Curved surfaces

CS 465 Lecture 16

From curves to surfaces

- So far have discussed spline curves in 2D
  - it turns out that this already provides of the mathematical machinery for several ways of building curved surfaces
- Building surfaces from 2D curves
  - extrusions and surfaces of revolution
- Building surfaces from 2D and 3D curves
  - generalized swept surfaces
- Building surfaces from spline patches
  - generalizing spline curves to spline patches
- Also to think about: generating triangles

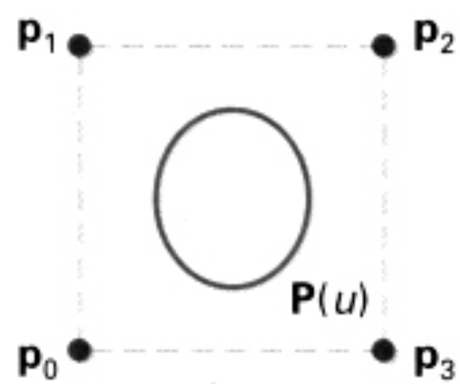
Extrusions

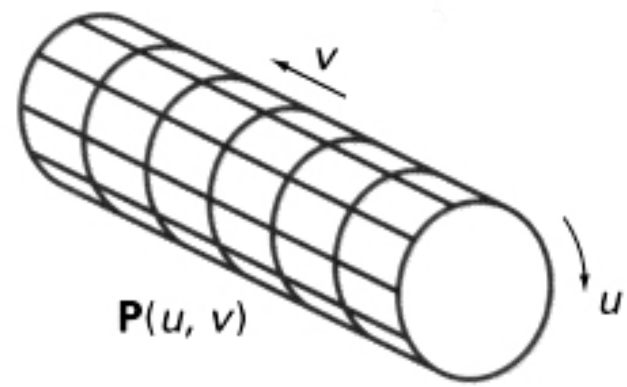
- Given a spline curve $C \in \mathbb{R}^2$, define $S \in \mathbb{R}^3$ by

$$S = C \times [a, b]$$

- This produces a "tube" with the given cross section
  - Circle: cylinder; "L": shelf bracket; "I": I beam
- It is parameterized by the spline *t* and the interval [*a, b*]

$$s(t, s) = [c_x(t), c_y(t), s]^T$$



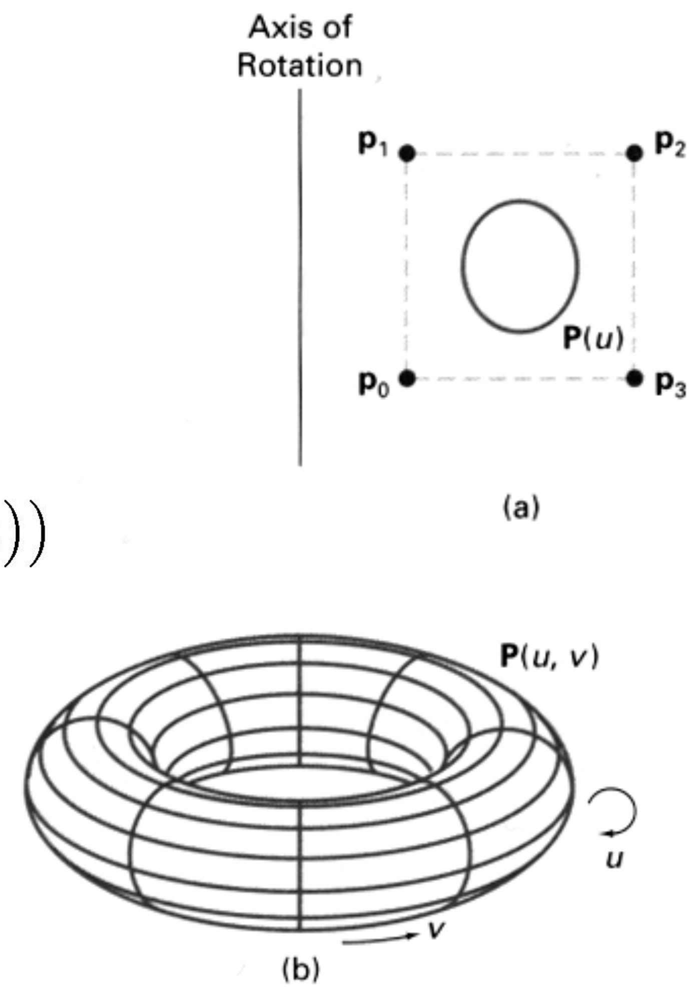


Surfaces of revolution

- Take a 2D curve and spin it around an axis
- Given curve **c**(*t*) in the plane, the surface is defined easily in cylindrical coordinates:

