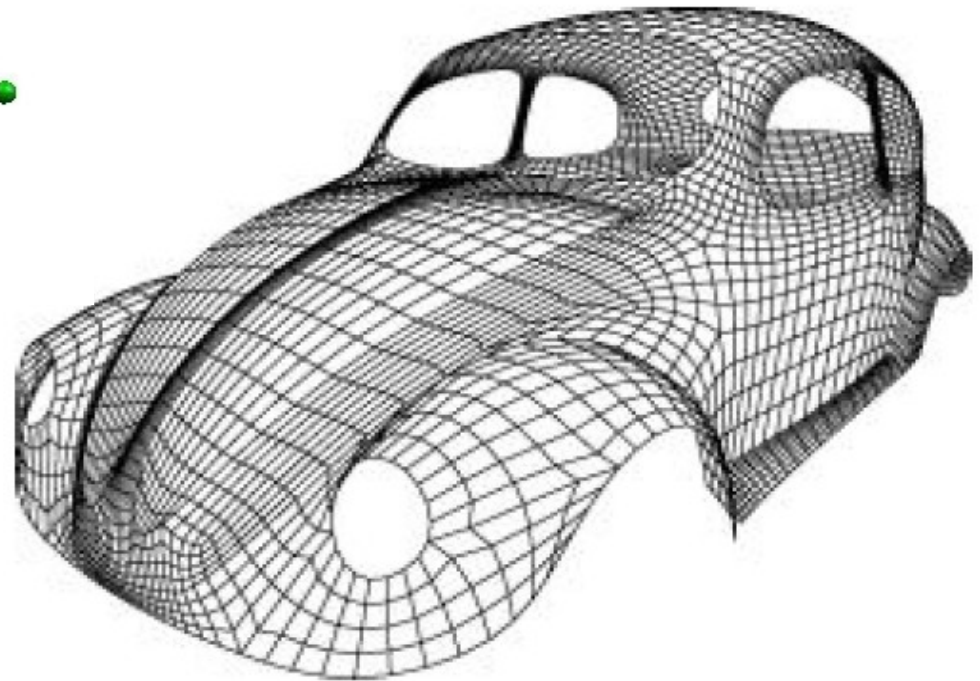
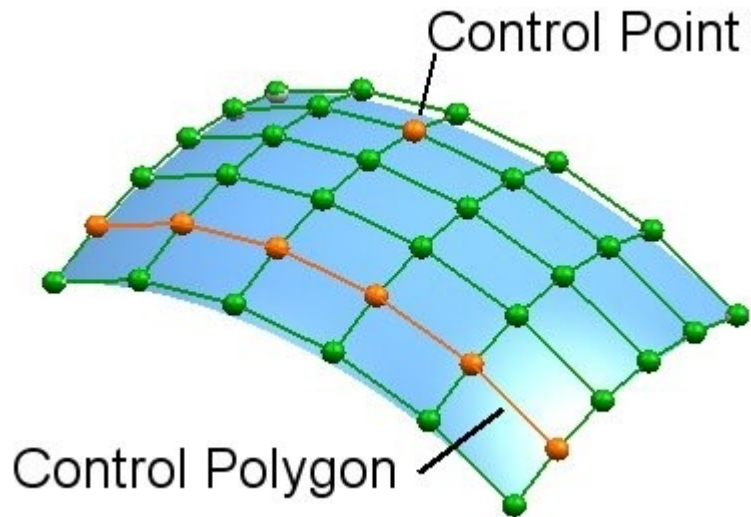




UPPSALA
UNIVERSITET

Curves and surfaces





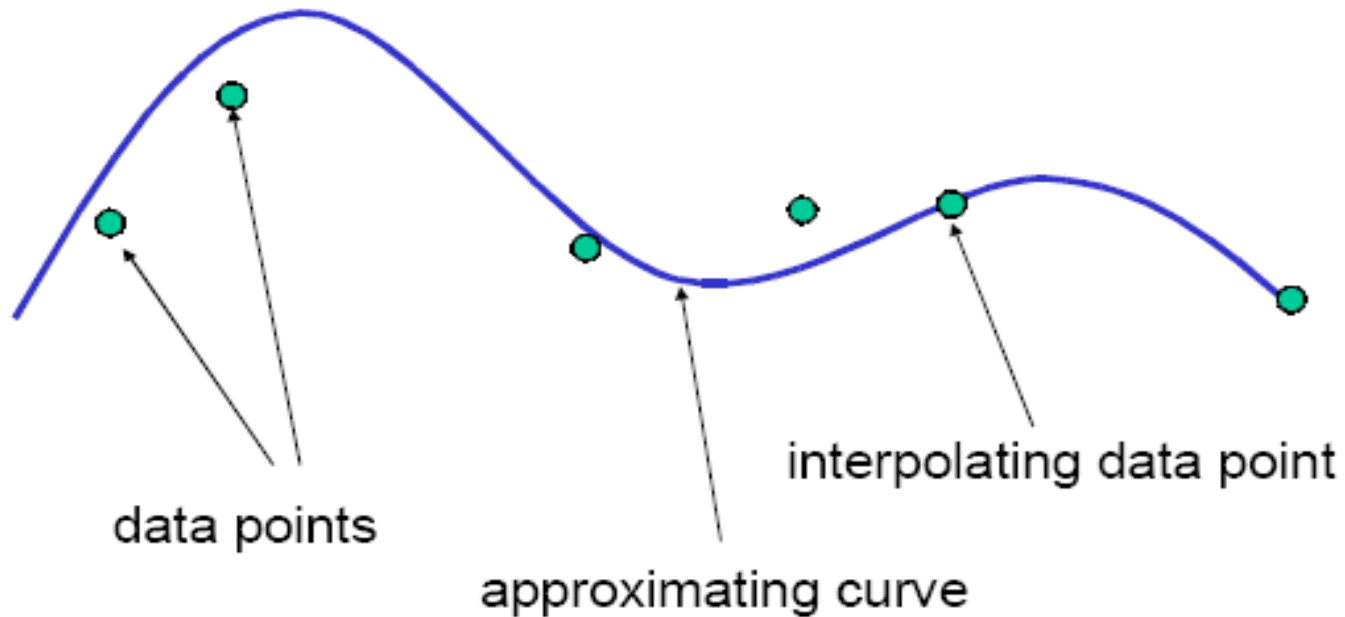
Escaping Flatland

- Until now we have worked with flat entities such as lines and flat polygons
 - Fit well with graphics hardware
 - = Mathematically simple
- But the world is not composed of flat entities
 - = Need curves and curved surfaces
 - = May only have need at the application level
 - = Implementation can render them approximately with flat primitives



UPPSALA
UNIVERSITET

Modelling with curves





What is a good representation?

- There are many ways to represent curves and surfaces
- Want a representation that is
 - Stable
 - = Smooth
 - = Easy to evaluate
 - = Must we interpolate or can we just come close to data?
 - Do we need derivatives?



Explicit representation

- Most familiar form of curve in 2D

$$y=f(x)$$

- Cannot represent all curves

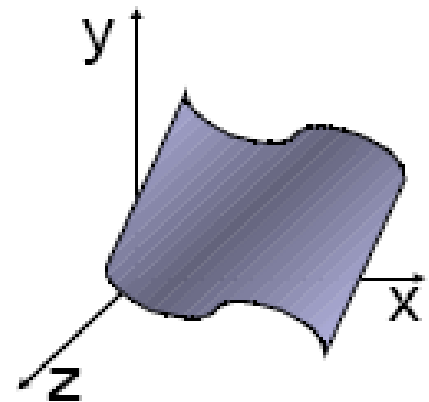
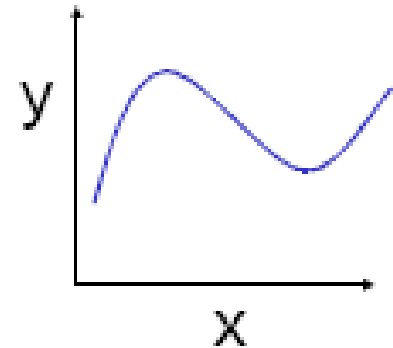
= Vertical lines

= Circles

- Extension to 3D

$$y=f(x), z=g(x)$$

The form $z = f(x,y)$ defines a surface





Implicit representation

- Two-dimensional curve(s)

$$g(x,y)=0$$

- Much more robust

- = Lines $ax+by+c=0$

- = Circles $x^2+y^2-r^2=0$

- In 3D, $g(x,y,z)=0$ defines a surface
 - Intersect two surface to get a curve
- In general, we cannot find y for a given x



Parametric curves

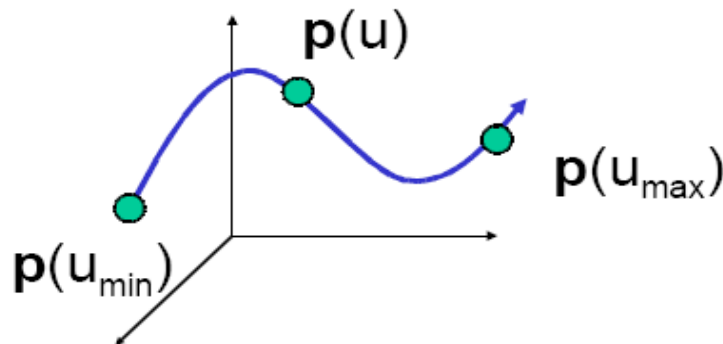
- Separate equation for each spatial variable

$$x=x(u)$$

$$y=y(u)$$

$$z=z(u)$$

$$\mathbf{p}(u)=[x(u), y(u), z(u)]^T$$





Selecting functions

- We want functions that
 - can approximate or interpolation known data
 - are easy to evaluate
 - are easy to differentiate
 - are smooth
 - ...



