize, we require at first:

- convexity

Additional requirements

To summarize, we require at first:

- convexity
- smoothness

Additional requirements

To summarize, we require at first:

- convexity
- smoothness
- steadiness

Additional requirements

To summarize, we require at first:

- convexity
- smoothness
- steadiness
- locality

Additional requirements

To summarize, we require at first:

- convexity
- smoothness
- steadiness
- locality

We want in addition

Additional requirements

To summarize, we require at first:

- convexity
- smoothness
- steadiness
- locality

We want in addition

- flexibility of shape

Additional requirements

To summarize, we require at first:

- convexity
- smoothness
- steadiness
- locality

We want in addition

- flexibility of shape
- intuitive shape parameter

Additional requirements

To summarize, we require at first:

- convexity
- smoothness
- steadiness
- locality

We want in addition

- flexibility of shape
- intuitive shape parameter
- smooth transition between approximation and interpolation

The Plan

The Plan

Will begin with polynomial blending functions $F_k(t)$, e.g. one such function for Bézier curves

$$f_k(t) = 3t^2(1-t)$$

The Plan

Will begin with polynomial blending functions $F_k(t)$, e.g. one such function for Bézier curves

$$f_k(t) = 3t^2(1-t)$$

To get convexity, (partition of unity) will normalise each blending function, i.e.

$$F_k(t) := \frac{f_k(t)}{\sum_{k=0}^{n} f_k(t)}$$

giving a *rational function* (a fraction of polynomials).

The Plan

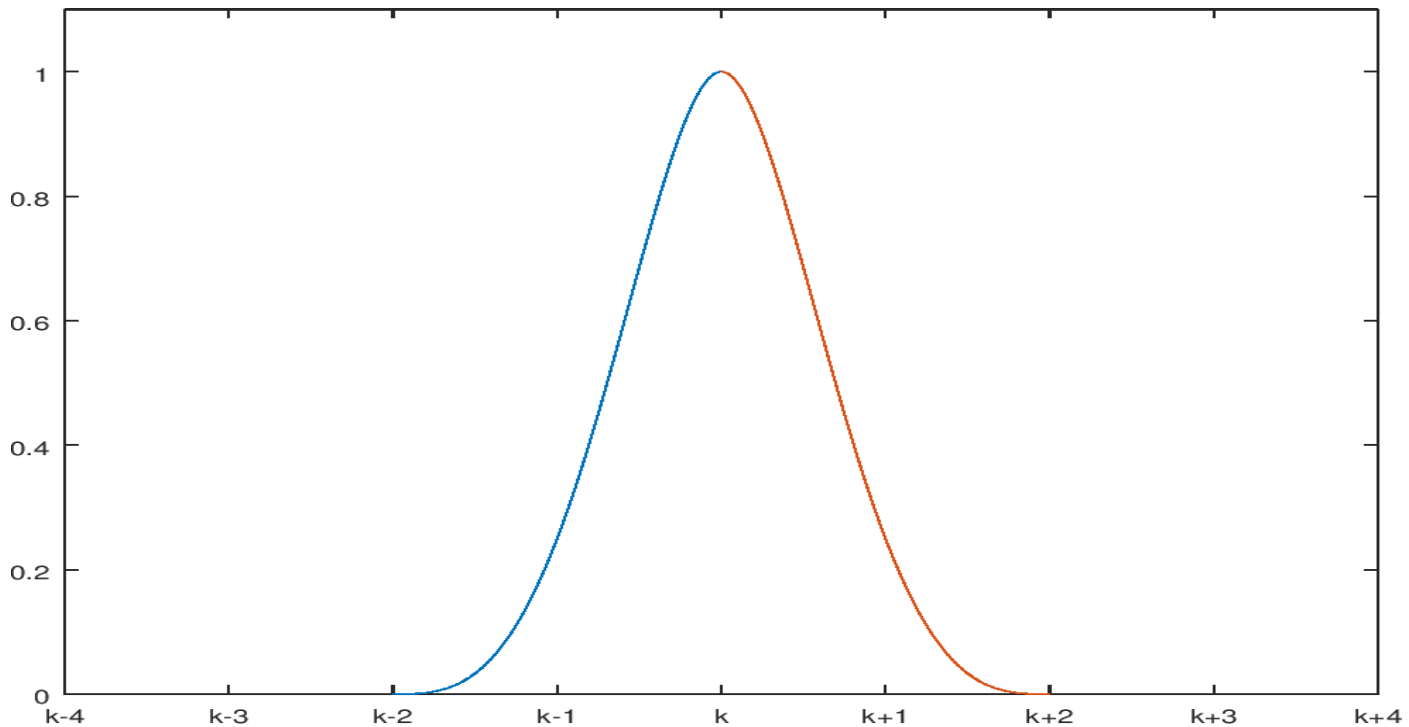
The functions will be defined piecewise on equally-spaced intervals of the parameter space, i.e.

$$f_k(t) := \begin{cases} f\left(\frac{t-k}{2}\right) & \text{if } t \in [k-2, k] \\ f\left(\frac{k-t}{2}\right) & \text{if } t \in [k, k+2] \\ 0 & \text{otherwise} \end{cases}$$