$$\mathbf{s}(t, s) = (r, \phi, z) = (c_x(t), s, c_y(t))$$

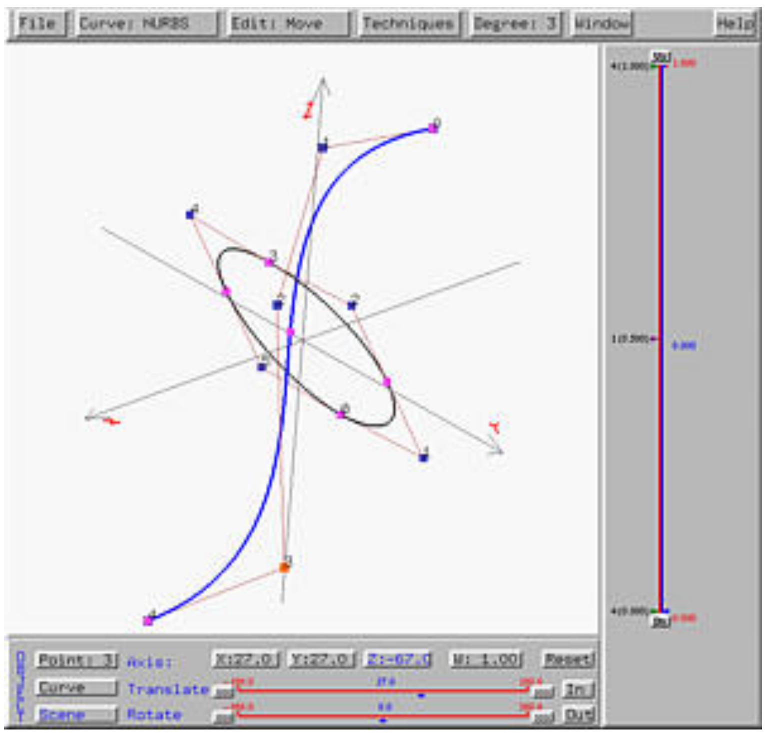
  – the torus is an example in which the curve **c** is a circle

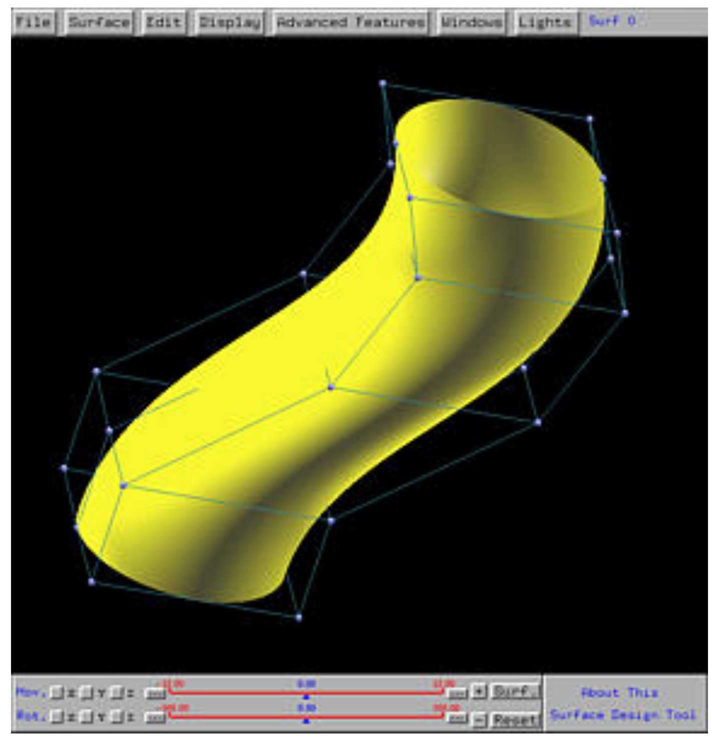


[Hearn & Baker]

