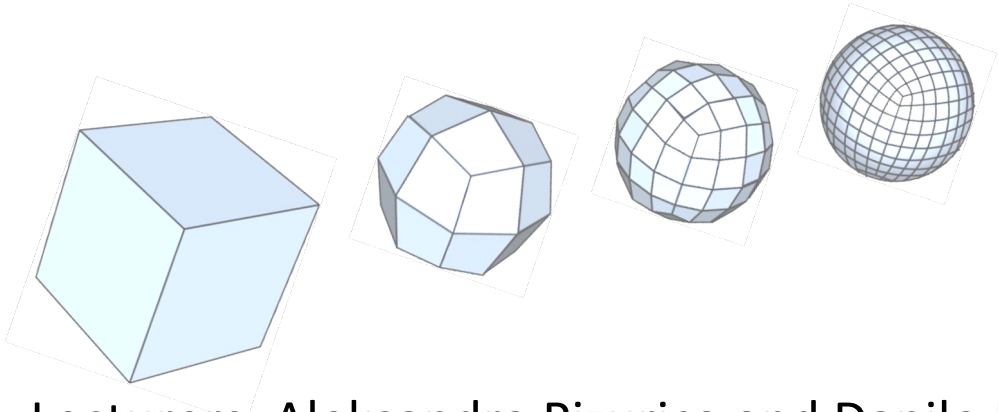
FACULTY OF ENGINEERING AND ARCHITECTURE

E016712: Computer Graphics

Curves and Surfaces

Part 2: Splines and Subdivision



Lecturers: Aleksandra Pizurica and Danilo Babin

GHENT UNIVERSITY

Overview

- Cubic B-Splines
- NURBS
- Rendering curves and surfaces
- Subdivision surfaces

Based on:

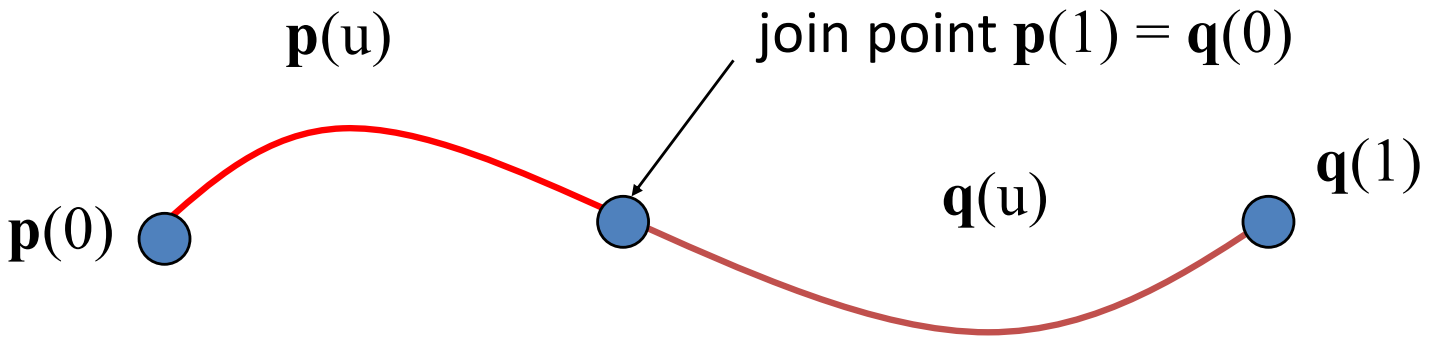
E. Angel and D. Shreiner: *Interactive Computer* Graphics – *A Top Down Approach with Shader-Based OpenGL* (6th ed). Chapter 10: Curves and Surfaces

Reminder: Curve Segments

After normalizing u, each curve is written

$$\mathbf{p}(u)=[x(u), y(u), z(u)]^{\mathrm{T}}, \quad 1 \geq u \geq 0$$

While in classical numerical methods we design a single global curve, in computer graphics it is better to design small connected curve segments



Advantages of designing each segment individually

- working interactively; affecting the shape only where we want
- local control implies stability: small changes in parameters (independent var.) cause small changes in dependent variables

Reminder: Parametric Polynomial Curves

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

$$\mathbf{p}(u) = \mathbf{c}_0 + \mathbf{c}_1 u + \mathbf{c}_2 u^2 + \mathbf{c}_3 u^3 = \sum_{k=0}^{3} \mathbf{c}_k u^k$$

$$\mathbf{u} = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}, \qquad \mathbf{c} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix}, \qquad \mathbf{c}_k = \begin{bmatrix} c_{kx} \\ c_{ky} \\ c_{kz} \end{bmatrix} \qquad \mathbf{p}(u) = \mathbf{u}^T \mathbf{c} = \mathbf{c}^T \mathbf{u}$$

Let $\mathbf{p} = [\mathbf{p}_0 \;\; \mathbf{p}_1 \;\; \mathbf{p}_2 \;\; \mathbf{p}_3]$ denote control points

From the particular conditions: $\mathbf{p} = \mathbf{A}\mathbf{c} \Rightarrow \mathbf{c} = \mathbf{A}^{-1}\mathbf{p} = \mathbf{M}\mathbf{p}$

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

Blending polynomials

Interpolating form, Bézier form and beyond

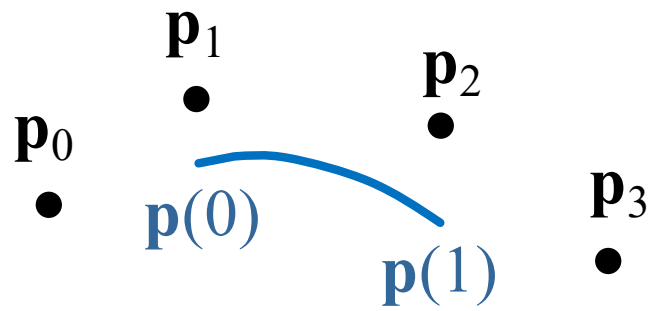
Although the Bézier form is much better than the interpolating form, the derivatives are not continuous at join points.

Can we do better?

- Go to higher order Bézier curves
  - More work
  - Derivative continuity still only approximate
  - Supported by OpenGL
- Splines --> this lesson

B-Splines

Cubic B-Splines



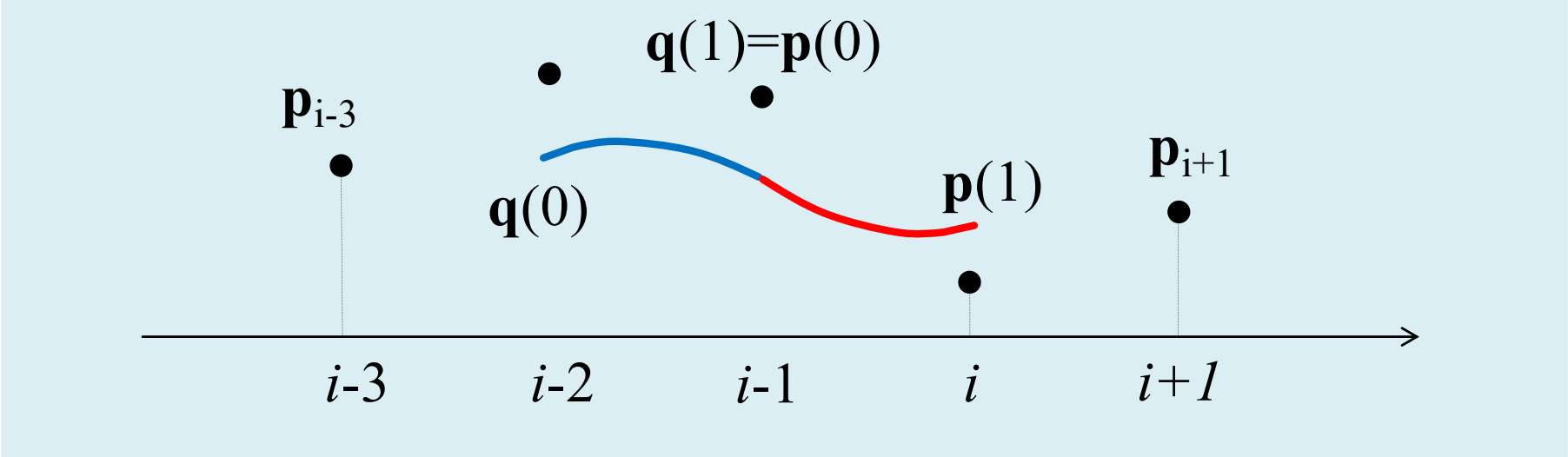
Use the data at $\mathbf{p}=[\mathbf{p}_{i-2}\ \mathbf{p}_{i-1}\ \mathbf{p}_i\ \mathbf{p}_{i-1}]^T$ to define a curve only between $\mathbf{p}_{i-1}$ and $\mathbf{p}_i$.

This allows to apply more continuity conditions to each segment.

For cubic splines, we can have continuity of function, first and second derivatives at join points.

