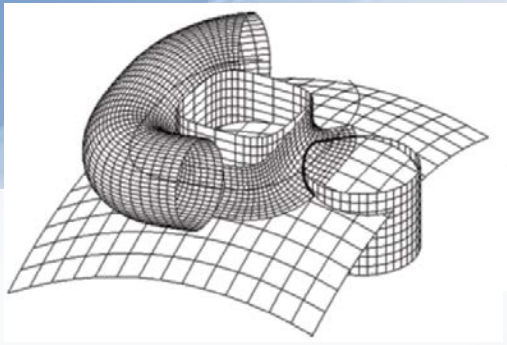
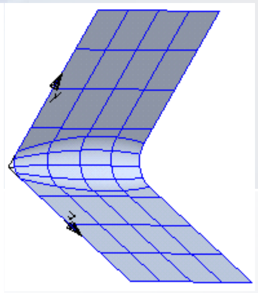
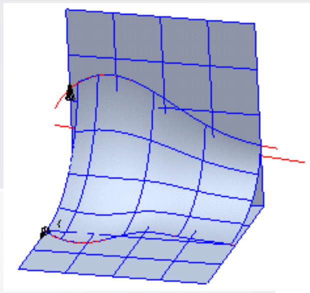
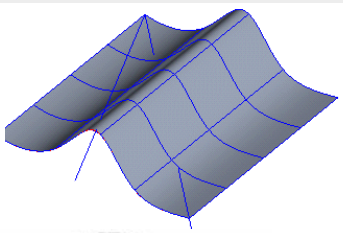
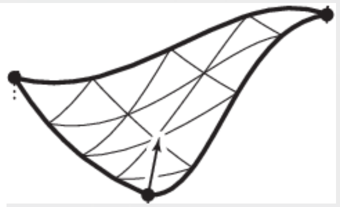
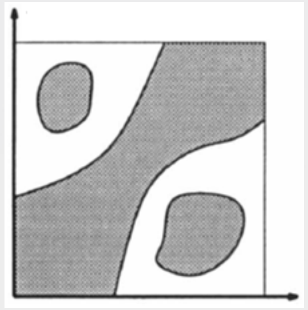
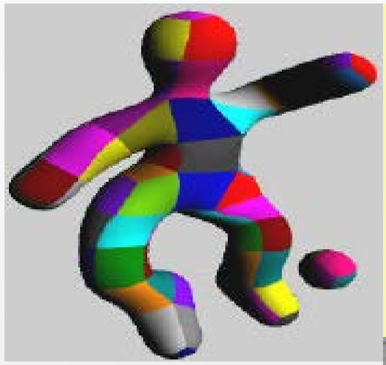
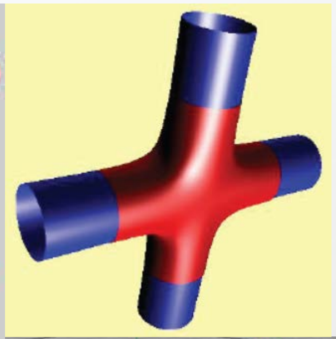
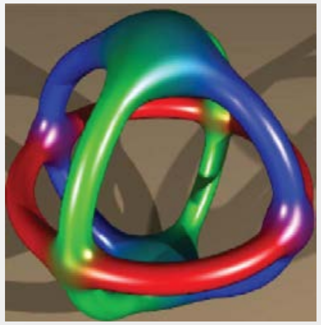
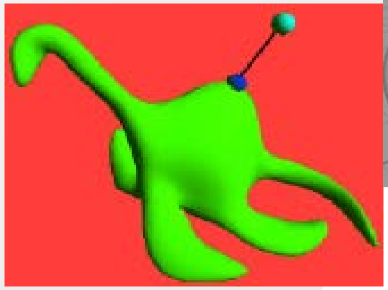
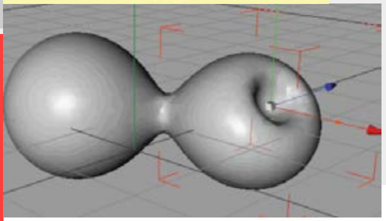
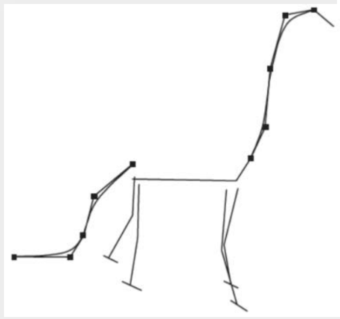
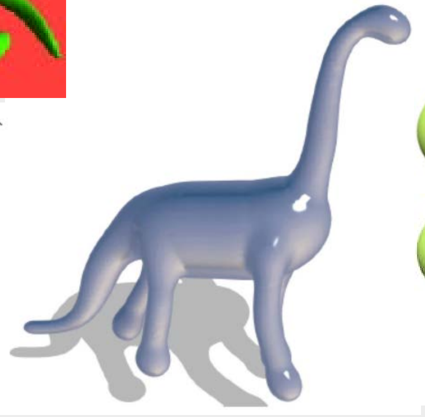
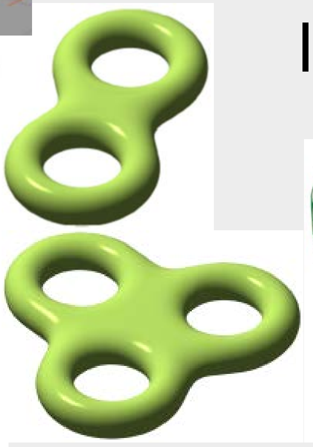
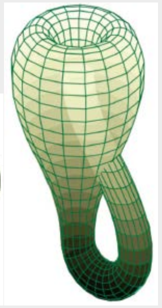
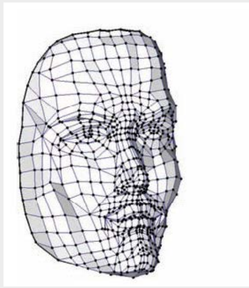
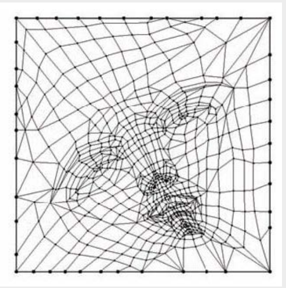






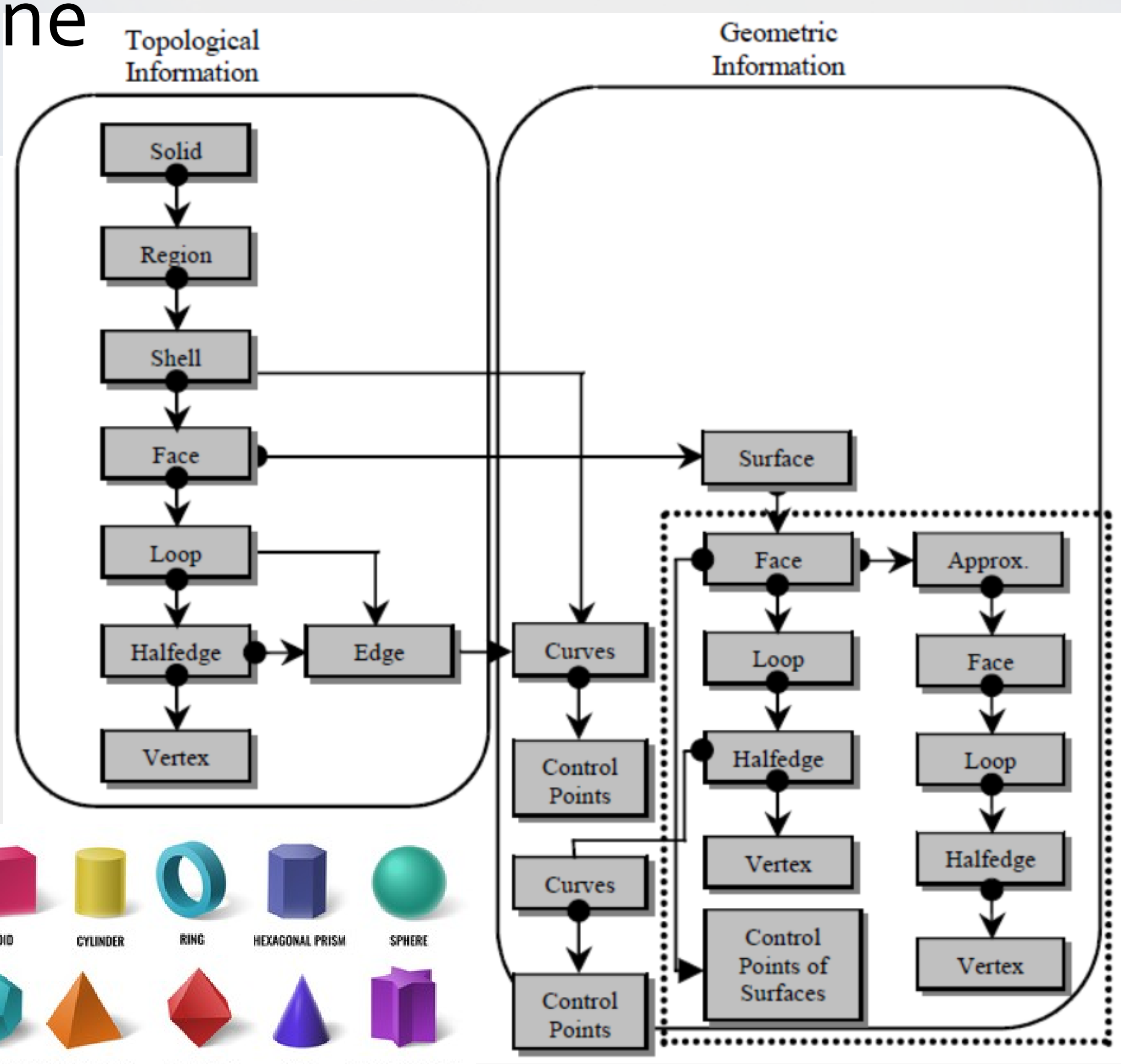













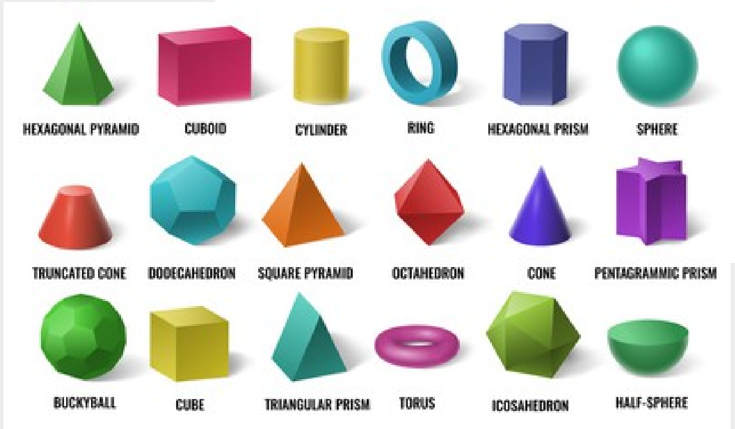
Advanced CAD Surfaces

Prof. PhD. Hikmet Kocabas
Istanbul Technical Univ.

Lectures, Outline of the course

1 Adv. CAD Techs
2 Geom. Modeling
3 Transformations
4 Curves
5 Splines, NURBS
6 Surfaces
7 Solid Modeling
8 API prog.



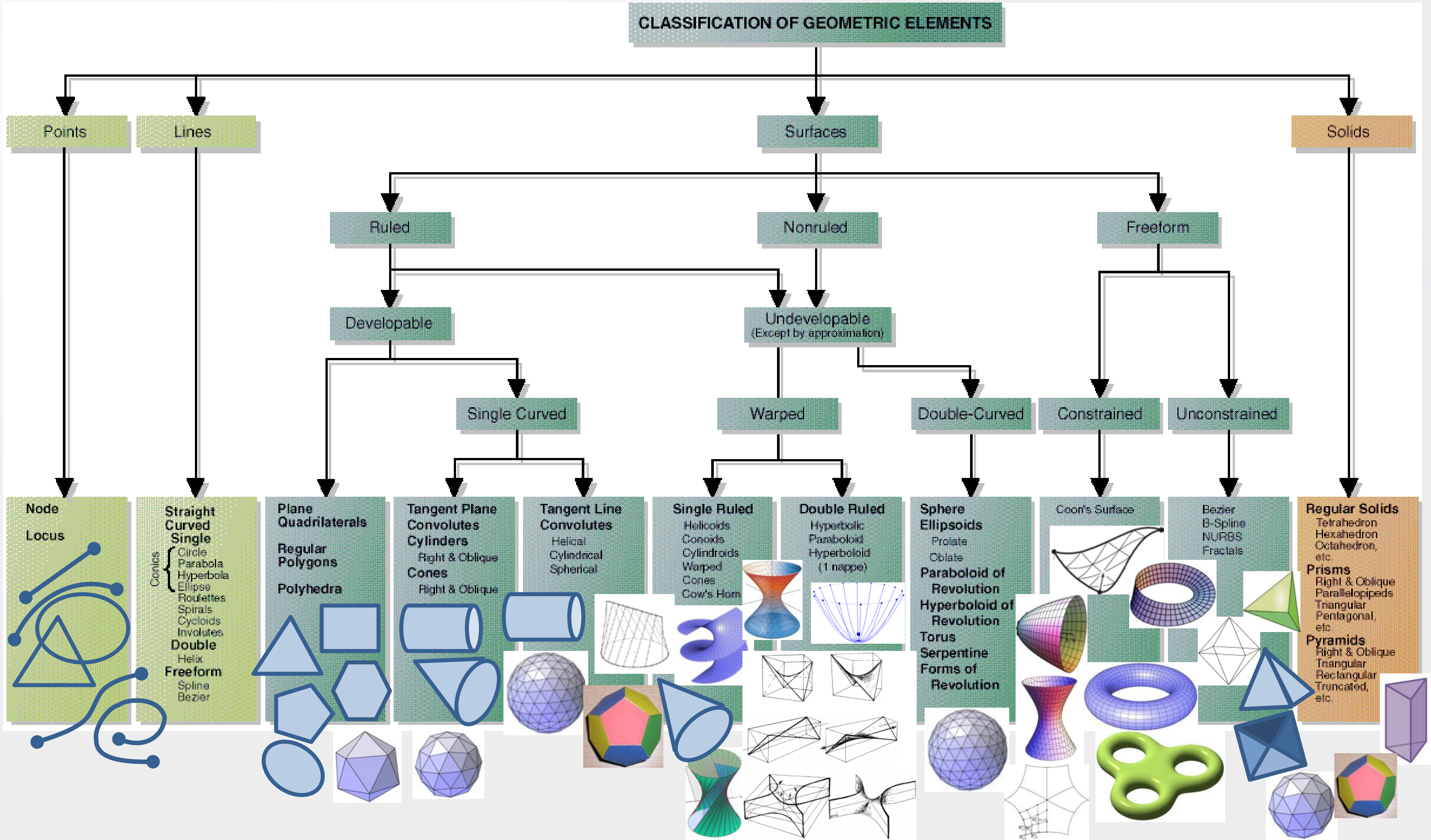


Intro. to Computer Graphic Systems

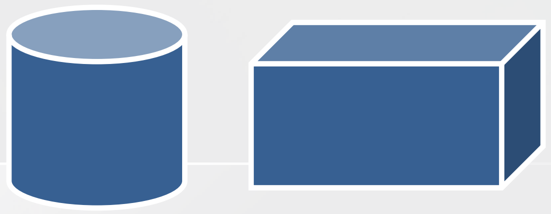
Textbooks:

- Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, Fourth Edition, Gerald Farin, September 1996
- Computer Aided Engineering Design, Anupam Saxena, Birendra Sahay, Springer, 2005
- CAD/CAM Theory and Practice , Ibrahim Zeid, McGraw Hill , 1991, Mastering CAD/CAM, ed. 2004
- The NURBS Book, Les Piegl, Springer-Verlag, 1997
- 3D CAD Principles and Applications, H Toriya, Springer-Verlag, 1991

geometric elements



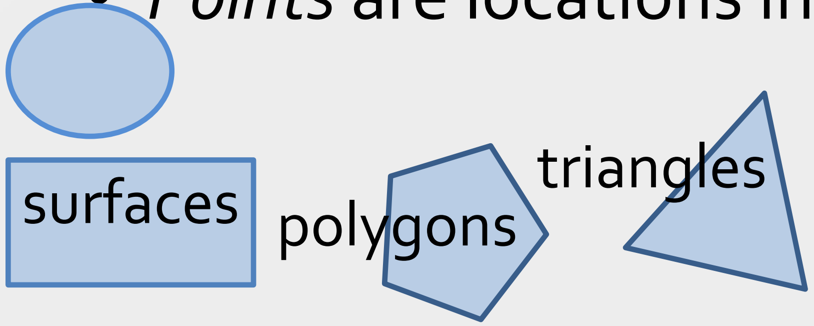
Geometric Modeling

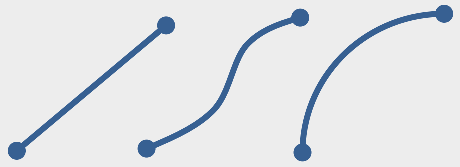


A typical solid model is defined by solids, surfaces, curves, and points.

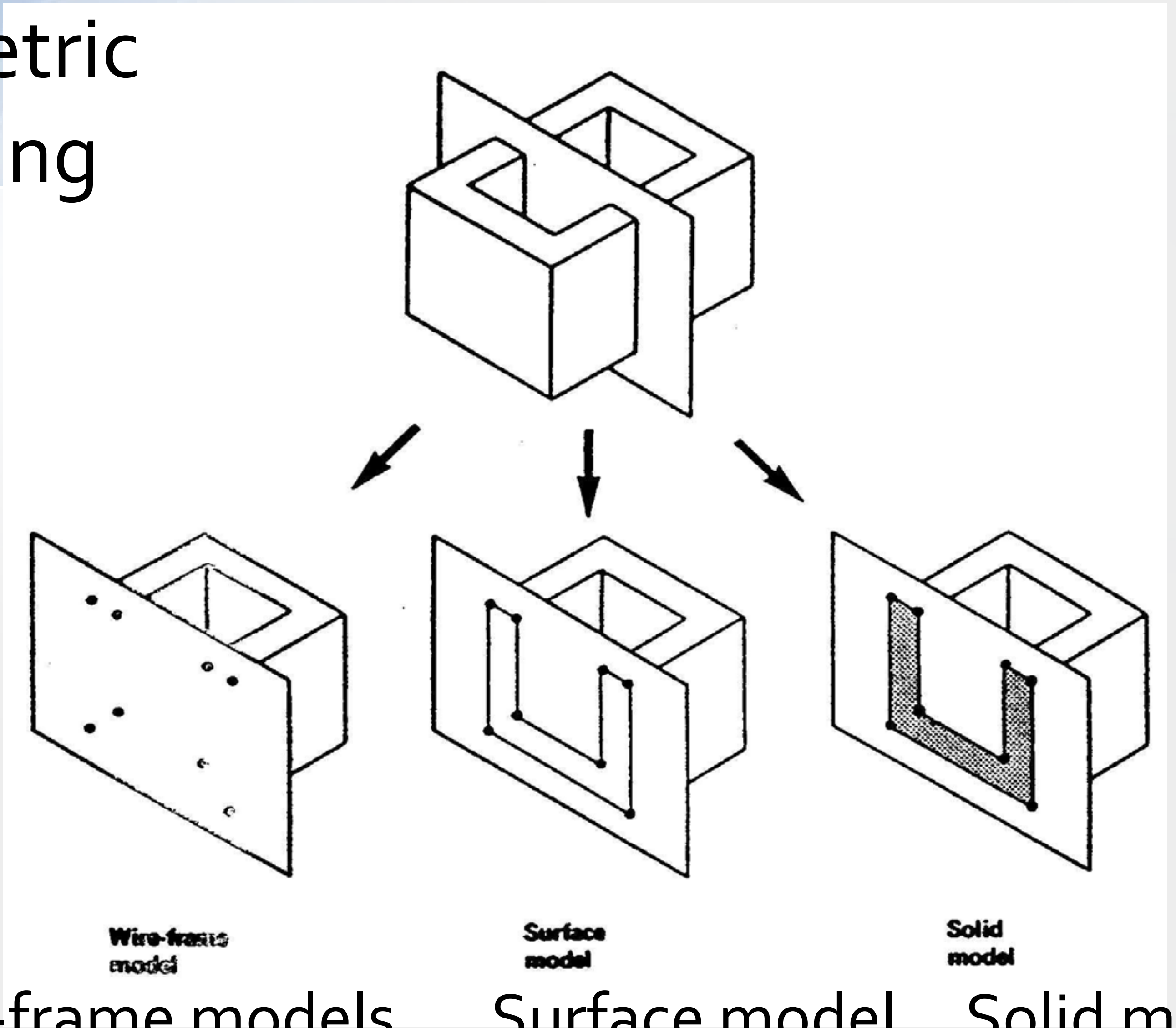
- *Solids* are bounded by surfaces. They represent solid objects. Analytic shape
- ***Surfaces* are bounded by lines.** Surfaces of solid objects, or planar or shell objects. Quadric surfaces, sphere, ellipsoid, torus

- *Curves* are bounded by points. Edges of objects. Lines, polylines, curves
- *Points* are locations in 3-D space. Vertices of objects.