Swept surfaces

- Surface defined by a *cross section* moving along a *spine*
- Simple version: a single 3D curve for spine and a single 2D curve for the cross section





[Ching-Kuang Shene]

More general surfaces

- Extrusions are fine but lack control
- In sketching, it's common to draw several cross sections rather than just one
  - the understanding is that there is a smooth surface that interpolates the cross sections
- So suppose we have several cross sections given as splines: how to define parametric surface?
  - know t cross section at, say, $s = 0, 1, 2$
  - define intermediate sections by interpolating control points
  - use more splines to interpolate smoothly!

Splines within splines

- For every *s* the *t* cross-section will be defined by a spline—say it's a cubic (4-point) Bézier segment

$$\mathbf{c}(t) = \sum_{i=1}^{4} \mathbf{p}_i b_i(t)$$

(a single curve)

$$\mathbf{s}(s, t) = \sum_{i=1}^{4} \mathbf{p}_i(s) b_i(t)$$

(a surface defined by making the curve a function of s)

- now suppose we choose the same type of spline to represent the functions $\mathbf{p}_i(s)$

Splines within splines

- Using a spline to define the control points of a spline

$$\mathbf{s}(s,t) = \sum_{i=1}^{4} \mathbf{p}_i(s) b_i(t)$$ (from prev. slide)

$$\mathbf{p}_i(s) = \sum_{j=1}^{4} \mathbf{p}_{ij} b_j(s)$$ (each $\mathbf{p}_i$ is a spline in *s*)

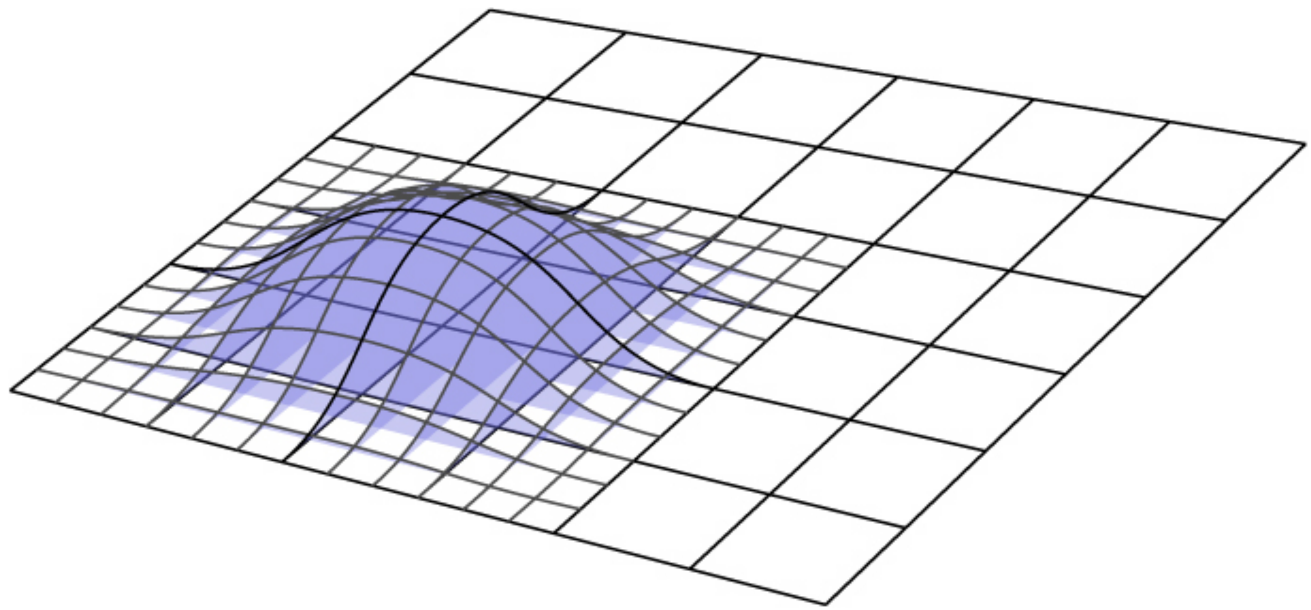
$$\mathbf{s}(s,t) = \sum_{i,j=1}^{4} \mathbf{p}_{ij} b_i(t) b_j(s)$$ (substitute)