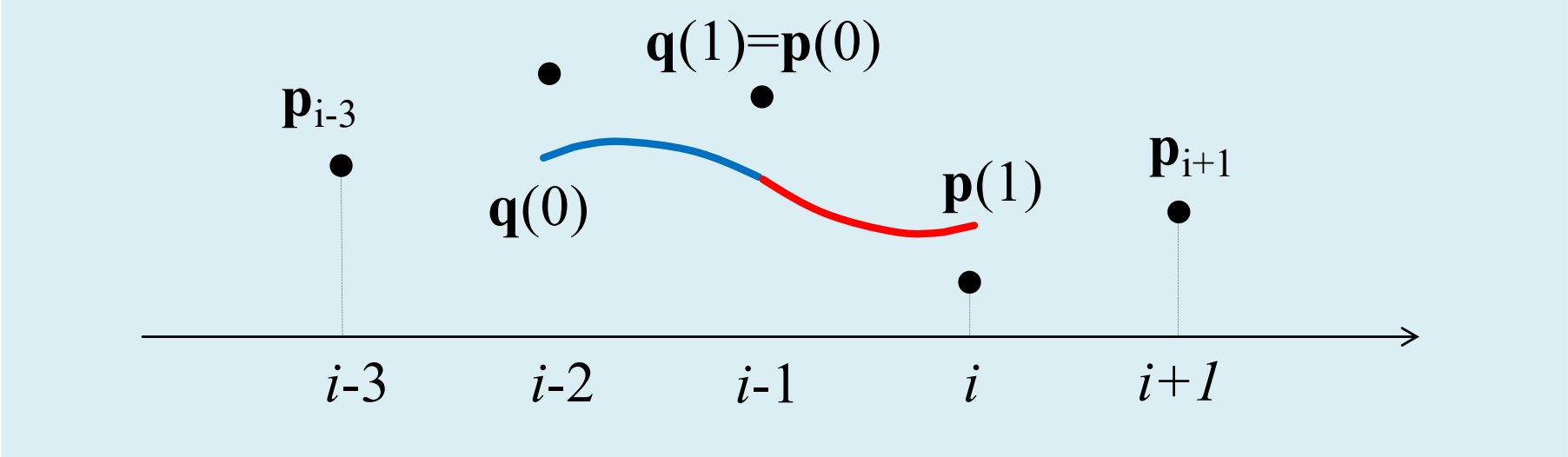
Cubic B-Splines

$$\mathbf{p}(u) = \mathbf{c}_0 + \mathbf{c}_1 u + \mathbf{c}_2 u^2 + \mathbf{c}_3 u^3$$



Cubic B-Splines

$$\mathbf{p}(u) = \mathbf{c}_0 + \mathbf{c}_1 u + \mathbf{c}_2 u^2 + \mathbf{c}_3 u^3$$



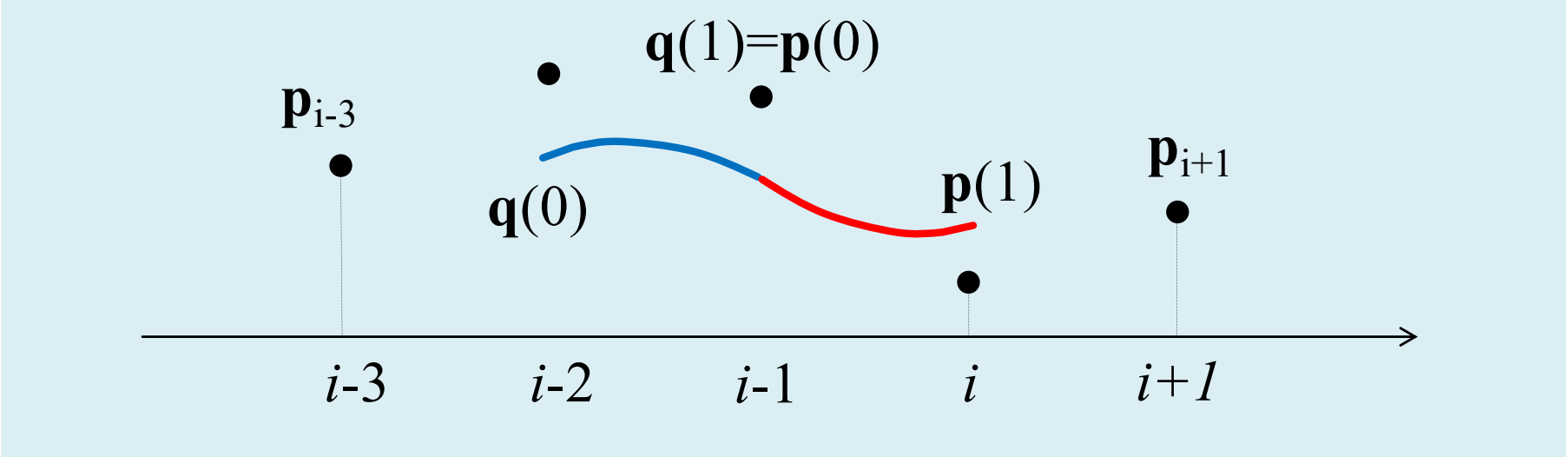
Two possible conditions:

$$\mathbf{p}(0) = \mathbf{q}(1) = \frac{1}{6}(\mathbf{p}_{i-2} + 4\mathbf{p}_{i-1} + \mathbf{p}_i)$$

$$\mathbf{p}'(0) = \mathbf{q}'(1) = \frac{1}{2}(\mathbf{p}_i - \mathbf{p}_{i-2})$$

Cubic B-Splines

$$\mathbf{p}(u) = \mathbf{c}_0 + \mathbf{c}_1 u + \mathbf{c}_2 u^2 + \mathbf{c}_3 u^3$$

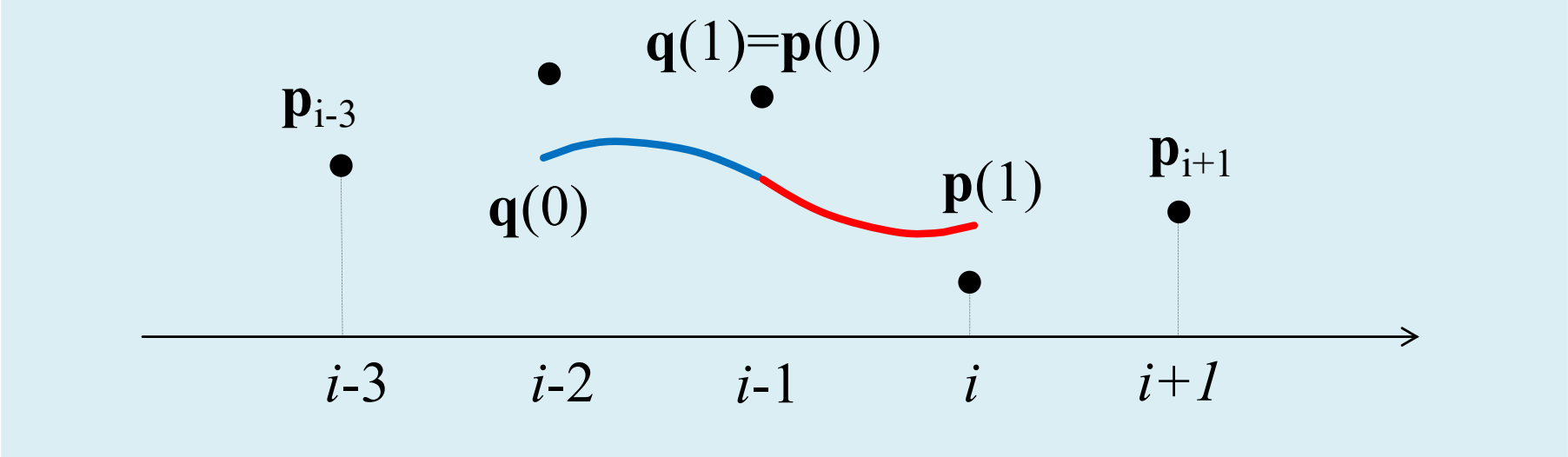


Two possible conditions:

$$\left.\begin{aligned} \mathbf{p}(0) &= \mathbf{q}(1) = \frac{1}{6}(\mathbf{p}_{i-2} + 4\mathbf{p}_{i-1} + \mathbf{p}_i) \\ \mathbf{p}'(0) &= \mathbf{q}'(1) = \frac{1}{2}(\mathbf{p}_i - \mathbf{p}_{i-2}) \end{aligned}\right\} \Rightarrow \boxed{\begin{aligned} \mathbf{c}_0 &= \frac{1}{6}(\mathbf{p}_{i-2} + 4\mathbf{p}_{i-1} + \mathbf{p}_i) \\ \mathbf{c}_1 &= \frac{1}{2}(\mathbf{p}_i - \mathbf{p}_{i-2}) \end{aligned}}$$

