Computer Graphics

(Curves and Surfaces)

Thilo Kielmann
Fall 2003
Vrije Universiteit, Amsterdam
kielmann@cs.vu.nl

http://www.cs.vu.nl/~graphics/

Outline for today

- Representation of curves
- Design criteria
- Interpolation
- Bezier curves
- B-Splines

Representation of curves

- Explicit
- Implicit
- Parametric

Explicit representation

$y = f(x)$, hopefully also $x = g(y)$

In 3D: $y = f_1(x), z = f_2(x)$

Line: $y = mx + h$
Circle: $y = \sqrt{r^2 - x^2}$ **and** $y = -\sqrt{r^2 - x^2}$

Problems: – partially undefined (vertical lines)
– ambiguity (upper/lower half of circle)

Implicit representation

$f(x, y) = 0$

Line: $ax + by + c = 0$
Circle: $x^2 + y^2 + z^2 = 0$

In 3D:
Plane: $ax + by + cz + d = 0$
Circle: $x^2 + y^2 + z^2 - r^2 = 0$

Problems:
- only **membership test** (not constructive)
- curves in 3D: intersecting 2 planes
  $\Rightarrow f(x, y, z) = 0 = g(x, y, z)$

Quadrics

Quadric surfaces are **algebraic surfaces** with a degree of up to 2. (special case of implicit representation)

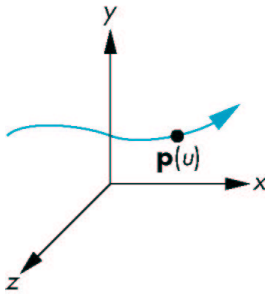
Algebraic surface: $f(x, y, z)$ is the sum of polynomials in x, y, z.

Quadrics:
- spheres, disks, cones
- at most 2 intersections with lines

Quadric: degree is the sum of powers of the individual terms. (≤ 2)
Examples: x xy z^2, **but not:** xy^2

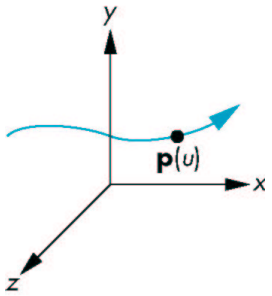
Parametric Form

Values expressed depending on an independent variable u, mostly $u \in [0, 1]$



Parametric Form the same both for 3D and 2D (just drop the z component)

Parametric Curve



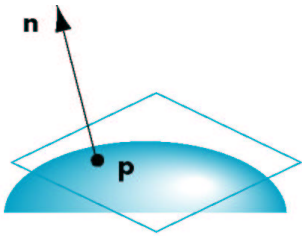
$$p(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}$$

Derivative:

$$\frac{dp(u)}{du} = \begin{bmatrix} \frac{dx(u)}{du} \\ \frac{dy(u)}{du} \\ \frac{dz(u)}{du} \end{bmatrix}$$

gives the tangent to the curve

Parametric Surface



$$p(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}$$

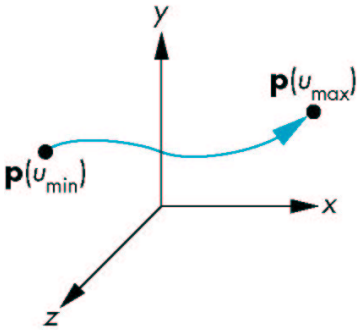
$$\frac{\partial p}{\partial u} = \begin{bmatrix} \frac{\partial x(u,v)}{\partial u} \\ \frac{\partial y(u,v)}{\partial u} \\ \frac{\partial z(u,v)}{\partial u} \end{bmatrix} \quad \frac{\partial p}{\partial v} = \begin{bmatrix} \frac{\partial x(u,v)}{\partial v} \\ \frac{\partial y(u,v)}{\partial v} \\ \frac{\partial z(u,v)}{\partial v} \end{bmatrix}$$

give the tangent plane

$n = \partial p/\partial u \times \partial p/\partial v$

(normal vector)