  – note that you can't tell which spline is on the outside any more: *s* and *t* are not different

From curves to surface patches

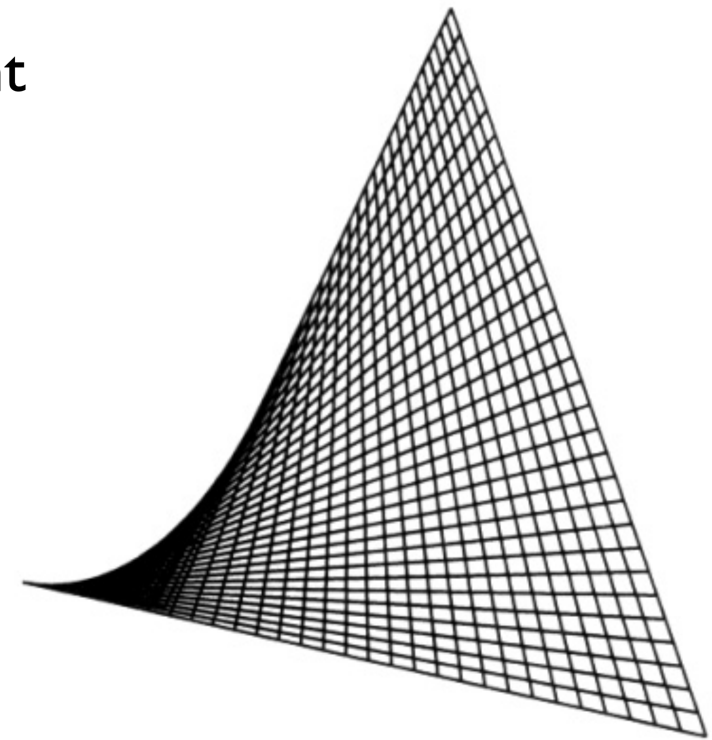
- Curve was sum of weighted 1D basis functions
- Surface is sum of weighted 2D basis functions
  - construct them as separable products of 1D fns.
  - choice of different splines
    - spline type
    - order
    - closed/open (B-spline)

Separable product construction



Bilinear patch

- Simplest case: 4 points, cross product of two linear segments
  - basis function is a 3D tent



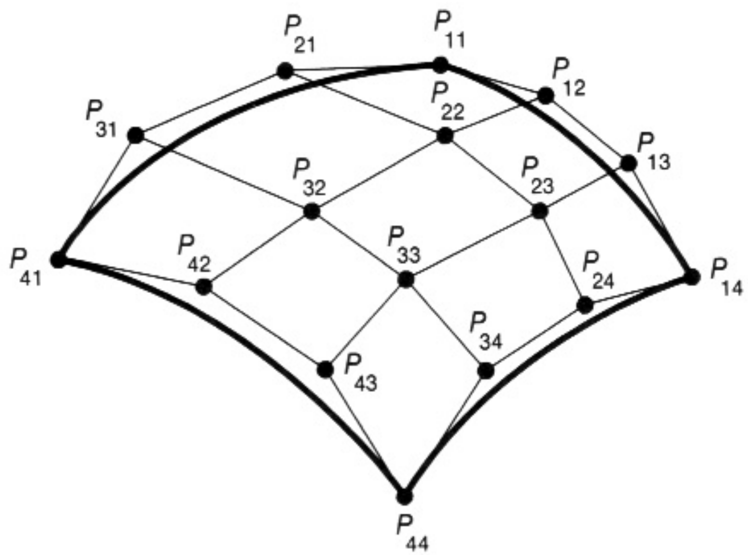
(c)