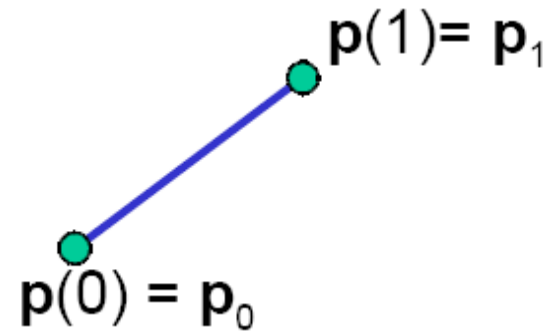
Parametric lines

UPPSALA
UNIVERSITET

We can normalize u to be over the interval $[0,1]$

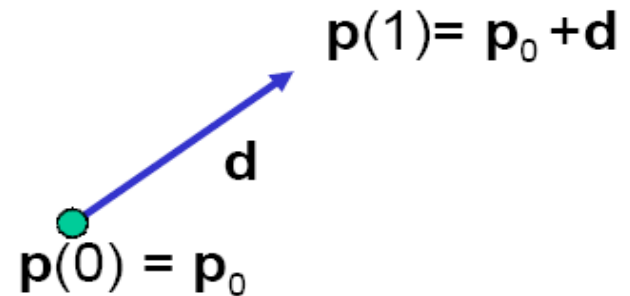
Line connecting two points \mathbf{p}_0 and \mathbf{p}_1

$$\mathbf{p}(u) = (1-u)\mathbf{p}_0 + u\mathbf{p}_1$$



Ray from \mathbf{p}_0 in the direction \mathbf{d}

$$\mathbf{p}(u) = \mathbf{p}_0 + u\mathbf{d}$$





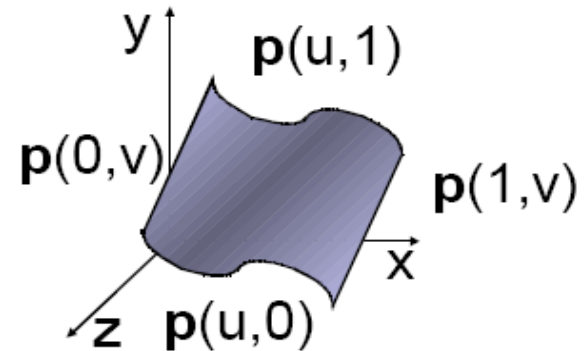
Parametric surfaces

Surfaces require 2 parameters

$$x=x(u,v)$$

$$y=y(u,v)$$

$$z=z(u,v)$$



$$\mathbf{p}(u,v) = [x(u,v), y(u,v), z(u,v)]^T$$



Normals

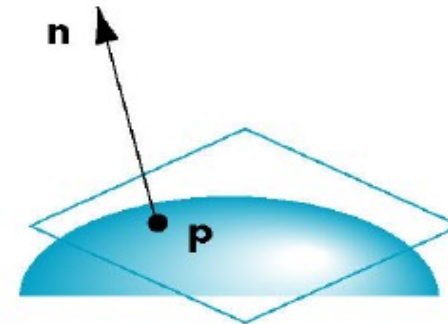
UPPSALA
UNIVERSITET

We can differentiate with respect to u and v to obtain the normal at any point \mathbf{p}

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

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

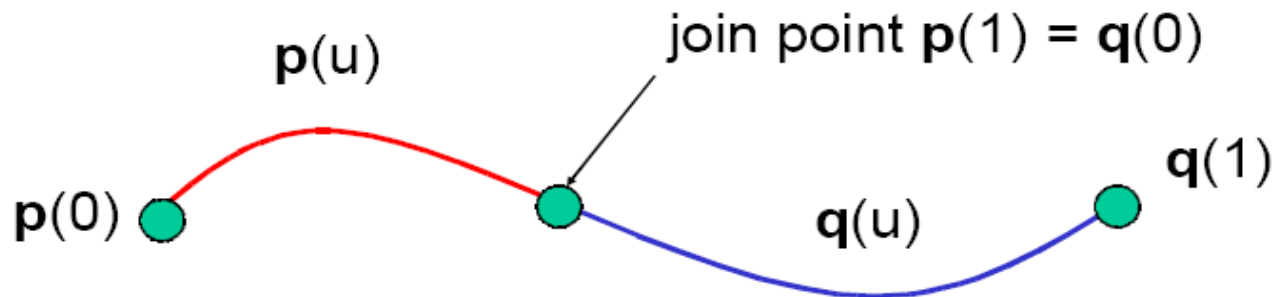
$$\mathbf{n} = \frac{\partial \mathbf{p}(u, v)}{\partial u} \times \frac{\partial \mathbf{p}(u, v)}{\partial v}$$





Curve segments

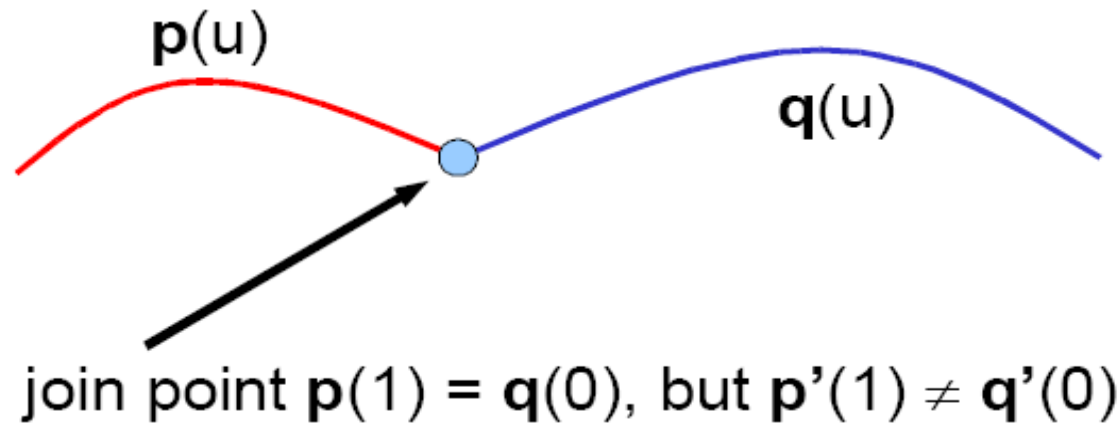
- After normalizing u , each curve is written
$$\mathbf{p}(u)=[x(u), y(u), z(u)]^T, 0 \leq u \leq 1$$
- In classical numerical methods, we design a single global curve.
- In computer graphics and CAD, it is better to design small connected curve *segments*.





Polynomials

- Easy to evaluate
- Continuous and differentiable everywhere
 - Must worry about continuity at join points, including continuity of derivatives





Cubic parametric polynomials

- Polynomials of degree three gives balance between ease of evaluation and flexibility in design

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

- Four coefficients to determine for each of x , y and z
- Seek four independent conditions for various values of u resulting in 4 equations in 4 unknowns for each of x , y and z
 - Conditions are a mixture of continuity requirements at the join points and conditions for fitting the data



Matrix-vector form

$$p(u) = \sum_{k=0}^3 c_k u^k$$

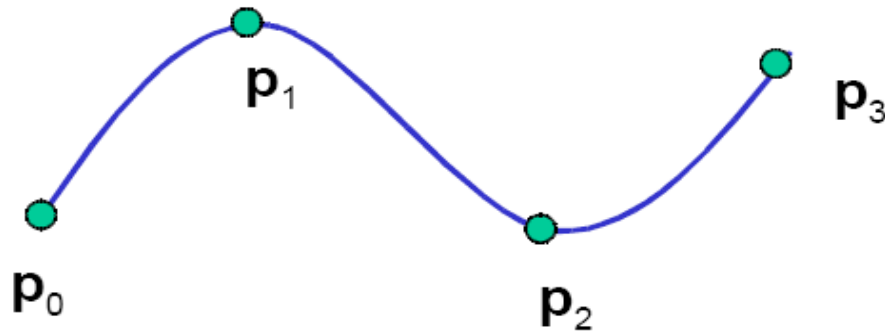
define $\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ $\mathbf{u} = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}$

then $p(u) = \mathbf{u}^T \mathbf{c} = \mathbf{c}^T \mathbf{u}$



Interpolating curve

- Given four data (control) points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$.
- Determine cubic $\mathbf{p}(u)$ which passes through them.
- Must find c_0, c_1, c_2, c_3 .