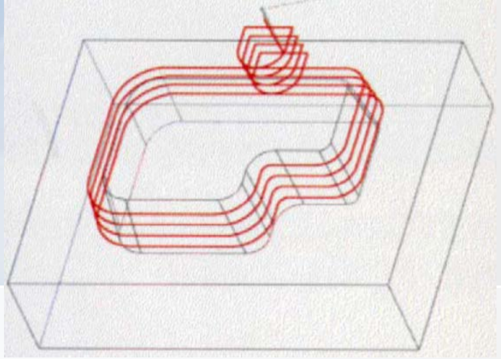
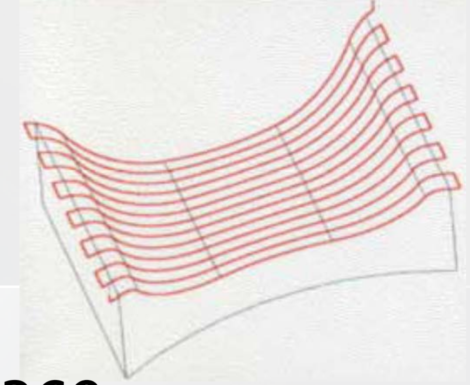
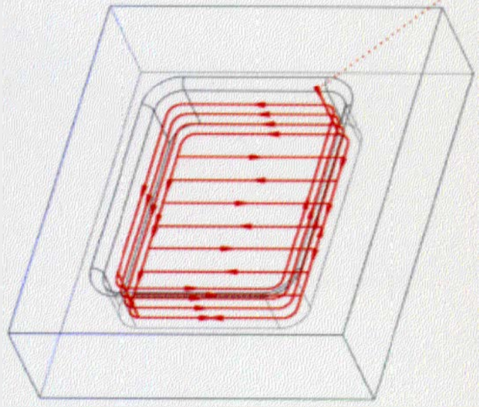
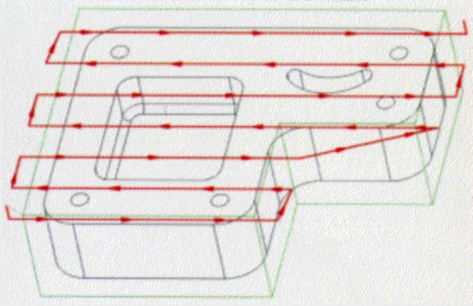


Geometric Modeling

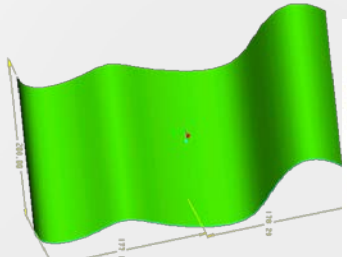
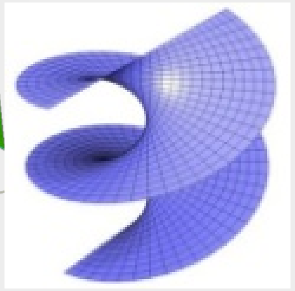
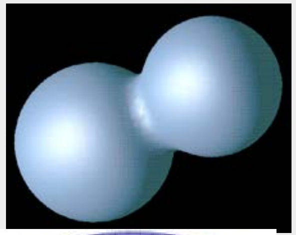
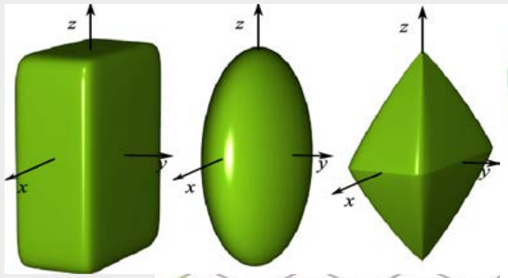
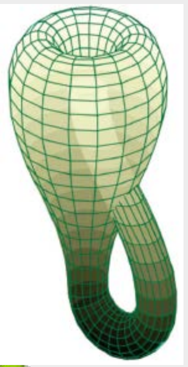
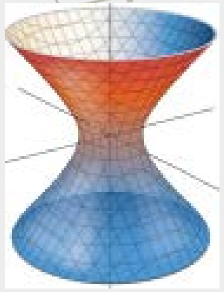
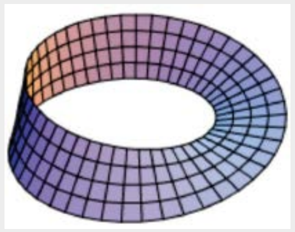
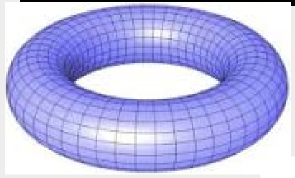
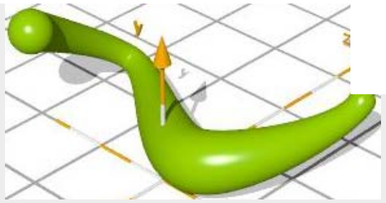
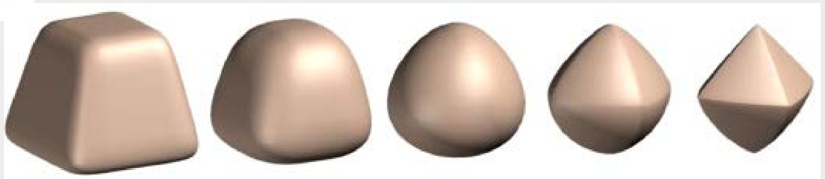


Wire-frame models Surface model Solid model

Surface Modeling

Models 2D surfaces in 3D space
All points on surface are defined
  useful for machining, visualization, etc.
Surfaces have **no thickness**,
objects have **no volume** or **solid properties.**
Surfaces may be **open** and **closed.**

Surfaces

Explicit : Value of dependent variable in terms of independent variable(s), e.g. **$z = f(x,y)$**

Implicit : e.g. **$f(x,y,z) = 0$**

Parametric : Express value of each spatial variable in terms of independent variables (the parameters)

e.g. for parameters u and w in 3D:

$x = x(u,w)$

$y = y(u,w)$

$z = z(u,w)$

Surface Modeling

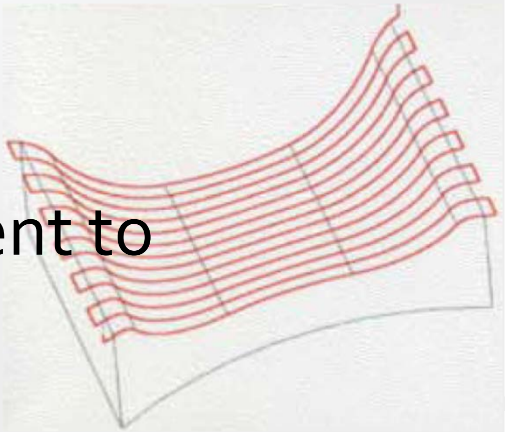
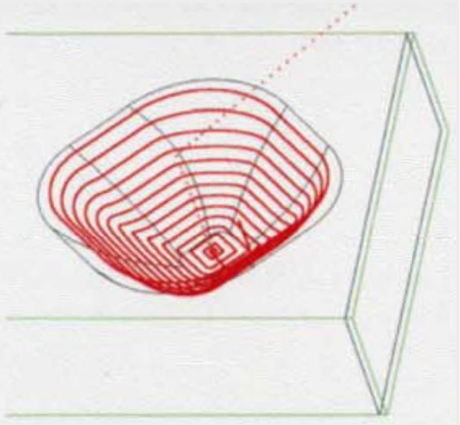
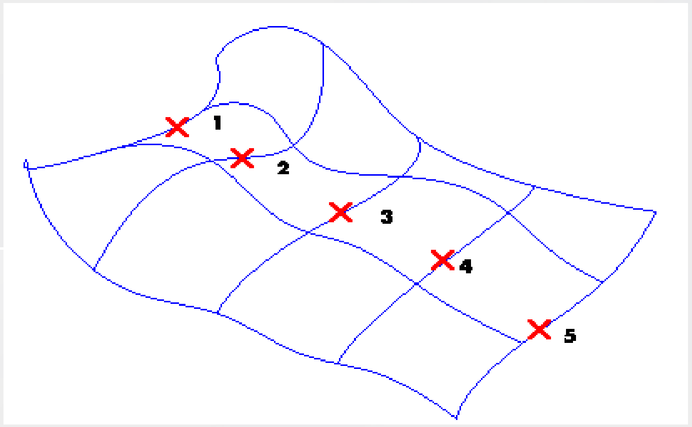
Extension of curve modeling

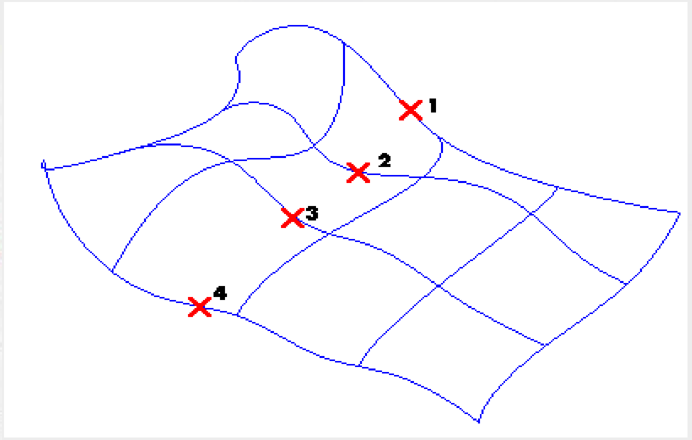
Parametric representation:

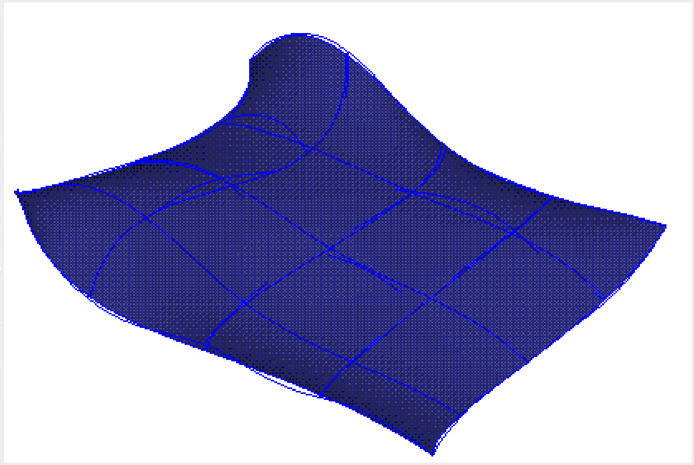
$$p = p(u,v)$$

which is equivalent to

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

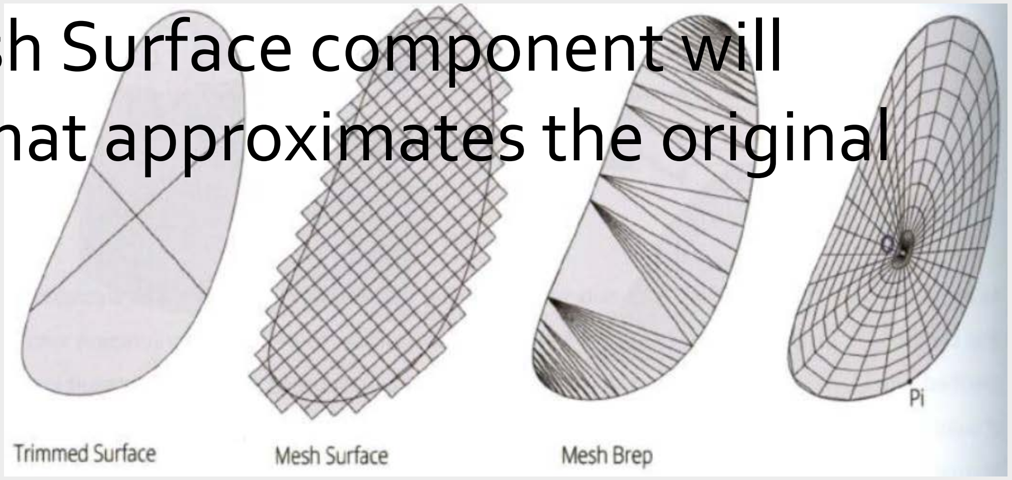








Trimmed surfaces return a polygon mesh with the original boundary geometry

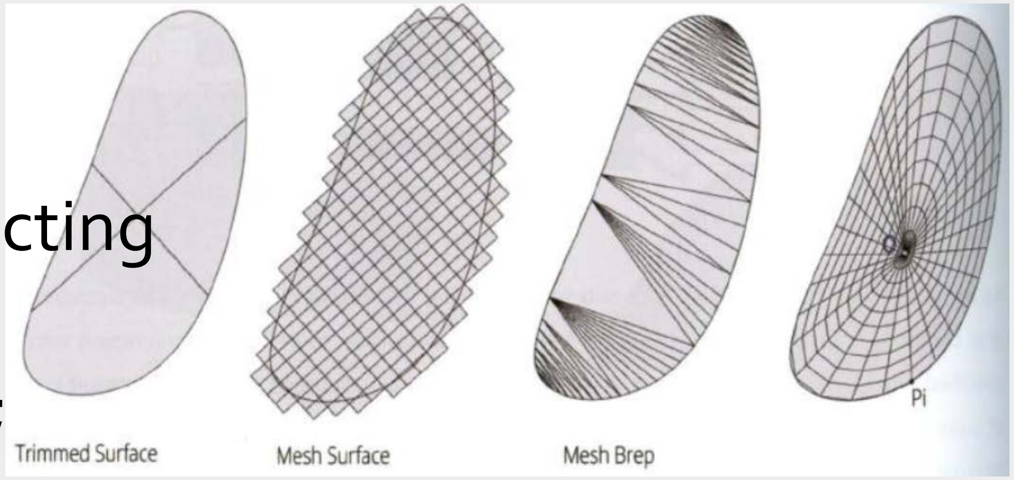
NURBS to mesh conversion is compatible with untrimmed surfaces, since untrimmed surfaces contain a rectangular isogrid that can be converted into a set of quad meshes. Unlike untrimmed surfaces, trimmed surfaces are not compatible with the NURBS to mesh conversion method. Trimmed surfaces converted using the Mesh Surface component will return a polygon mesh that approximates the original boundary geometry.



Trimmed surfaces return a polygon mesh with the original boundary geometry

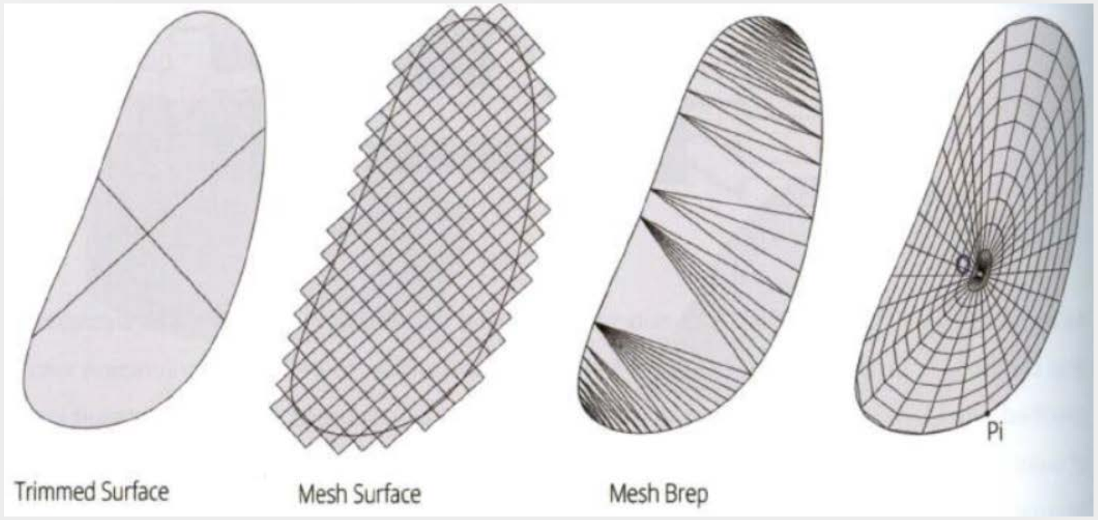
A possible strategy to convert a trimmed surface into a mesh is summarized as follows:

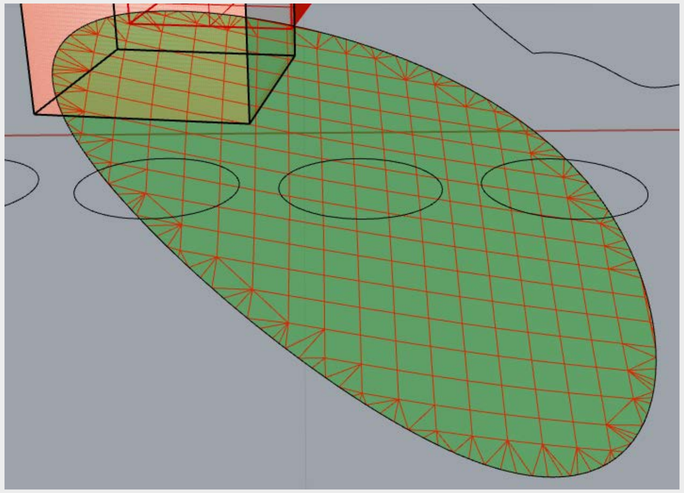
1. Adding an arbitrary point O;
2. Dividing the surface's boundary into N parts getting N points P;
3. Creating a set of lines from O to the subdivision points P.;
4. Dividing the lines into equal parts and connecting the resulting points with a set of polylines;



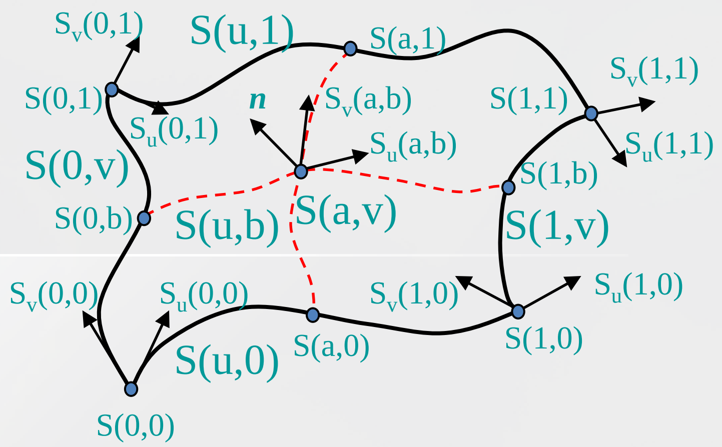
Trimmed surfaces return a polygon mesh with the original boundary geometry but allows for thin triangles

5. Splitting the initial surface using the network of lines obtained in step 3 and step 4, to generate a set of trimmed sub-surfaces;
6 . Extract the sub-surfaces' vertices and create the mesh relying on topology method.