Parametric polynomial curves

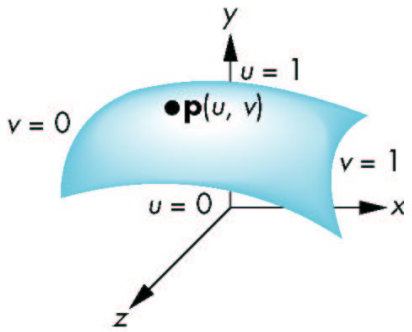


$u_{min} \leq u \leq u_{max}$

usually $0 \leq u \leq 1$

$$p(u) = \sum_{k=0}^{n} u^k c_k, \quad \text{degree } n, \quad \text{and } c_k = \begin{bmatrix} c_{xk} \\ c_{yk} \\ c_{zk} \end{bmatrix}$$

Parametric polynomial surfaces



$$p(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} = \sum_{i=0}^{n} \sum_{j=0}^{m} c_{ij} u^i v^j$$

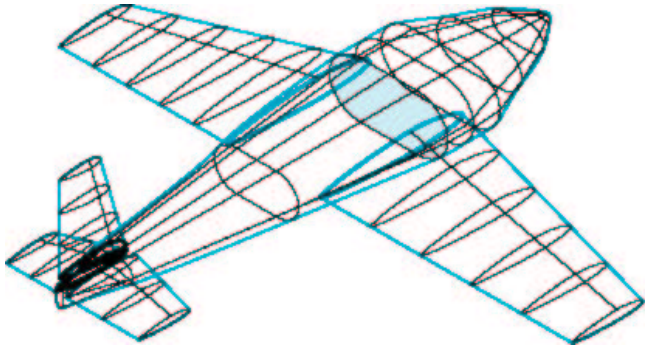
$0 \leq u, v \leq 1$ defines a **surface patch**

Design Criteria

Desirable properties for curves:

- local control of shape
- smoothness and continuity
- ability to compute derivatives
- stability
- ease of rendering

Example, model airplane



(cross section piece in blue)

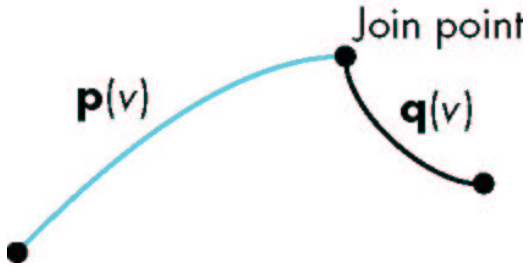
Cross section curve

We want to model the cross section piece suitably:

- smooth
- smooth join points for adjacent pieces
- producable (possibly only a good approximation)

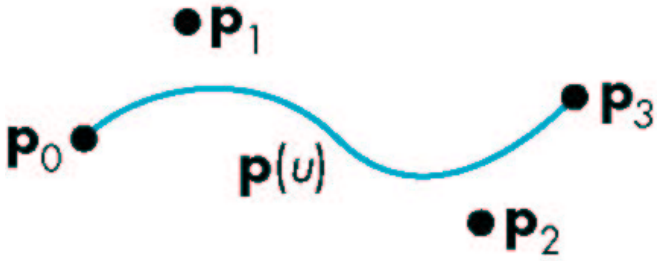
Join points

Derivative discontinuity:



- We want to align derivatives at join points.
- For polynomials, all derivatives exist and can be computed.

Local control of change



- curve segment $p(u)$ defined via control points
- (here,) $p(u)$ **interpolates** (passes through) p_0 and p_3
- no global curve description, just individual segments

Choosing the degree of polynomial

High degree: many parameters to fit desired shape, but possibly very rough

Low degree: easy to compute, possibly hard to match desired shape

Cubic polynomial curves:
degree 3, combining many short curve segments

Cubic polynomial curves

$$p(u) = c_0 + c_1 u + c_2 u^2 + c_3 u^3 = \sum_{k=0}^{3} c_k u^k = u^T c$$

$$c = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad u = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix} \quad c_k = \begin{bmatrix} c_{kx} \\ c_{ky} \\ c_{kz} \end{bmatrix}$$

Cubic interpolation

Assume 4 control points p_0, p_1, p_2, p_3, search cubic polynomial $p(u) = u^T c$ that interpolates (passes through) them.