Interpolating equations

apply the interpolating conditions at $u=0, 1/3, 2/3, 1$

$$p_0 = p(0) = c_0$$

$$p_1 = p\left(\frac{1}{3}\right) = c_0 + \left(\frac{1}{3}\right)c_1 + \left(\frac{1}{3}\right)^2 c_2 + \left(\frac{1}{3}\right)^3 c_3$$

$$p_2 = p\left(\frac{2}{3}\right) = c_0 + \left(\frac{2}{3}\right)c_1 + \left(\frac{2}{3}\right)^2 c_2 + \left(\frac{2}{3}\right)^3 c_3$$

$$p_3 = p(1) = c_0 + c_1 + c_2 + c_3$$



Interpolating equations

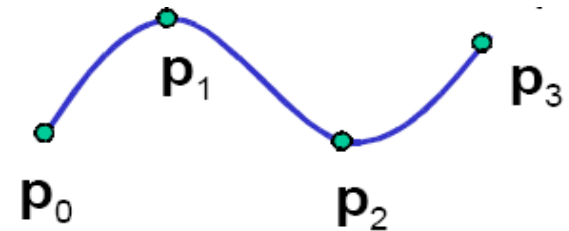
UPPSALA
UNIVERSITET

...or in matrix form with $p = Ac$, where

$$p = [p_0 \ p_1 \ p_2 \ p_3]^T$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \left(\frac{1}{3}\right) & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \\ 1 & \left(\frac{2}{3}\right) & \left(\frac{2}{3}\right)^2 & \left(\frac{2}{3}\right)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$p(u) = \sum_{k=0}^3 c_k u^k$$





Interpolating matrix

$$\mathbf{M}_I = \mathbf{A}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -5.5 & 9 & -4.5 & 1 \\ 9 & -22.5 & 18 & -4.5 \\ -4.5 & 13.5 & -13.5 & 4.5 \end{bmatrix}$$

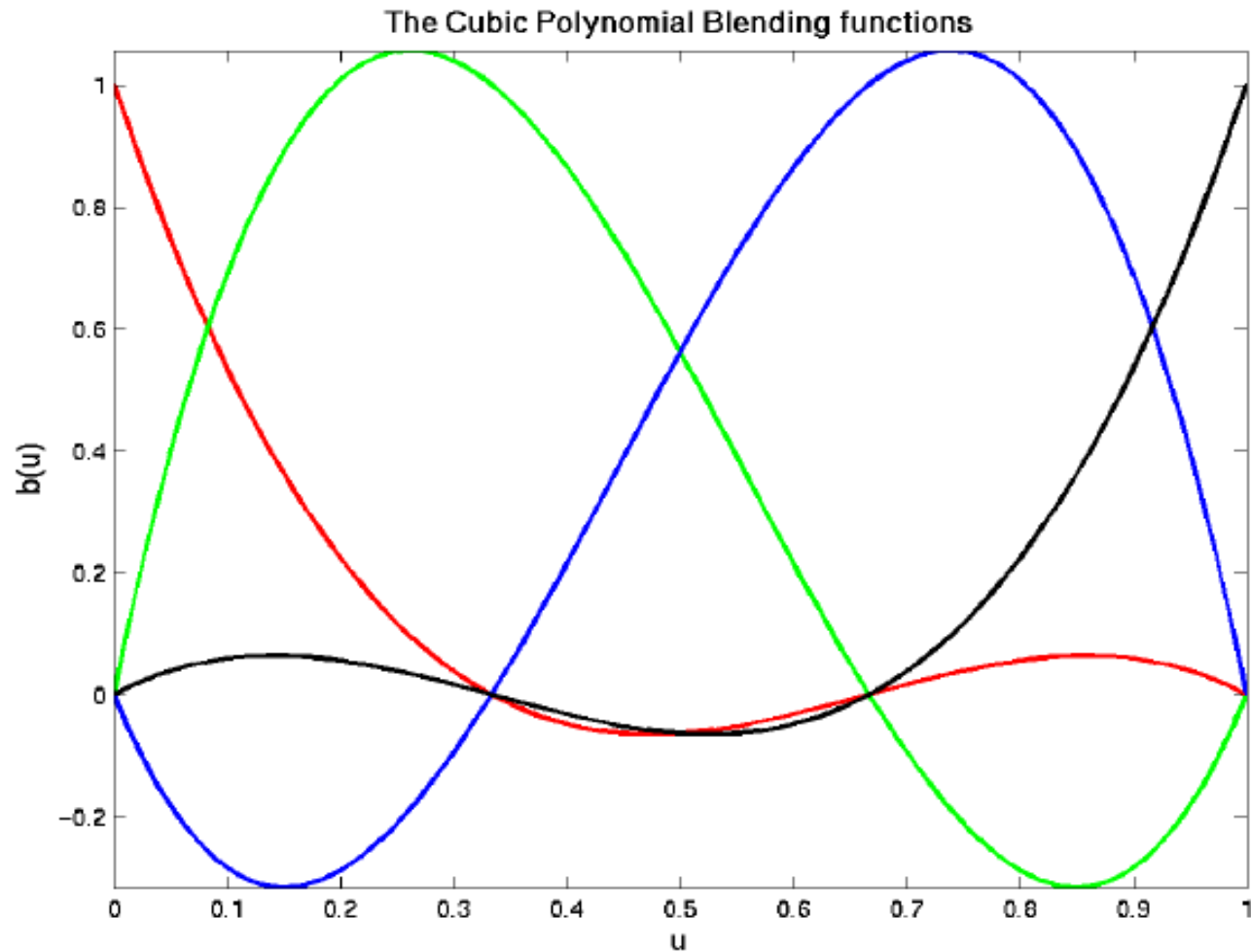
The resulting curve is $\mathbf{p}(u) = u^T \mathbf{c} = u^T \mathbf{M}_I \mathbf{p}$, which we can write $\mathbf{p}(u) = \mathbf{b}(u)^T \mathbf{p}$ where $\mathbf{b}(u) = \mathbf{M}_I^T u$. The polynomials $\mathbf{b}_i(u)$ are called blending polynomials, where each polynomial is a cubic.

$$\mathbf{p}(u) = b_0(u) \mathbf{p}_0 + b_1(u) \mathbf{p}_1 + b_2(u) \mathbf{p}_2 + b_3(u) \mathbf{p}_3 = \sum_{i=0}^3 b_i(u) \mathbf{p}_i$$