The Plan

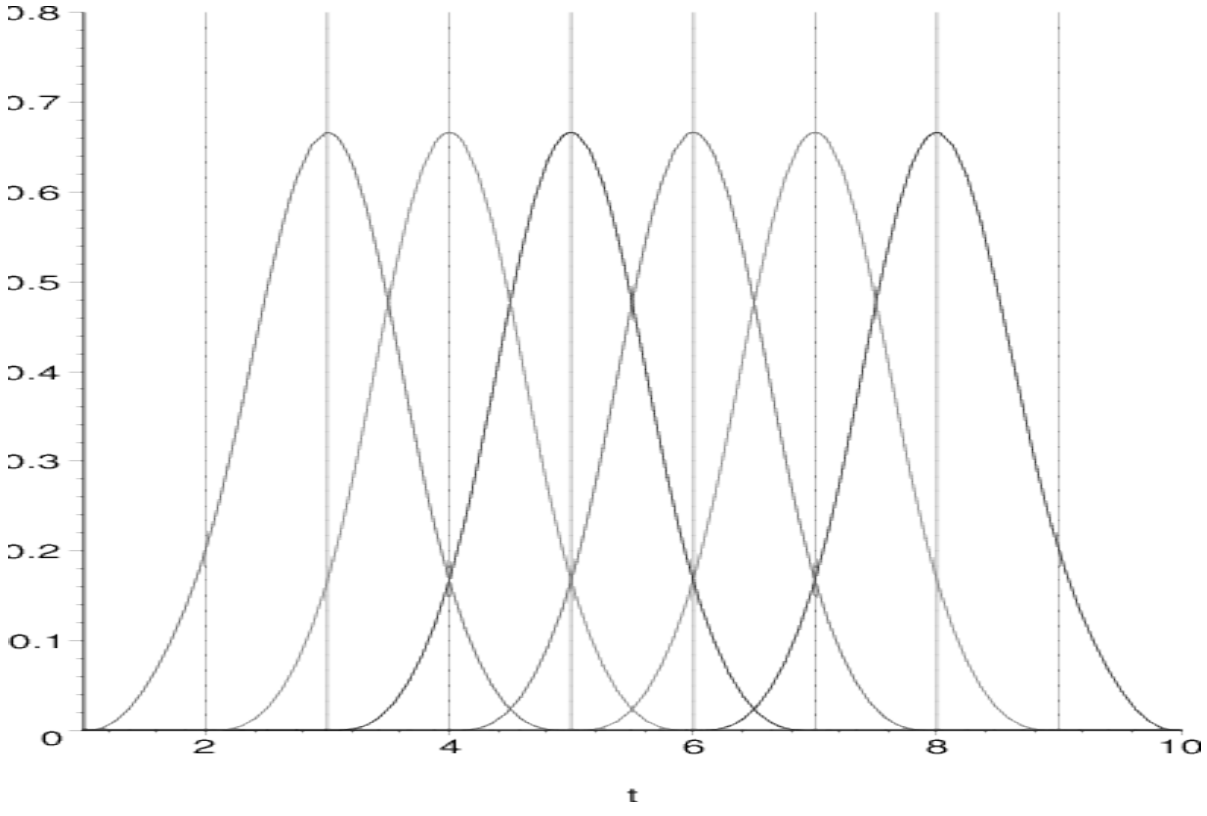
The functions will be defined piecewise on equally-spaced intervals of the parameter space, i.e.

$$f_k(t) := \begin{cases} f\left(\frac{t-k}{2}\right) & \text{if } t \in [k-2, k] \\ f\left(\frac{k-t}{2}\right) & \text{if } t \in [k, k+2] \\ 0 & \text{otherwise} \end{cases}$$



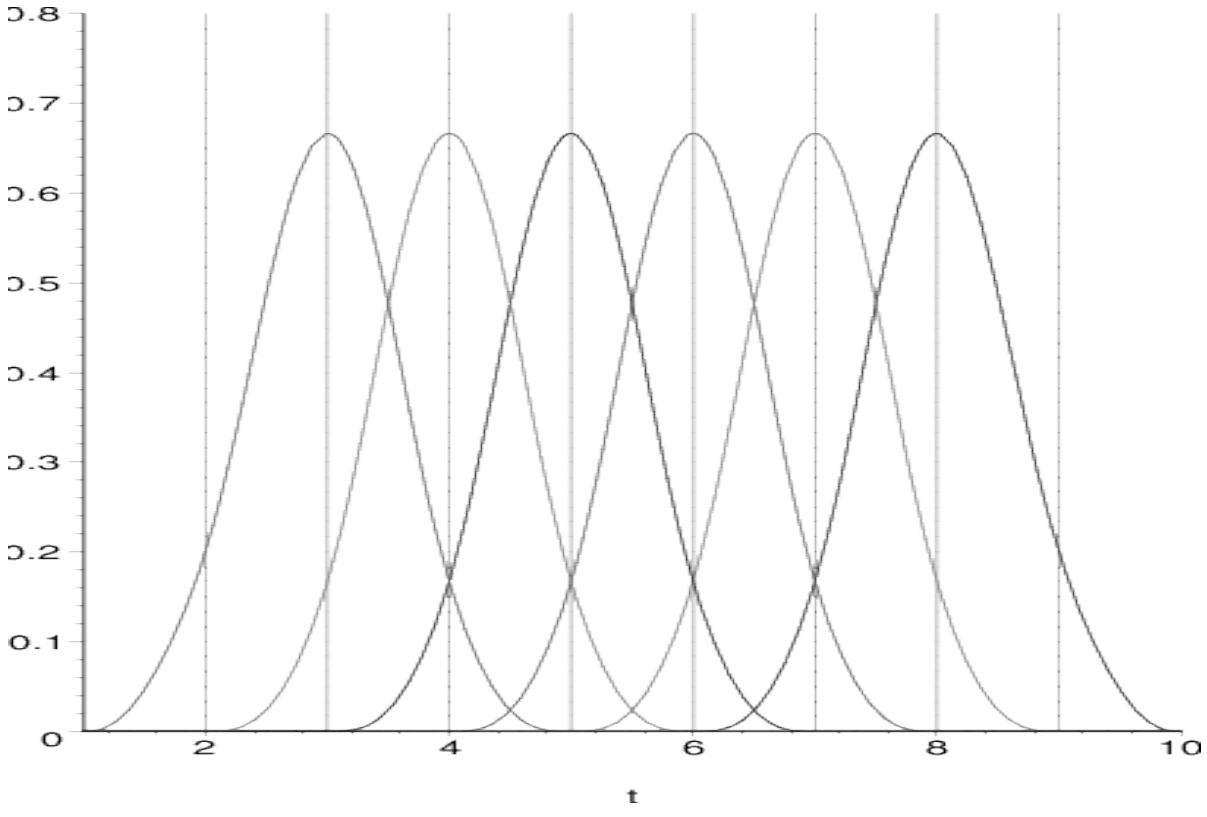
The Plan

Recall: blending functions



The Plan

Recall: blending functions



Neighbours of $F_k(t)$ cross at k, hence *cross-splines* or x-splines.