Cubic B-Splines

$$\mathbf{p}(u) = \mathbf{c}_0 + \mathbf{c}_1 u + \mathbf{c}_2 u^2 + \mathbf{c}_3 u^3$$



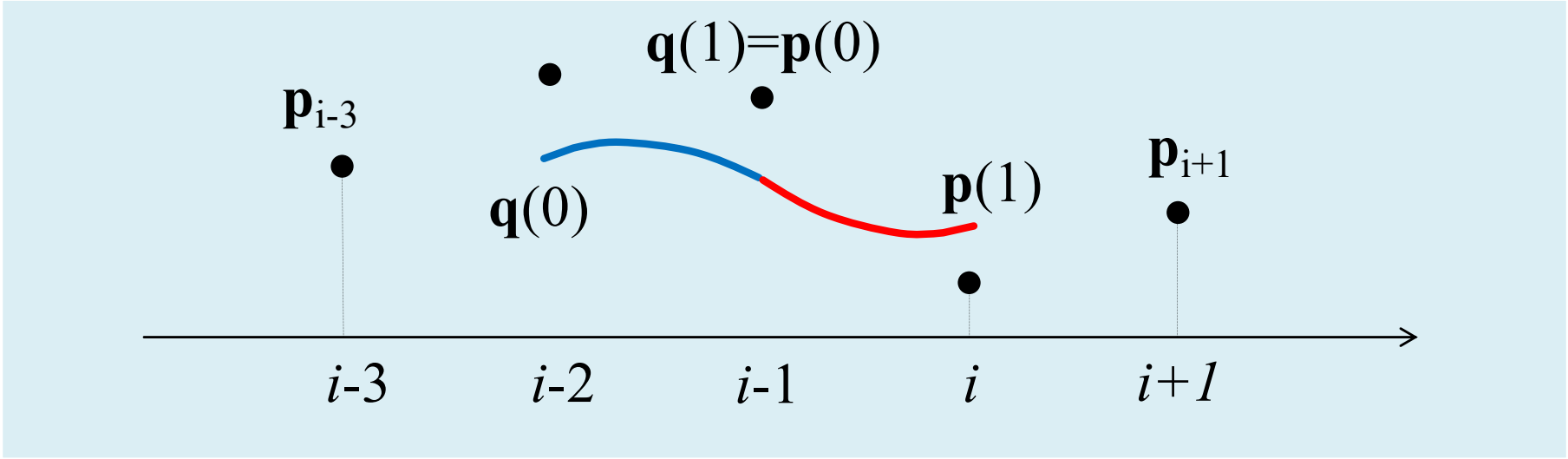
We can apply symmetric conditions at $\mathbf{p}(1)$:

$$\mathbf{p}(1) = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3 = \frac{1}{6}(\mathbf{p}_{i-1} + 4\mathbf{p}_i + \mathbf{p}_{i+1})$$

$$\mathbf{p}'(1) = \mathbf{c}_1 + 2\mathbf{c}_2 + 3\mathbf{c}_3 = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_{i-1})$$

Cubic B-Splines

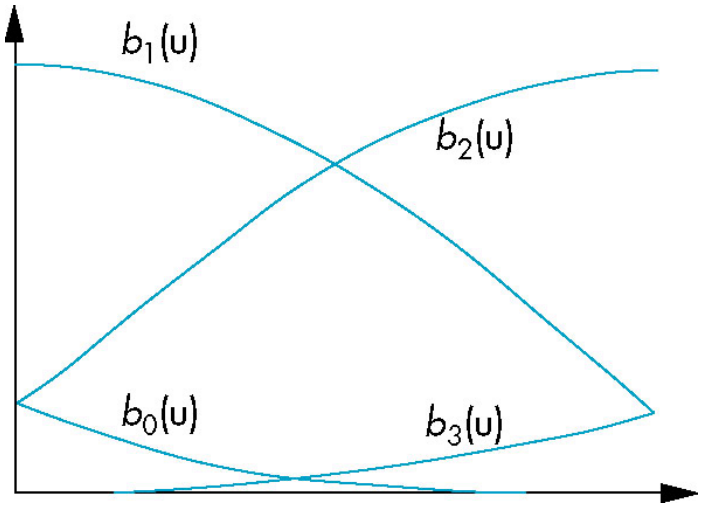
$$\mathbf{p}(u) = \mathbf{c}_0 + \mathbf{c}_1 u + \mathbf{c}_2 u^2 + \mathbf{c}_3 u^3$$



$$\left.\begin{aligned}
\mathbf{c}_0 &= \frac{1}{6}(\mathbf{p}_{i-2} + 4\mathbf{p}_{i-1} + \mathbf{p}_i) \\
\mathbf{c}_1 &= \frac{1}{2}(\mathbf{p}_i - \mathbf{p}_{i-2}) \\
\mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3 &= \frac{1}{6}(\mathbf{p}_{i-1} + 4\mathbf{p}_i + \mathbf{p}_{i+1}) \\
\mathbf{c}_1 + 2\mathbf{c}_2 + 3\mathbf{c}_3 &= \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_{i-1})
\end{aligned}\right\} \Rightarrow \mathbf{M}_S = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

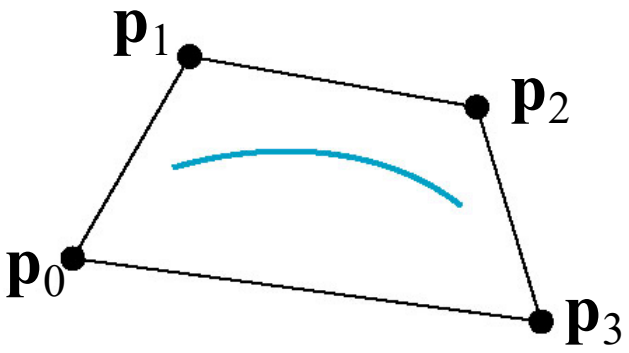
Cubic B-Splines: blending polynomials

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

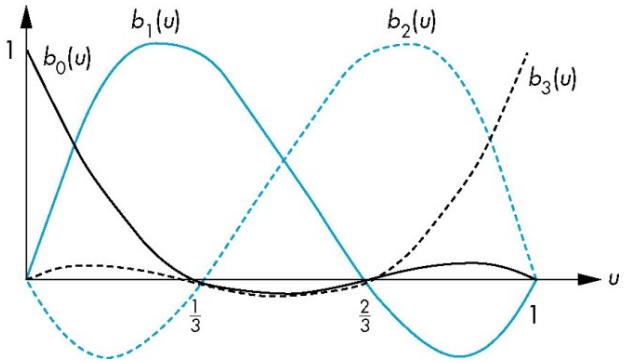


It can be shown that $\sum_{i=0}^{3} b_i(u) = 1$ and $0 < b_i(u) < 1$ for $0 < u < 1$.

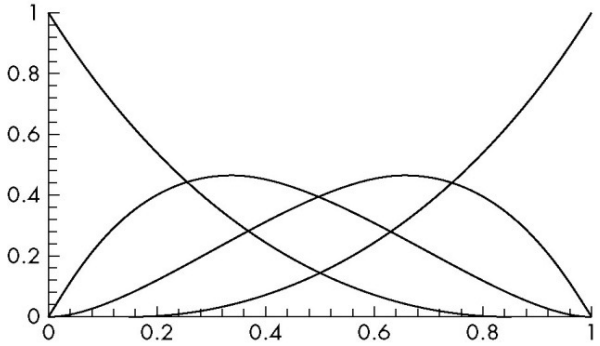
Thus, the curve must lie in the convex hull of the control points.