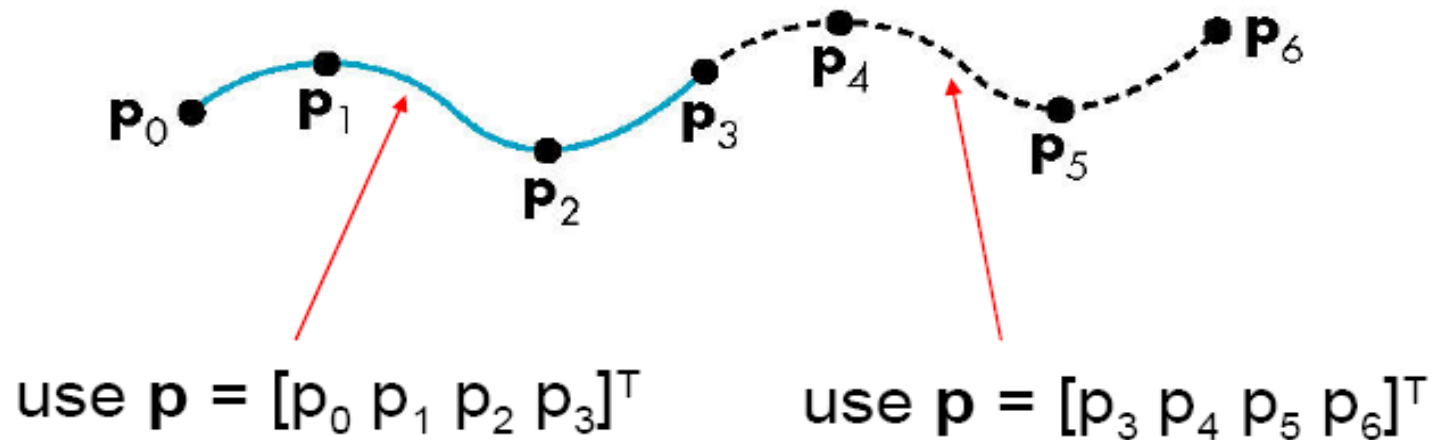
UPPSALA
UNIVERSITET

Interpolating blending functions





Multiple segments



We get continuity at join points but not continuity of derivatives

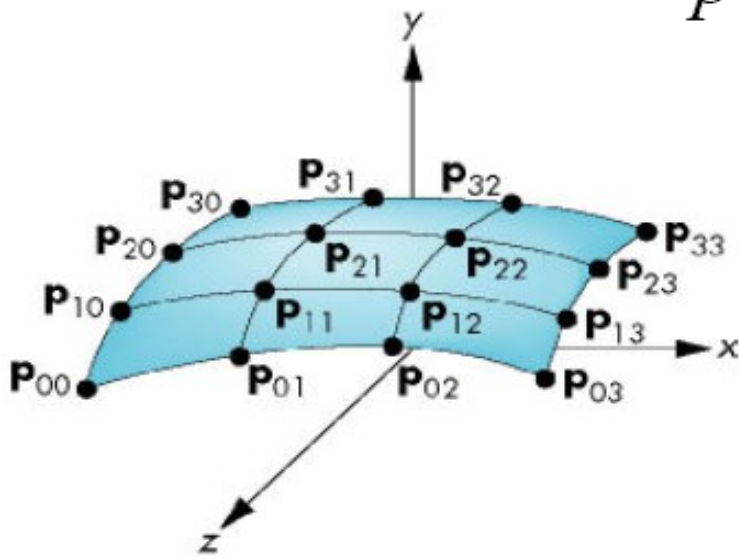


Interpolating patch

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

Need 16 conditions to determine the 16 coefficients c_{ij}

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





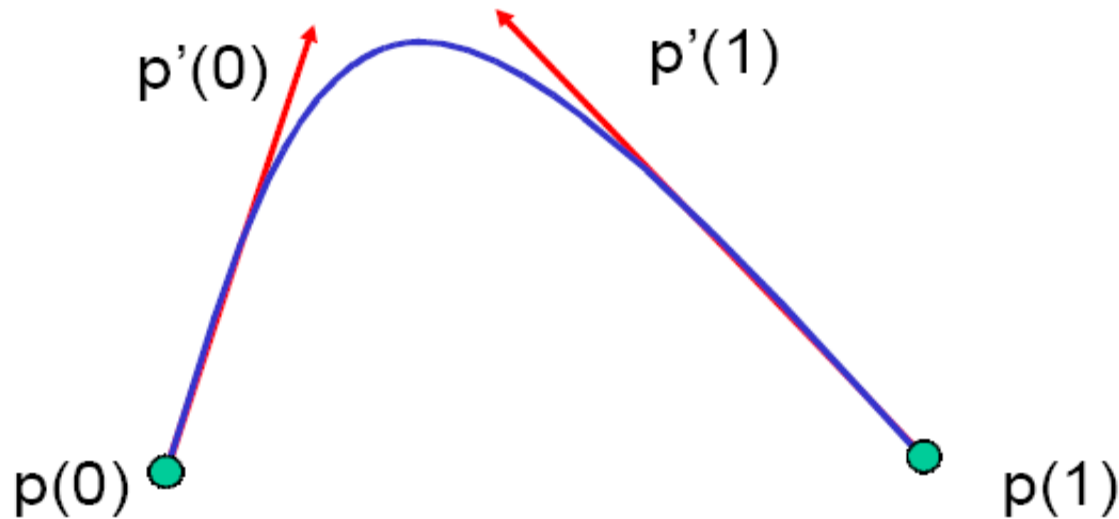
Other types of curves and surfaces

- How can we get around the limitations of the interpolating form
 - Lack of smoothness
 - Discontinuous derivatives at join points
- We have four conditions (for cubics) that we can apply to each segment
 - Use them other than for interpolation
 - Need only come close to the data



Hermite form

UPPSALA
UNIVERSITET



Use two interpolating conditions and two derivative conditions per segment.

Ensures continuity and first derivative continuity between segments



Hermite equations

Interpolating conditions are the same at ends

$$p(0) = p_0 = c_0$$

$$p(1) = p_3 = c_0 + c_1 + c_2 + c_3$$

Differentiating we find

$$p'(u) = c_1 + 2uc_2 + 3u^2c_3$$

Evaluating at endpoints

$$p'(0) = p'_0 = c_1$$

$$p'(1) = p'_3 = c_1 + 2c_2 + 3c_3$$



Hermite matrix

$$\mathbf{q} = \begin{bmatrix} p_0 \\ p_3 \\ p'_0 \\ p'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{c}$$

Solving, we find $\mathbf{c} = \mathbf{M}_H \mathbf{q}$ where \mathbf{M}_H is the Hermite matrix

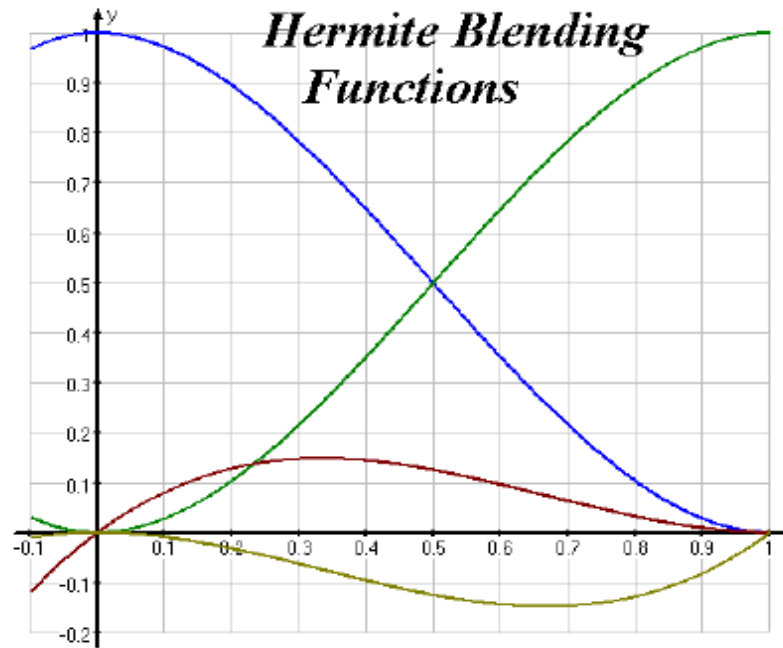
$$\mathbf{M}_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$



Hermite blending polynomials

$$p(u) = \mathbf{b}(u)^T \mathbf{q}$$

$$\mathbf{b}(u) = \begin{bmatrix} 2u^3 - 3u^2 + 1 \\ -2u^3 + 3u^2 \\ u^3 - 2u^2 + u \\ u^3 - u^2 \end{bmatrix}$$



Although these functions are smooth, the Hermite form is not used directly in Computer Graphics and CAD because we usually have control points but not derivatives. However, the Hermite form is the basis of the Bézier form



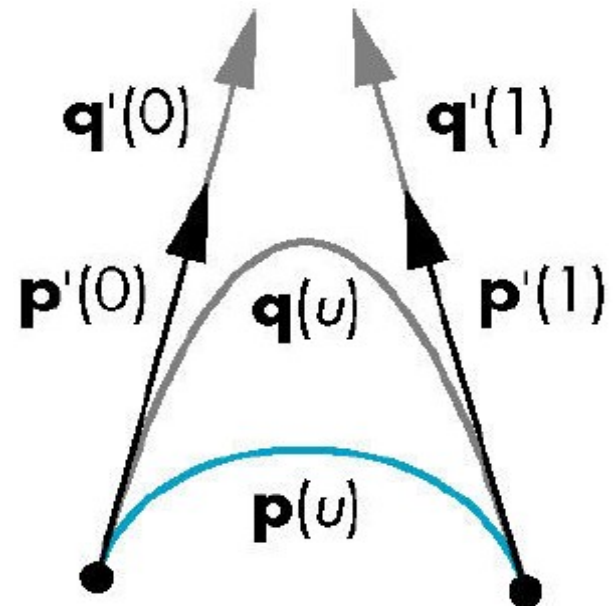
Parametric and geometric continuity

- We can require the derivatives of x , y , and z to each be continuous at join points (parametric continuity).
- Alternatively, we can only require that the tangents of the resulting curve be continuous (geometry continuity)
- The latter gives more flexibility as we need to satisfy only two conditions rather than three at each join point



Example

- Here the p and q have the same tangents at the ends of the segment but different derivatives.
- Generate different Hermite curves.
- This technique is used in drawing applications.



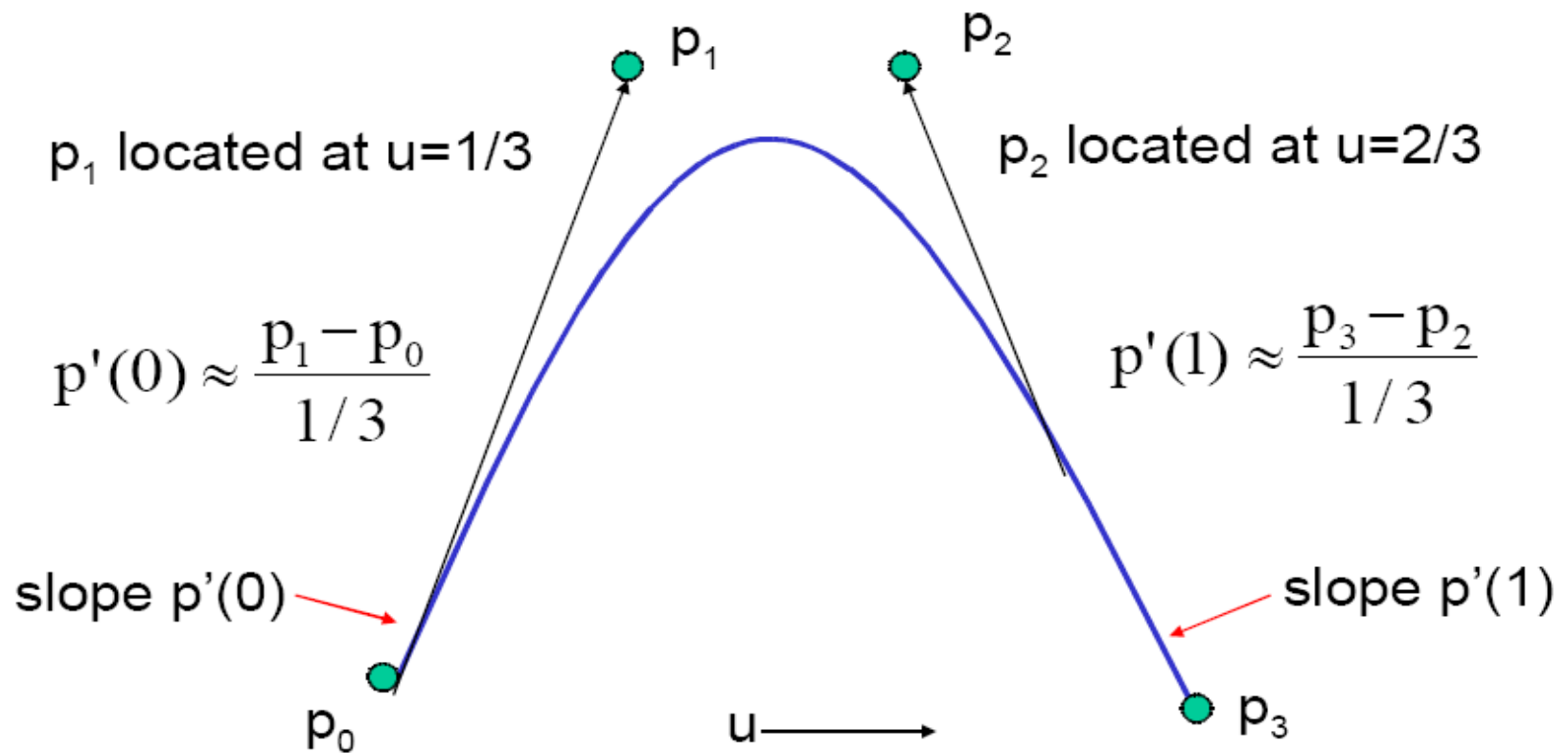


Bezier's idea

- In graphics and CAD, we do not usually have derivative data
- Bezier suggested using the same 4 data points as with the cubic interpolating curve to approximate the derivatives in the Hermite form