The Plan

We will impose continuity and smoothness (derivatives) conditions at the parameter knots.

The Plan

We will impose continuity and smoothness (derivatives) conditions at the parameter knots.

- Values of each segment must be equal

The Plan

We will impose continuity and smoothness (derivatives) conditions at the parameter knots.

- Values of each segment must be equal
- First derivatives (tangent lines) must be equal

The Plan

We will impose continuity and smoothness (derivatives) conditions at the parameter knots.

- Values of each segment must be equal
- First derivatives (tangent lines) must be equal
- Second derivatives (curvatures) must be equal

The basic ingredient

Let's build the basic x-spline function $f(u)$.

The basic ingredient

Let's build the basic x-spline function $f(u)$.
For simplicity, let's work on the interval $[0,1]$ and require

$$f(0)=0 \qquad f'(0)=0 \qquad f''(0)=0$$
$$f(1)=1 \qquad f'(1)=0 \qquad f''(1)=-2p$$

The basic ingredient

Let's build the basic x-spline function $f(u)$.
For simplicity, let's work on the interval $[0,1]$ and require

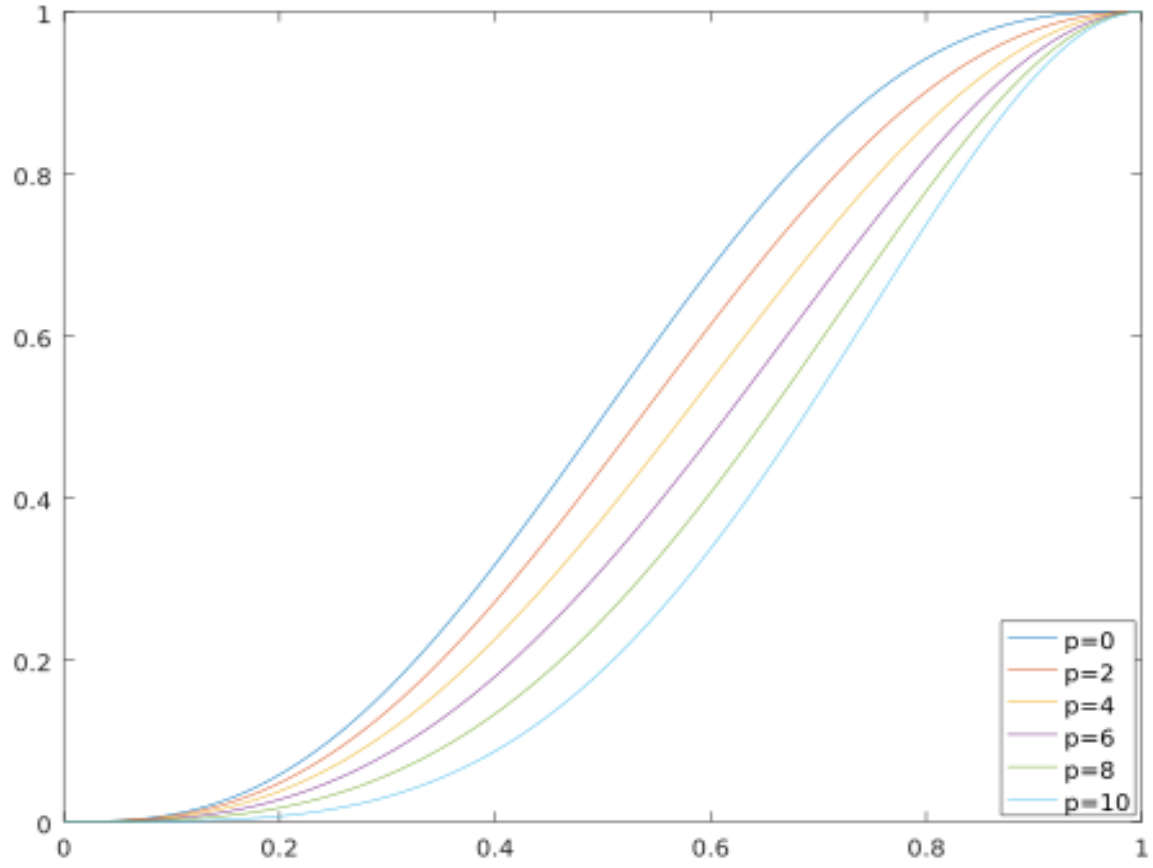
$$f(0)=0 \qquad f'(0)=0 \qquad f''(0)=0$$
$$f(1)=1 \qquad f'(1)=0 \qquad f''(1)=-2p$$

Six equations, so degree 5 polynomial. Solve to get

$$f(u)=(6-p)u^5+(2p-15)u^4+(10-p)u^3$$

The basic ingredient

As free parameter *p* varies from 0 to 10, we have some flexibility in shape:



$$f(u) = (6-p)u^5 + (2p-15)u^4 + (10-p)u^3$$

The basic x-spline

Summarising, x-spline formula for $F_k(t)$ is:

$$F_K(t) = \frac{f_k(t)}{\sum_{k=0}^{n} f_k(t)}$$

The basic x-spline

Summarising, x-spline formula for $F_k(t)$ is:

$$F_K(t) = \frac{f_k(t)}{\sum_{k=0}^{n} f_k(t)}$$

where

$$f_k(t) := \begin{cases} f\left(\frac{t-k}{2}, p_k^{\text{Left}}\right) & \text{if } t \in [k-2, k] \\ f\left(\frac{k-t}{2}, p_k^{\text{Right}}\right) & \text{if } t \in [k, k+2] \\ 0 & \text{otherwise} \end{cases}$$