Surface Modeling

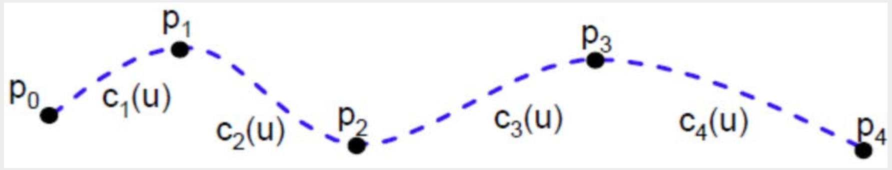


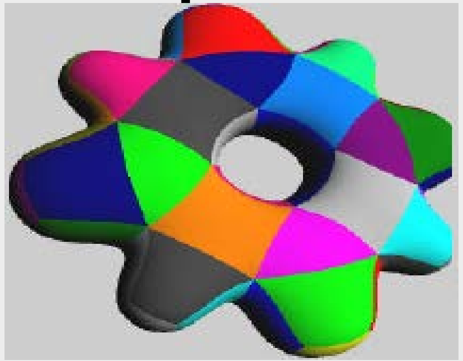
In three-dimensional space,

a surface would have the following parametric description: $x = x(u,v)$ $y = y(u,v)$ $z = z(u,v)$

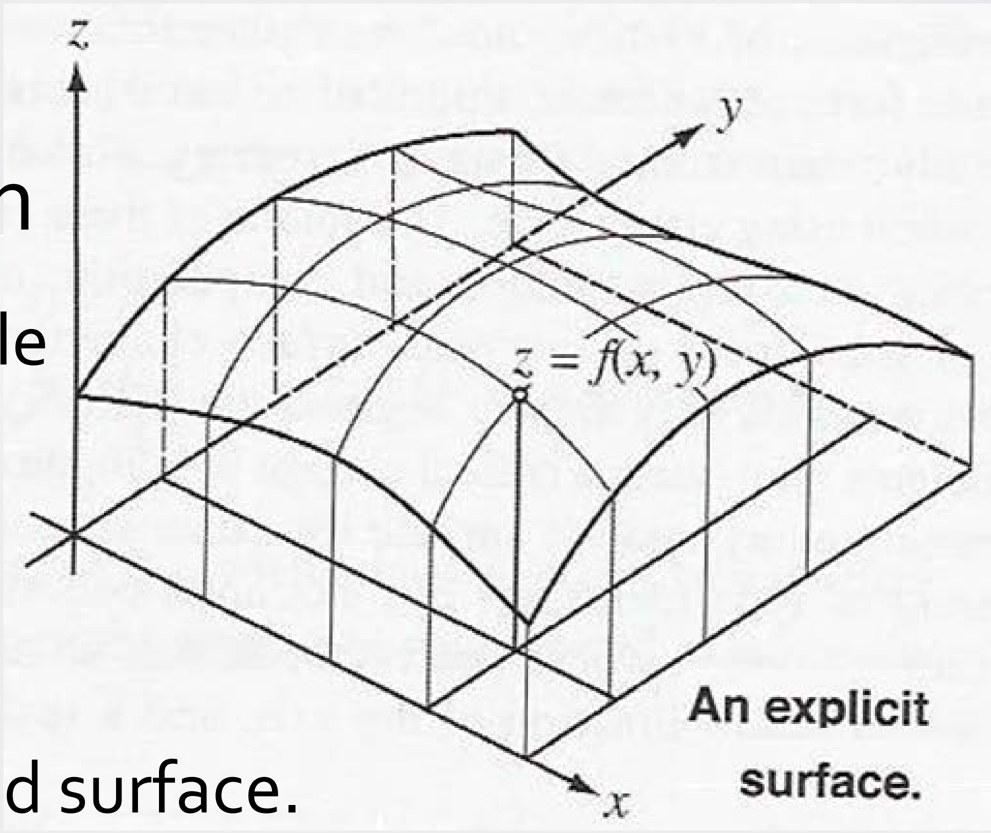
where a point is defined by $\mathbf{p}(u,v)$, and $u, v \in [0, 1]$.

Some possible approaches to **surface fitting** to a set of data points: **Least-squares fit** of a single surface , Bézier and Spline fit of a **number of patched surfaces**.





Surface, Explicit form



An explicit surface.

Value of dependent variable in terms of independent variables, e.g. **z = f (x,y)**

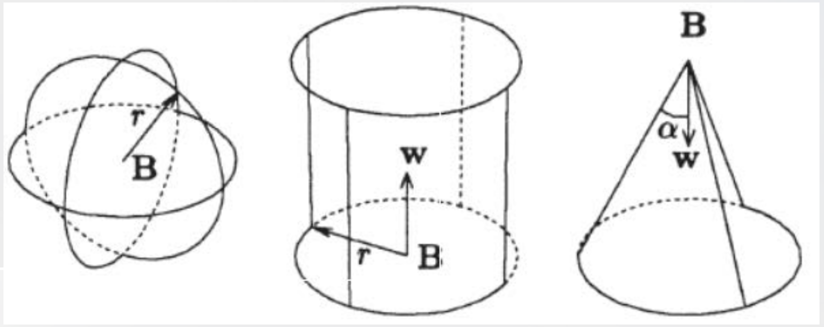
Axis-dependent

Can be hard to represent a transformed and bounded surface.

Sample surface-fitting procedure: determine $\mathbf{a}_{ij}$ coefficients from data points:

$$z(x,y)=\sum_{i=0}^{m}\sum_{j=0}^{n}a_{ij}x^{i}y^{i}$$

Surface, Implicit form



General form: f (x,y,z) = 0 , f (x,y,z) is polynomial in x, y, z such that: Axis-dependent $\sum_{i,j,k} a_{ijk} x^i y^j z^k = 0$

Examples:

Plane: Equation is linear in all its variables.

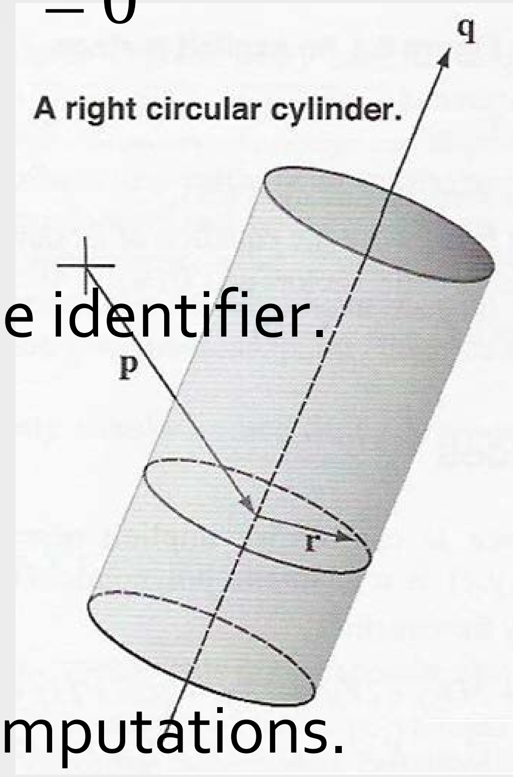
Quadric: Second-degree equation.

Can represent using vectors, scalars and a type identifier.

Right circular cylinder

- One vector gives a point on its axis
- One vector defines axis direction
- Scalar gives radius

Type testing requires robust floating-point computations.



Quadric Surfaces, Implicit form

Implicit form:

$$f(x,y,z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k = 0$$

Alternative to rational surfaces if only quadric surfaces are being represented.
Particularly useful for molecular modeling.

Reasons to use them:

- easy to compute normal
- easy to **test point inclusion**
- easy to compute z given x and y
- easy to **compute intersections** of one surface with another

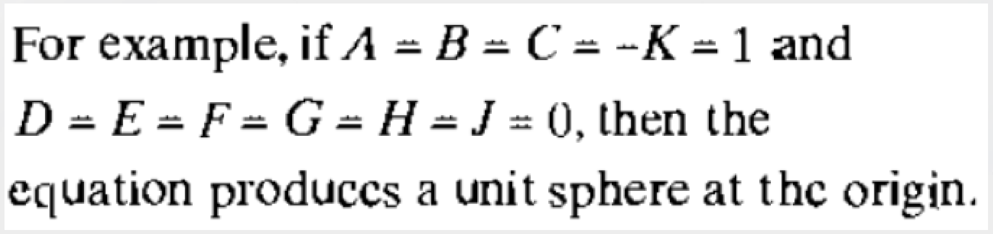
Implicit Form: Quadric Surfaces

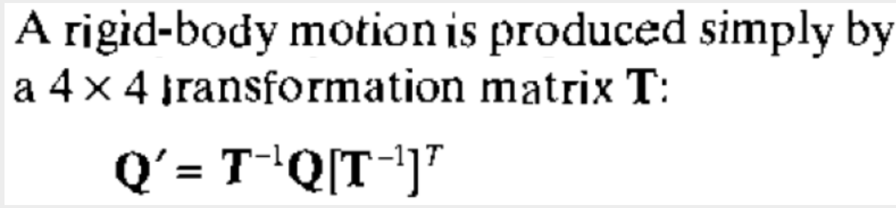
Type testing: f (x,y,z) = 0

$$Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyz + 2Fxz + 2Gx + 2Hy + 2Jz + K = 0$$

$$PQP^T = 0$$

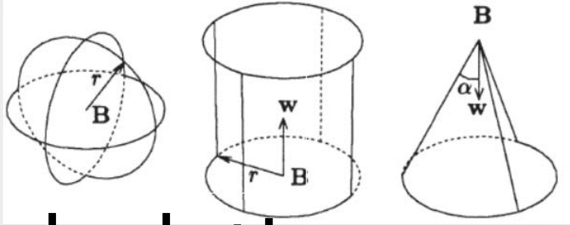
$$\begin{array}{ccc} P & Q & P^T \\ \begin{bmatrix} x & y & z & 1 \end{bmatrix} & \begin{bmatrix} A & D & F & G \\ D & B & E & H \\ F & E & C & J \\ G & H & J & K \end{bmatrix} & \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{array} = 0$$


For example, if $A = B = C = -K = 1$ and $D = E = F = G = H = J = 0$, then the equation produces a unit sphere at the origin.


A rigid-body motion is produced simply by a 4×4 transformation matrix $\mathbf{T}$:

$$\mathbf{Q}' = \mathbf{T}^{-1}\mathbf{Q}[\mathbf{T}^{-1}]^T$$

r*$_1$** = rank of ***Q (maximum number of linearly independent rows (or columns))



Surface intersection, Algebric calculation

$$q(x,y,z) = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} A & D & F & G \\ D & B & E & H \\ F & E & C & J \\ G & H & J & K \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \equiv \mathbf{x}Q\mathbf{x}^T$$

quadratic surfaces defined by $\mathbf{x}Q_1\mathbf{x}^T = 0, \qquad \mathbf{x}Q_2\mathbf{x}^T = 0$

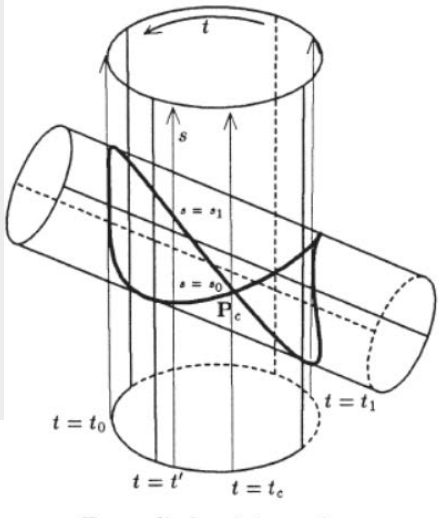
Suppose α is an arbitrary real number, $\mathbf{x}(Q_1 - \alpha Q_2)\mathbf{x}^T = 0$

its intersection line becomes: $a(t)s^2 + b(t)s + c(t) = 0$

When a quadratic surface forms a cylinder

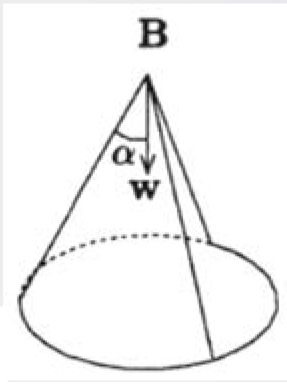
$x^2 + y^2 = r^2$, the functions $a(t)$, $b(t)$, and $c(t)$ are given by

$$\begin{aligned} a(t) &= C \\ b(t) &= 2Er\sin t + 2Fr\cos t + 2J \\ c(t) &= Ar^2\cos^2 t + Br^2\sin^2 t + 2Dr^2\sin t\cos t \\ &\quad +2Gr\cos t + 2Hr\sin t + K \end{aligned}$$

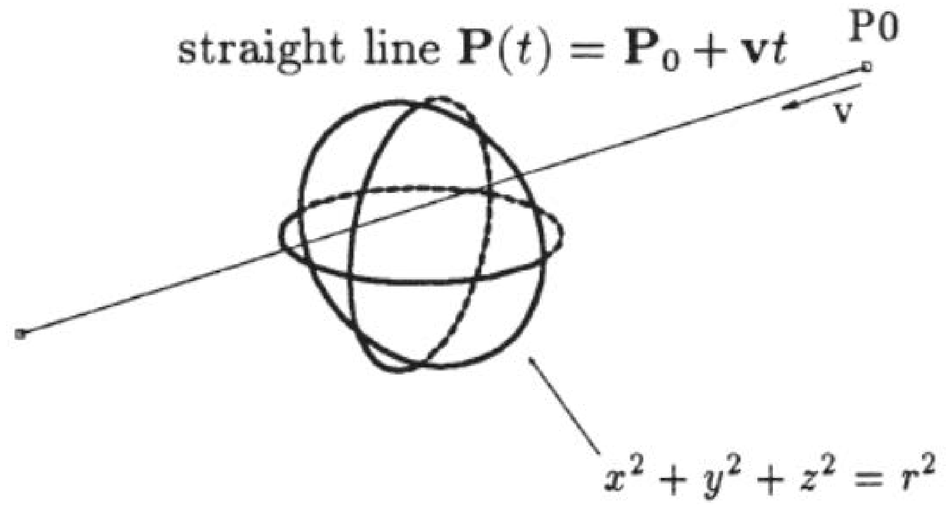


Two cylinders intersecting

Sphere and line intersection



straight line $\mathbf{P}(t) = \mathbf{P}_0 + \mathbf{v}t$



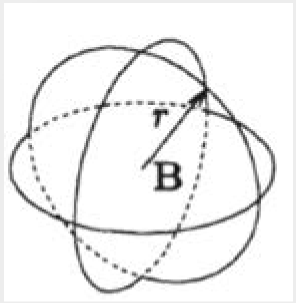
$x^2 + y^2 + z^2 = r^2$

Intersection calculation between a straight line and a sphere

$$a(t)s^2 + b(t)s + c(t) = 0$$

When it forms a cone $x^2 + y^2 - m^2z^2 = 0$,

$$\begin{aligned} a(t) &= Am^2\cos^2 t + Bm^2\sin^2 t + C + 2Dm^2\sin t\cos t \\ &\quad +2Em\sin t + 2Fm\cos t \\ b(t) &= 2Gm\cos t + 2Hm\sin t + 2J \\ c(t) &= K \end{aligned}$$



$$\mathbf{P}(t) = \mathbf{P}_0 + \mathbf{v}t$$

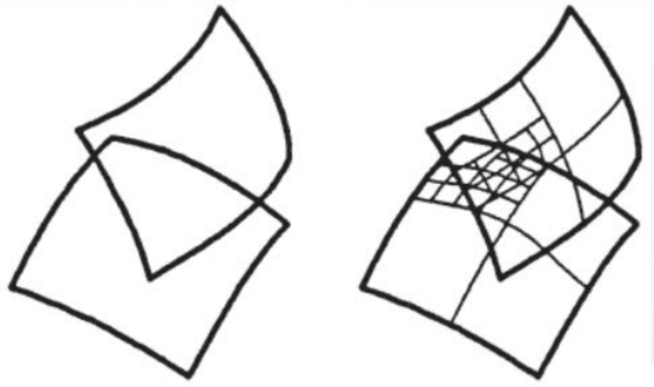
$$x^2 + y^2 + z^2 = r^2$$

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{0x} \\ \mathbf{P}_{0y} \\ \mathbf{P}_{0z} \end{bmatrix} + t \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{v}_z \end{bmatrix}$$

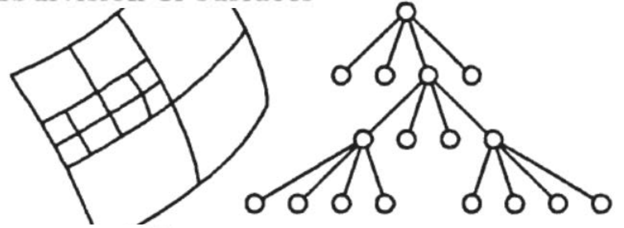
$$(\mathbf{P}_{0x} + t\mathbf{v}_x)^2 + (\mathbf{P}_{0y} + t\mathbf{v}_y)^2 + (\mathbf{P}_{0z} + t\mathbf{v}_z)^2 = r^2$$

This equation is a quadratic expression about t and can be easily solved.