We assume $0 \leq u \leq 1$, and equally spaced control points at $u = 0, \frac{1}{3}, \frac{2}{3}, 1$.

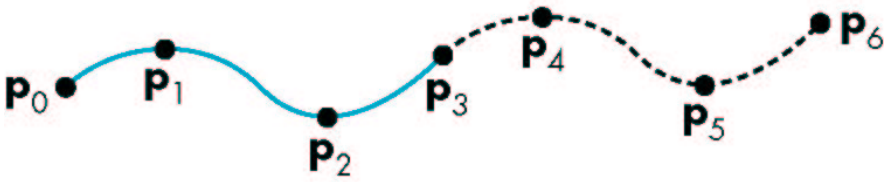
$$\begin{aligned} p_0 &= p(0) = c_0 \\ p_1 &= p\left(\tfrac{1}{3}\right) = c_0 + \tfrac{1}{3}c_1 + \left(\tfrac{1}{3}\right)^2 c_2 + \left(\tfrac{1}{3}\right)^3 c_3 \\ p_2 &= p\left(\tfrac{2}{3}\right) = c_0 + \tfrac{2}{3}c_1 + \left(\tfrac{2}{3}\right)^2 c_2 + \left(\tfrac{2}{3}\right)^3 c_3 \\ p_3 &= p(1) = c_0 + c_1 + c_2 + c_3 \end{aligned}$$

Cubic interpolation (2)

We can write the equations for p_0, p_1, p_2, p_3 in matrix form as $p = Ac$.

With $M_I = A^{-1}$ we get $c = M_I p$, our polynomial coefficients.

Joining interpolating segments



First segment: p_0, p_1, p_2, p_3
Second segment: p_3, p_4, p_5, p_6
Each segment is defined for $0 \leq u \leq 1$, and M_I is always the same.

Joins are continuous, but join point derivatives are not continuous.

Blending functions

Goal: understanding the interpolation

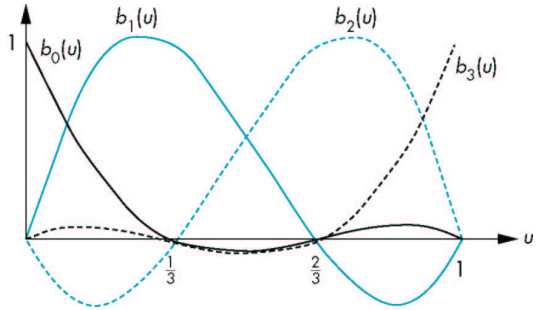
We have: $p(u) = u^T c = u^T M_I p$

$p(u) = b(u)^T p$, where $b(u) = M_I^T u = \begin{bmatrix} b_0(u) \\ b_1(u) \\ b_2(u) \\ b_3(u) \end{bmatrix}$

the column matrix of blending polynomials, yielding:

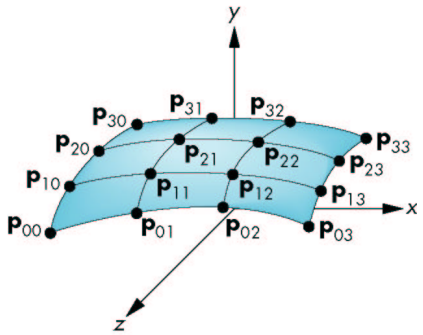
$p(u) = b_0(u)p_0 + b_1(u)p_1 + b_2(u)p_2 + b_3(u)p_3$