Approximating derivatives





Bézier equations

Interpolating conditions are the same

$$p(0) = p_0 = c_0$$

$$p(1) = p_3 = c_0 + c_1 + c_2 + c_3$$

Approximating derivative conditions

$$p'(0) = 3(p_1 - p_0) = c_0$$

$$p'(1) = 3(p_3 - p_2) = c_1 + 2c_2 + 3c_3$$

Solve four linear equations for $\mathbf{c} = \mathbf{M}_B \mathbf{p}$




Bézier matrix

$$\mathbf{M}_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

$$\mathbf{p}(u) = \mathbf{u}^T \mathbf{M}_B \mathbf{p} = \mathbf{b}(u)^T \mathbf{p}$$

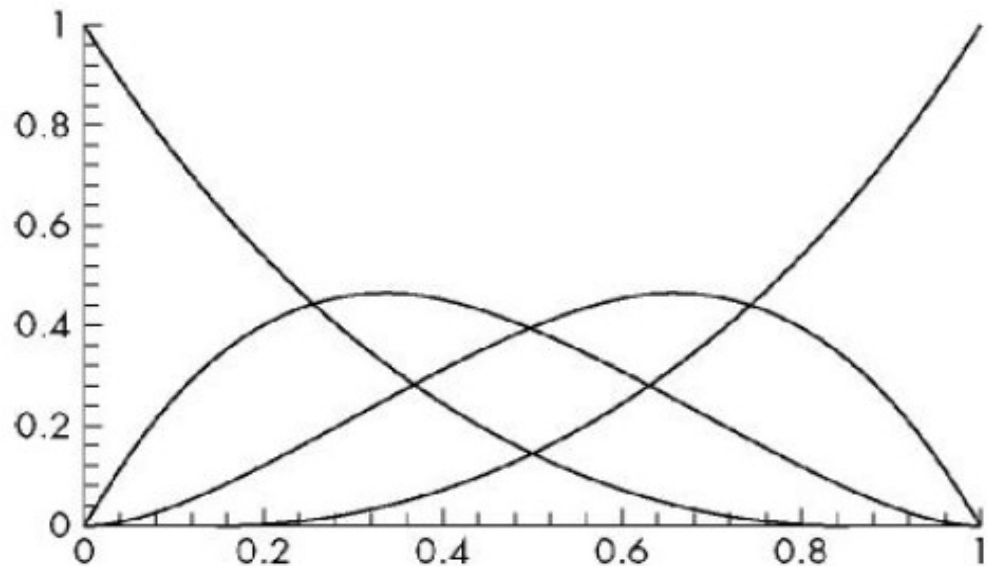
blending functions

A red arrow points from the text "blending functions" to the term $\mathbf{b}(u)^T$ in the equation above.



Bézier blending functions

$$\mathbf{b}(u) = \begin{bmatrix} (1-u)^3 \\ 3u(1-u)^2 \\ 2u^2(1-u) \\ u^3 \end{bmatrix}$$



Note that all zeros are at 0 and 1 which forces the functions to be smooth over (0,1)



Bernstein polynomials

- The blending functions are a special case of the Bernstein polynomials

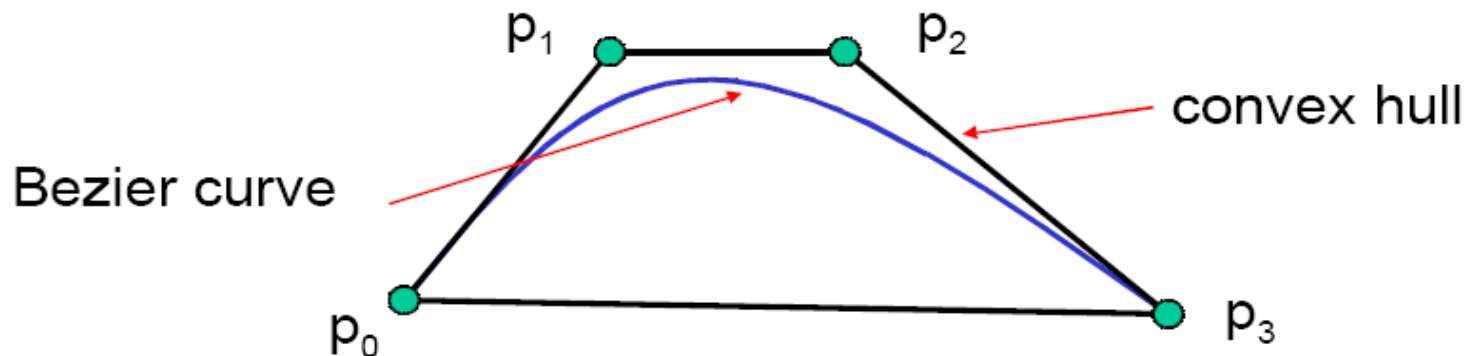
$$b_{kd}(u) = \frac{d!}{k!(d-k)!} u^k (1-u)^{d-k}$$

- These polynomials give the blending polynomials for any degree Bezier form
 - All zeros at 0 and 1
 - = For any degree they all sum to 1
 - They are all between 0 and 1 inside (0,1)



Convex hull property

- The properties of the Bernstein polynomials ensure that all Bezier curves lie in the convex hull of their control points
- Hence, even though we do not interpolate all the data, we cannot be too far away





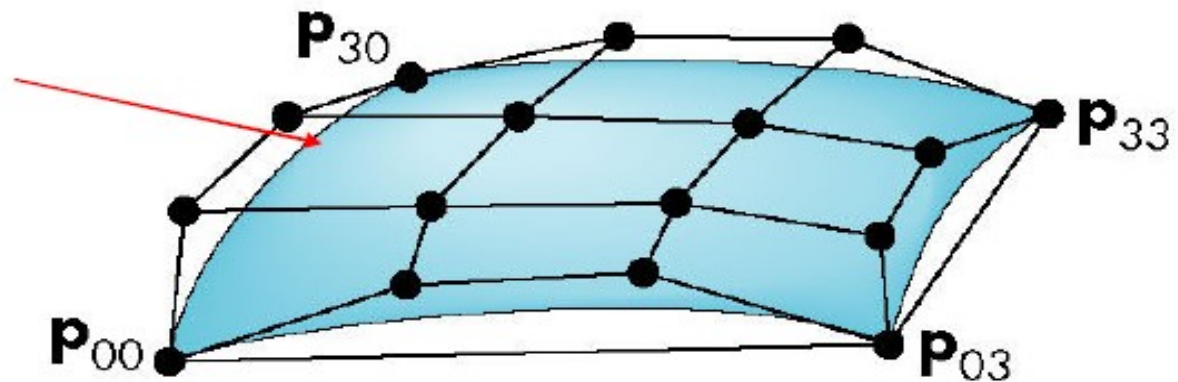
Bézier patches

UPPSALA
UNIVERSITET

Using same data array $\mathbf{P}=[p_{ij}]$ as with
interpolating form

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

Patch lies in
convex hull





Analysis

- Although the Bézier form is much better than the interpolating form, we have discontinuous derivatives at join points
- We need to have another representation to represent curves that even have continuous 2nd derivatives
- We need to do more work per segment and use more control points to apply more continuity conditions to each segment



B-splines

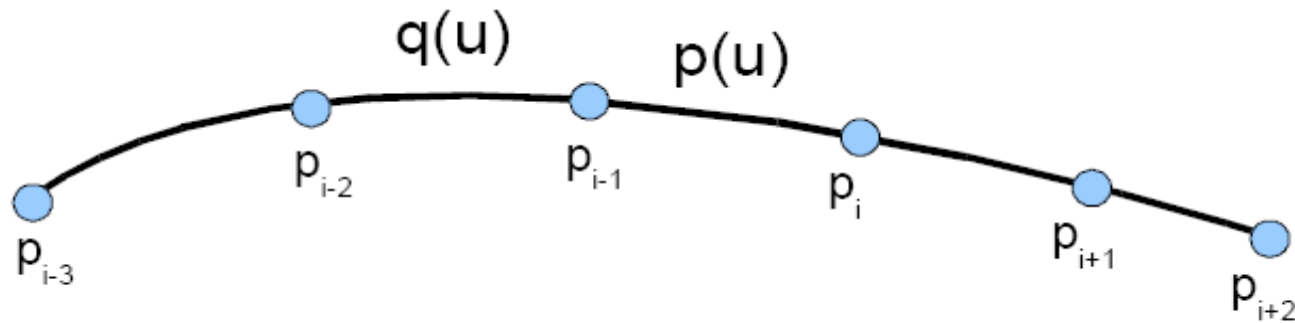
UPPSALA
UNIVERSITET

- **B**asis splines: use the data at $p=[p_{i-2} \ p_{i-1} \ p_i \ p_{i+1}]^T$ to define curve only between p_{i-1} and p_i
- We relax the condition that the curve segments should interpolate any of the control points and we utilize symmetry in approximating the tangents at the joints
- For cubics, we can have C^2 continuity at join points
- Three times as much work for curves and nine times as much work for surfaces