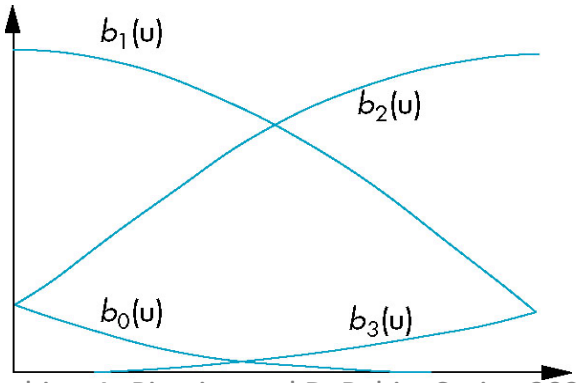
Comparison of blending polynomials



Cubic Interpolating polynomial



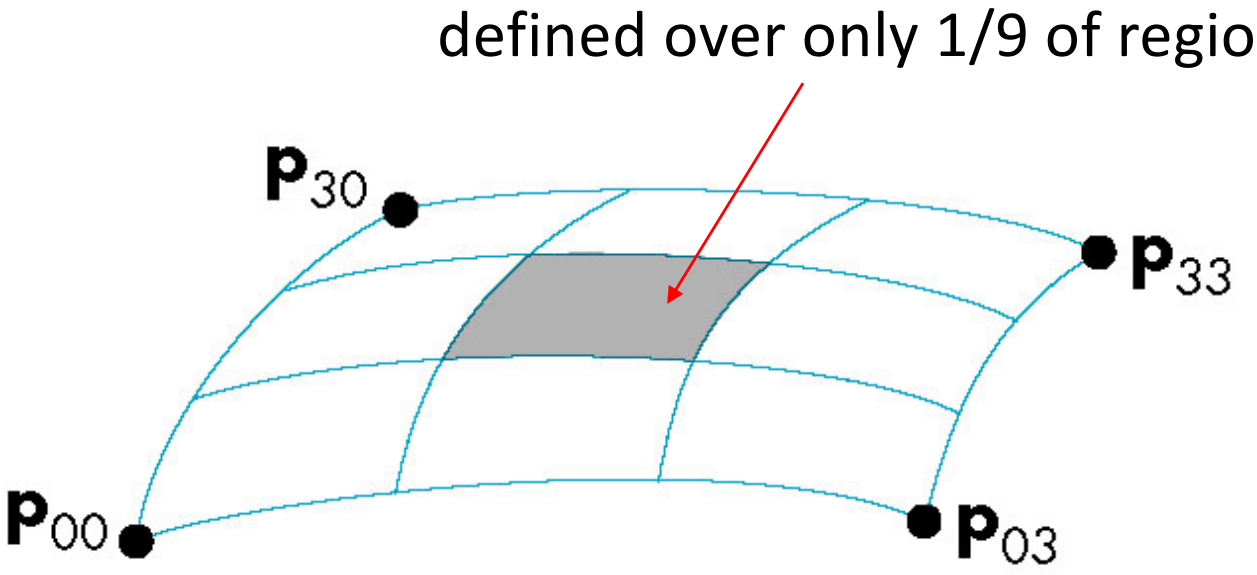
Bézier curves



Qubic B-splines

B-Spline Patches

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



Splines and Basis

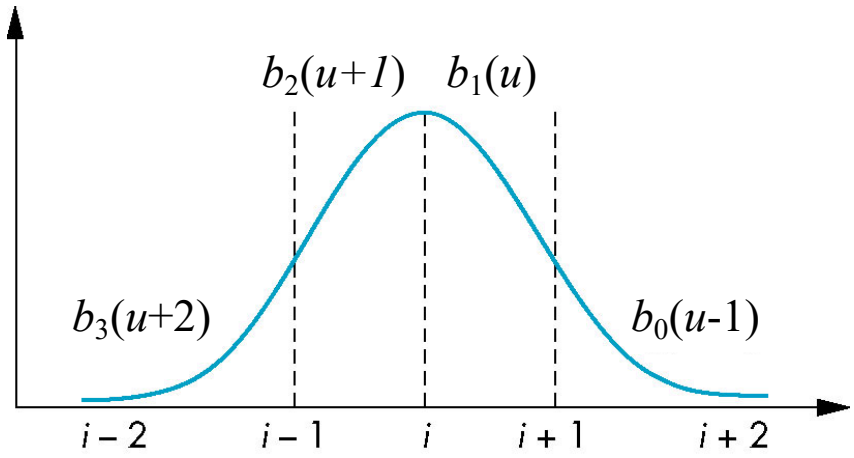
Note that each interior control point contributes (through the blending functions) to four segments.

We can rewrite $\mathbf{p}(u)$ in terms of the data points as

$$\mathbf{p}(u) = \sum_{i=1}^{m-1} B_i(u)\,\mathbf{p}_i$$

defining the basis functions $\{B_i(u)\}$. The name **B**-spline comes from "**basis** spline".

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



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*)

- Non-uniform and uniform splines
- Can have repeated knots
  - repeating knots can force the spline to interpolate the points

There are a number of ways to define basis splines – of particular importance is the set of splines defined by Cox-deBoor recursion.

Recursively defined splines

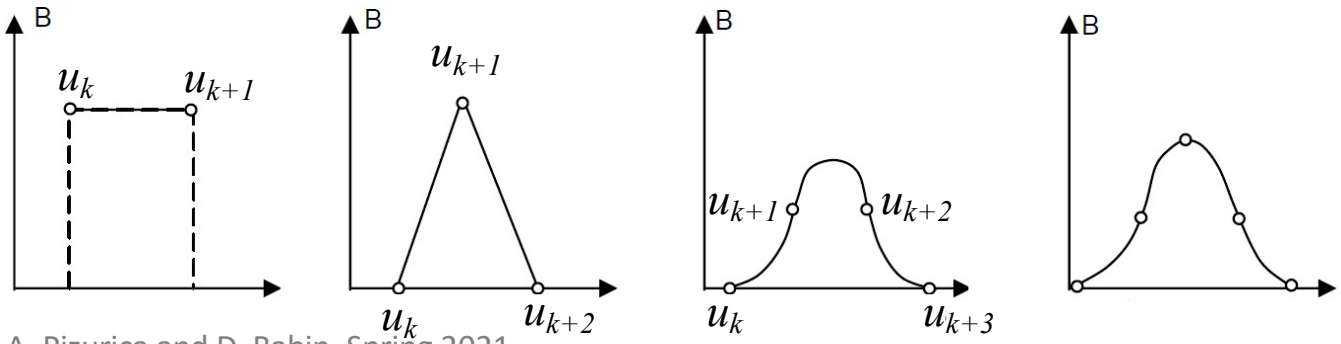
A B-spline is defined in terms of a set of basis (blending) functions each of which is nonzero over the region spanned by a few knots

$$\mathbf{p}(u) = \sum_{i=0}^{m} B_{id}(u)\mathbf{p}_i$$

$B_{id}(u)$ is a polynomial of degree d, except at the knots, and is zero outside the interval (u_{imin}, u_{imax}).

Cox-deBoor recursion

$$B_{k0} = \begin{cases} 1, & u_k \le u \le u_{k+1} \\ 0, & \text{otherwise} \end{cases}, \qquad B_{kd} = \frac{u - u_k}{u_{k+d} - u_k} B_{k,d-1}(u) + \frac{u_{k+d+1} - u}{u_{k+d+1} - u_{k+1}} B_{k+1,d-1}(u)$$



NURBS

NURBS

Nonuniform Rational B-Spline curves and surfaces add an extra variable w, which acts as a weight to change importance of some control points

A 3D control point $\mathbf{p}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$ $\longrightarrow$ $\mathbf{q}_i = w_i \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}$ weighted homogeneous-coordinate representation

The first three components are the B-spline representation of the weighted points

$$\mathbf{q}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \sum_{i=0}^{n} B_{i,d}(u) w_i \mathbf{p}_i$$

The w component is the scalar B-spline polynomial derived from the set of weights

$$w(u) = \sum_{i=0}^{n} B_{i,d}(u) w_i$$

NURBS, contd.

In homogeneous coordinates, this representation can have w component different from 1. Thus, a perspective division is needed to derive the 3D points:

$$\mathbf{p}(u) = \frac{1}{w(u)}\mathbf{q}(u) = \frac{\sum_{i=0}^{n} B_{i,d}(u) w_i \mathbf{p}_i}{\sum_{i=0}^{n} B_{i,d}(u) w_i}$$

Each component of $\mathbf{p}(u)$ is now a rational function of u.

NURBS retain all the properties of 3D B-splines, such as convex hull and continuity properties.

B-splines are invariant under affine transformation.

Perspective transformations are not affine. NURBS will be handled properly under perspective viewing as well, while most other splines not!

Rendering curves and surfaces

Polynomial evaluation methods

Suppose we have

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{c}_i u^i, \quad 0 \le u \le 1$$

The 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.

Rather than evaluating each term independently, we can group the terms using the Horner's method.

$$\mathbf{p}(u) = \mathbf{c}_0 + u(\mathbf{c}_1 + u(\mathbf{c}_2 + u(\ldots + \mathbf{c}_n u)))$$

For the cubic polynomial, we need 3 multiplications

$$\mathbf{p}(u) = \mathbf{c}_0 + u(\mathbf{c}_1 + u(\mathbf{c}_2 + u\mathbf{c}_3))$$

Polynomial evaluation methods, contd.