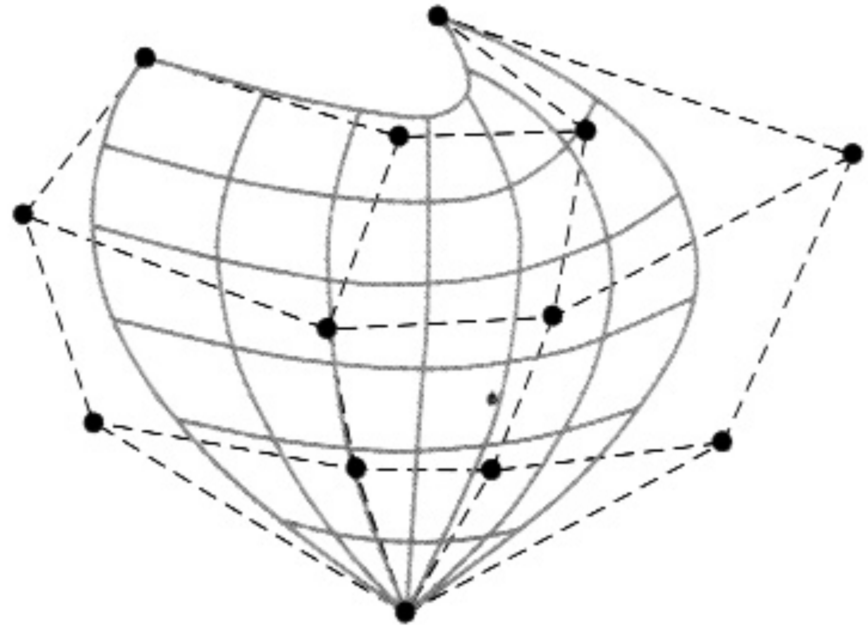
[Rogers]

Bicubic Bézier patch

- Cross product of two cubic Bézier segments

[Foley et al.]





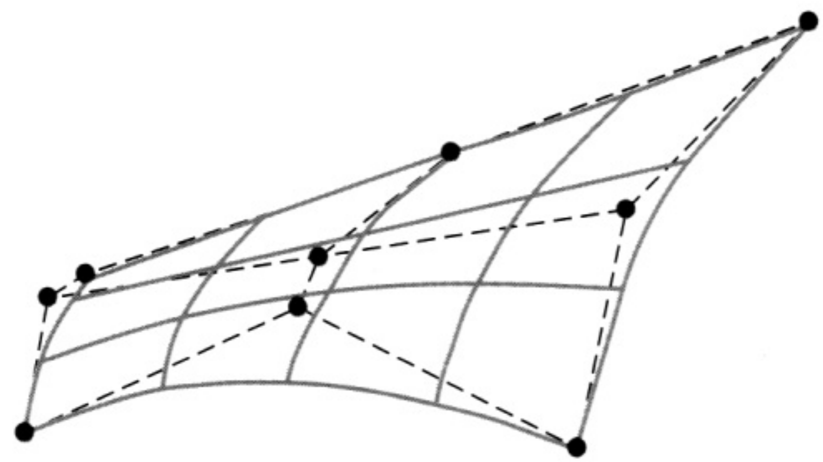
(b)

[Hearn & Baker]

  - properties that carry over
    - interpolation at corners, edges
    - tangency at corners, edges
    - convex hull