Cubic B-Spline

UPPSALA
UNIVERSITET



$$p(0) = q(1) = \frac{1}{6}(p_{i-2} + 4p_{i-1} + p_i) = c_0$$

$$p'(0) = q'(1) = \frac{1}{2}(p_i - p_{i-2}) = c_1$$

$$p(u) = \sum_{k=0}^3 c_k u^k$$

$$p(1) = q(1) = \frac{1}{6}(p_{i-1} + 4p_i + p_{i+1}) = c_0 + c_1 + c_2 + c_3$$

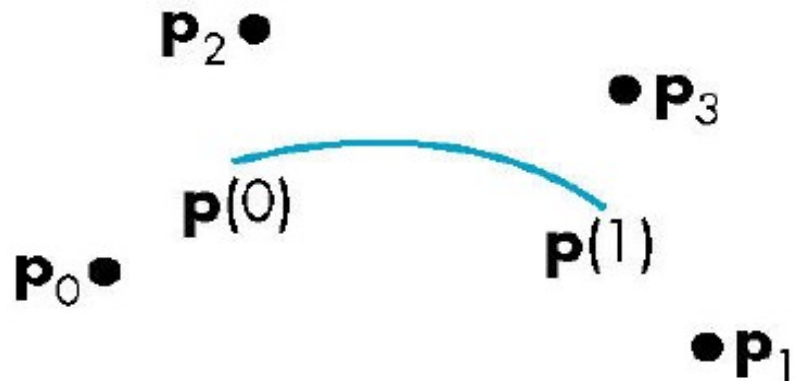
$$p'(1) = q'(1) = \frac{1}{2}(p_{i+1} - p_{i-1}) = c_1 + 2c_2 + 3c_3$$



Cubic B-Spline

$$p(u) = \mathbf{u}^T \mathbf{M}_s \mathbf{p} = \mathbf{b}(u)^T \mathbf{p}$$

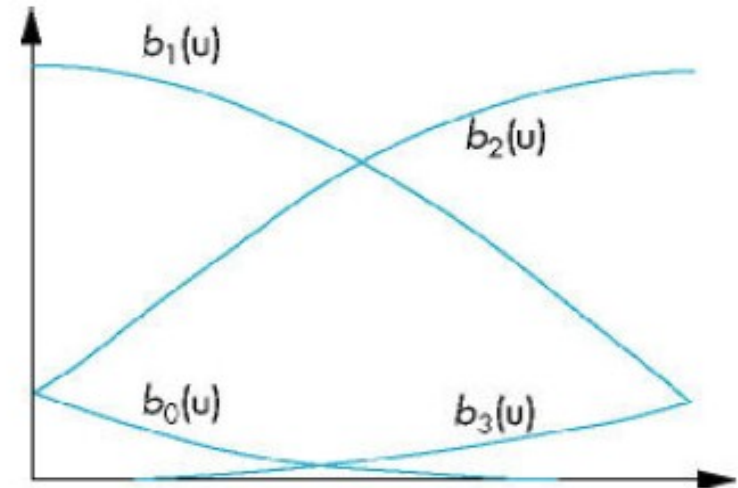
$$\mathbf{M}_s = \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$



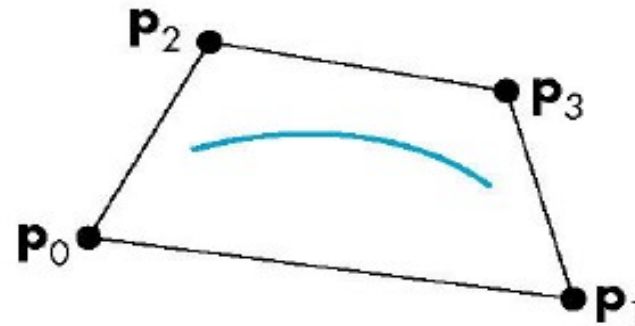


Blending functions

$$\mathbf{b}(u) = \frac{1}{6} \begin{bmatrix} (1-u)^3 \\ 4-6u^2+3u^3 \\ 1+3u+3u^2-3u^2 \\ u^3 \end{bmatrix}$$



convex hull property

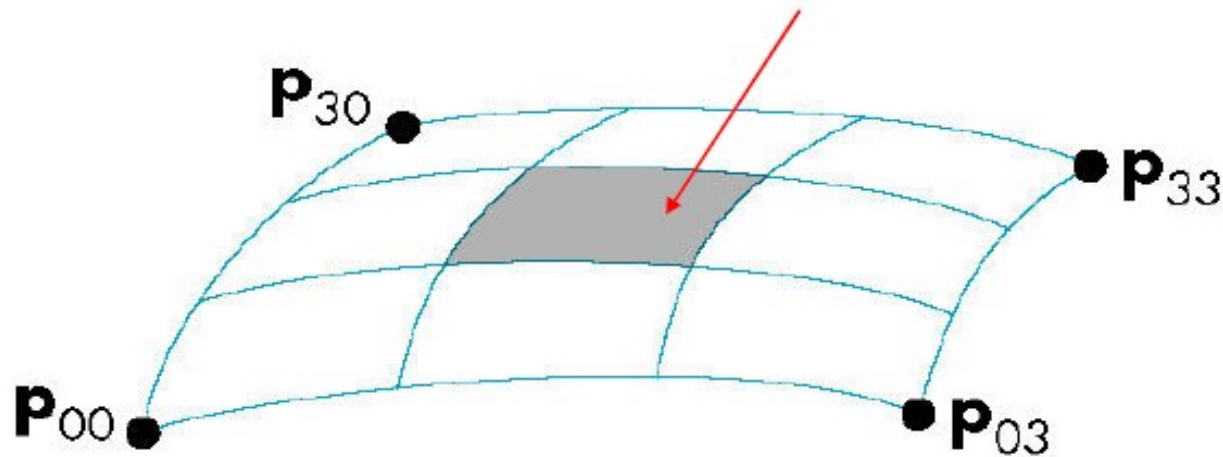




B-spline patches

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

defined over only 1/9 of region





Splines and basis

- If we examine the cubic B-spline from the perspective of each control (data) point, each interior point contributes (through the blending functions) to four segments
- We can rewrite $p(u)$ in terms of the data points as

$$p(u) = \sum B_i(u) p_i$$

defining the basis functions $\{B_i(u)\}$

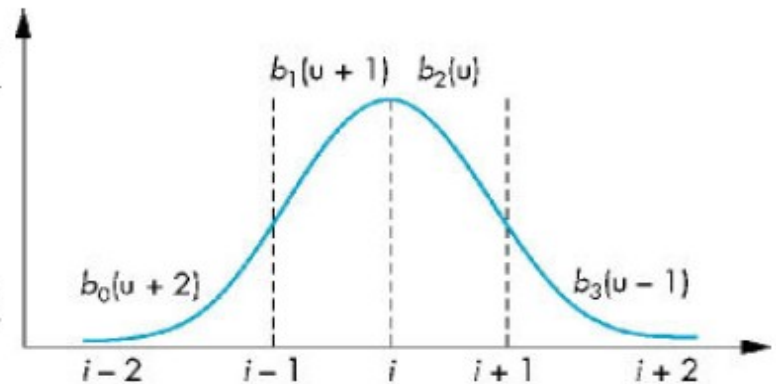


Basis functions

UPPSALA
UNIVERSITET

In terms of the blending polynomials

$$B_i(u) = \begin{cases} 0 & u < i-2 \\ b_0(u+2) & i-2 \leq u < i-1 \\ b_1(u+1) & i-1 \leq u < i \\ b_2(u) & i \leq u < i+1 \\ b_3(u-1) & i+1 \leq u < i+2 \\ 0 & u \geq i+2 \end{cases}$$





Generalizing Splines

- We can extend to splines of any degree
- Data and conditions do not have to be given at equally spaced values (the *knots*)
 - Nonuniform and uniform splines
 - = Can have repeated knots
 - Can force spline to interpolate points



NURBS

- Non-uniform Rational B-Spline curves and surfaces add a fourth variable w to x, y, z
 - = Can interpret as weight to give more importance to some control data
 - = Can also interpret as moving to homogeneous coordinate
- Requires a perspective division
 - = NURBS act correctly for perspective viewing
- Quadrics are a special case of NURBS



Rendering

UPPSALA
UNIVERSITET

- Simplest method to render a polynomial curve is to evaluate the polynomial at many points and form an approximating polyline
- For surfaces we can form an approximating mesh of triangles or quadrilaterals
- Use Horner's method to evaluate polynomials
$$p(u) = c_0 + u(c_1 + u(c_2 + uc_3))$$