The basic x-spline

Summarising, x-spline formula for $F_k(t)$ is:

$$F_K(t) = \frac{f_k(t)}{\sum_{k=0}^{n} f_k(t)}$$

where

$$f_k(t) := \begin{cases} f\left(\frac{t-k}{2}, p_k^{\text{Left}}\right) & \text{if } t \in [k-2, k] \\ f\left(\frac{k-t}{2}, p_k^{\text{Right}}\right) & \text{if } t \in [k, k+2] \\ 0 & \text{otherwise} \end{cases}$$

and

$$f(u,p) = (6-p)u^5 + (2p-15)u^4 + (10-p)u^3.$$

The basic x-spline

Summarising, x-spline formula for $F_k(t)$ is:

$$F_K(t) = \frac{f_k(t)}{\sum_{k=0}^{n} f_k(t)}$$

where

$$f_k(t) := \begin{cases} f\left(\frac{t-k}{2}, p_k^{\text{Left}}\right) & \text{if } t \in [k-2, k] \\ f\left(\frac{k-t}{2}, p_k^{\text{Right}}\right) & \text{if } t \in [k, k+2] \\ 0 & \text{otherwise} \end{cases}$$

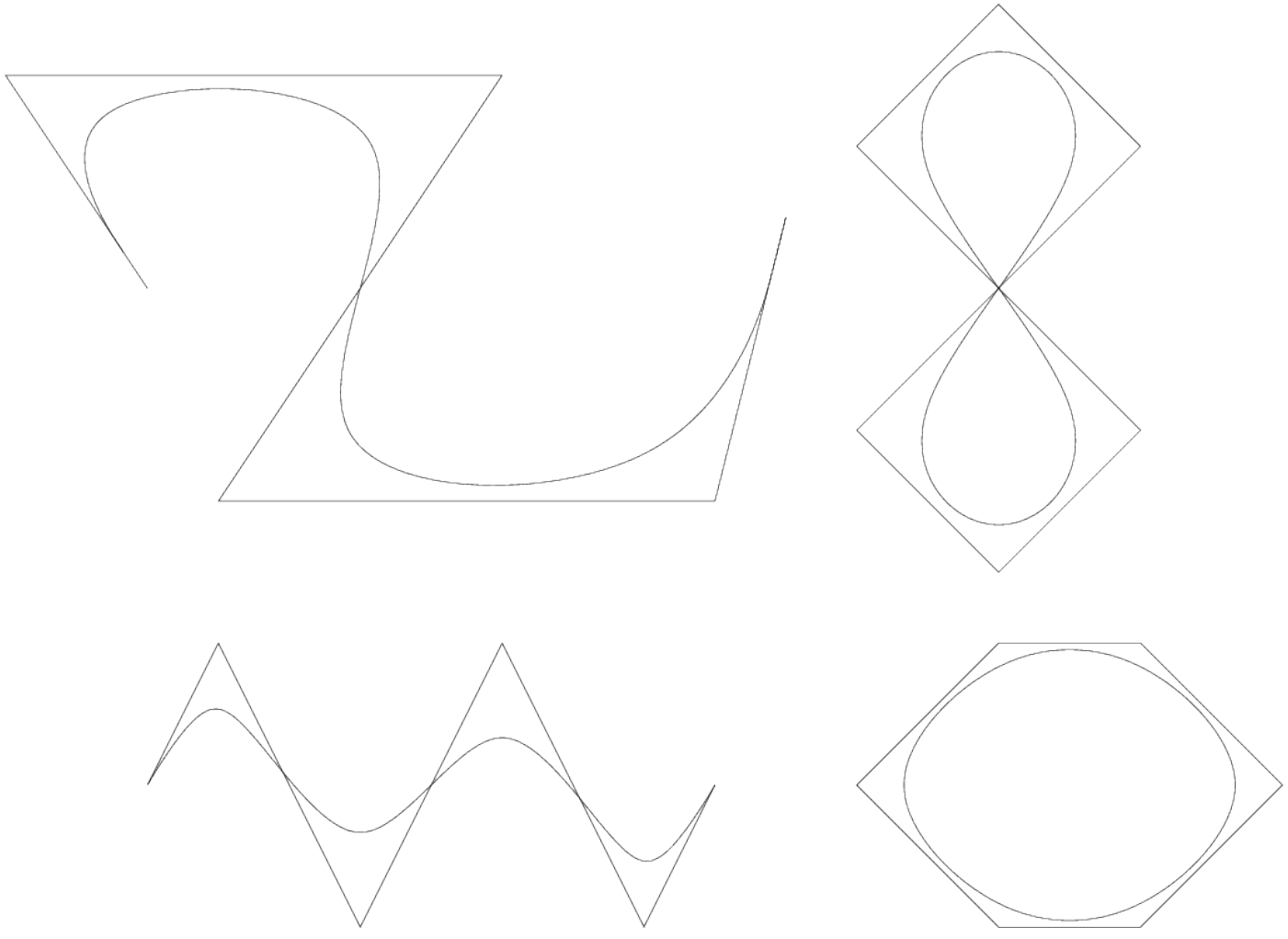
and

$$f(u,p) = (6-p)u^5 + (2p-15)u^4 + (10-p)u^3.$$

Observe that each $F_k(t)$ has its own p_k^{Left} and p_k^{Right}parameters too!

The basic x-spline

Setting all $p = 8$, we get a very good approximation of B-splines.