Biquadratic Bézier patch

- Cross product of quadratic Bézier curves

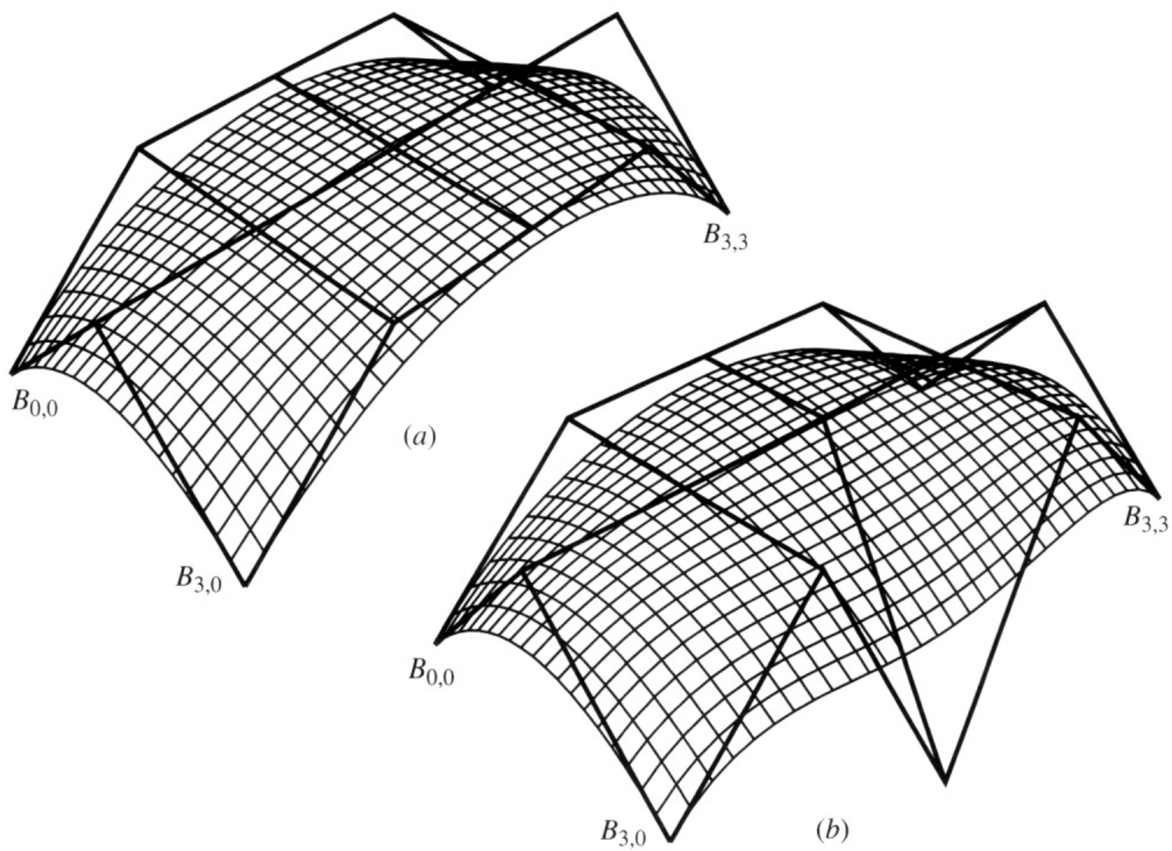


(a)

[Hearn & Baker]

3x5 Bézier patch

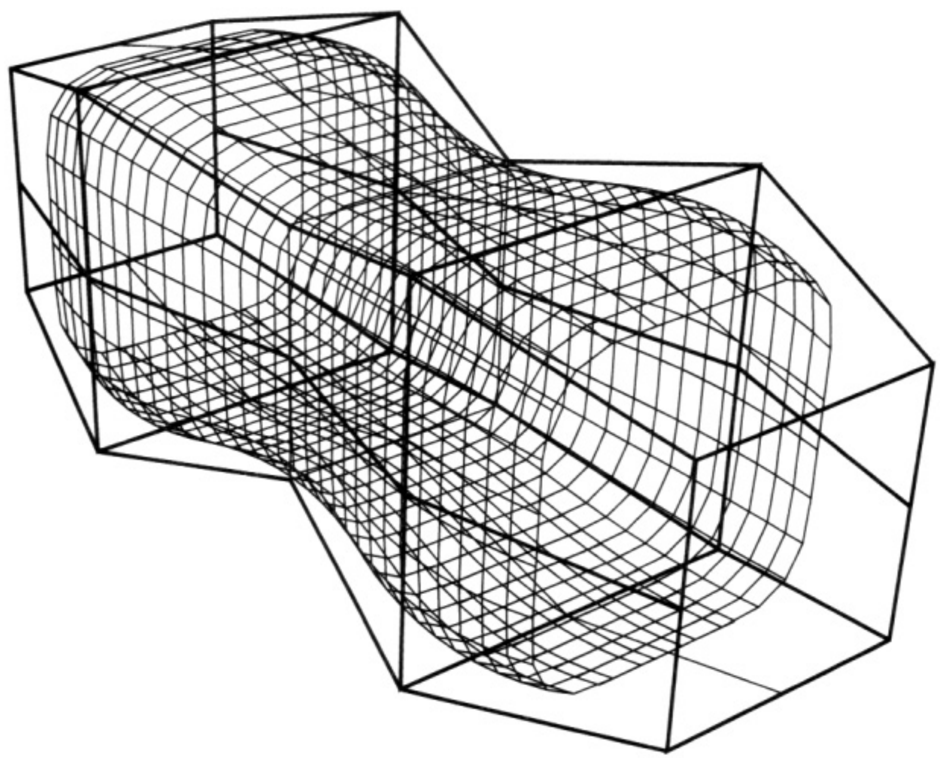
- Cross product of quadratic and quartic Béziers