For equally spaced $\{u_k\}$ we define finite differences

$$\Delta^{(0)} p(u_k) = p(u_k)$$

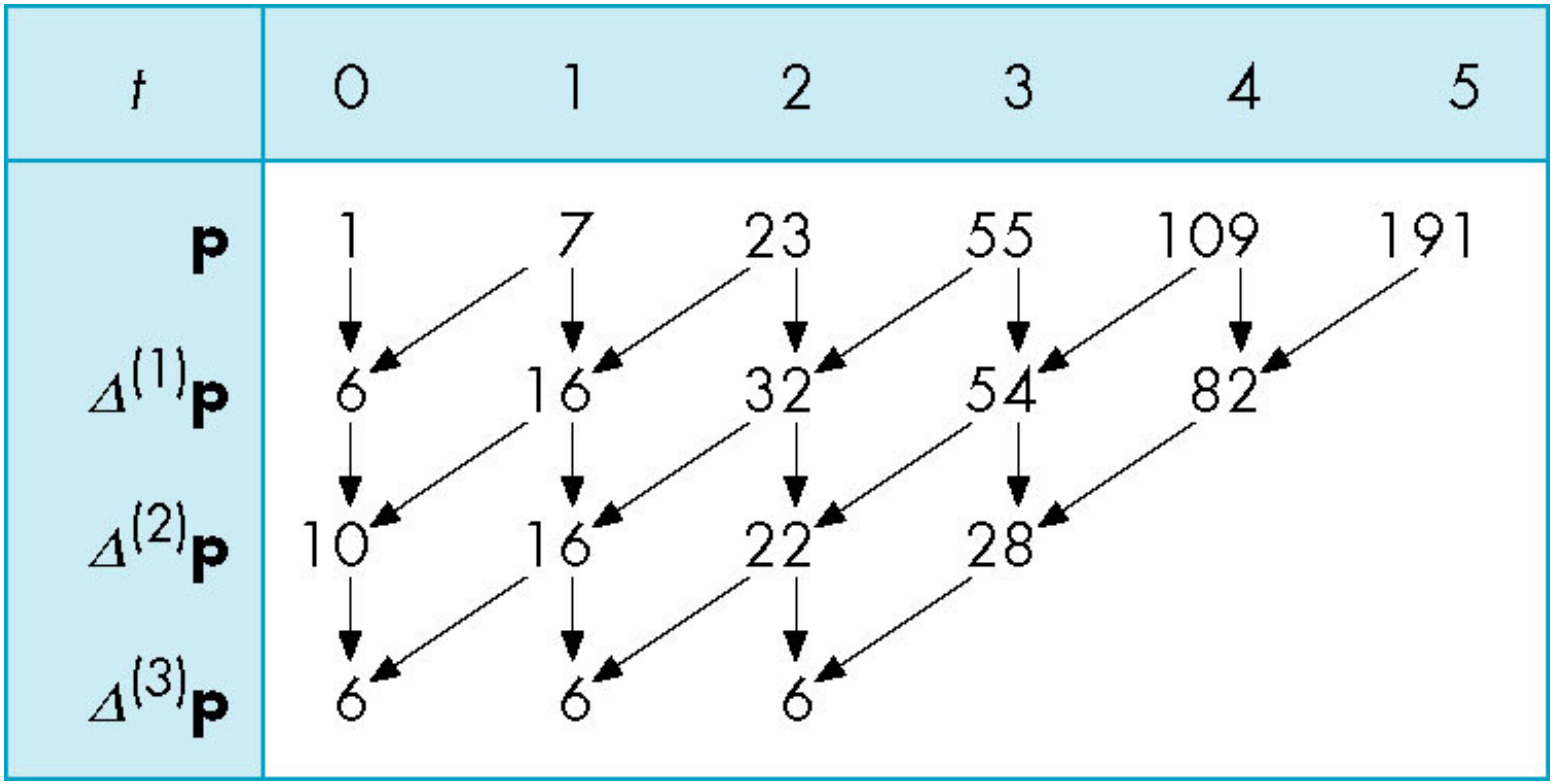
$$\Delta^{(1)} p(u_k) = p(u_{k+1}) - p(u_k)$$

$$\Delta^{(m+1)} p(u_k) = \Delta^{(m)} p(u_{k+1}) - \Delta^{(m)} p(u_k)$$

For a polynomial of degree n, the n^{th} finite difference is constant.

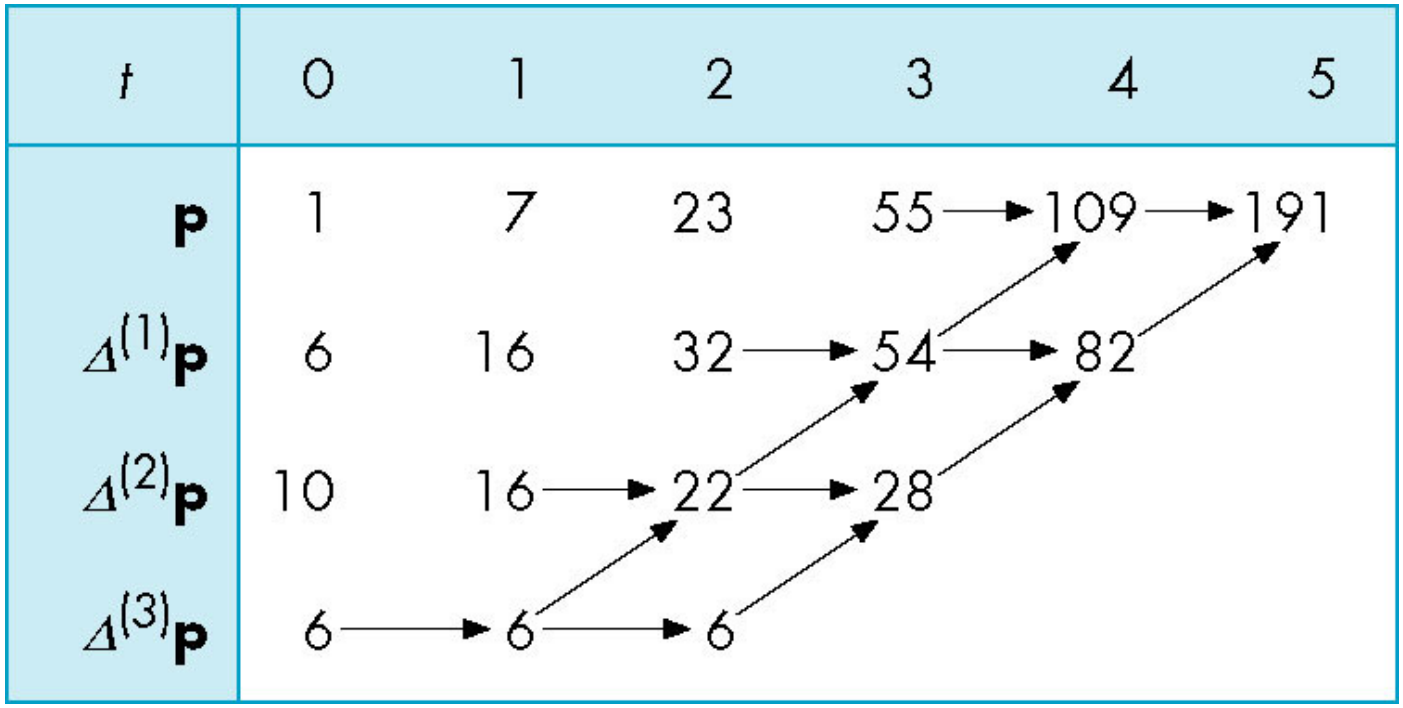
Building a Finite Difference Table

$p(u)=1+3u+2u^2+u^3$



Finding the Next Values

Starting at the bottom, we can work up generating new values for the polynomial



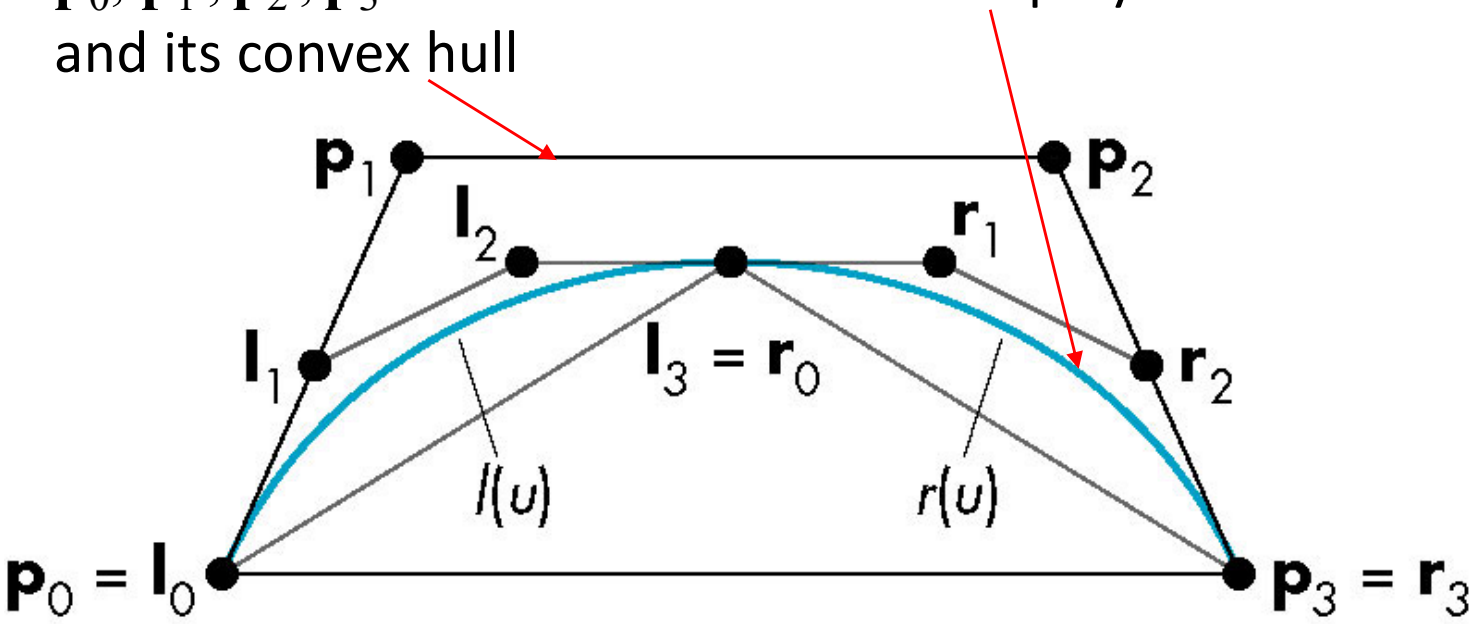
E. Angel and D.Shreiner: Interactive Computer Graphics 6E © Addison-Wesley 2012