Blending polynomials for interpolation



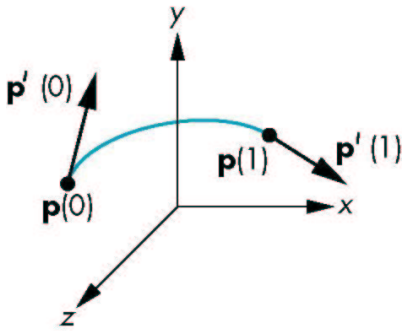
- blend together the contributions of the control points on the curve
- 1 in "their" point, 0 in the others
- not very smooth, consequence of interpolation

Cubic interpolation patch



$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} u^i v^j c_{ij} = u^T C v \quad C = [c_{ij}] \quad v = \begin{bmatrix} 1 & v & v^2 & v^3 \end{bmatrix}^T$$

Hermite form



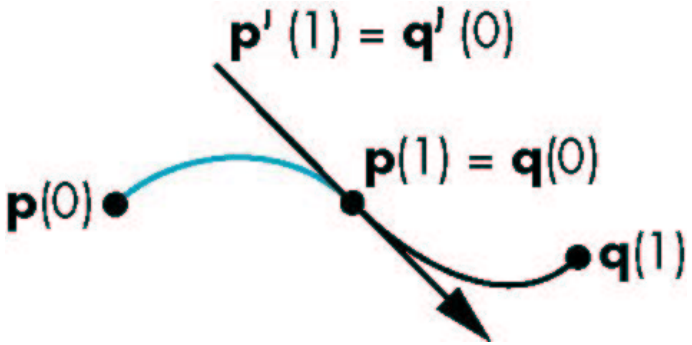
We interpolate only points p_0 and p_3 and require in addition to fix the derivatives at p_0 and p_3.

Hermite form (2)

$$
\begin{aligned}
p(0) &= p_0 = c_0 \\
p(1) &= p_3 = c_0 + c_1 + c_2 + c_3 \\
\\
p'(u) &= c_1 + 2uc_2 + 3u^2c_3 \\
\\
p'_0 &= p'(0) = c_1 \\
p'_3 &= p'(1) = c_1 + 2c_2 + 3c_3
\end{aligned}
$$

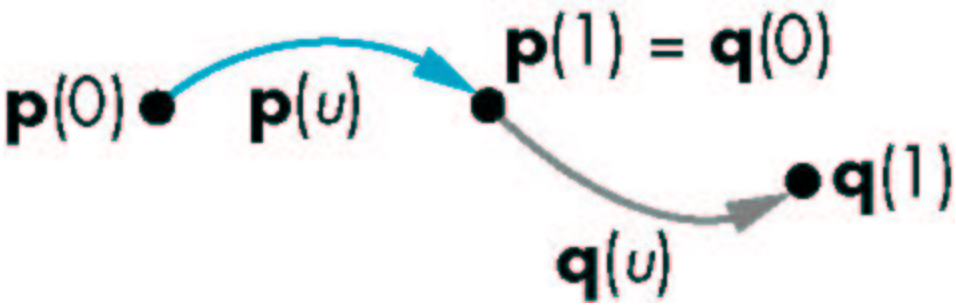
With $q^T = [p_0 \;\; p_3 \;\; p'_0 \;\; p'_3]$ we get $p(u) = u^T M_H q$

Hermite form at join point



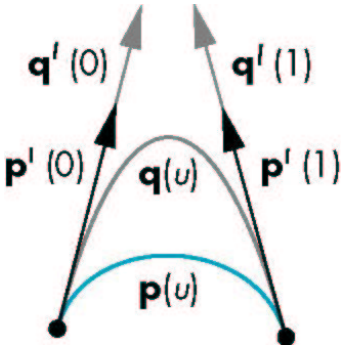
gives much smoother joins

Parametric continuity



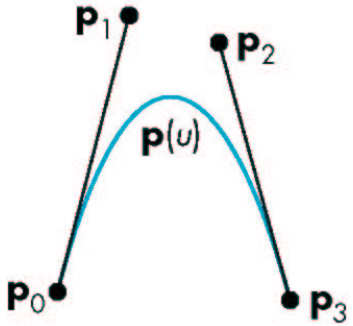
C^0 continuity: $p(1) = q(0)$
C^1 continuity: $p(1) = q(0)$ and $p'(1) = q'(0)$