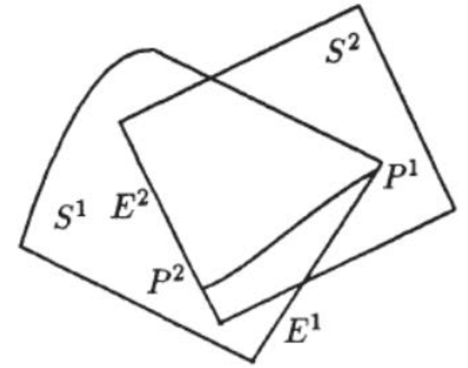
Surface intersection



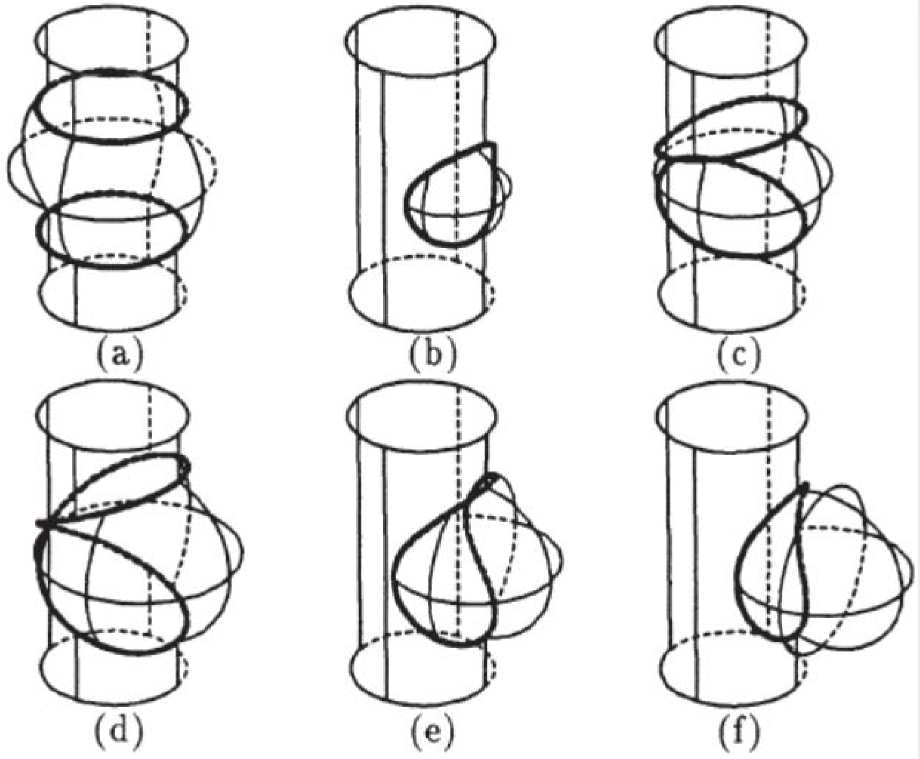
Recursive subdivision of surfaces



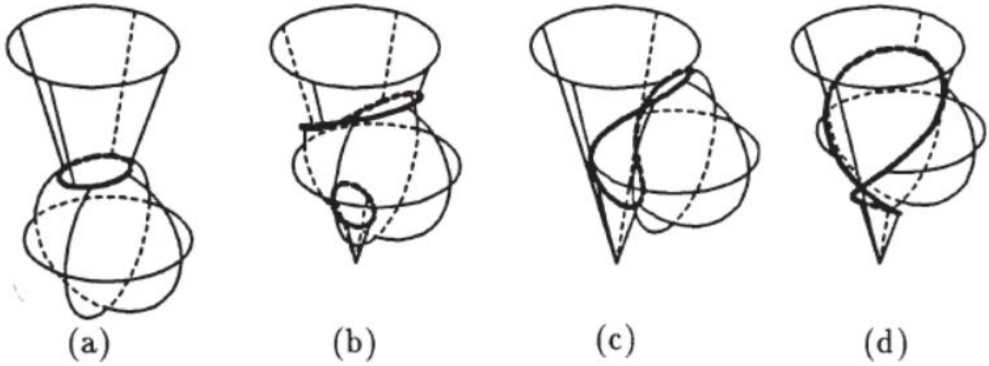
data in a quadtree



Calculating intersection between surfaces using the marching method



Intersection curves between a sphere and a cylinder

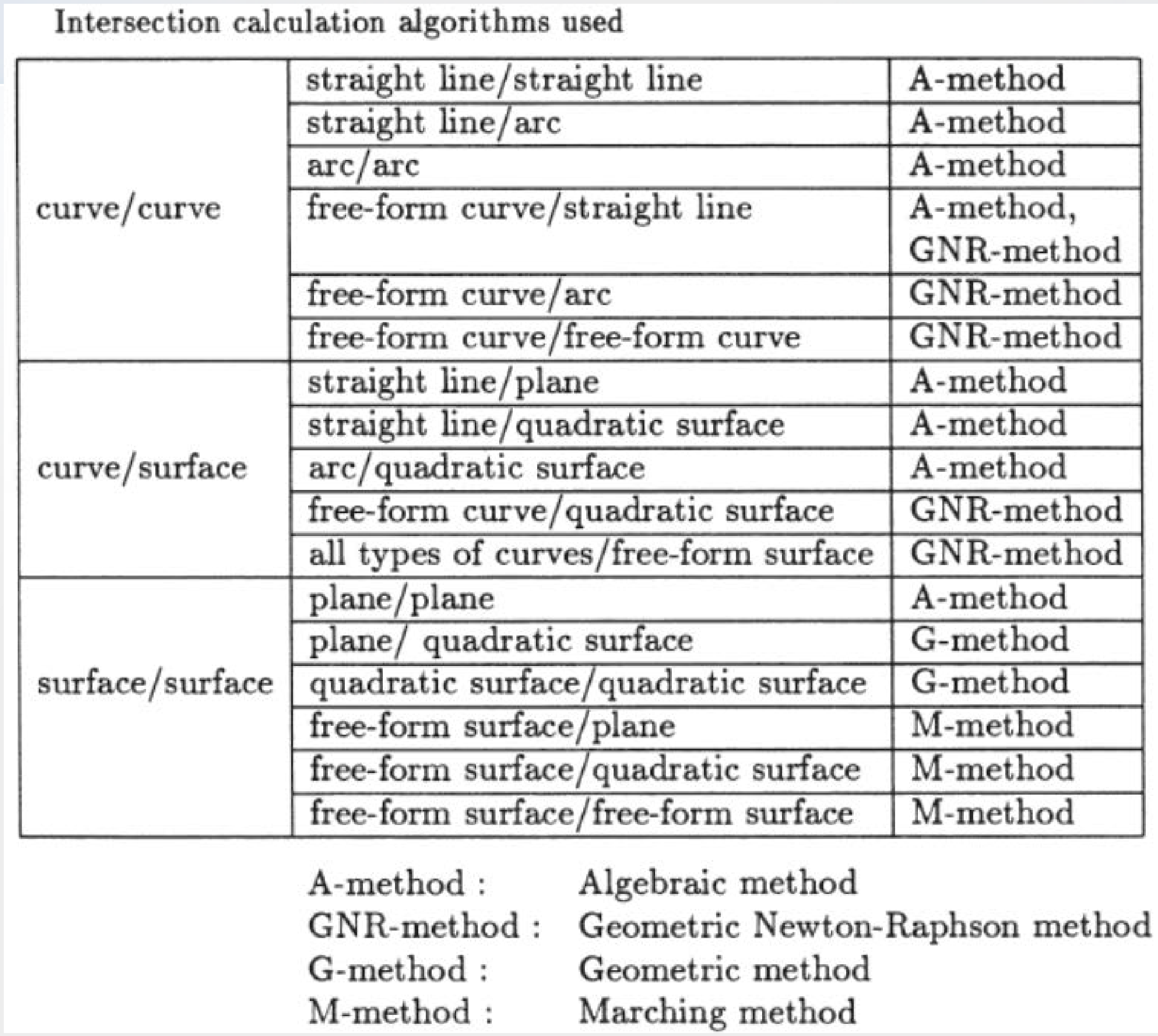


Intersection curves between a sphere and a cone

Intersections

Intersection Calculations

ref. book:
3d CAD Principles and Applications, Toriya, 1993.



Intersection calculation algorithms used

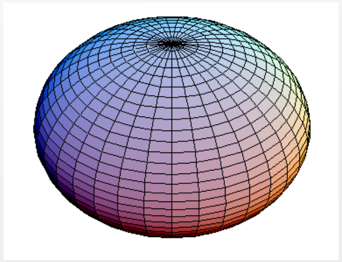
curve/curve	straight line/straight line	A-method
	straight line/arc	A-method
	arc/arc	A-method
	free-form curve/straight line	A-method, GNR-method
	free-form curve/arc	GNR-method
	free-form curve/free-form curve	GNR-method
curve/surface	straight line/plane	A-method
	straight line/quadratic surface	A-method
	arc/quadratic surface	A-method
	free-form curve/quadratic surface	GNR-method
	all types of curves/free-form surface	GNR-method
surface/surface	plane/plane	A-method
	plane/ quadratic surface	G-method
	quadratic surface/quadratic surface	G-method
	free-form surface/plane	M-method
	free-form surface/quadratic surface	M-method
	free-form surface/free-form surface	M-method

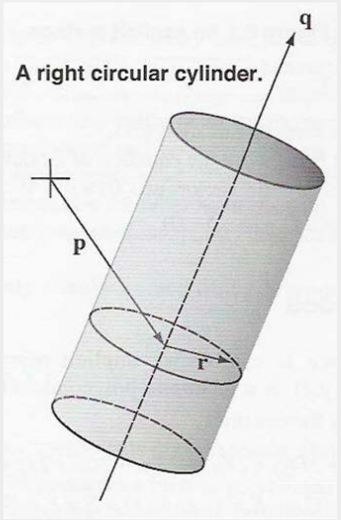
A-method :	Algebraic method
GNR-method :	Geometric Newton-Raphson method
G-method :	Geometric method
M-method :	Marching method

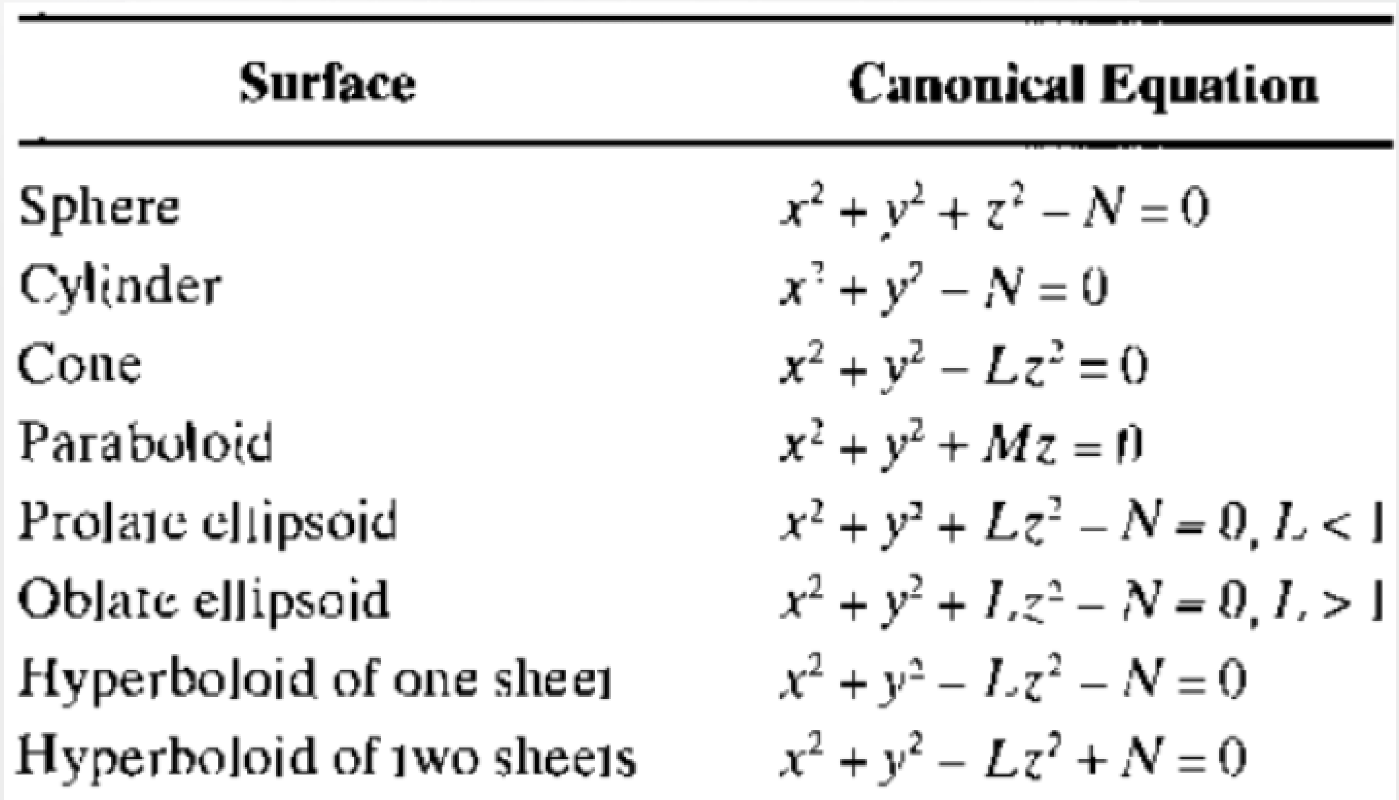
Quadric Surfaces of Revolution

Rotate conic curve about its axis. Center or vertex at origin. Axes of symmetry coincide with coordinate axes.

$$x^2 + y^2 + Lz^2 + Mz + N = 0 \text{ where } LM = 0$$

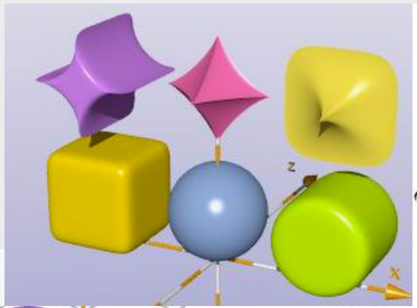


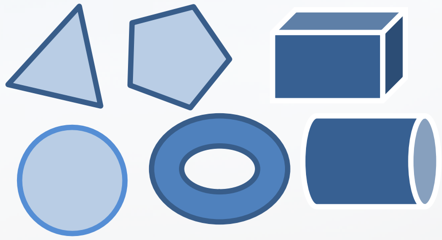
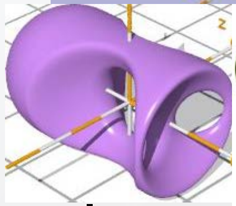
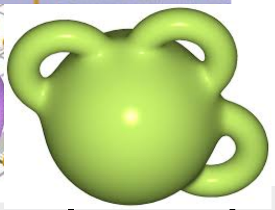




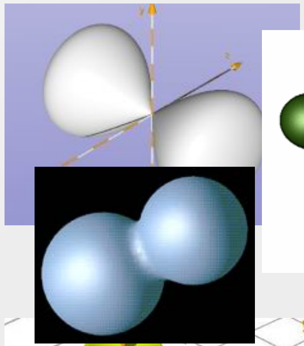
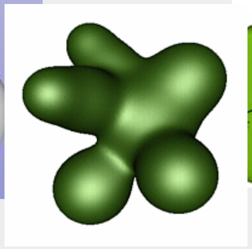
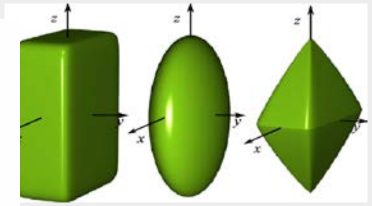
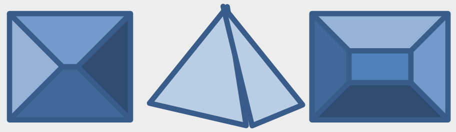
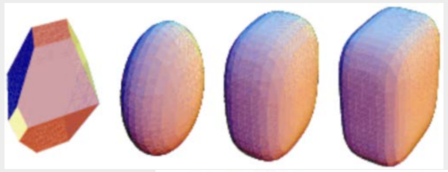
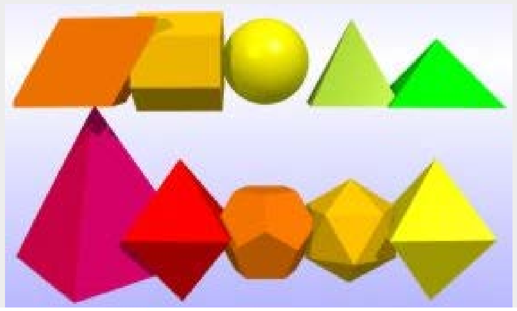
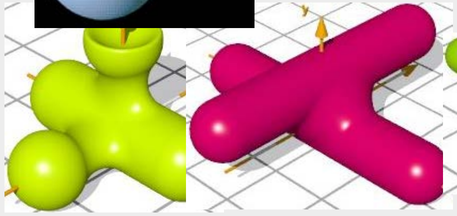
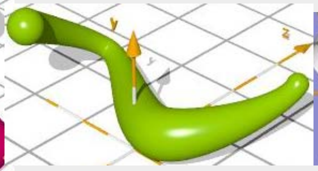
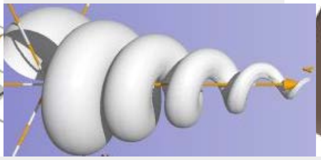
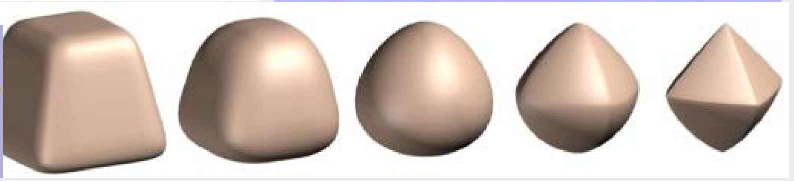
Surface	Canonical Equation
Sphere	$x^2 + y^2 + z^2 - N = 0$
Cylinder	$x^2 + y^2 - N = 0$
Cone	$x^2 + y^2 - Lz^2 = 0$
Paraboloid	$x^2 + y^2 + Mz = 0$
Prolate ellipsoid	$x^2 + y^2 + Lz^2 - N = 0, L < 1$
Oblate ellipsoid	$x^2 + y^2 + Lz^2 - N = 0, L > 1$
Hyperboloid of one sheet	$x^2 + y^2 - Lz^2 - N = 0$
Hyperboloid of two sheets	$x^2 + y^2 - Lz^2 + N = 0$

3D Analytic Surfaces



2.0
1.5
1.0
ϵ_2
0.8
0.2
0.2
0.8
1.0
1.5
2.0
ϵ_1




- Triangular meshes, polygonal meshes
- Quadric surfaces, sphere, ellipsoid, torus
- Superquadric surfaces, superellipse, superellipsoid
- Blobby models, tetrahedron, pyramid, hexahedron

Surface, Parametric Form

Express value of each spatial variable in terms of independent variables (the parameters)

e.g. for parameters **u, w** in 3D:

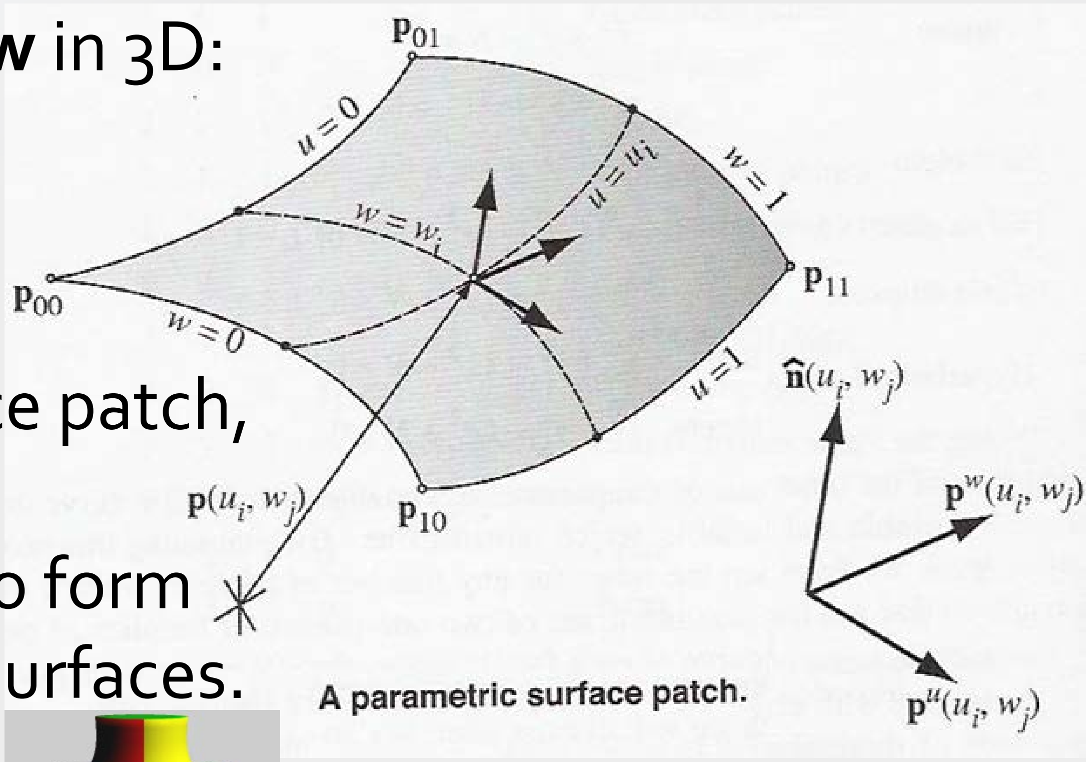
x = x (u,w)

y = y (u,w)

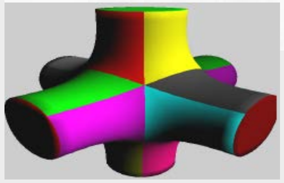
z = z (u,w)

For a rectangular surface patch, typically $u, w \in [0,1]$

Patches can be joined to form composite parametric surfaces.



A parametric surface patch.



Surface, Parametric Form

Sample patch: rectangular segment of x, y plane

$x = (c - a)u + a$

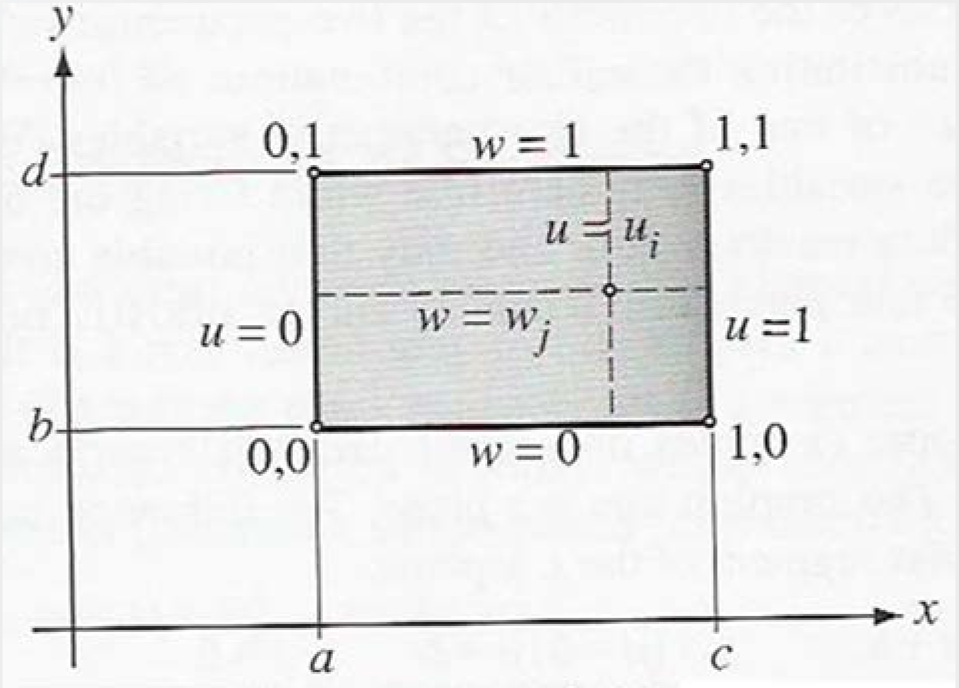
$y = (d - b)w + b$

$z = 0$

Here:

Curves of constant $\boldsymbol{w}$ are horizontal lines.