Efficient method, but applies only to uniform grid and is prone to accumulation of numerical errors.

deCasteljau Recursion

We can use the convex hull property of Bézier curves to obtain an efficient recursive method that does not require any function evaluations, but uses only the values at the control points

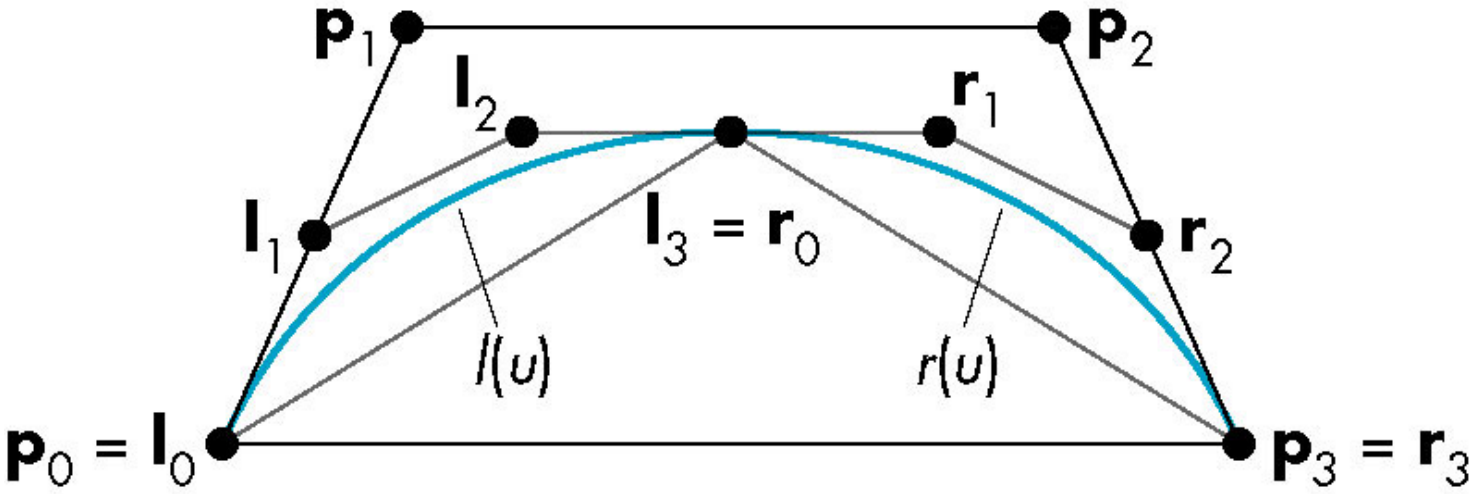
$\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ determine a cubic Bézier polynomial and its convex hull



Consider left half $\mathbf{l}(u)$ and right half $\mathbf{r}(u)$

deCasteljau Recursion, contd.

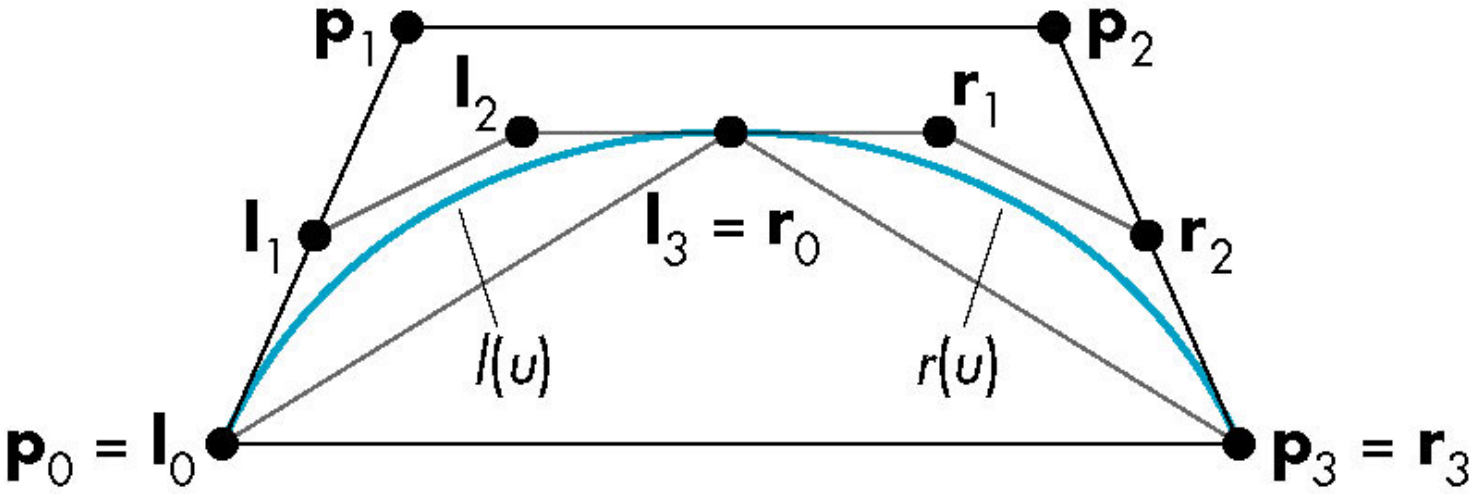
Since $\mathbf{l}(u)$ and $\mathbf{r}(u)$ are Bézier curves, we should be able to find two sets of control points $\{\mathbf{l}_0, \mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3\}$ and $\{\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3\}$ that determine them



deCasteljau Recursion, contd.

$\{\mathbf{l}_0, \mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3\}$ and $\{\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3\}$ each have a convex hull that is closer to $\mathbf{p}(u)$ than the convex hull of $\{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$. This is known as the variation diminishing property.

The polyline from $\mathbf{l}_0$ to $\mathbf{l}_3$ $(=\mathbf{r}_0)$ to $\mathbf{r}_3$ is an approximation to $\mathbf{p}(u)$. Repeating recursively we get better approximations.



deCasteljau Recursion, contd.

Bézier polynomial: $\mathbf{p}(u)=\mathbf{b}(u)^T\mathbf{p}$, $\mathbf{b}(u)=[(1-u)^3 \;\; 3u(1-u)^2 \;\; 3u^2(1-u) \;\; u^3]^T$

$$\mathbf{p}(u)=(1-u)^3\mathbf{p}_0+3u(1-u)^2\mathbf{p}_1+3u^2(1-u)\mathbf{p}_2+u^3\mathbf{p}_3$$

$\mathbf{l}(u)$ must interpolate $\mathbf{p}(0)$ and $\mathbf{p}(1/2)$

$$\mathbf{l}(0)=\mathbf{l}_0=\mathbf{p}_0$$

$$\mathbf{l}(1)=\mathbf{l}_3=\mathbf{p}(1/2)=1/8(\mathbf{p}_0+3\mathbf{p}_1+3\mathbf{p}_2+\mathbf{p}_3)$$

Matching slopes, taking into account that $\mathbf{l}(u)$ and $\mathbf{r}(u)$ only go over half the distance as $\mathbf{p}(u)$