= 3 multiplications/evaluation for cubic



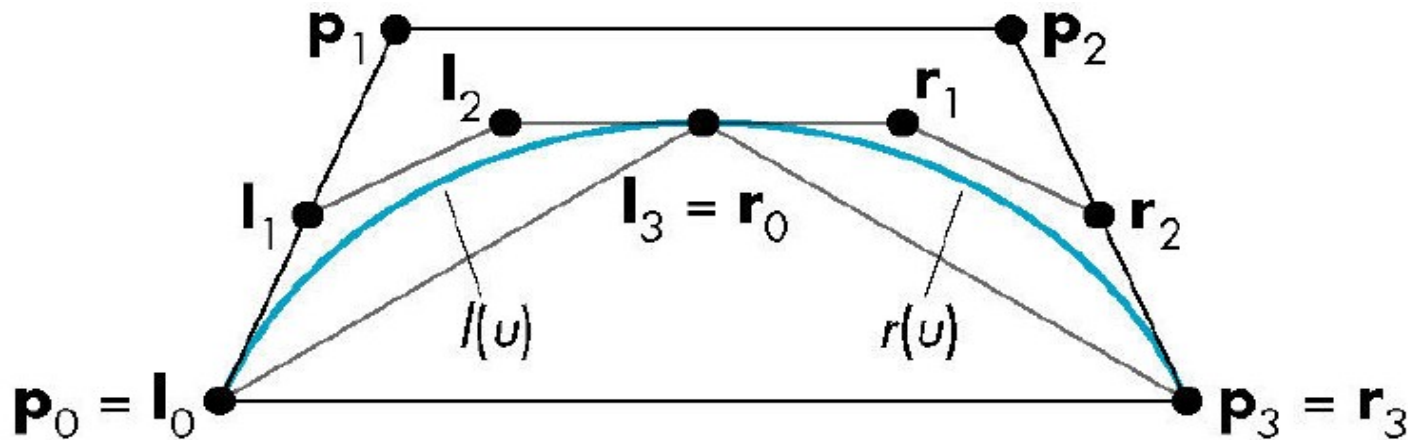
Recursive subdivision

- We can use the convex hull property of Bézier curves to obtain an efficient recursive method that does not require any function evaluations
- Uses only the values at the control points
- Based on the idea that “any polynomial and any part of a polynomial is a Bézier polynomial for properly chosen control data”



Splitting a cubic Bézier

p_0, p_1, p_2, p_3 determine a cubic Bézier polynomial and its convex hull

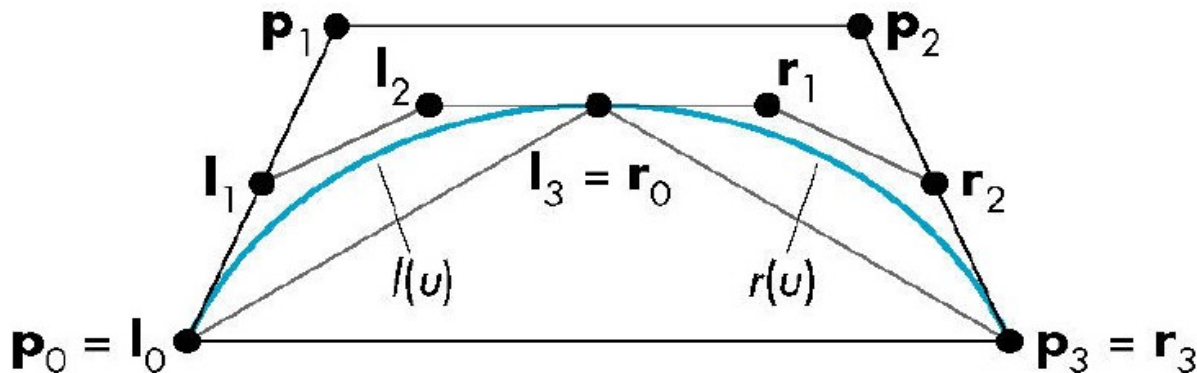


Consider left half $l(u)$ and right half $r(u)$



$l(u)$ and $r(u)$

- Since $l(u)$ and $r(u)$ are Bezier curves, we should be able to find two sets of control points $\{l_0, l_1, l_2, l_3\}$ and $\{r_0, r_1, r_2, r_3\}$ that determine them.
- $\{l_0, l_1, l_2, l_3\}$ and $\{r_0, r_1, r_2, r_3\}$ each have a convex hull that is closer to $p(u)$ than the convex hull of $\{p_0, p_1, p_2, p_3\}$
- Repeating recursively, we get a good approximation





Efficient algorithm

UPPSALA
UNIVERSITET

$$l_0 = p_0$$

$$r_3 = p_3$$

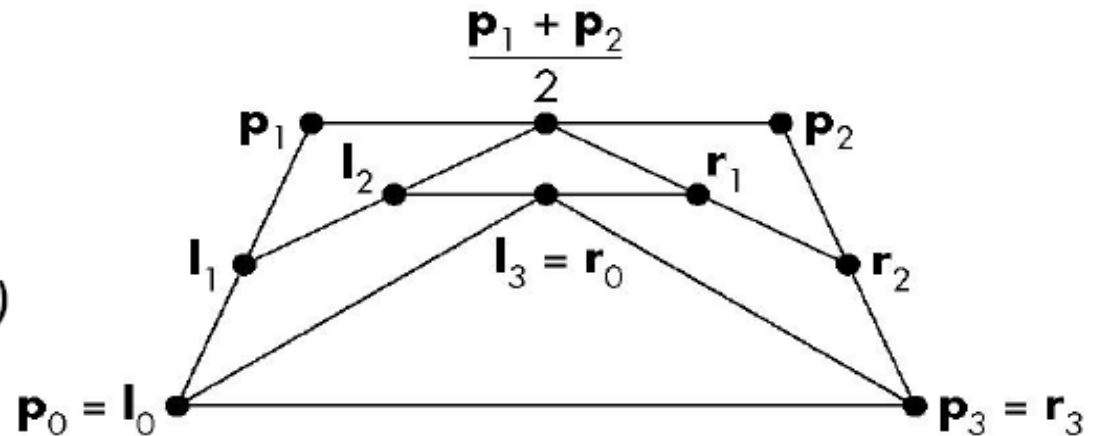
$$l_1 = \frac{1}{2}(p_0 + p_1)$$

$$r_1 = \frac{1}{2}(p_2 + p_3)$$

$$l_2 = \frac{1}{2}(l_1 + \frac{1}{2}(p_1 + p_2))$$

$$r_1 = \frac{1}{2}(r_2 + \frac{1}{2}(p_1 + p_2))$$

$$l_3 = r_0 = \frac{1}{2}(l_2 + r_1)$$



Requires only shifts and adds!



Every curve is a Bézier curve

- We can render a given polynomial using the recursive method if we find control points for its representation as a Bezier curve
- Suppose that $p(u)$ is given as an interpolating curve with control points \mathbf{q}

$$p(u) = \mathbf{u}^T \mathbf{M}_I \mathbf{q}$$

- There exist Bezier control points \mathbf{p} such that

$$p(u) = \mathbf{u}^T \mathbf{M}_B \mathbf{p}$$

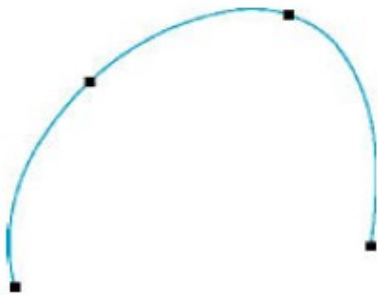
- Equating and solving, we find $\mathbf{p} = \mathbf{M}_B^{-1} \mathbf{M}_I$



Matrices

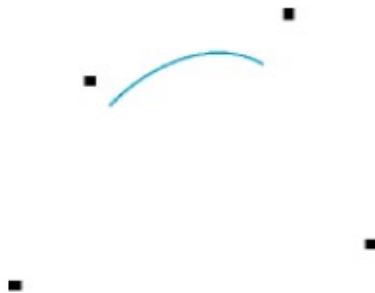
UPPSALA
UNIVERSITET

Interpolating to Bezier



$$\mathbf{M}_B^{-1} \mathbf{M}_I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{5}{6} & 3 & -\frac{3}{2} & \frac{1}{3} \\ \frac{1}{3} & -\frac{3}{2} & 3 & -\frac{5}{6} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

B-spline to Bezier



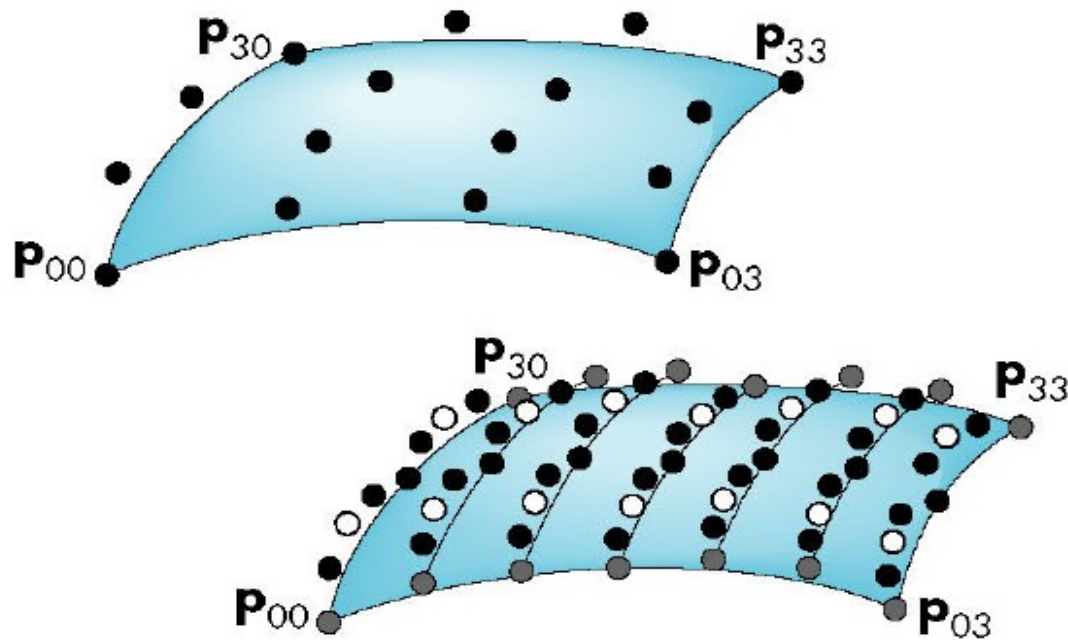
$$\mathbf{M}_B^{-1} \mathbf{M}_S = \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix}$$