Curves of constant $\boldsymbol{u}$ are vertical lines.



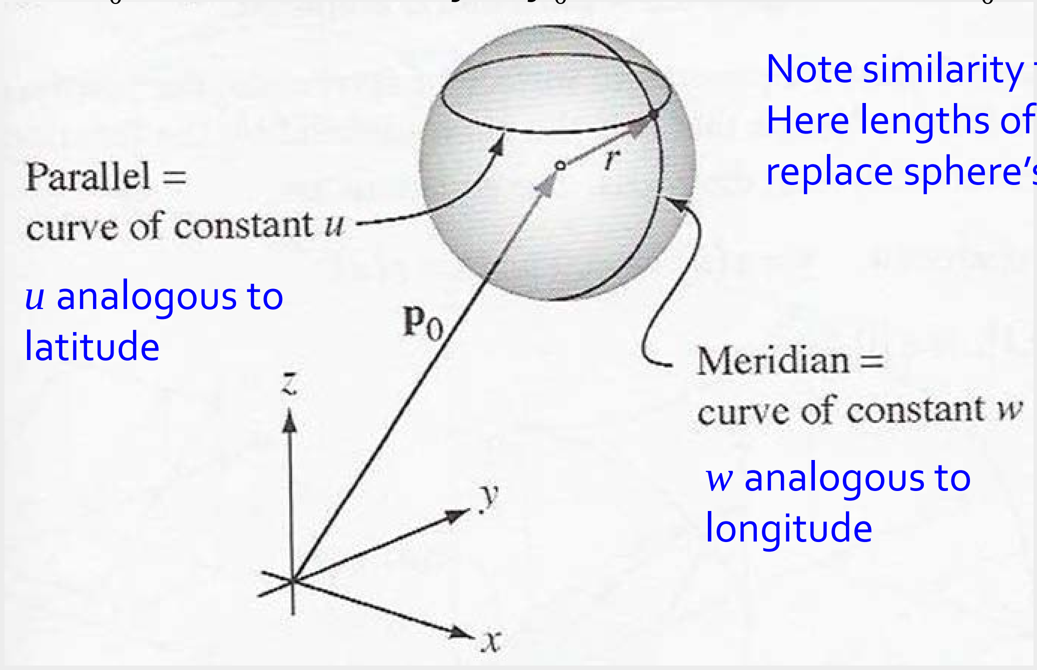
Parametric and **x, y** coordinates of a plane.

Surface, Parametric Form

Parametric sphere and ellipsoid of radius *r*, centered on (x_0, y_0, z_0):

$$x = x_0 + r\cos u\cos w \quad y = y_0 + r\cos u\sin w \quad z = z_0 + r\sin u \quad \text{where } u \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \quad w \in [0, 2\pi]$$

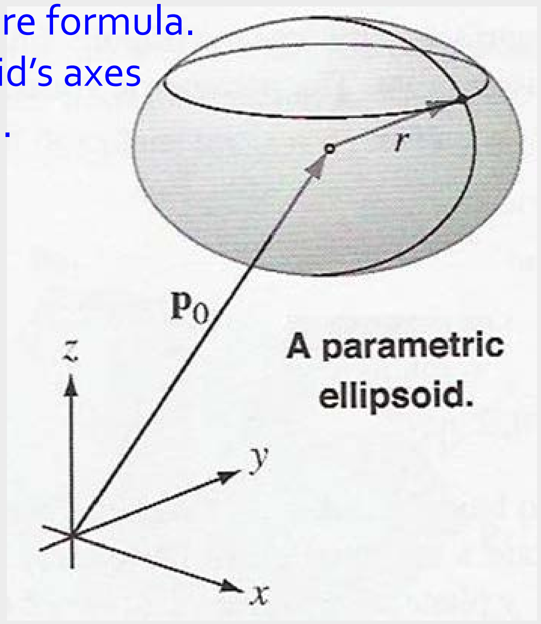
$$x = x_0 + a\cos u\cos w \quad y = y_0 + b\cos u\sin w \quad z = z_0 + c\sin u \quad \text{where } u \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \quad w \in [0, 2\pi]$$



u analogous to latitude

w analogous to longitude

Note similarity to sphere formula. Here lengths of ellipsoid's axes replace sphere's radius.



A parametric ellipsoid.

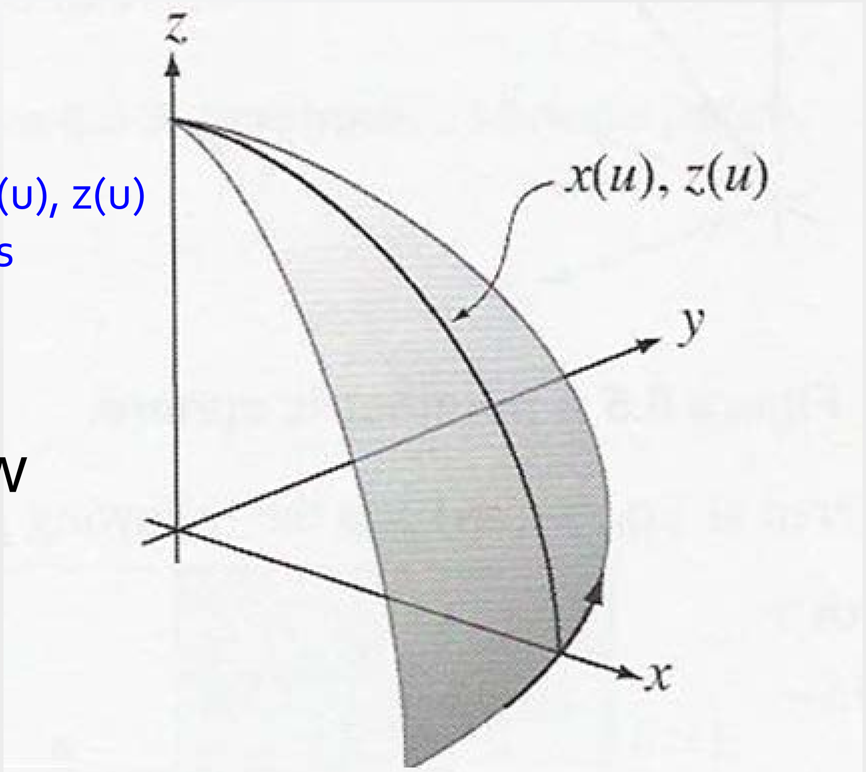
Surface, Parametric Form

Parametric surface of revolution:

$$x = x(u)\cos w \quad y = x(u)\sin w \quad z = z(u) \quad \text{where } u \in [0,1], \quad w \in [0,2\pi]$$

revolving x(u), z(u)
about z axis

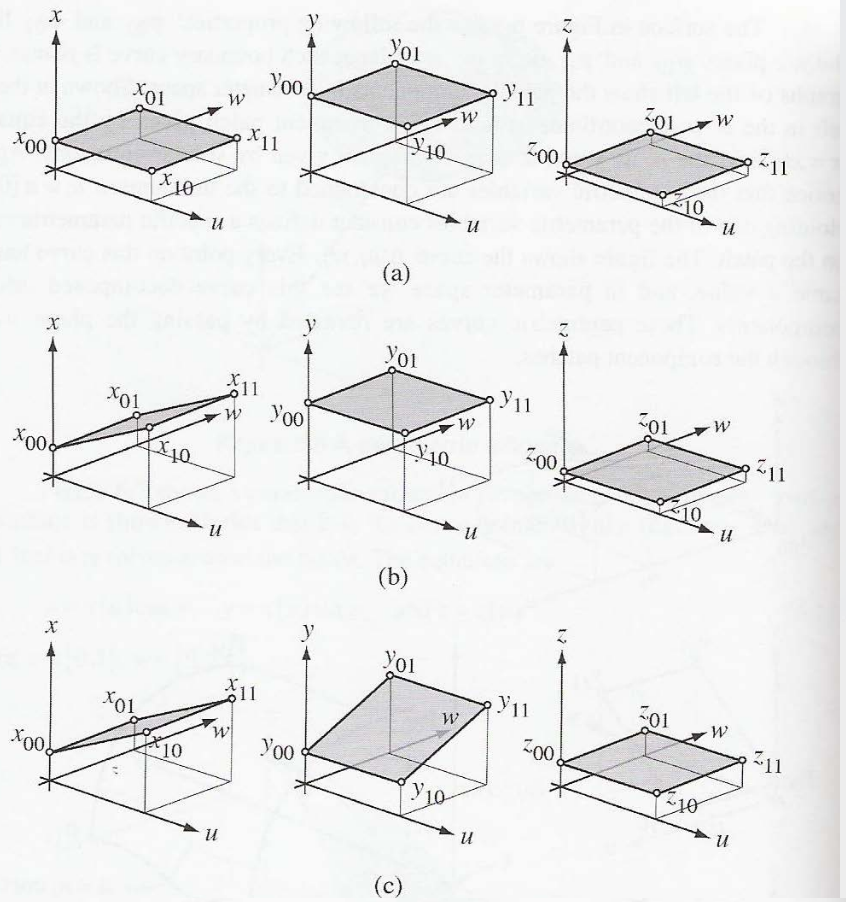
partial view



A parametric surface of revolution.

Bicubic surface patch in Parameter Space

point, line, and planar patch in parameter space



Bicubic surface patch's components in parameter space

Bicubic surface patch in model space

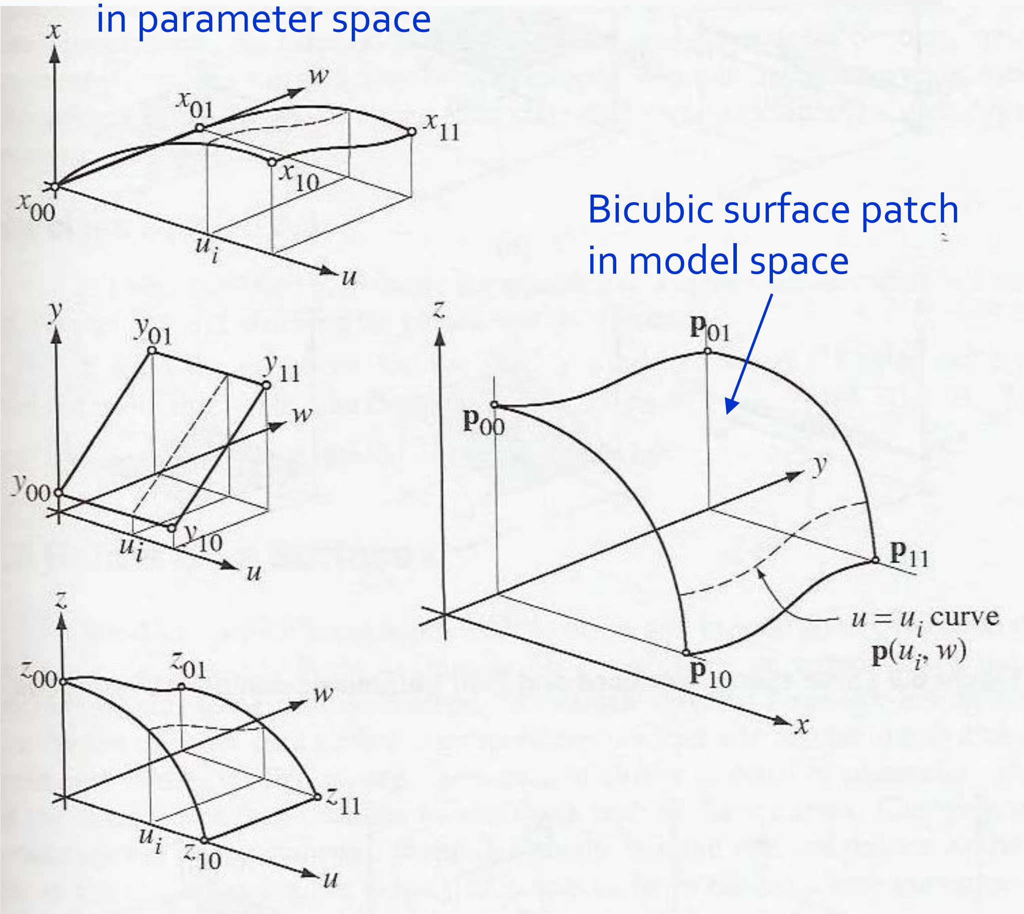


Figure 6.8 The parameter space of a surface.

point, line, and planar patch in model space

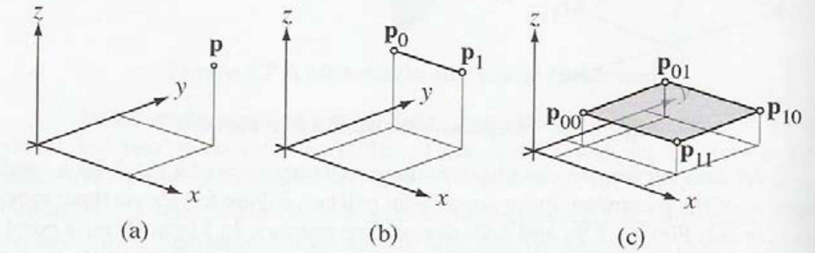


Figure 6.10 Three more special surfaces.

Curves on Surfaces

Isoparametric curve:
One parameter varies while the other is constant.

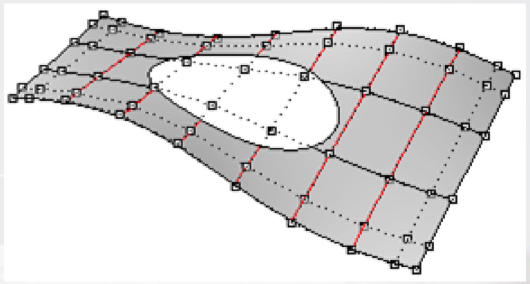
Parametric curve net on a patch

Two 1-parameter families of curves such that through each point there passes just one of each family.

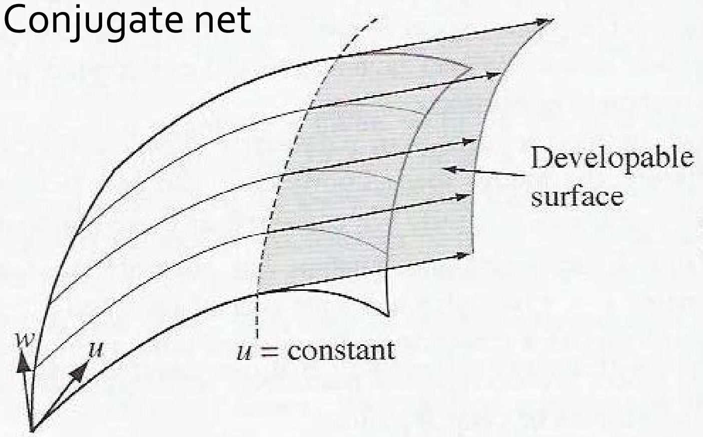
2 tangents of the curves at each point must be distinct.
Orthogonal tangents produce orthogonal net.

source: Mortenson

Curves on Parametric Surfaces



Conjugate net



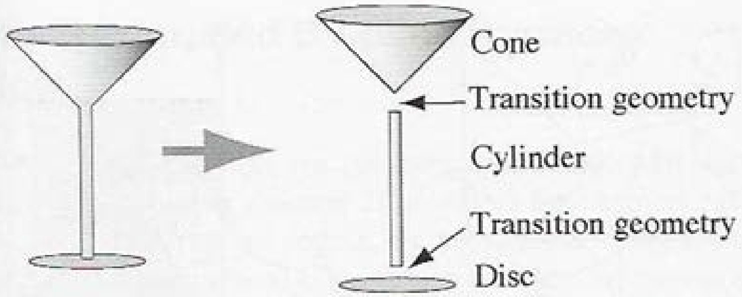


Figure 6.13 Decomposition of a complex shape.

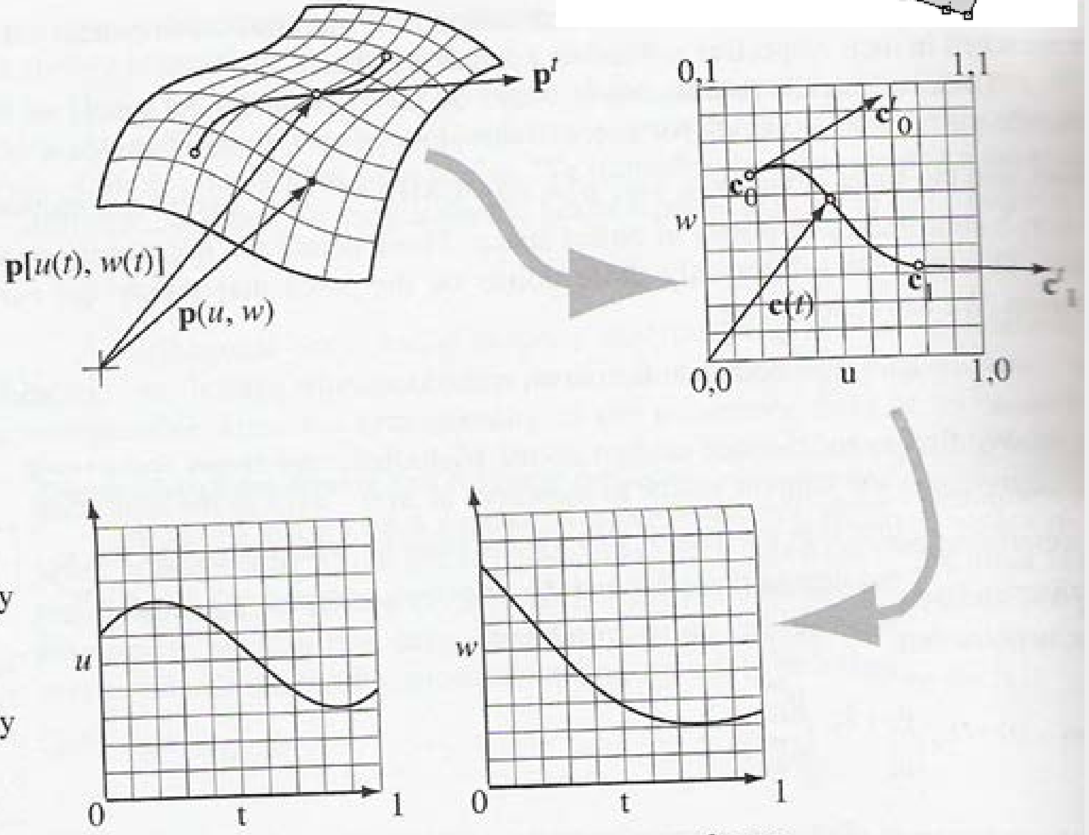
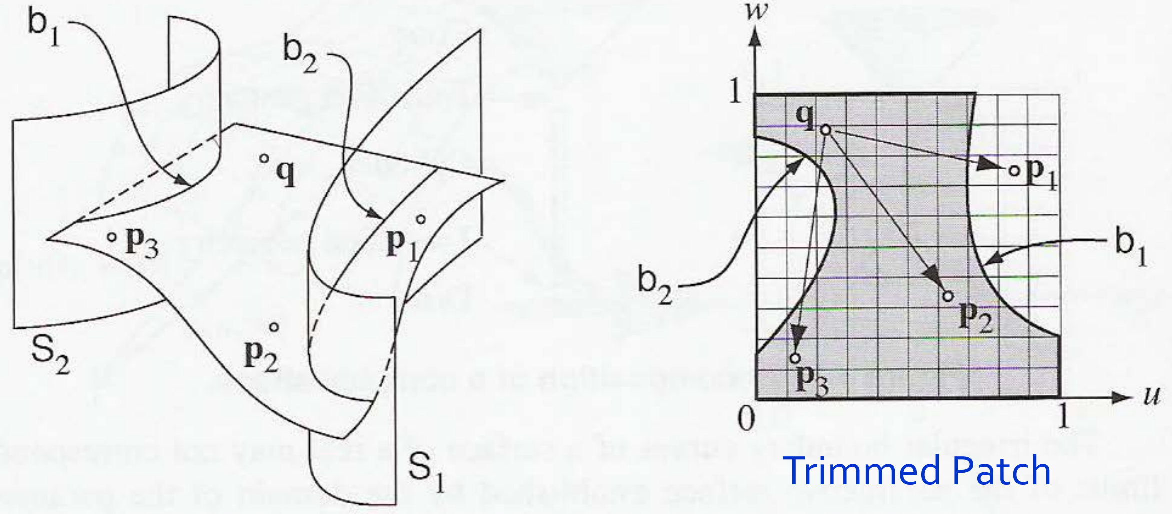


Figure 6.12 Curves on surfaces.