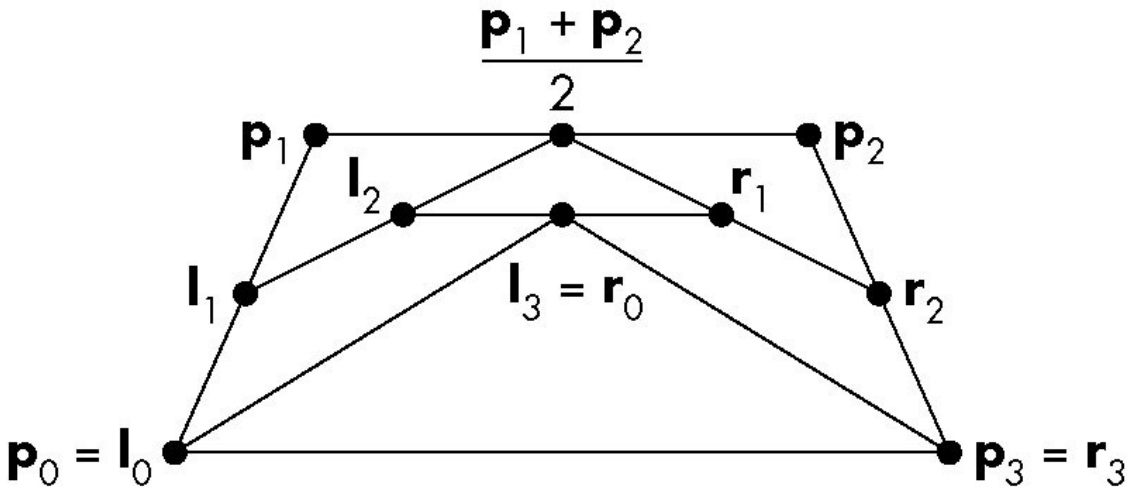
$$\mathbf{l}'(0)=3(\mathbf{l}_1-\mathbf{l}_0)=\mathbf{p}'(0)=3/2(\mathbf{p}_1-\mathbf{p}_0)$$

$$\mathbf{l}'(1)=3(\mathbf{l}_3-\mathbf{l}_2)=\mathbf{p}'(1/2)=3/8(-\mathbf{p}_0-\mathbf{p}_1+\mathbf{p}_2+\mathbf{p}_3)$$

Symmetric equations hold for $\mathbf{r}(u)$.

deCasteljau Recursion, contd.

$$\mathbf{l}_0 = \mathbf{p}_0$$
$$\mathbf{r}_3 = \mathbf{p}_3$$
$$\mathbf{l}_1 = \tfrac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1)$$
$$\mathbf{r}_2 = \tfrac{1}{2}(\mathbf{p}_2 + \mathbf{p}_3)$$
$$\mathbf{l}_2 = \tfrac{1}{2}(\mathbf{l}_1 + \tfrac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2))$$
$$\mathbf{r}_1 = \tfrac{1}{2}(\mathbf{r}_2 + \tfrac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2))$$
$$\mathbf{l}_3 = \mathbf{r}_0 = \tfrac{1}{2}(\mathbf{l}_2 + \mathbf{r}_1)$$



Requires only shifts and additions!

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 Bézier curve

Suppose that $\mathbf{p}(u)$ is given as an interpolating curve with control points $\mathbf{q}$

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

There exist Bézier control points $\mathbf{p}$ such that

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

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

Transforming to Bézier form

Interpolating to Bézier

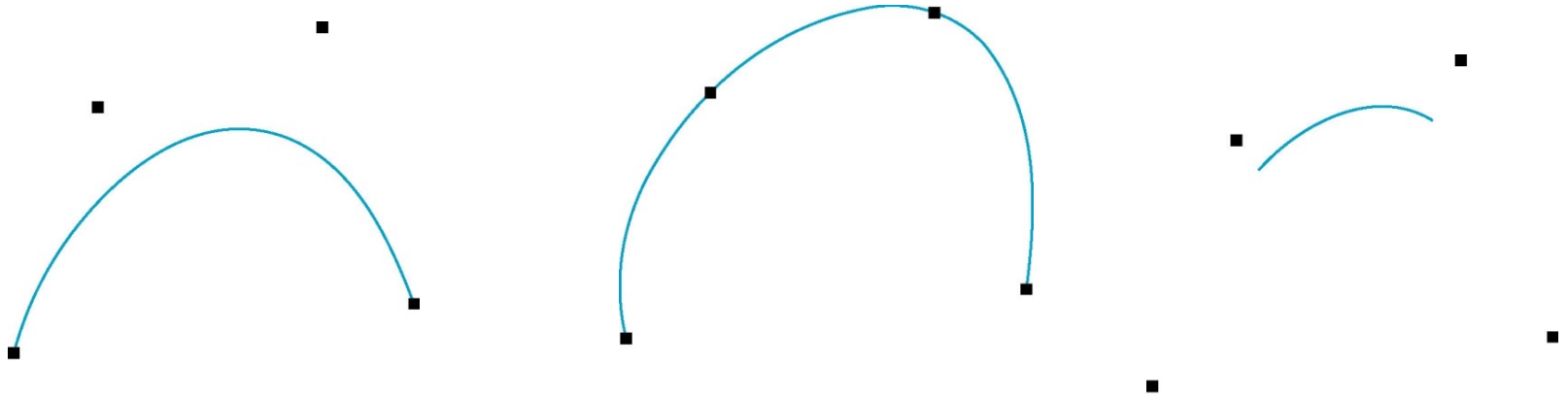
$$\mathbf{M}_B^{-1}\mathbf{M}_I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{5}{6} & 3 & -\frac{3}{2} & \frac{1}{3} \\ \frac{1}{3} & -\frac{3}{2} & 3 & -\frac{5}{6} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

B-Spline to Bézier

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

Example

These three curves were all generated from the same original data using Bezier recursion by converting all control point data to Bezier control points.



Bézier

Interpolating

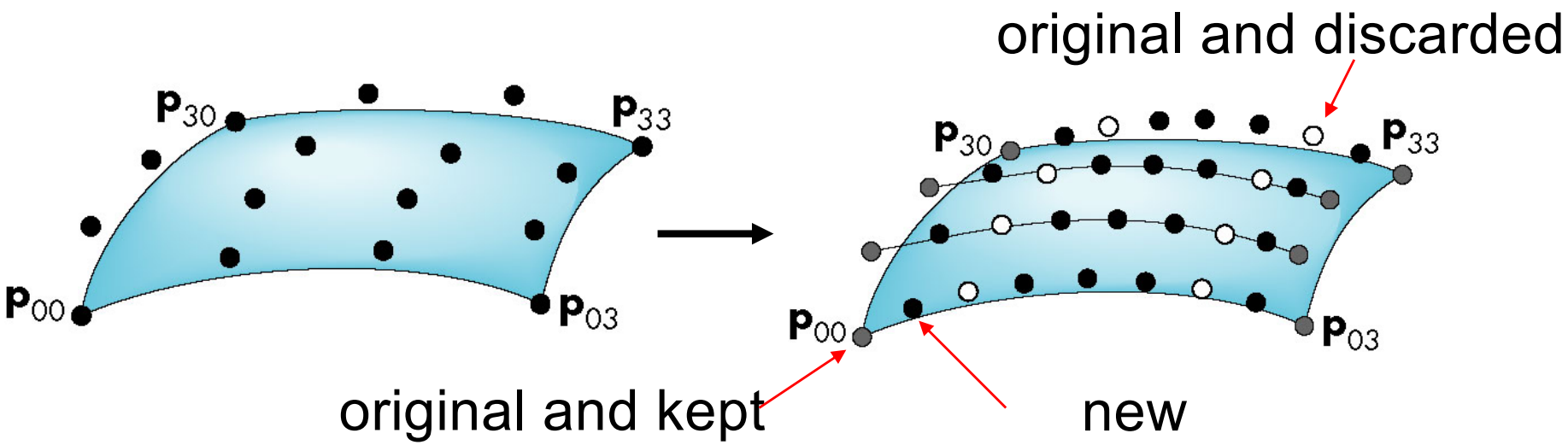
B Spline

Application to surfaces

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)