Geometric continuity



G^1 continuity: $p(1) = q(0)$ and $p'(1) \propto q'(0)$
derivatives are proportional: same direction, different magnitude

Bezier curves



Use p_0 and p_3 for interpolation
Use p_1 and p_2 to approximate tangents
(like with Hermite curves)

Bezier curves (2)

Approximating tangents:

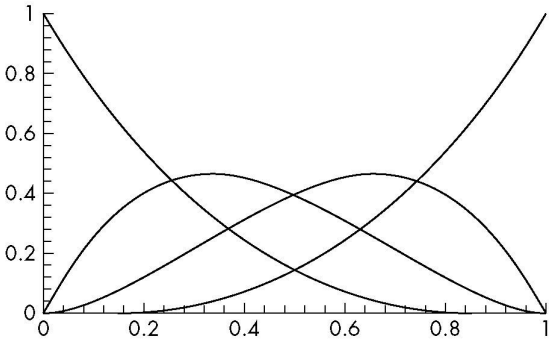
$p'(0) = \frac{p_1 - p_0}{\frac{1}{3}} = 3(p_1 - p_0)$
$p'(1) = \frac{p_3 - p_2}{\frac{1}{3}} = 3(p_3 - p_2)$
apply these to the derivatives of $p(u) = u^T c$ at the endpoints gives
$3p_1 - 3p_0 = c_1$
$3p_3 - 3p_2 = c_1 + 2c_2 + 3c_3$
in addition to
$p_0 = c_0$ and $p_3 = c_0 + c_1 + c_2 + c_3$
we get $p(u) = u^T M_B p$

Joins of Bezier curves

We use again $p_0 \ldots p_3$ for the first curve and $p_3 \ldots p_6$ for the second.

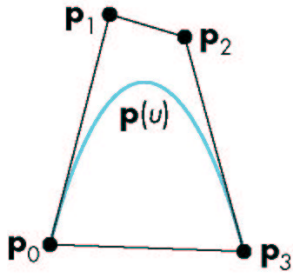
Bezier curves are C^0 continuous, but **not** C^1

Bezier blending polynomials



Much smoother than for interpolation.

Bezier polynomial and its convex hull



The Bezier polynomial in terms of its blending polynomials forms a convex sum. (property of the blending polynomials).

Thus the Bezier polynomial lies within the convex hull of its control points, which is close to by not exactly interpolating all control points.

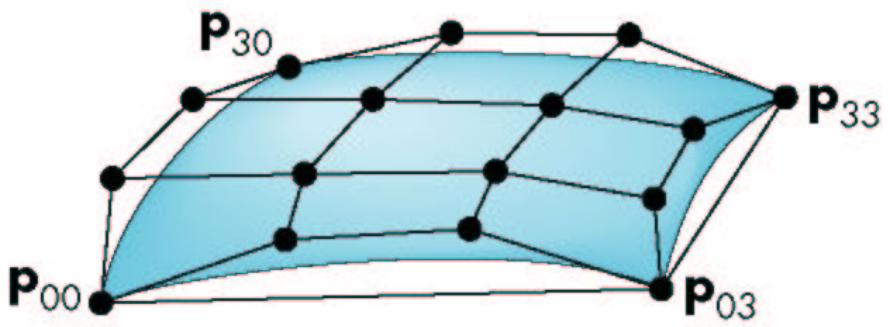
Bezier surface patch

Given $P = [p_{ij}]$ a 4×4 array of control points.