[Rogers]

Cylindrical B-spline surfaces

- Cross product of closed and open cubic B-splines

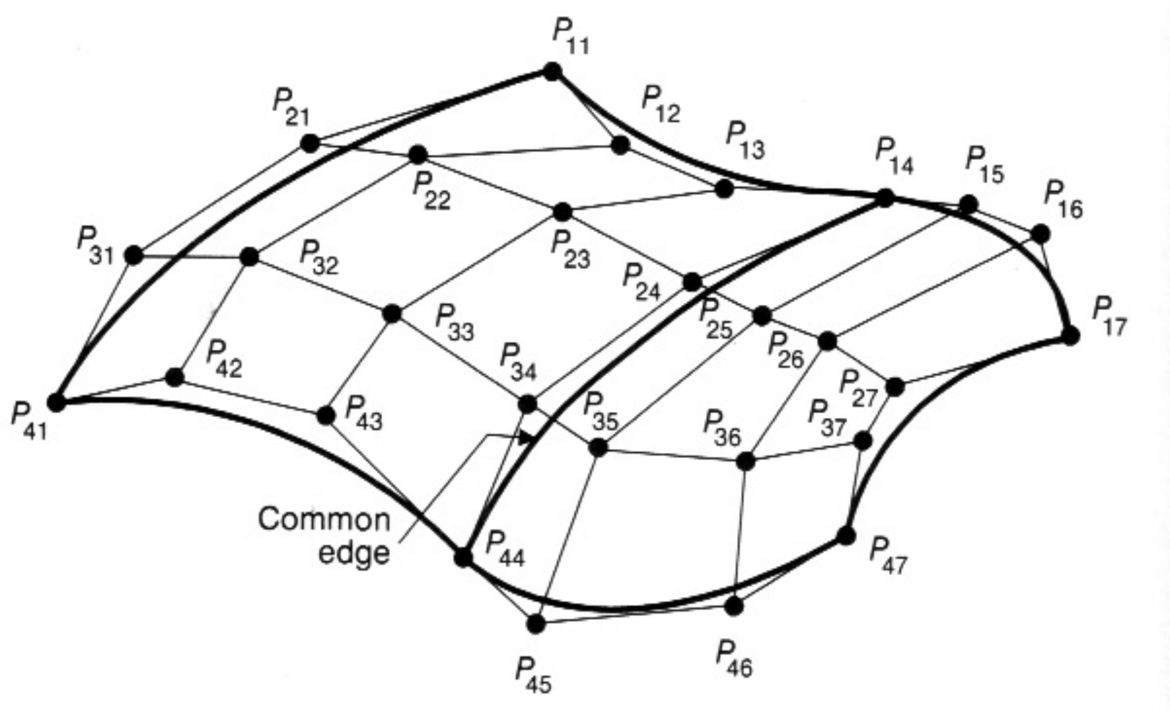


(b)

[Rogers]

Joining spline patches

- Example: Bézier patches
  - conditions for C1 are similar to curve
  - conditions for G1 allow any coplanar structure…



[Foley et al.]

Joining multiple patches

- Joining a 4-way corner is fairly simple
  - constraints from multiple joins are compatible
  - this makes patches tileable into large sheets
- Joining at irregular corners is quite messy
  - constraints become contradictory
    - e.g. at 3-way corner
  - this is problematic for building anything with topology different from plane, cylinder, or torus
    - it is possible, though—just messy

Approximating spline surfaces

- Similarly to curves, approximate with simple primitives
  - in surface case, triangles or quads
  - quads widely used because they fit in parameter space
    - generally eventually rendered as pairs of triangles
- adaptive subdivision
  - basic approach: recursively test flatness
    - if the patch is not flat enough, subdivide into four using curve subdivision twice, and recursively process each piece
  - as with curves, convex hull property is useful for termination testing (and is inherited from the curves)

Approximating spline surfaces

- With adaptive subdivision, must take care with cracks
  - (at the boundaries between degrees of subdivision)