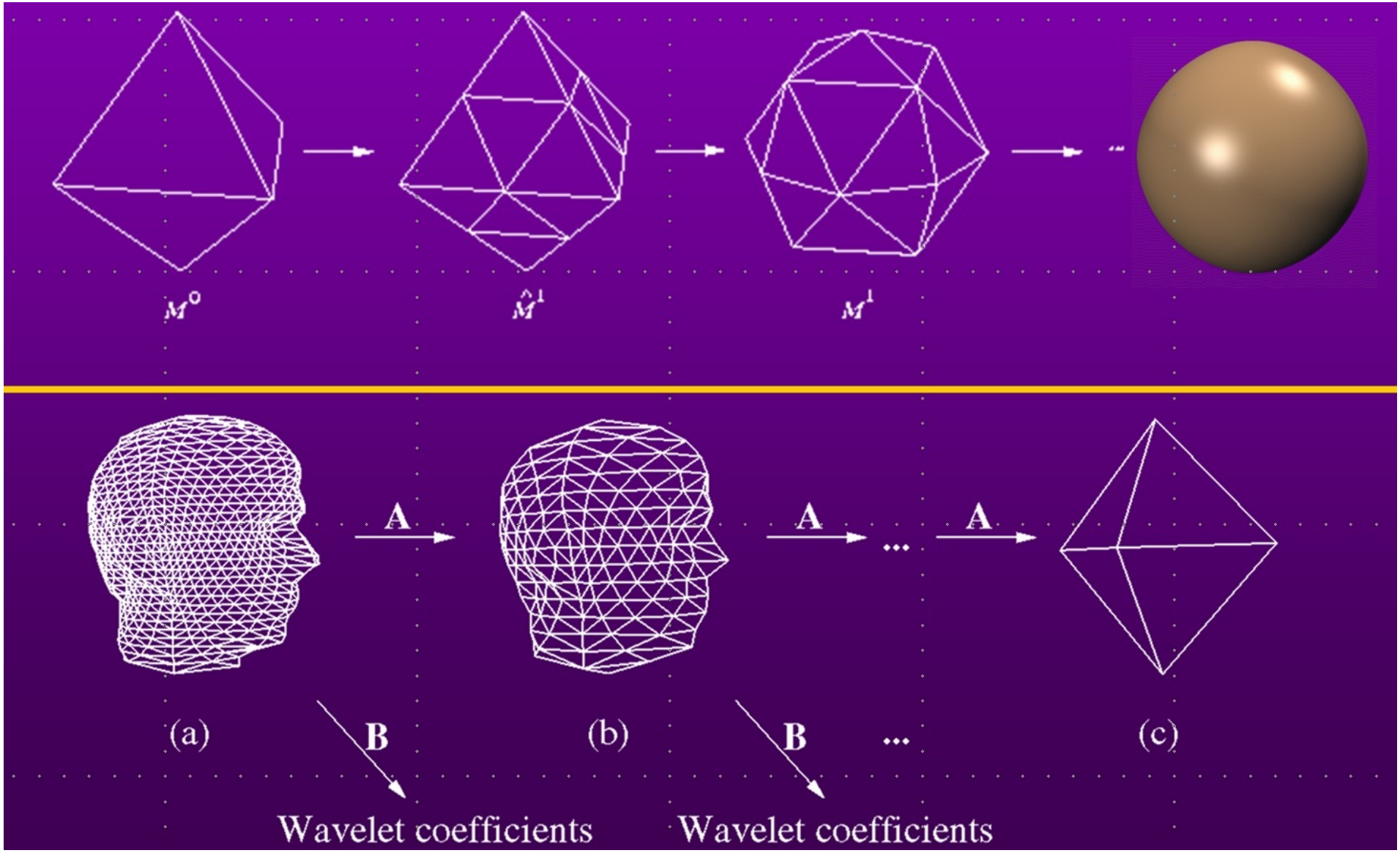
First subdivide in u

- Process creates new points
- Some of the original points are discarded



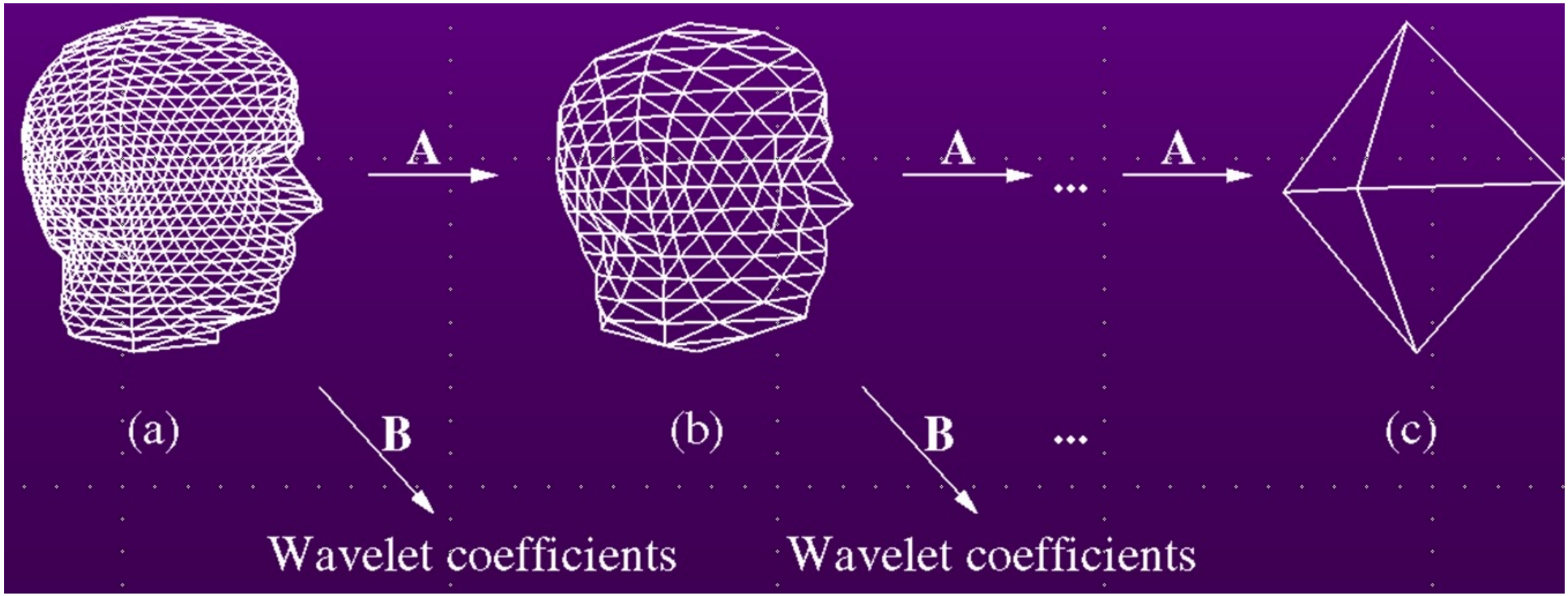
Wavelet Subdivision Surfaces

Wavelets: subdivision runs backwards



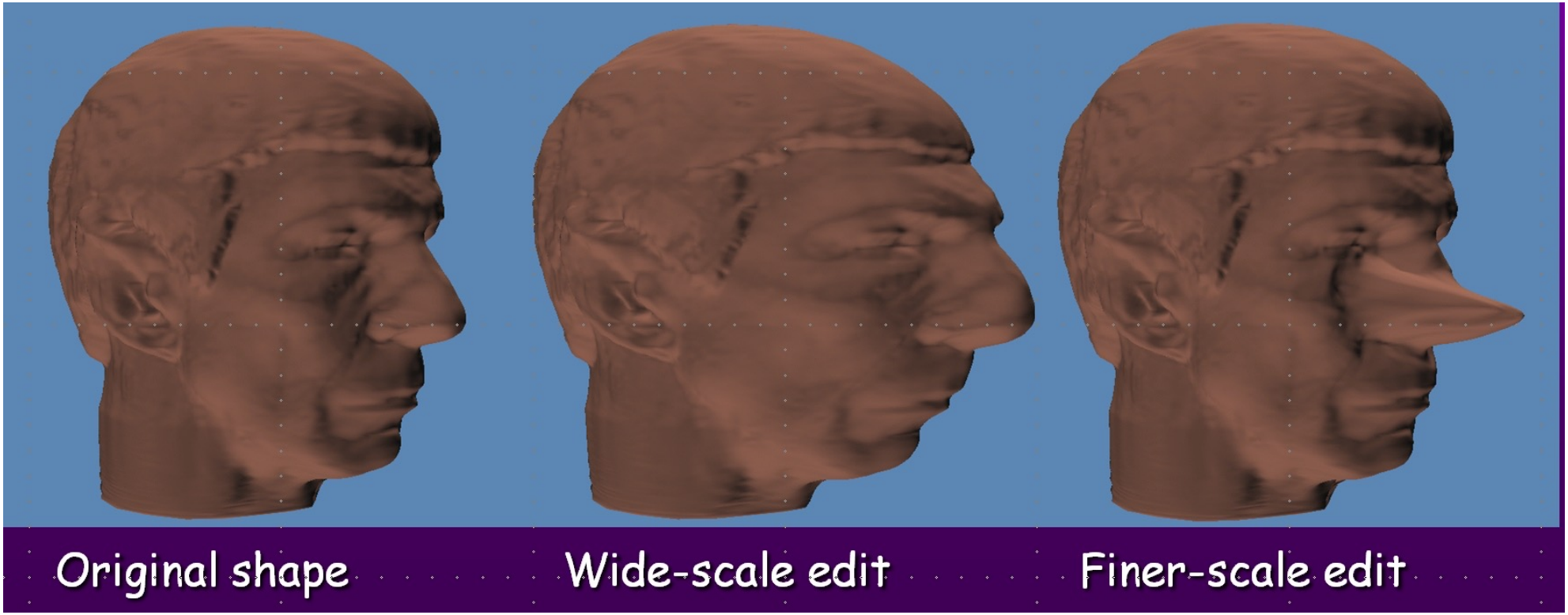
Lounsbery, DeRosse and Warren: Multiresolution Analysis for Surfaces of Arbitrary Type

Wavelet subdivision surfaces



Lounsbery, DeRosse and Warren: Multiresolution Analysis for Surfaces of Arbitrary Type

Editing surface



Lounsbery, DeRosse and Warren: Multiresolution Analysis for Surfaces of Arbitrary Type