Surface with Irregular Boundary

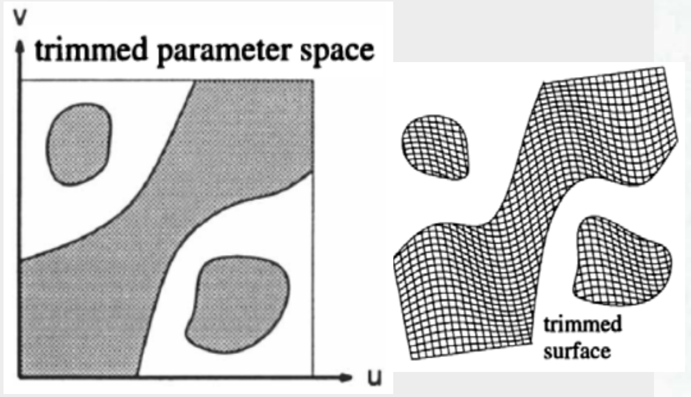


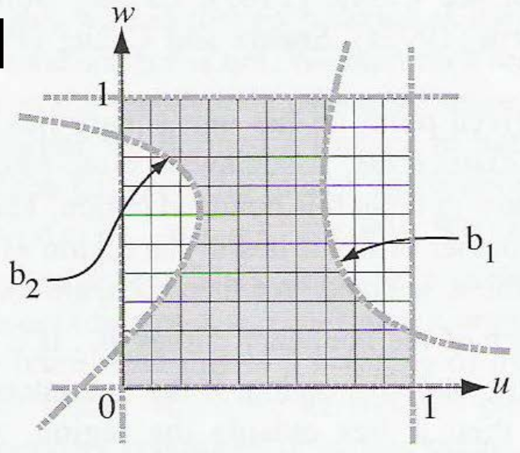
Trimmed Patch

Figure 6.14 Surface with irregular boundary.

Point Classification: count number of times a line segment to interior point q crosses each boundary curve.

$$\mathbf{b}_i = [u_i(t) \quad w_i(t)] \qquad t \in [0,1]$$

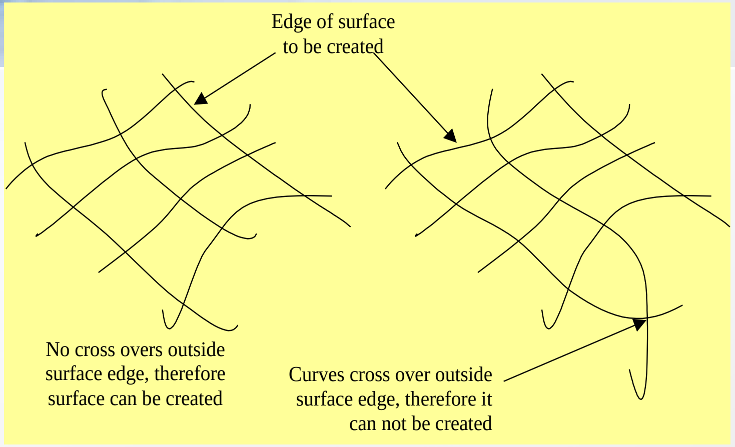


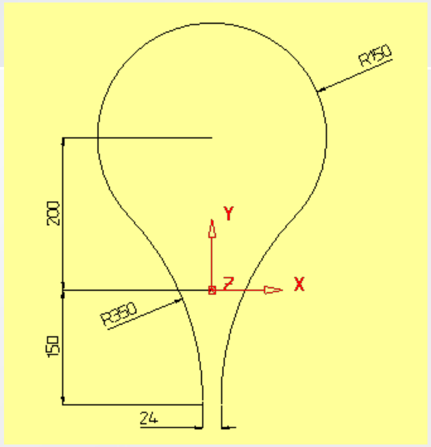


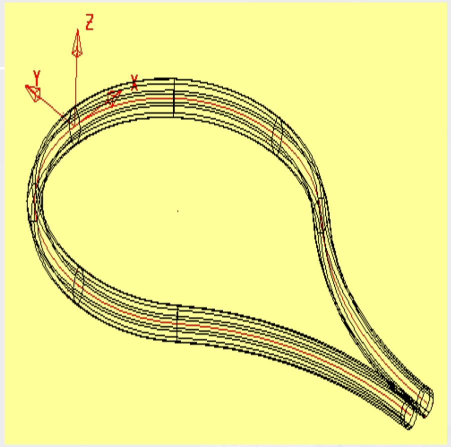
$$u \ge 0 \quad w \ge 0 \quad b_1 \ge 0$$
$$1-u \ge 0 \quad 1-w \ge 0 \quad b_2 \ge 0$$

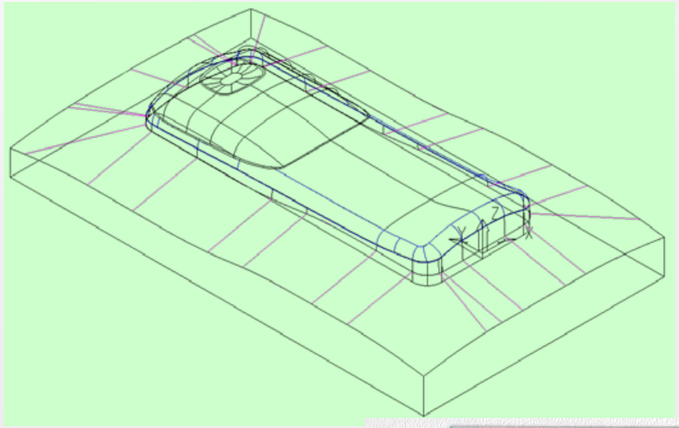
Figure 6.15 Irregular surface defined by the intersection of halfspaces.

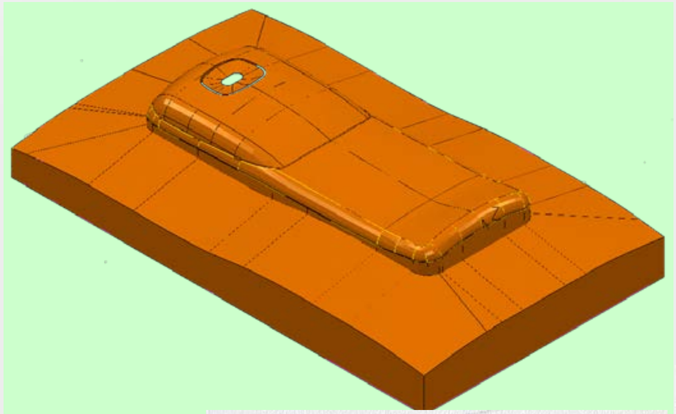
Surfaces from Curves

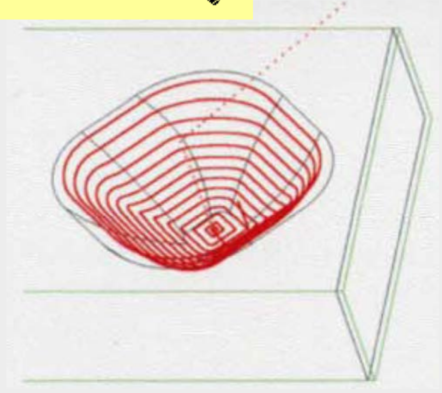


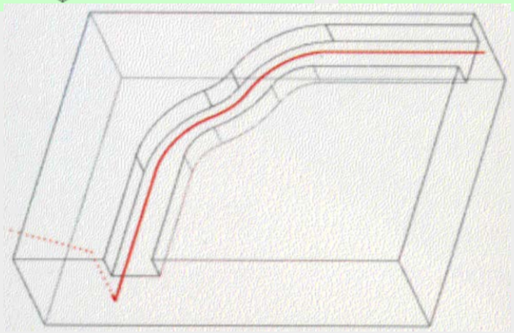


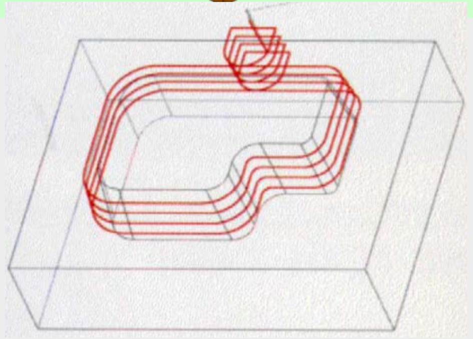


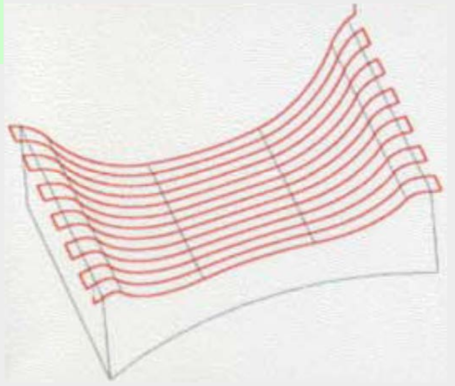










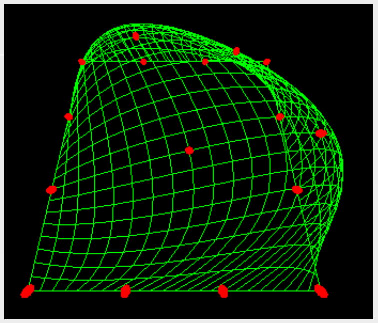


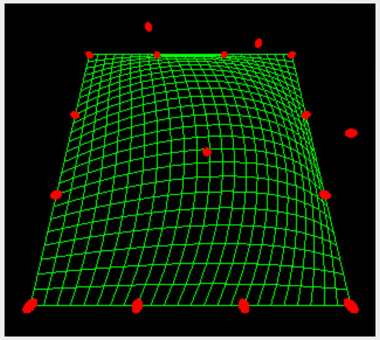
4 Typical Types of Parametric Surface Patches

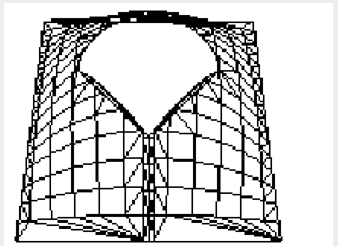
Control points influence surface shape.

- Interpolating
  - Defined by rectangular array of control points.
  - Surface patch passes through all control points.
- Hermite (bicubic)
  - Defined by 4 corner points, tangent vectors at 4 boundary curves, and “twist vectors” at corner points.
  - Interpolates all its corner points.
  - Not invariant under affine transformations.
- Bezier
  - Defined by rectangular array of control points.
  - Interpolates all its corner points.
  - Starting and ending tangents of each boundary curve are determined by control polyhedron at corner points.
  - Invariant under affine transformations.
  - Surface patch lies within convex hull of control polygon.
  - *Not* necessarily variation-diminishing.
  - Degrees of basis functions related to number of control points.
- B-Spline
  - Defined by rectangular array of control points.
  - Not guaranteed to interpolate control points.
  - Invariant under affine transformations.
  - Surface patch lies within convex hull of control polygon.
  - *Not* necessarily variation-diminishing.
  - Degrees of basis functions independent of number of control points.
  - More local control than Bezier.

source: Mortenson, Angel and more…


Bicubic Interpolating Patch


Bicubic Bezier Patch


Trimmed NURBS Surface: courtesy of Silicon Graphics

Bezier Surface Patch

Quadratic case (left), Cubic case (right)

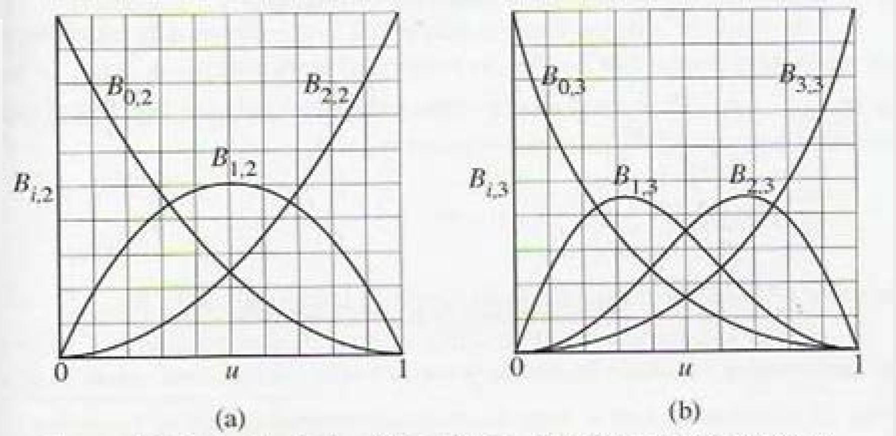


Figure 4.6 Bézier basis functions for 3 points (a), and 4 points (b).

Geometric form:

$$\mathbf{p}(u,w)=\sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{p}_{ij}B_{i,m}(u)B_{j,n}(w)$$

$$B_{i,m}(u)=\binom{m}{i}u^i(1-u)^{m-i}$$

Bernstein polynomials.

Degree in *u* parameter = *m*.

Degree in *w* parameter = *n*.

$$B_{i,n}(u)=\frac{n!}{i!(n-i)!}u^i(1-u)^{n-i}$$

$$\sum_{i=0}^{m}B_{i,m}(u)=1$$

Degree elevation to (*m*+1,*n*+1) is reduced to series of univariate degree elevation problems.

Convex combination, so Bezier surface points all lie within convex hull of control polyhedron.

Rational form is invariant under perspective transformation: where h_{ij} are projective space coordinates (weights)

$$\mathbf{p}(u,w)=\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}h_{ij}\mathbf{p}_{ij}B_{i,m}(u)B_{j,n}(w)}{\sum_{i=0}^{m}\sum_{j=0}^{n}h_{ij}B_{i,m}(u)B_{j,n}(w)}$$

source: Mortenson

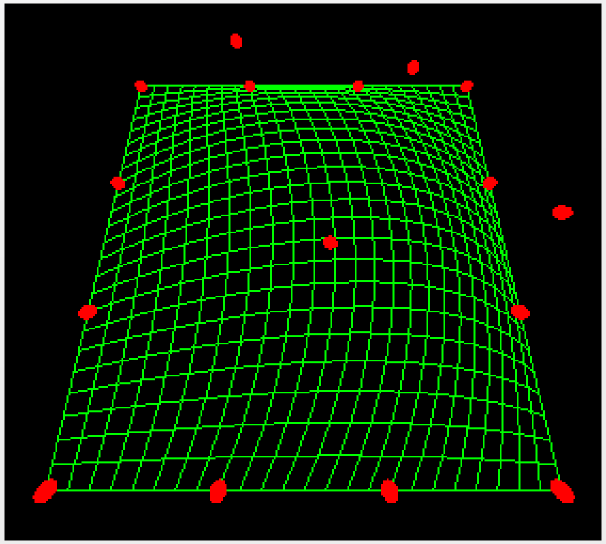
$$p(u,w) = U^T[M_B][P][M_B]^T W$$

$$[M_B] = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Bicubic Bezier Patch

$$U^T = [u^3 \quad u^2 \quad u \quad 1]$$

$$\mathbf{p}(u,w) = [(1-u)^3 \quad 3u(1-u)^2 \quad 3u^2(1-u) \quad u^3] \begin{bmatrix} \mathbf{p}_{11} & \mathbf{p}_{12} & \mathbf{p}_{13} & \mathbf{p}_{14} \\ \mathbf{p}_{21} & \mathbf{p}_{22} & \mathbf{p}_{23} & \mathbf{p}_{24} \\ \mathbf{p}_{31} & \mathbf{p}_{32} & \mathbf{p}_{33} & \mathbf{p}_{34} \\ \mathbf{p}_{41} & \mathbf{p}_{42} & \mathbf{p}_{43} & \mathbf{p}_{44} \end{bmatrix} \begin{bmatrix} (1-w)^3 \\ 3w(1-w)^2 \\ 3w^2(1-w) \\ w^3 \end{bmatrix}$$



Bicubic Bezier Patch: courtesy of Shu Ye

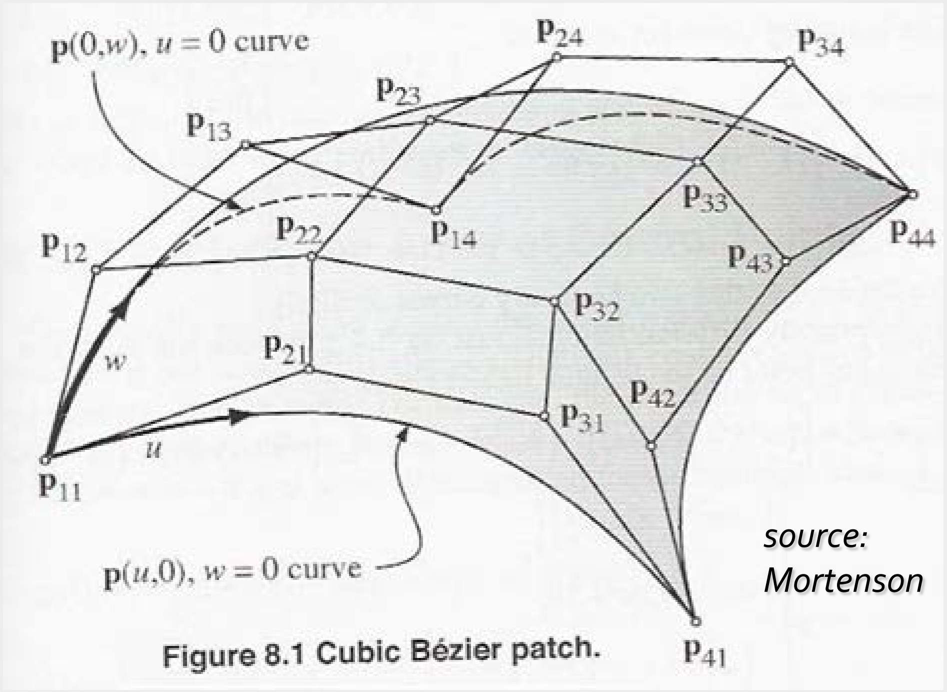


Figure 8.1 Cubic Bézier patch.

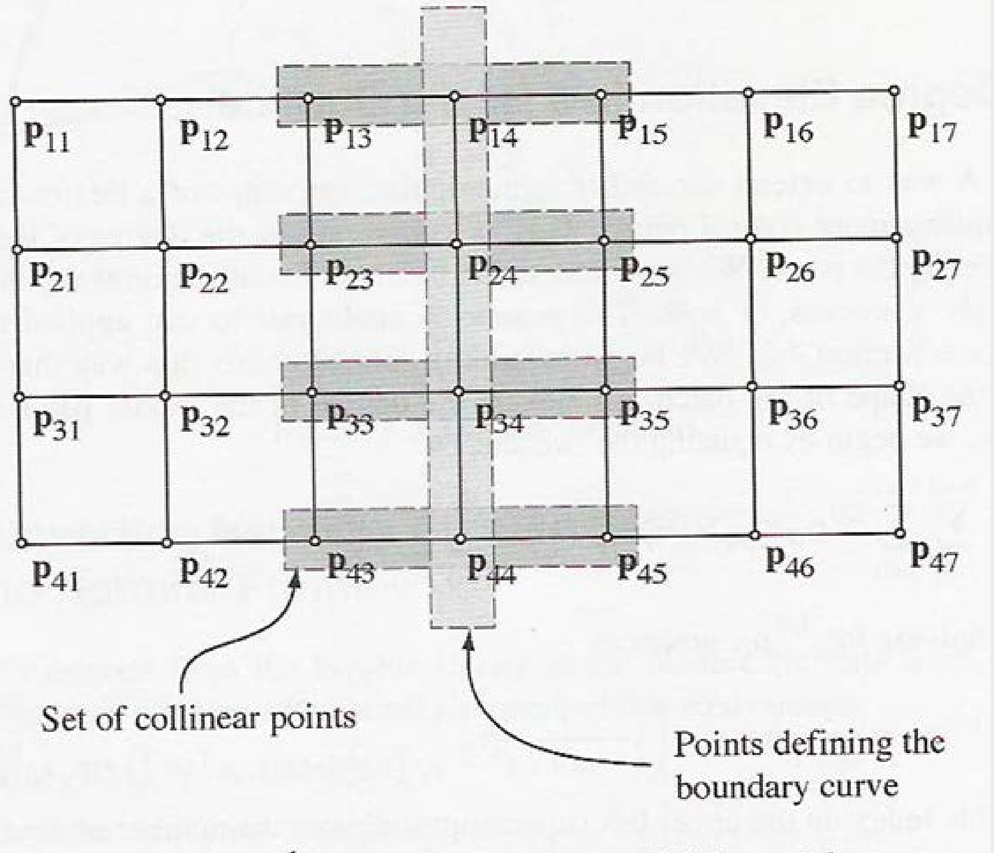
source: Mortenson

Composite Bezier Surface

- Bezier surface patches can provide G^1 continuity at patch boundary curves.
- For common boundary curve defined by control points $\mathbf{p}_{14}$, $\mathbf{p}_{24}$, $\mathbf{p}_{34}$, $\mathbf{p}_{44}$, need collinearity of:

$$\{\mathbf{p}_{i,3}, \mathbf{p}_{i,4}, \mathbf{p}_{i,5}\}, \quad i \in [1:4]$$

- Two adjacent patches are C^r across their common boundary if all rows of control net vertices are interpretable as polygons of C^r piecewise Bezier curves.