Outline

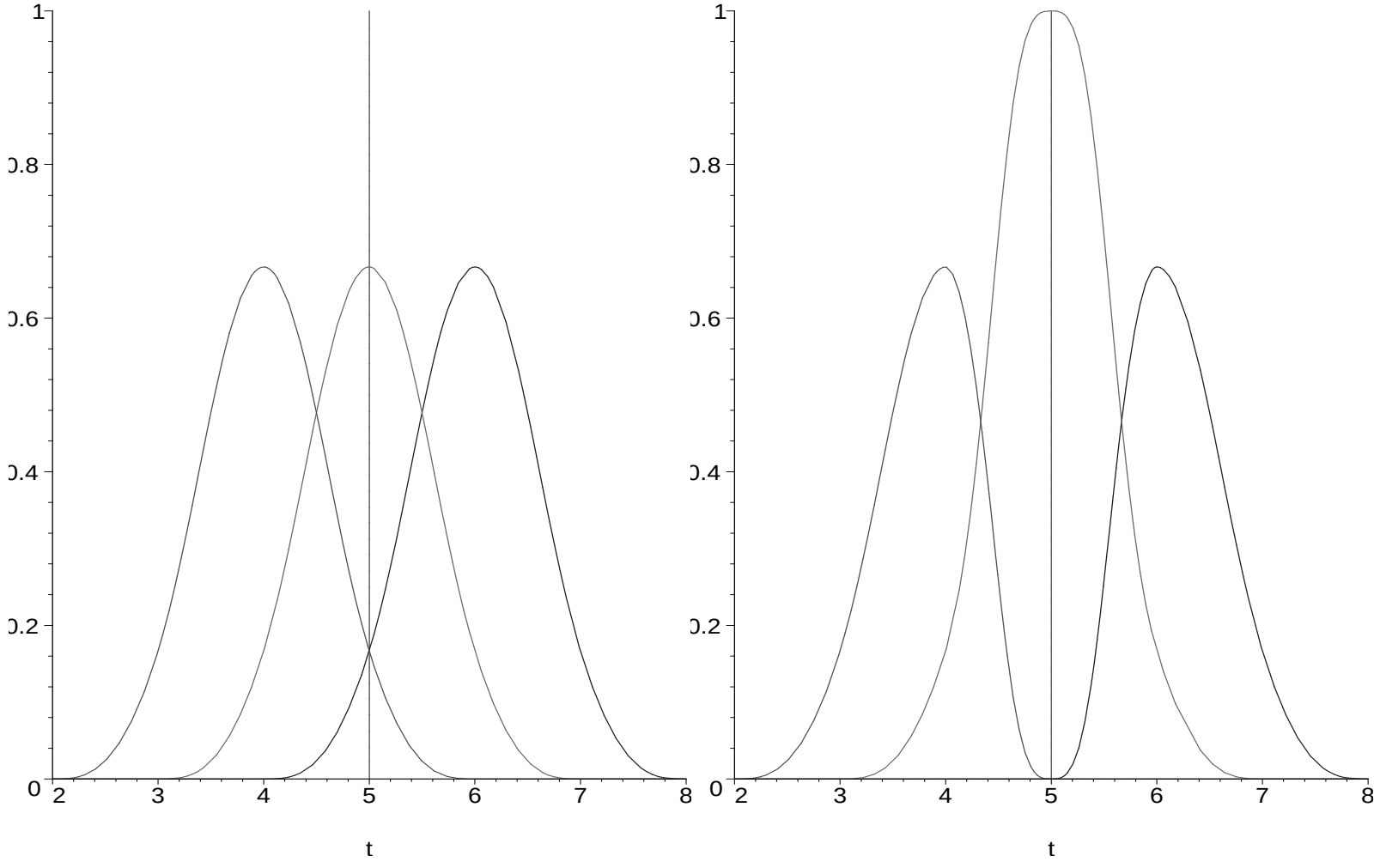
Pushing the blending function upwards

The parameters p_k^{Left} and p_k^{Right} give us some freedom to modify the spline more.

Pushing the blending function upwards

The parameters p_k^{Left} and p_k^{Right} give us some freedom to modify the spline more.

Will use them to modify the blending functions $F_k(t)$ like so.



Exercise for the audience

Calculations and notation gets a bit messy, but the idea is

Exercise for the audience

Calculations and notation gets a bit messy, but the idea is

1. Assign a new parameter $s_k \in [0,1]$ to each blending function $F_k(t)$.

Exercise for the audience

Calculations and notation gets a bit messy, but the idea is

1. Assign a new parameter $s_k \in [0,1]$ to each blending function $F_k(t)$.
2. When all $s_k = 1$, we have all $p = 8$ and a basic approximating x-spline.

Exercise for the audience

Calculations and notation gets a bit messy, but the idea is

1. Assign a new parameter $s_k \in [0,1]$ to each blending function $F_k(t)$.
2. When all $s_k = 1$, we have all $p = 8$ and a basic approximating x-spline.
3. Define new zeroing points for the neighbours of $F_k(t)$:

$$T_{k-1}^{\text{Right}} := k + s_k$$
$$T_{k+1}^{\text{Left}} := k - s_k$$

Exercise for the audience

Calculations and notation gets a bit messy, but the idea is

1. Assign a new parameter $s_k \in [0,1]$ to each blending function $F_k(t)$.
2. When all $s_k = 1$, we have all $p = 8$ and a basic approximating x-spline.
3. Define new zeroing points for the neighbours of $F_k(t)$:

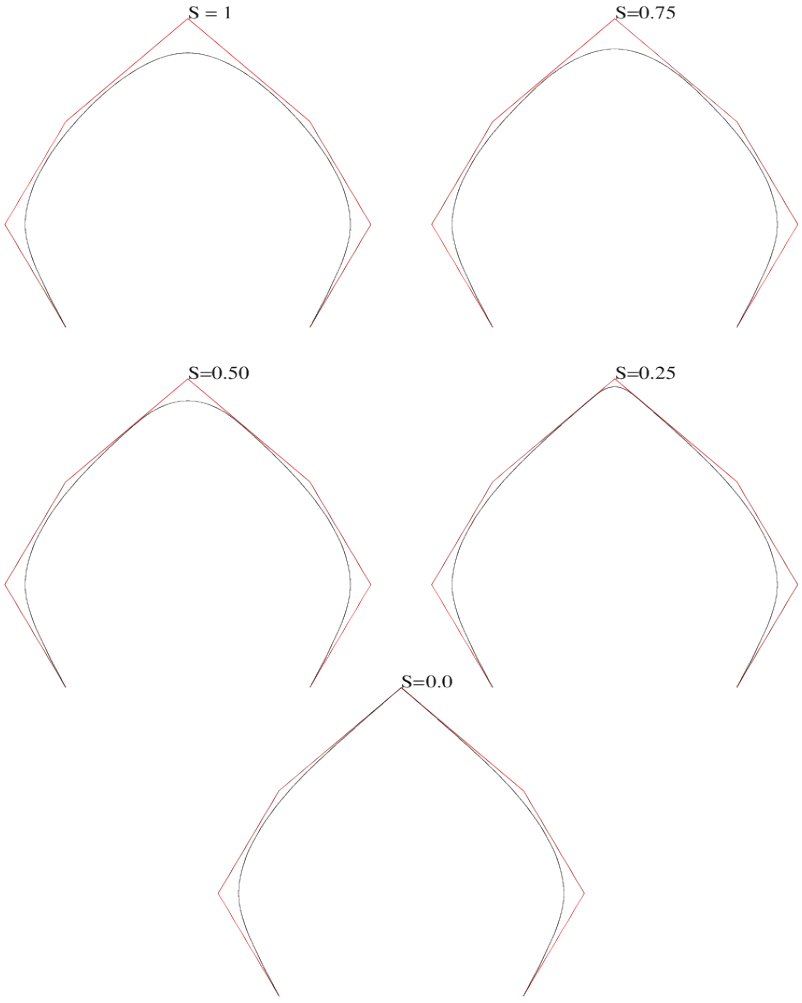
$$T_{k-1}^{\text{Right}} := k + s_k$$
$$T_{k+1}^{\text{Left}} := k - s_k$$