Bezier surface patch:

$$p(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} b_i(u) b_j(v) p_{ij} = u^T M_B P M_B^T v$$

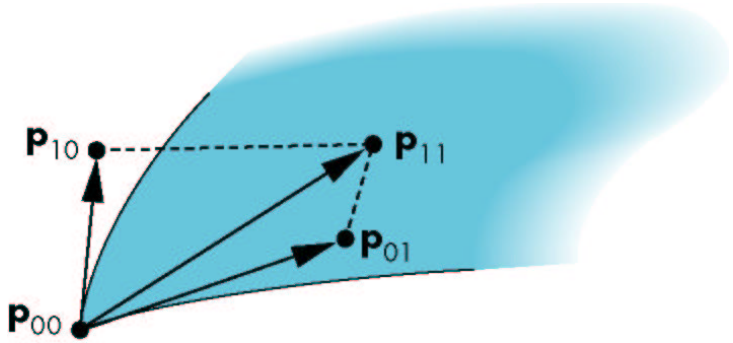
Bezier surface patch



completely contained in the convex hull of its control points

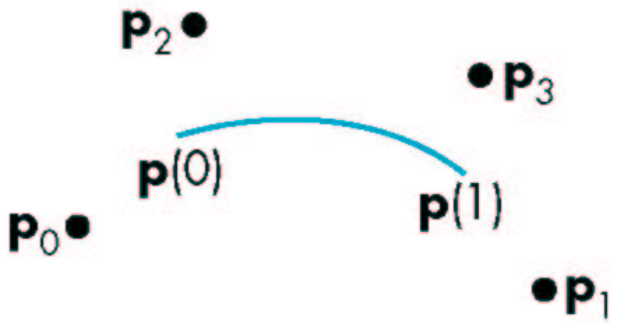
interpolates $p_{00}, p_{03}, p_{30}, p_{33}$

Corner twist of Bezier patch



$p_{00}, p_{01}, p_{10}, p_{11}$ are only in the same plane if the Bezier patch is flat (generally not)

Cubic B-Splines



To improve smoothness, give up interpolation completely. Let control points define curve only between middle control points.

Cubic B-Splines (2)

Assume:
$q(u)$ is defined by $q = [p_{i-3} \;\; p_{i-2} \;\; p_{i-1} \;\; p_i]^T$
lies between p_{i-2} and p_{i-1}

$p(u)$ is defined by $p = [p_{i-2} \;\; p_{i-1} \;\; p_i \;\; p_{i+1}]^T$
lies between p_{i-1} and p_i

We are looking for a matrix M such that
$p(u) = u^T M p$ and $q(u) = u^T M q$

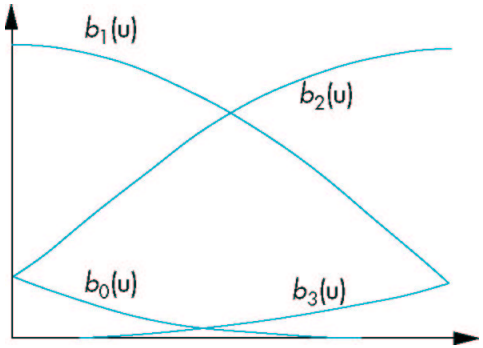
Cubic B-Splines (3)

We can use
$p(0) = q(1) = \frac{1}{6}(p_{i-2} + 4p_{i-1} + p_i)$
$p'(0) = q'(1) = \frac{1}{2}(p_i - p_{i-2})$

which satisfy the symmetry conditions that $p(0)$ does not depend on p_{i-3} and $q(1)$ is independent from p_{i+1}