Surfaces

UPPSALA
UNIVERSITET

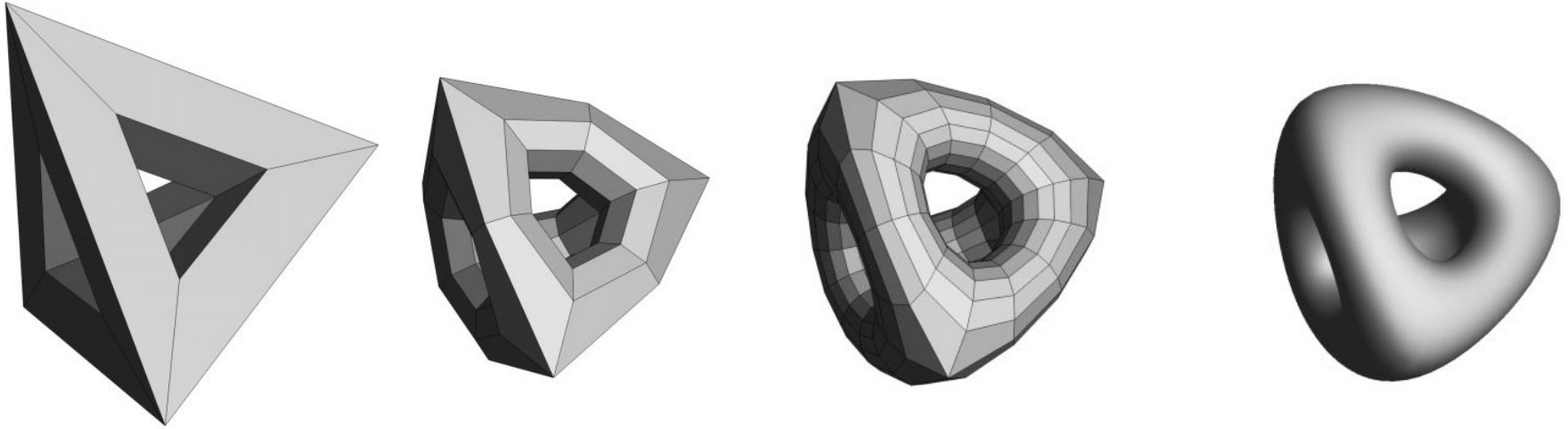
- Can apply the recursive method to surfaces if we recall that for a Bezier patch curves of constant u (or v) are Bezier curves in u (or v)





Subdivision surfaces

UPPSALA
UNIVERSITET



E. Catmull and J. Clark.

Recursively generated B-spline surfaces on arbitrary topological meshes.
Computer Aided Design, 10(6):350–355, 1978.

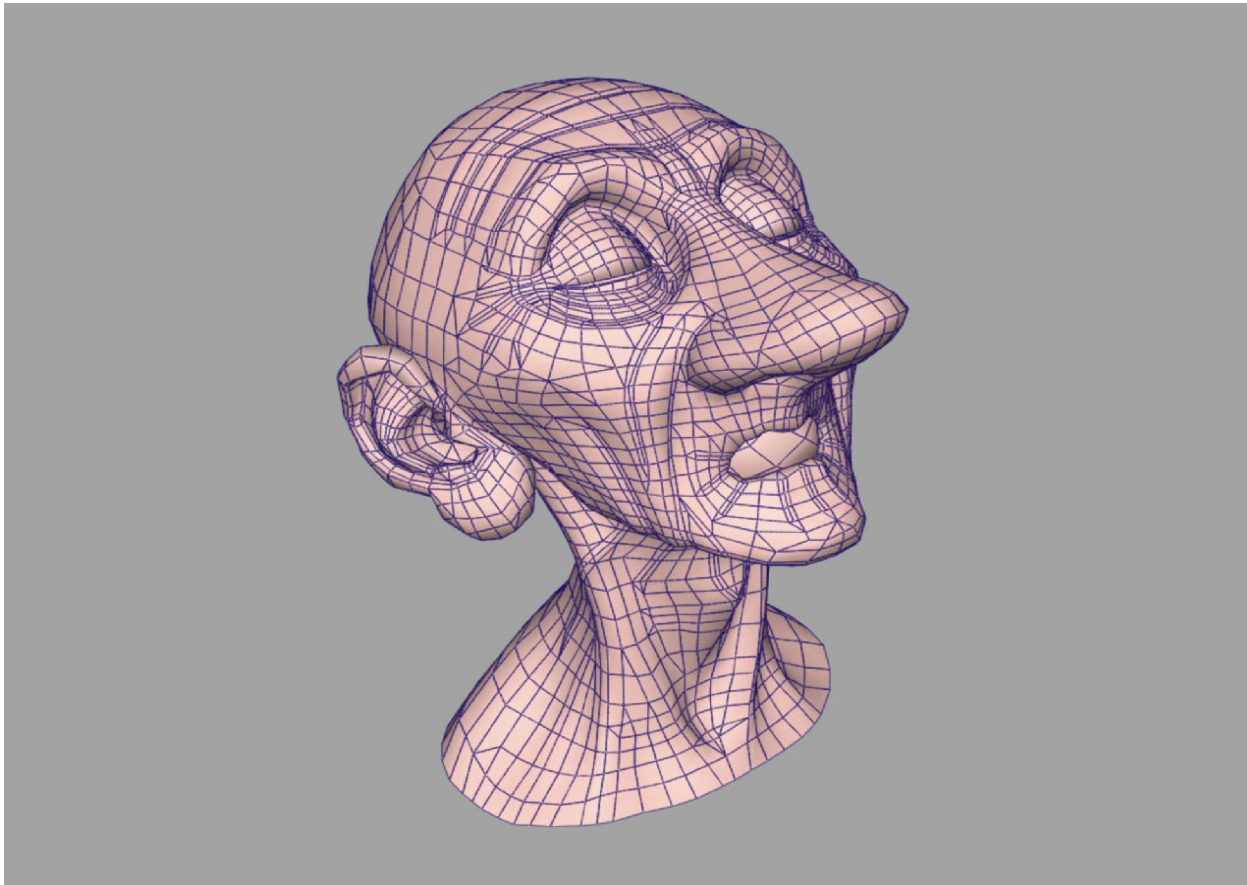
DeRose, T., Kass, M., and Truong, T.

Subdivision surfaces in character animation.
In Proceedings of SIGGRAPH '98.



UPPSALA
UNIVERSITET

Subdivision surfaces

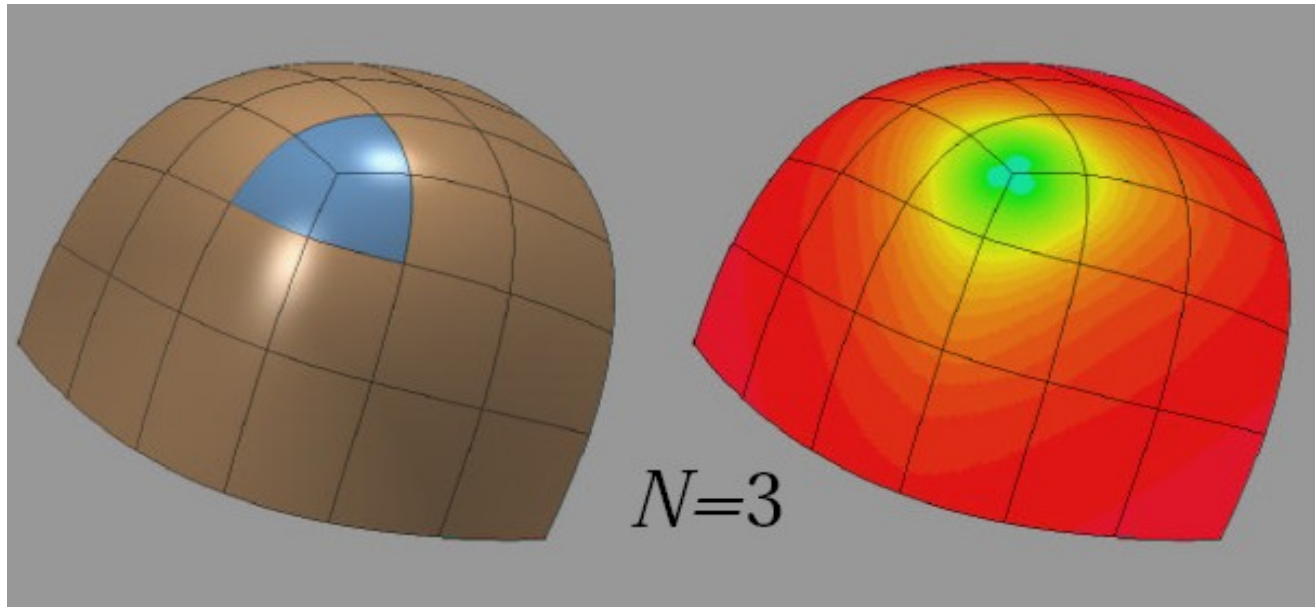


“Geri’s game”, Pixar 1997



UPPSALA
UNIVERSITET

Exact evaluation of SDS



Stam, J. 1998.

Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values.

In Proceedings of SIGGRAPH '98. ACM.



UPPSALA
UNIVERSITET

Available thesis projects at CBA

- 2 new projects will be announced this week.
- If you are interested, contact professor Ewert Bengtsson (ewert@cb.uu.se)



UPPSALA
UNIVERSITET

Using image analysis for identifying old coins and medals.





UPPSALA
UNIVERSITET

Using image analysis to track face motion.



In cooperation with Northern Light Studios