4. To preserve curvature (equality of second derivatives at $k-1$ and $k+1$), this requires setting special values of p

$$p_{k-1}^{\text{Right}} = 2(k - T_{k-1}^{\text{Right}})^2$$
$$p_{k+1}^{\text{Left}} = 2(k - T_{k+1}^{\text{Left}})^2$$

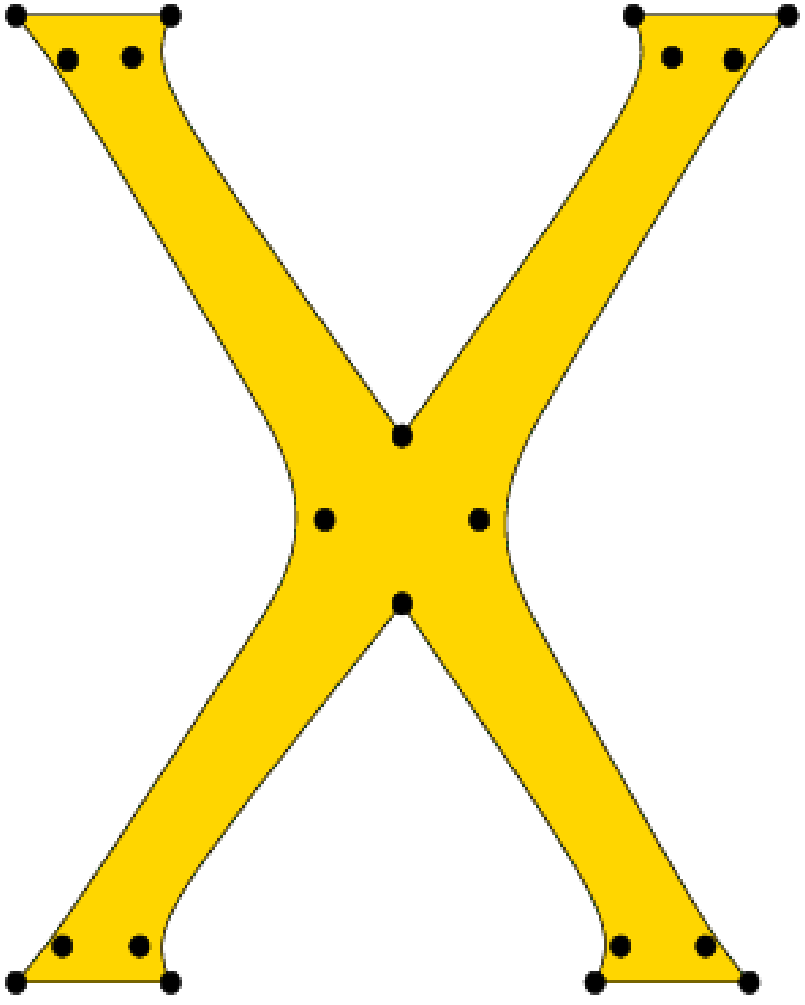
Can now interpolate sharply

End result: as s_k goes from 1 to 0 at a particular node, the spline gets closer to the node until it is interpolated sharply.



Fonts!

Good for drawing some basic fonts.



Outline

Smooth interpolation

To have smooth interpolation, we have to drop the convexity requirement.

Smooth interpolation

To have smooth interpolation, we have to drop the convexity requirement.
Recall, partition of unity:

$$1 = \sum_{k=0}^{n} F_k(t)$$

$$0 \leq F_k(t)$$

Smooth interpolation

To have smooth interpolation, we have to drop the convexity requirement.
Recall, partition of unity:

$$1 = \sum_{k=0}^{n} F_k(t)$$

$$0 \leq F_k(t)$$

We'll keep the sum equal to 1, but we'll let the blending functions go negative now.