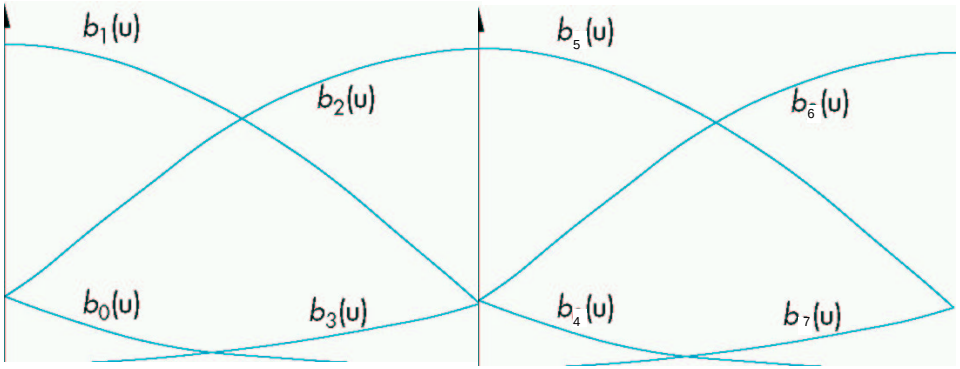
(some gymnastics then result in the desired matrix M_S)

Blending polynomials

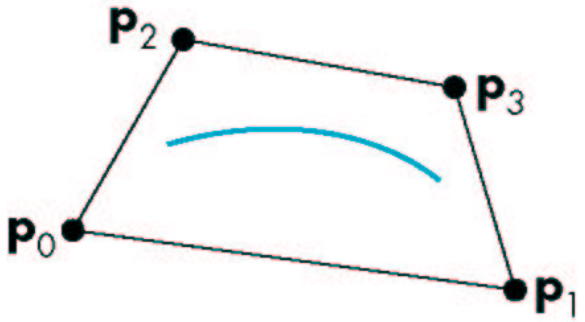


Very smooth. . .

Joining blending polynomials



Convex hull for B-spline curve



Again, control points define convex hull for the curve.

Continuity of B-Splines

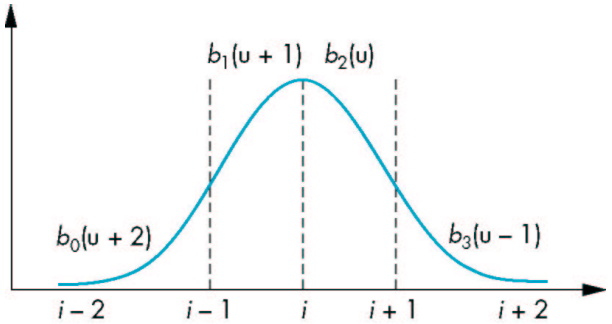
B-Splines have been designed to be C^0 and C^1 continuous.

They happen to be also C^2 continuous.

This is very smooth, like blending metal.

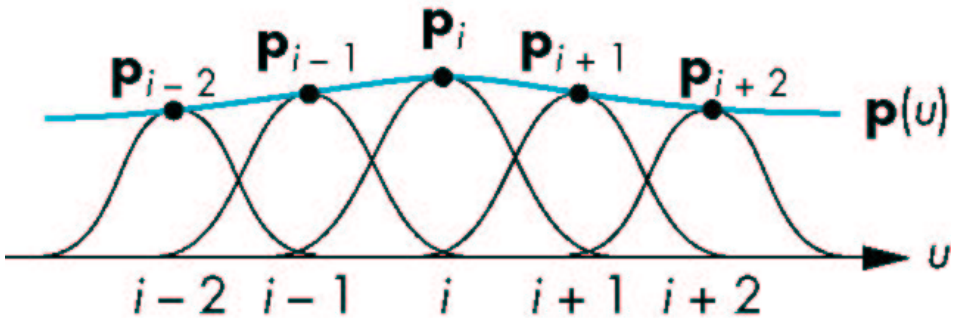
Compared to Bezier curves, B-splines need 3 times as much computation for the same number of control points, because we add one control point at a time, rather than 3 points.

Spline basis function



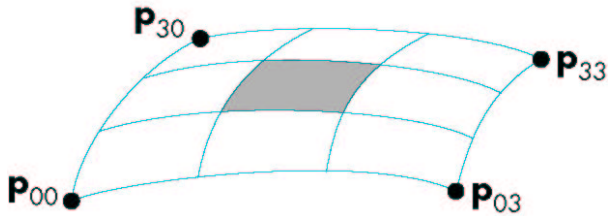
Each control point has impact on 4 consecutive intervalls.
$\Longrightarrow$ locality

Spline approximation function



$p(u)$ can be defined piecewise by the linear combination of the basis functions.