$\mathbf{p}_{11}$ $\mathbf{p}_{12}$ $\mathbf{p}_{13}$ $\mathbf{p}_{14}$ $\mathbf{p}_{15}$ $\mathbf{p}_{16}$ $\mathbf{p}_{17}$
$\mathbf{p}_{21}$ $\mathbf{p}_{22}$ $\mathbf{p}_{23}$ $\mathbf{p}_{24}$ $\mathbf{p}_{25}$ $\mathbf{p}_{26}$ $\mathbf{p}_{27}$
$\mathbf{p}_{31}$ $\mathbf{p}_{32}$ $\mathbf{p}_{33}$ $\mathbf{p}_{34}$ $\mathbf{p}_{35}$ $\mathbf{p}_{36}$ $\mathbf{p}_{37}$
$\mathbf{p}_{41}$ $\mathbf{p}_{42}$ $\mathbf{p}_{43}$ $\mathbf{p}_{44}$ $\mathbf{p}_{45}$ $\mathbf{p}_{46}$ $\mathbf{p}_{47}$
Set of collinear points
Points defining the boundary curve

Figure 8.4 G^1 continuity across two Bézier patches.

source: Mortenson, Farin

$$M_{\text{Bspline}} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 3 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

Features: Local control and variation of degree

$$P(u,v) = [N_{0,k}(u)\ \ N_{1,k}(u)\ \cdots\ N_{n,k}(u)] \begin{bmatrix} P_{00} & P_{01} & \cdots & P_{0m} \\ P_{10} & P_{11} & \cdots & P_{1m} \\ \vdots & \vdots & & \vdots \\ P_{n0} & P_{n1} & \cdots & P_{nm} \end{bmatrix} \begin{bmatrix} N_{0,l}(v) \\ N_{1,l}(v) \\ \vdots \\ N_{m,l}(v) \end{bmatrix}$$

B-Spline Surface Patch

$$U^T = [u^3 \quad u^2 \quad u \quad 1]$$

$$p(u,w) = U^T[M_B][P][M_B]^T W$$

Geometric form (non-uniform, non-rational case), where K controls degree (K -1) of basis functions for parameter u and L controls degree (L -1) of basis functions for parameter w:

$$\mathbf{p}(u,w) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{p}_{ij} N_{i,K}(u) N_{j,L}(w)$$

$$N_{i,k}(u) = \frac{(u-t_i)N_{i,k-1}(u)}{t_{i+k-1}-t_i} + \frac{(t_{i+k}-u)N_{i+1,k-1}(u)}{t_{i+k}-t_{i+1}}$$

Quadratic case (left), Cubic case (right)

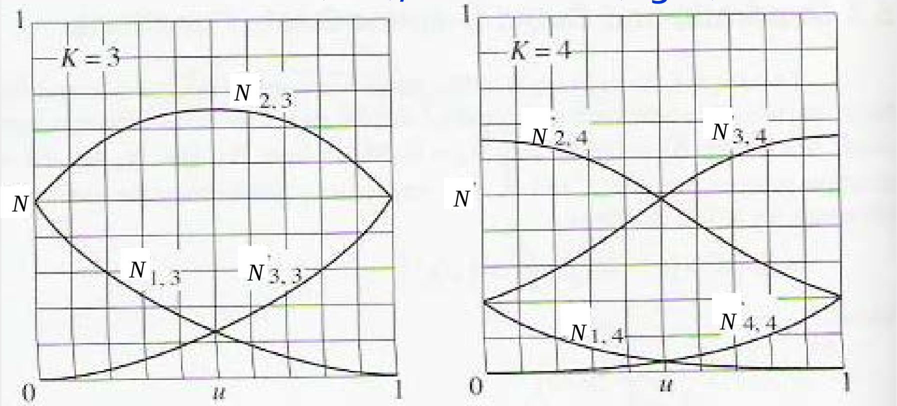


Figure 5.7 Uniform B-spline basis functions on the unit interval for $K = 3$ and $K = 4$.

$$\sum_{i=0}^{m} N_{i,K}(u) = 1$$

2 sets of knot vectors are required: 1 for each parameter.

Cubic B-splines can provide C^2 continuity at surface patch boundary curves.

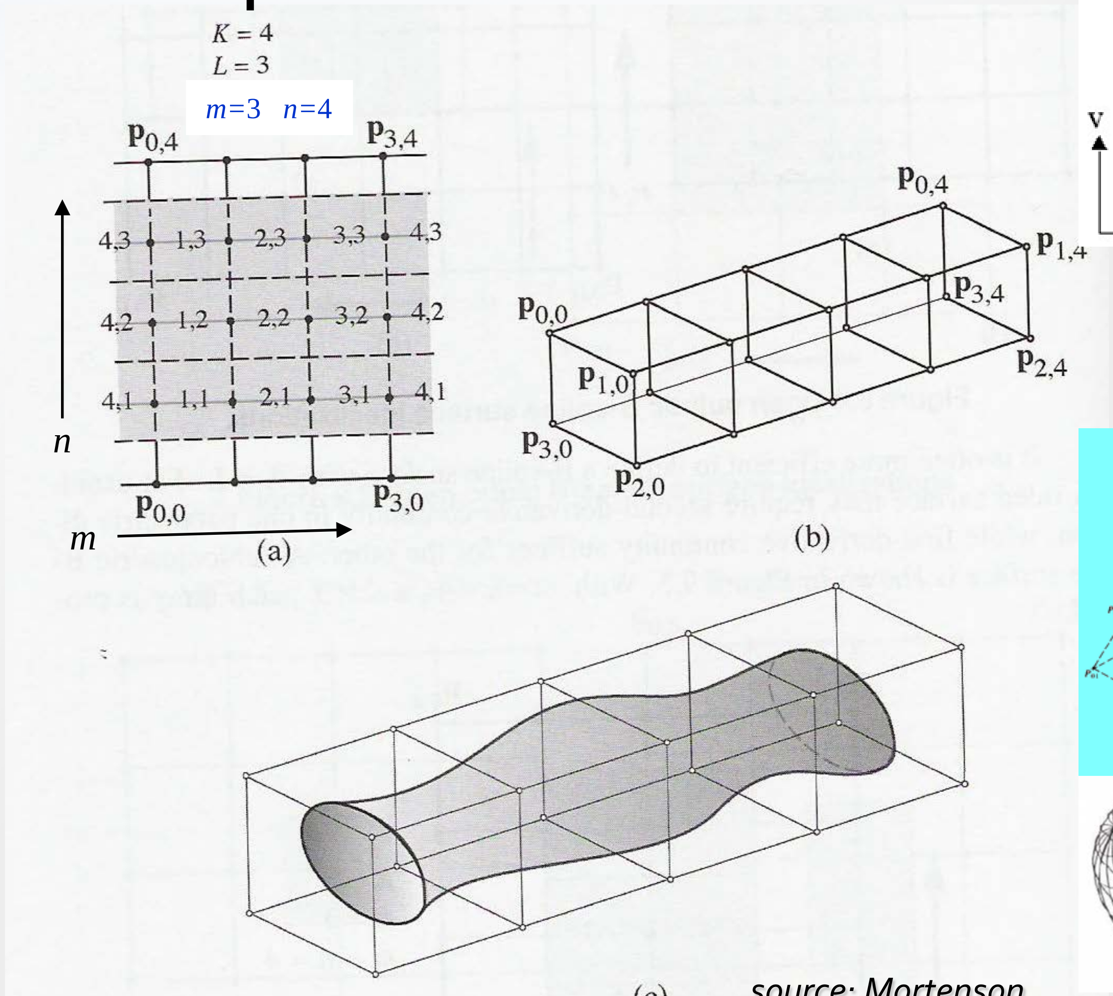
Convex combination, so B-spline surface points all lie within convex hull of control polyhedron.

Rational form (NURBS) is invariant under perspective transformation, where h_{ij} are projective space coordinates (weights).

$$\mathbf{p}(u,w) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} h_{ij}\mathbf{p}_{ij} N_{i,K}(u) N_{j,L}(w)}{\sum_{i=0}^{m}\sum_{j=0}^{n} h_{ij} N_{i,K}(u) N_{j,L}(w)}$$

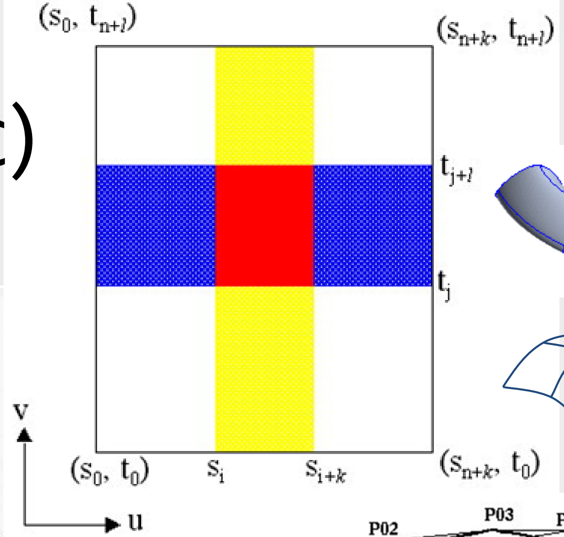
source: Mortenson

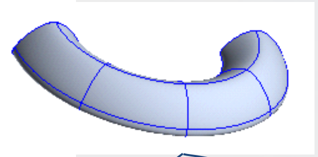
Partially Closed (Periodic) B-Spline Surface

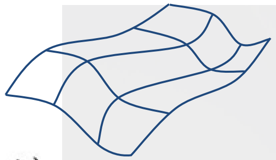


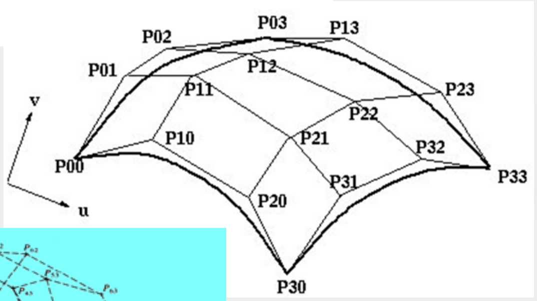
source: Mortenson

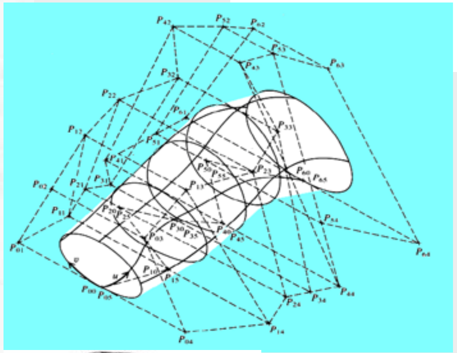
Figure 9.6 Partially closed B-spline surface.

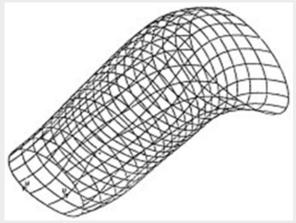


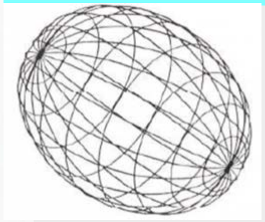


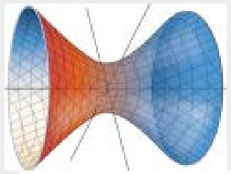


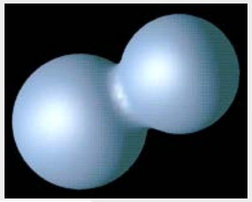


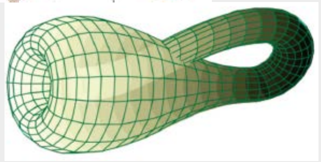






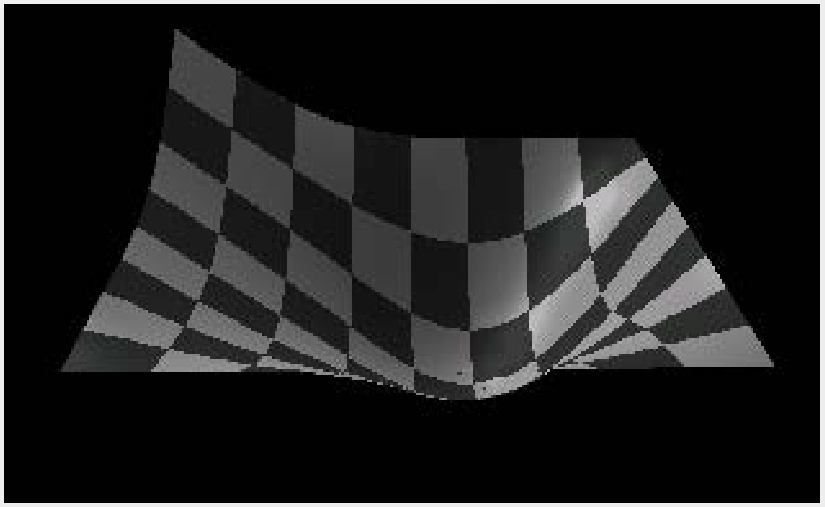
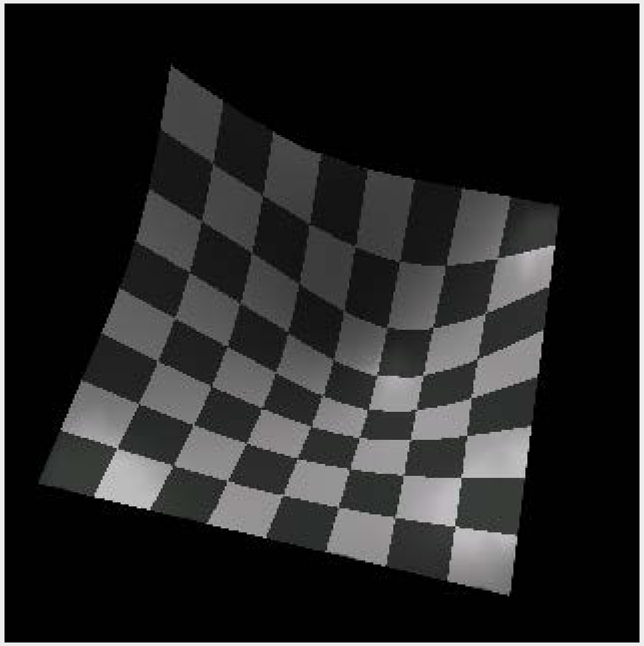






Untrimmed NURBS Surface Patch

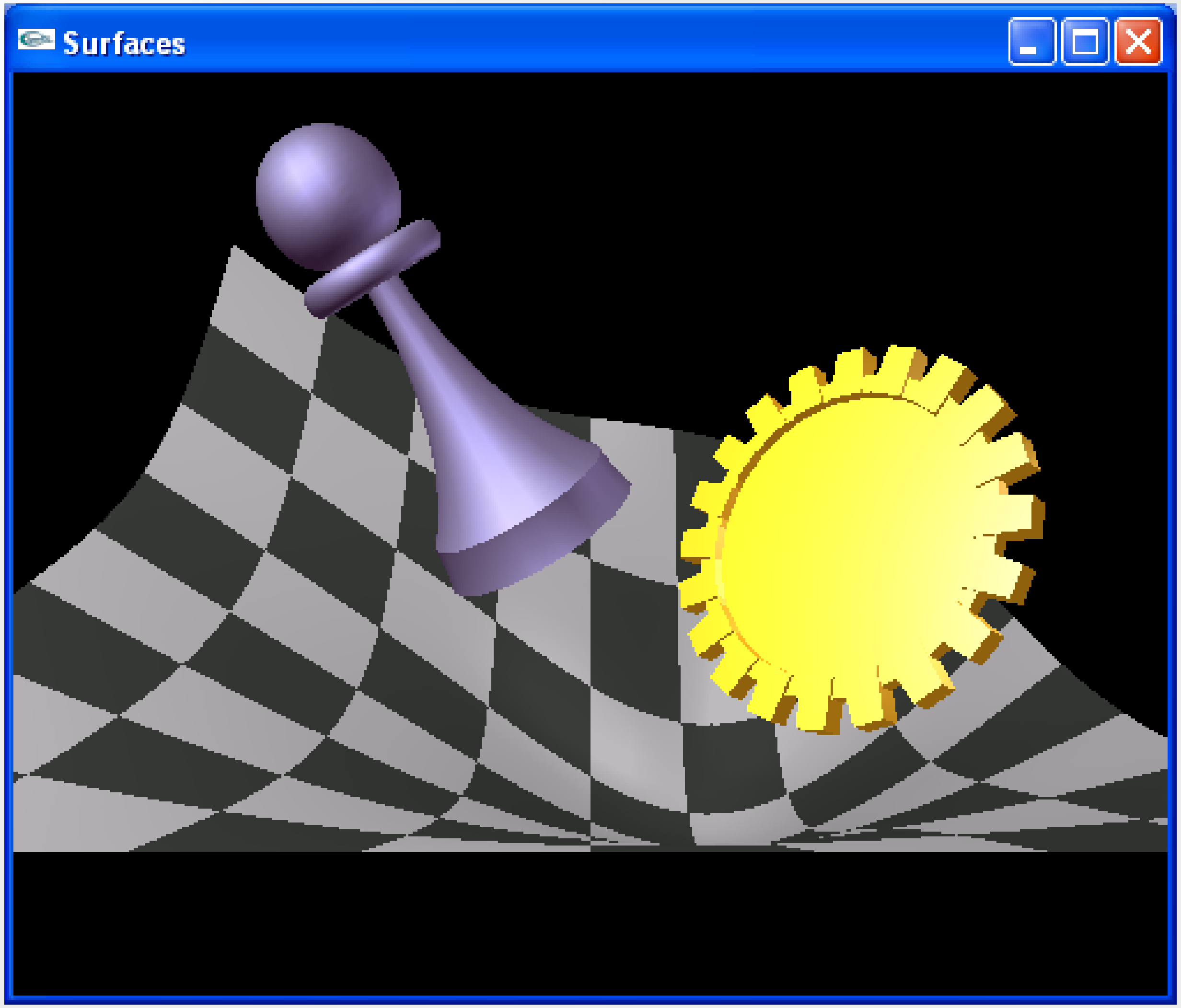
- Jason's NURBS surface
  - Checkerboard texture map
  - 8 control points in each dimension
  - 12 uniformly spaced knots for each parameter
  - u stride = 8*3, v stride = 3



2 different views of Jason's patch

Untrimmed Surface Patch examples

<u>Consists of</u>:

- NURBS surfaces
    - Checkerboard
    - Gear (body: front and back faces)
- Surfaces of revolution using cubic Bezier curves
    - Pawn
- Extrusion
    - -Gear's teeth and gaps



Trimmed NURBS Surface Patch

Start with a NURBS Patch, as in Red Book:

GLfloat knots[8] = {0.0, 0.0, 0.0, 0.0, 1.0, 10, 1.0, 1.0};