Smooth interpolation

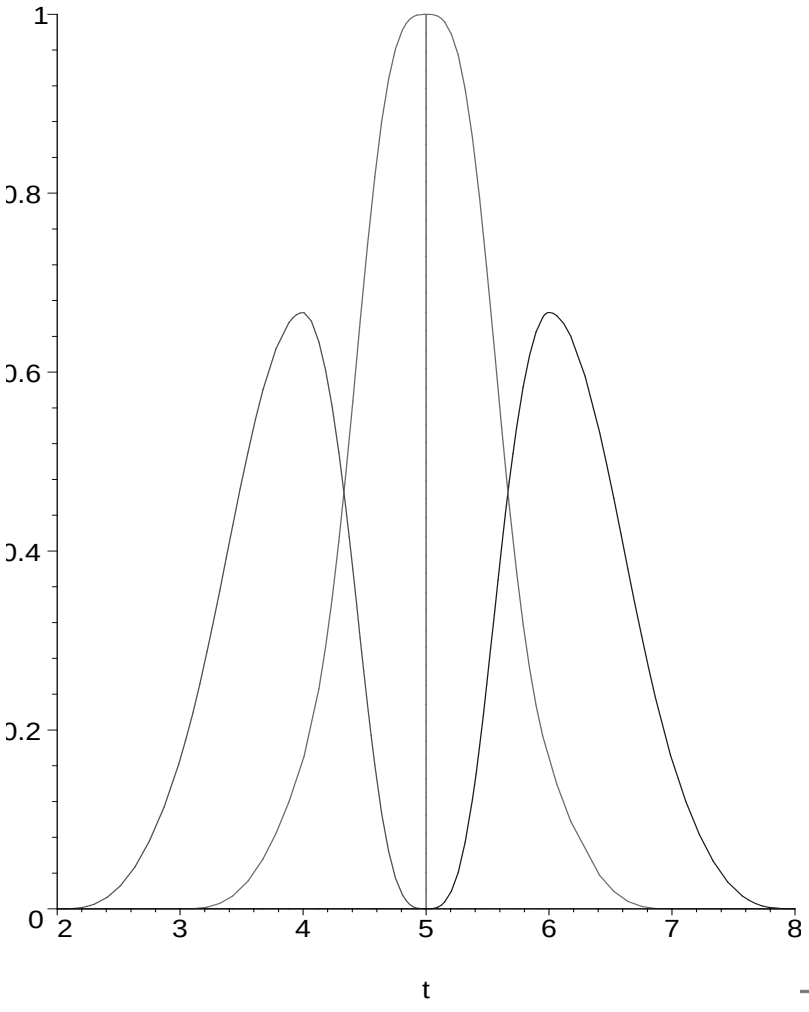
To have smooth interpolation, we have to drop the convexity requirement.
Recall, partition of unity:

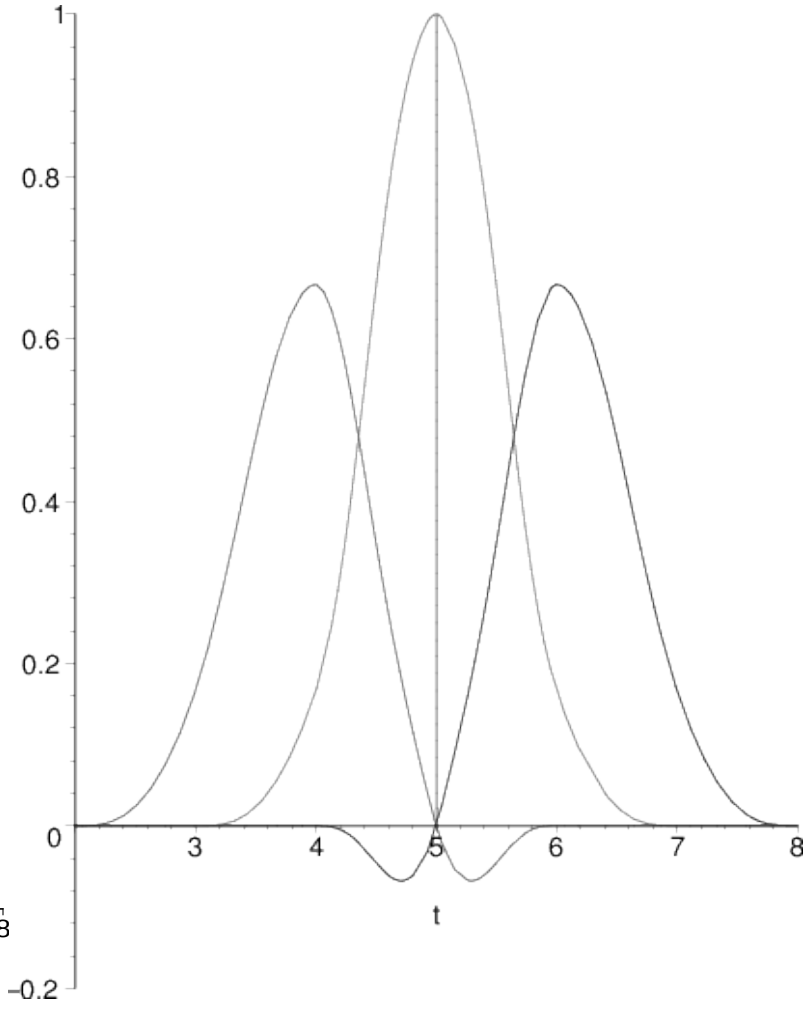
$$1 = \sum_{k=0}^{n} F_k(t)$$
$$0 \leq F_k(t)$$

We'll keep the sum equal to 1, but we'll let the blending functions go negative now.
We've pushed the tails of the blending functions down to zero (via s_k parameters). We'll push those tails further down into the negative.

Pushing way down

Blending functions $F_k(t)$ will now look like this:





Polynomial requirements again

This time use two polynomials, $g(u)$ for the positive part and $h(u)$ for the negative part.

Polynomial requirements again

This time use two polynomials, $g(u)$ for the positive part and $h(u)$ for the negative part.
If we pick the domain of $h(u)$ to be from -1 to 0 and that of $g(u)$ to go from 0 to 1...

Polynomial requirements again

This time use two polynomials, $g(u)$ for the positive part and $h(u)$ for the negative part.
If we pick the domain of $h(u)$ to be from -1 to 0 and that of $g(u)$ to go from 0 to 1...
12 conditions on continuity, tangent line, and curvature:

$$
\begin{aligned}
h(-1) &= 0 & h'(-1) &= 0 & h''(-1) &= 0 \\
h(0) &= 0 & h'(0) &= q & h''(0) &= 4q \\
g(0) &= 0 & g'(0) &= q & g''(0) &= 4q \\
g(1) &= 1 & g'(1) &= 0 & g''(1) &= -2p
\end{aligned}
$$

Polynomial requirements again

This time use two polynomials, $g(u)$ for the positive part and $h(u)$ for the negative part.
If we pick the domain of $h(u)$ to be from -1 to 0 and that of $g(u)$ to go from 0 to 1...
12 conditions on continuity, tangent line, and curvature:

$$\begin{array}{lll} h(-1)=0 & h'(-1)=0 & h''(-1)=0 \\ h(0)=0 & h'(0)=q & h''(0)=4q \\ g(0)=0 & g'(0)=q & g''(0)=4q \\ g(1)=1 & g'(1)=0 & g''(1)=-2p \end{array}$$