Spline surface patch



from the blending functions:

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)b_j(v)p_{ij}$$

defines only the middle patch of the surface

General B-Splines

Set of control points $p_0, \ldots, p_m$
approximation problem: find $p(u) = [x(u) \;\; y(u) \;\; z(u)]^T$
over $u_{\min} \leq u \leq u_{\max}$

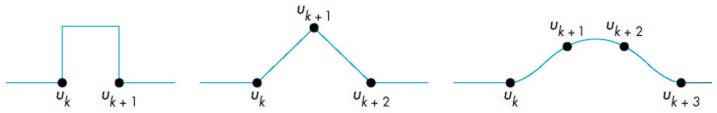
Set of values $\{u_k\}$ called **knot array**
$u_{\min} = u_0 \leq u_1 \leq \ldots \leq u_n = u_{\max}$

$$p(u) = \sum_{j=0}^{d} c_{jk}u^j, \quad u_k < u < u_{k+1}$$

(for $d = 3$ and n knots we have to solve $4n$ equations)

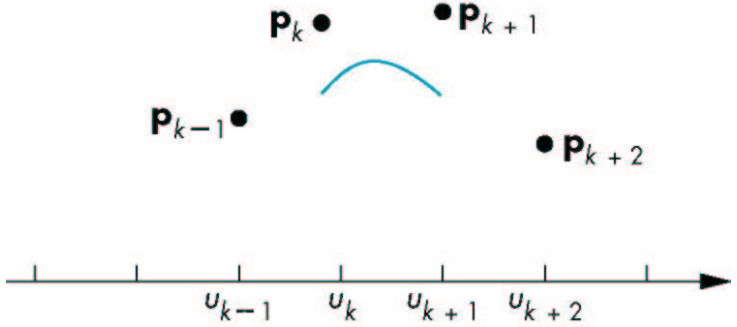
Recursively defined B-Splines

Local solution, by combining basis functions for each interval



These **Basis splines** ($\rightarrow$ name B-splines) of degree d are non-zero only in $d+1$ intervals.
(and they are recursively defined, needs $d-1$ extra knots for the ends)
C^{d-1} continuity at the knots

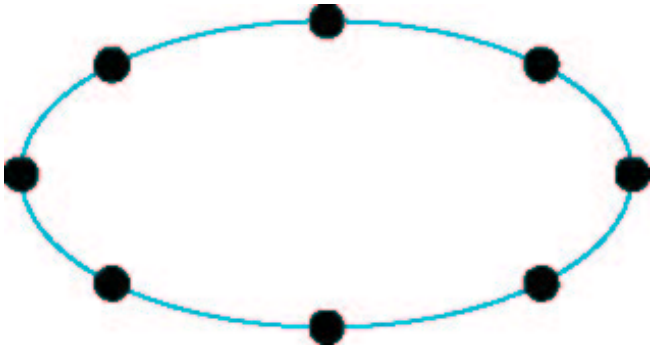
Uniform B-Splines



Splines are called **uniform** if their knots are equally spaced.

Also non-uniform spacing and repeated knots possible.

Periodic uniform spline



(example, periodicity depends on knot array)

Open splines

If a knot has a multiplicity $d+1$, a spline of degree d must interpolate it. (We can use this to define the endpoints of a spline.)

Example: $\{0,0,0,0,1,2,\ldots,n-1,n,n,n,n\}$

$\{0,0,0,0,1,1,1,1\}$ is the cubic Bezier curve

Summary

- Representation of curves
- Basics of Bezier curves and splines
- Next week:
  - ⋆ NURBS
  - ⋆ Rendering splines
  - ⋆ (some) procedural methods