Degree 5 solutions:

$$\begin{aligned} g(u) &= qu+2qu^2+(10-12q-p)u^3 \\ &\quad +(2p+14q-15)u^4+(6-5q-p)u^5 \\ h(u) &= qu+2qu^2-2qu^4+qu^5 \end{aligned}$$

Recovering Catmull-Rom

New q parameter to play with, corresponding to the s_k parameters:

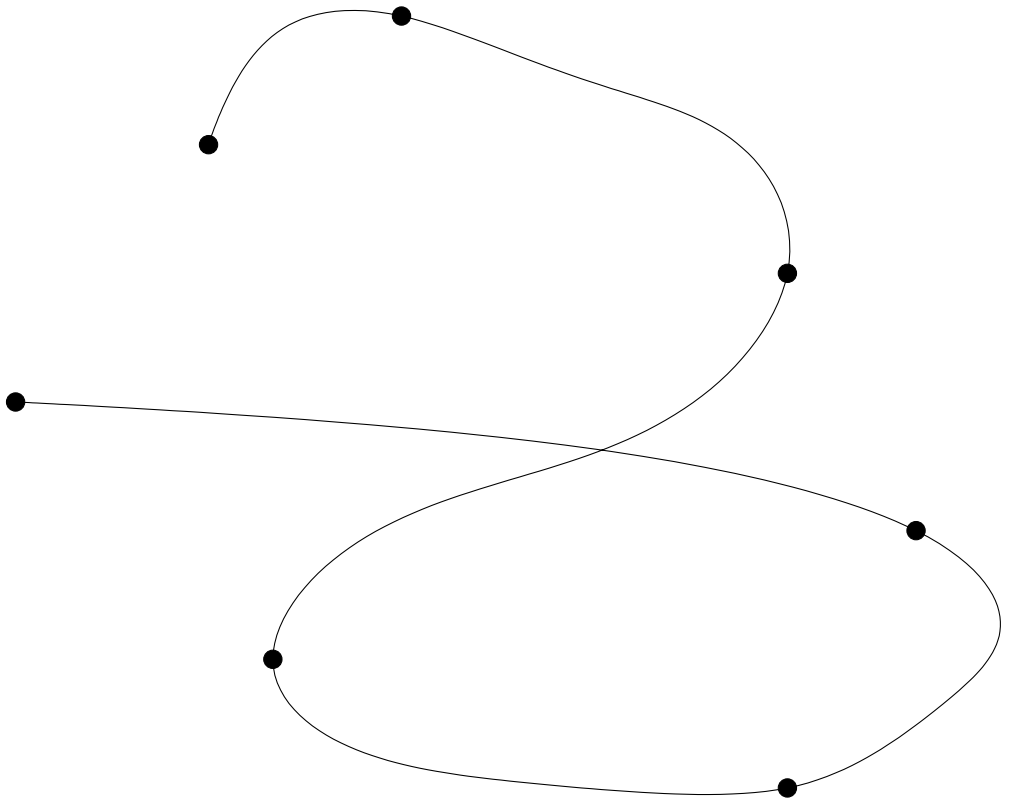
$$q_{k-1}^{\text{Right}} := -\frac{s_k}{2} =: q_{k+1}^{\text{Left}}$$

Recovering Catmull-Rom

New q parameter to play with, corresponding to the s_k parameters:

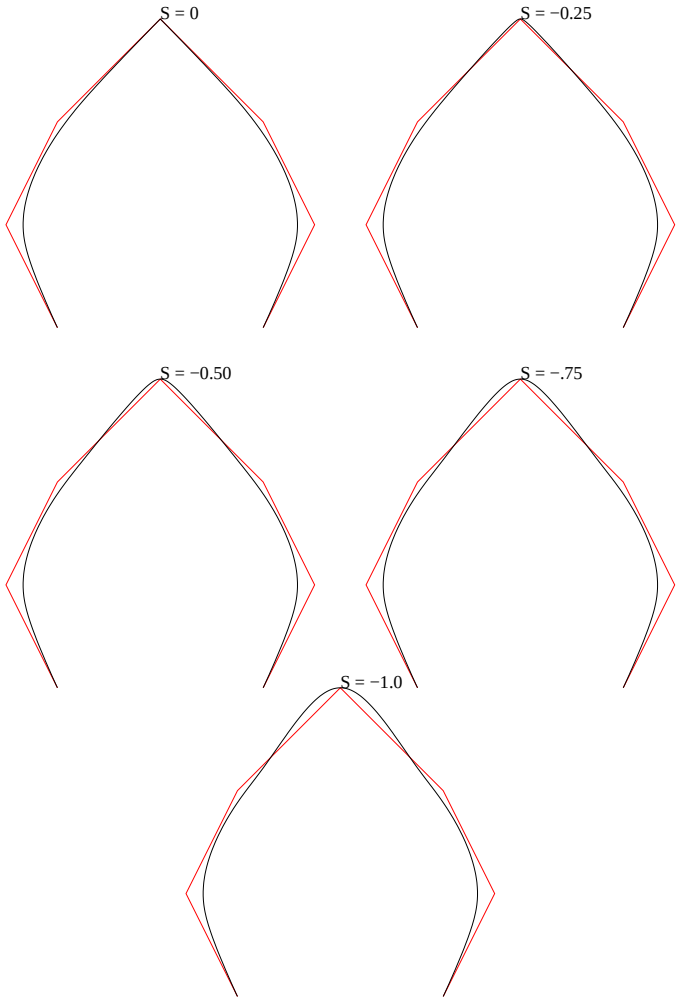
$$q_{k-1}^{\text{Right}} := -\frac{s_k}{2} =: q_{k+1}^{\text{Left}}$$

When $q = 1/2$ everywhere (i.e. $s = -1$ everywhere), we get an almost exact copy of Catmull-Rom.



Letting *s* vary slowly

Smoothly goes from sharp interpolation to smooth interpolation:



Thanks!

jordigh@octave.org or @JordiGH@mathstodon.xyz on Mastodon

Thanks!

jordigh@octave.org or @JordiGH@mathstodon.xyz on Mastodon

Remember xfig?