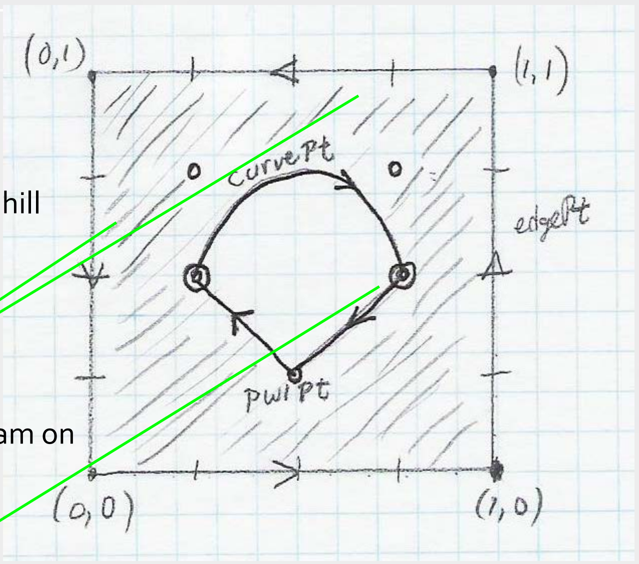
Control points: GFfloat ctlpoints[4][4][3] (4 in each parametric direction) yield symmetric hill ranging from -3.0 to 3.0.

gluNurbsSurface(the Nurb, 8, knots, 8, knots, 4*3, 3, &ctlpoints[0][0][0], 4, 4, GL_MAP2_VERTEX_3);

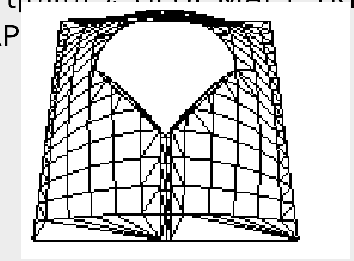
Add trimming curves (interior on left) before gluEndSurface(theNurb), according to diagram on right:

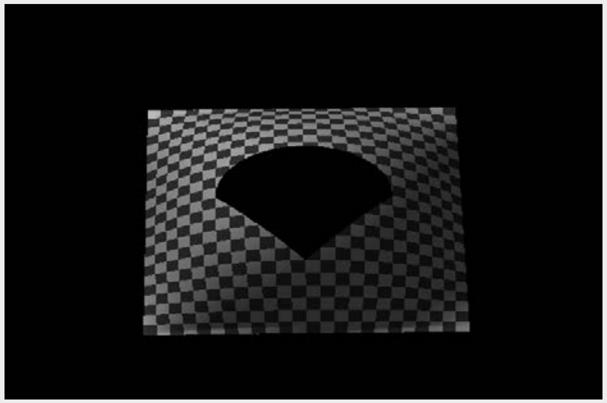
```
  gluBeginTrim(theNurb);
     gluPwlCurve(theNurb, 5, &edgePt[0][0], 2, GLU_MAP1_TRIM_2);
  gluEndTrim(theNurb);

  gluBeginTrim(theNurb);
     gluNurbsCurve(theNurb, 8, curveKnots, 2, &curvePt[0][0], 4, GLU_MAP1_TRIM_2);
     gluPwlCurve(theNurb, 3, &pwlPt[0][0], 2, GLU_MAP
  gluEndTrim(theNurb);
```



Note: From Jason's experience, more than 4 control points in each dimension may be problematic for trimming curves.


Trimmed NURBS Surface: courtesy of Silicon Graphics


checkerboard NURBS courtesy of Jason

Parametric bicubic surfaces

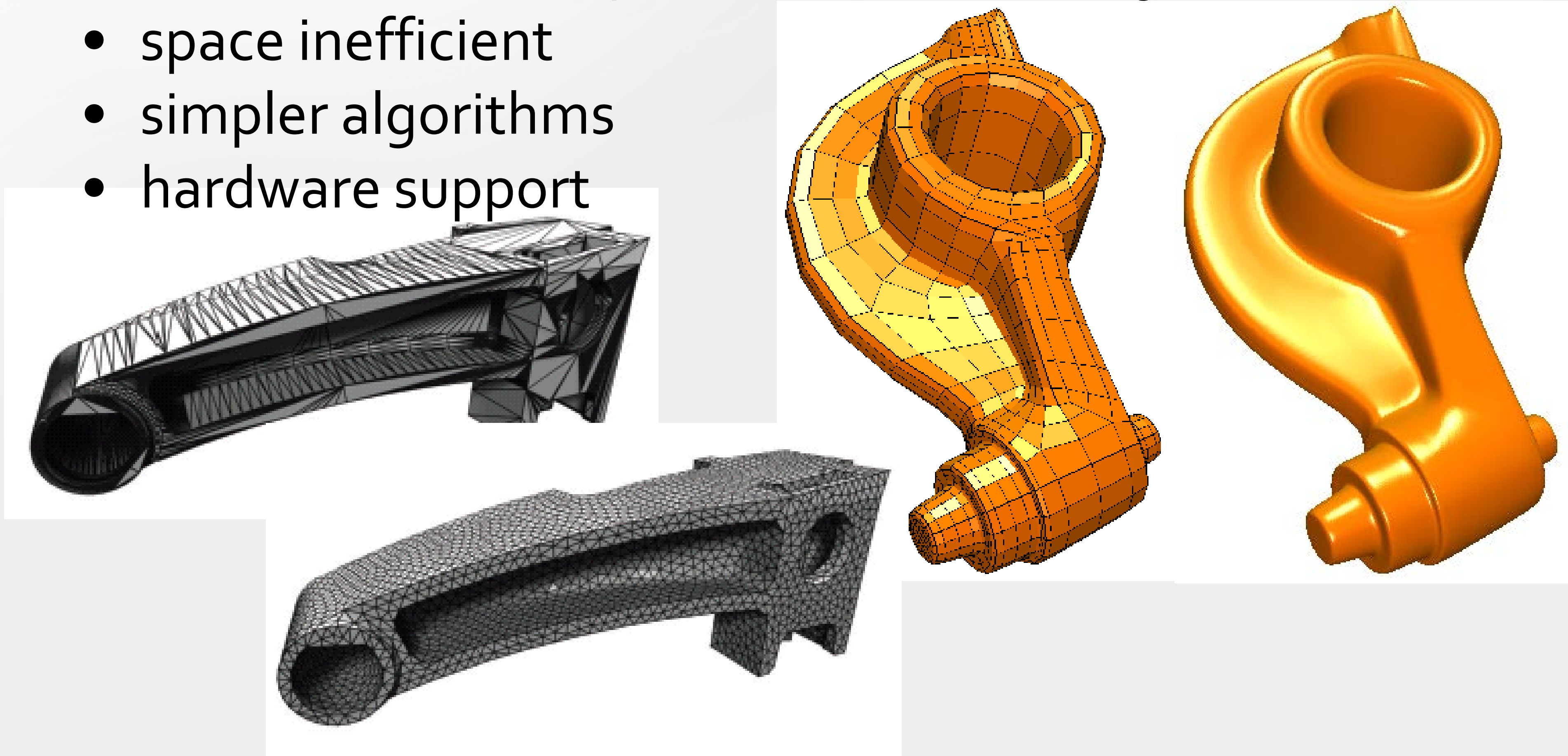
Major kinds of surfaces: Hermit, Bezier, B-spline

Displaying bicubic surfaces:

- brute-force **iterative evaluation is very expensive** (the surface is evaluated 20,000 times if step in parameters is 0.01)
- **forward-difference methods are better**, but still expensive
- fastest is **adaptive subdivision**, but it might create cracks

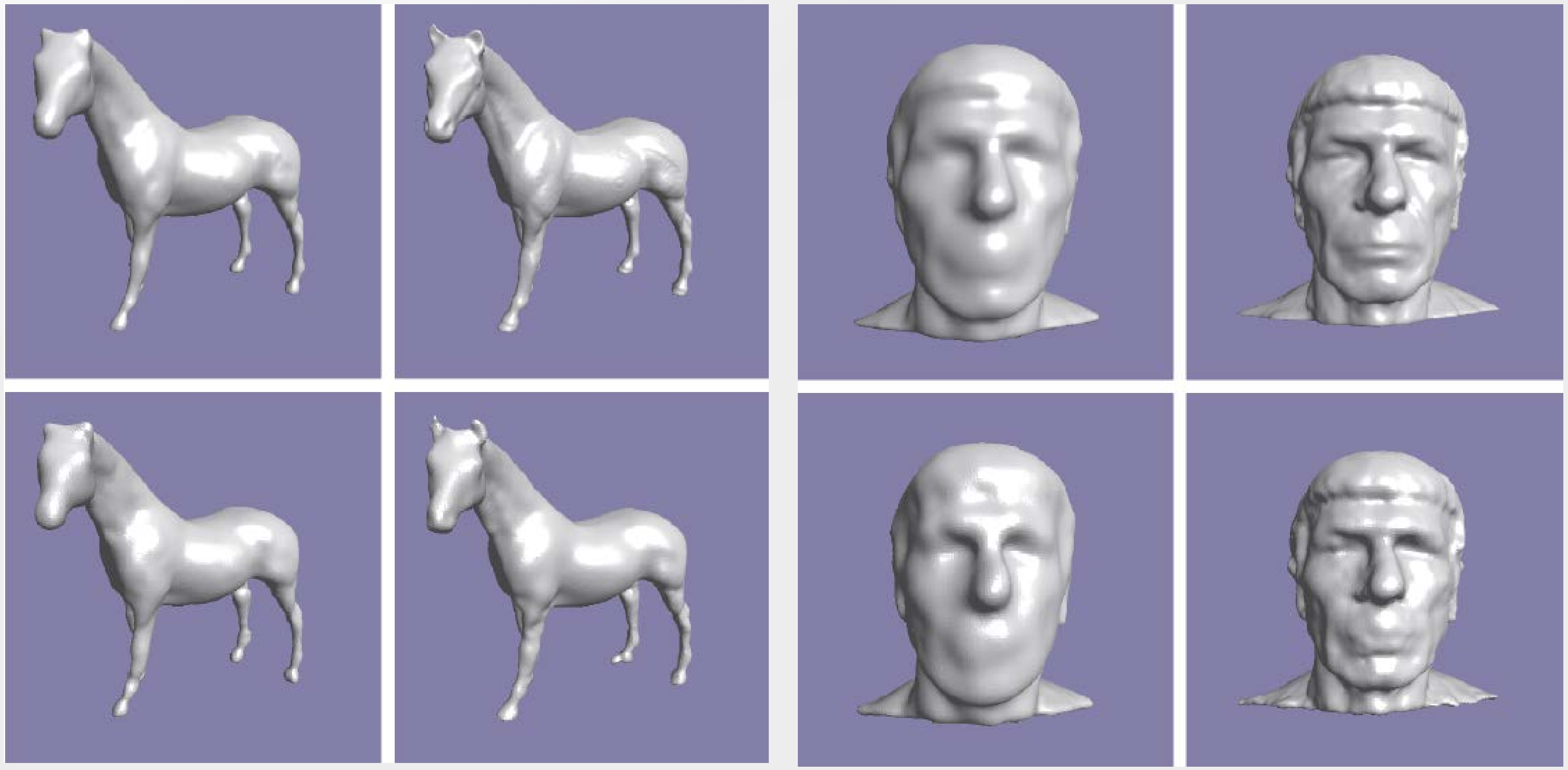
Polygon meshes

- well suited for **representing flat-faced objects**
- seldom satisfactory for curved-faced objects
- space inefficient
- simpler algorithms
- hardware support

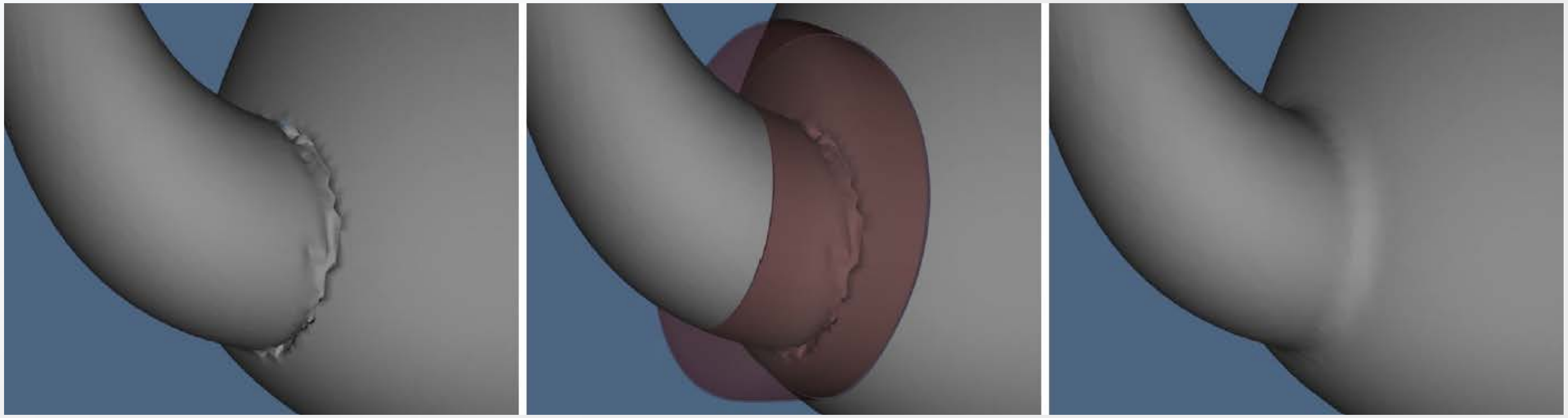


Surfaces from point cloud

Top row: low-resolution volumetric representations of the horse (104 x 70 x 119, 195 x 120 x 228) and the head (69 x 69 x 76, 118 x 120 x 135).
Bottom row: respective implicit surfaces generated by algorithm.

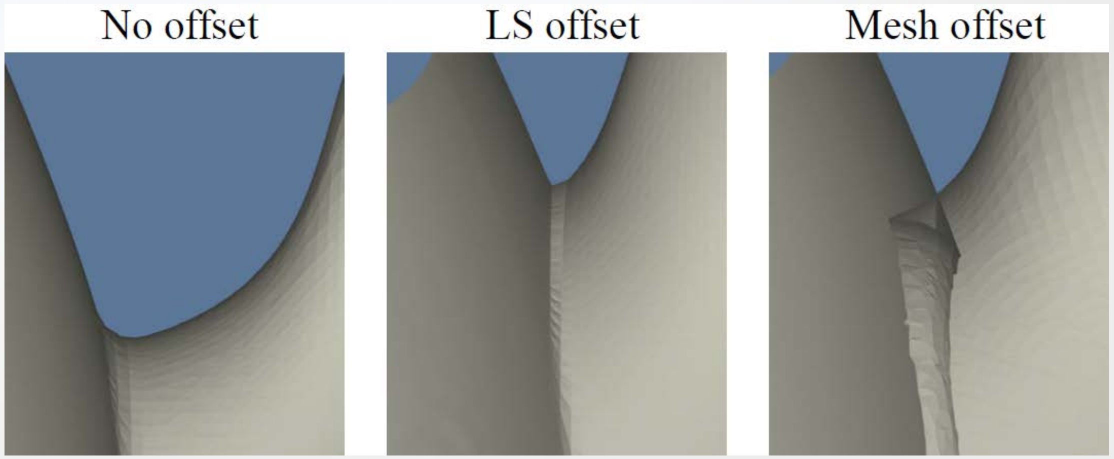


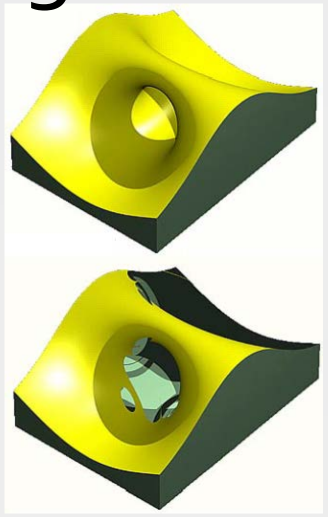
Localized LS Smoothing/Sharpening



(left) Scan conversion errors near the teapot spout.
(middle) Placing a (red) superellipsoid around the errors.
(right) The errors are smoothed away in 15 seconds. The surface is constrained to only move outwards.

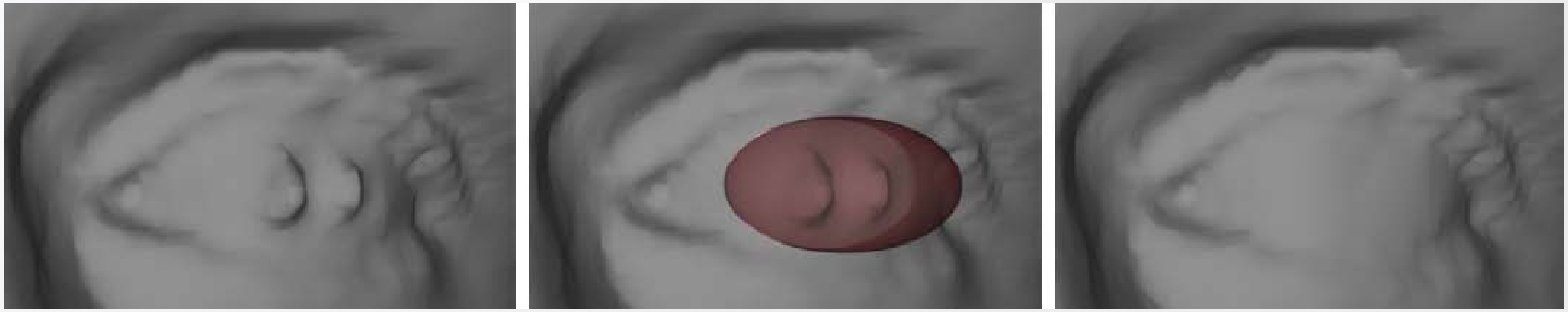
Localized LS Smoothing/Sharpening





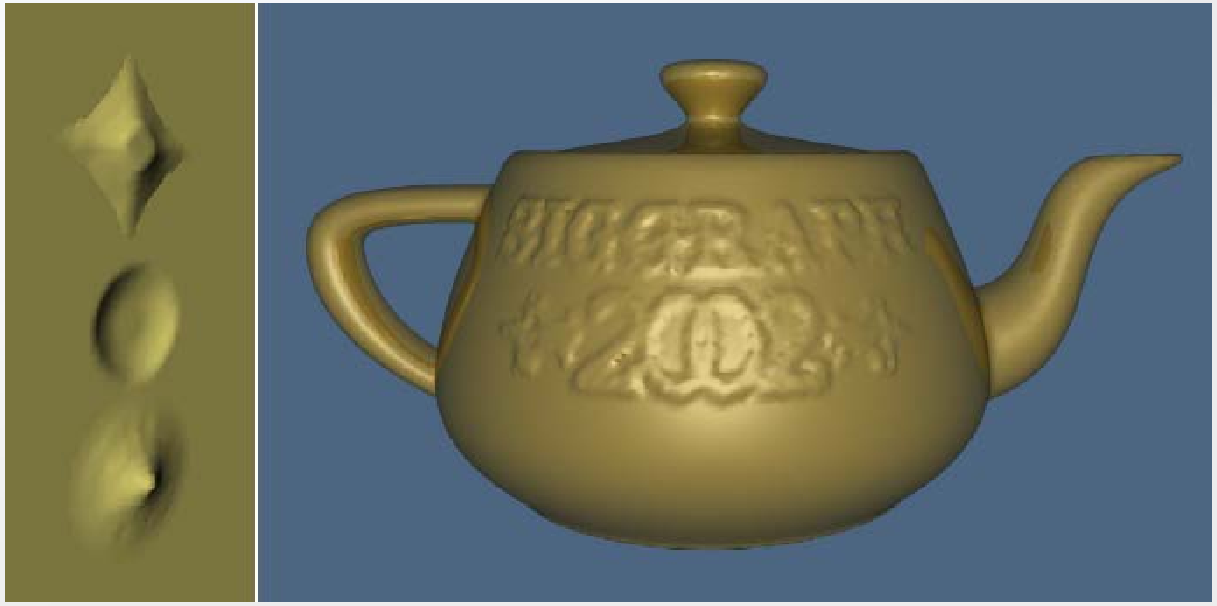
(left) A cross-section of the teapot model near the spout.
(middle) No self-intersection occurs, by construction, when performing a level set (LS) offset, i.e. dilation, of the surface.
(right) Self-intersections may occur when offsetting a mesh model.

Regionally constrained smoothing

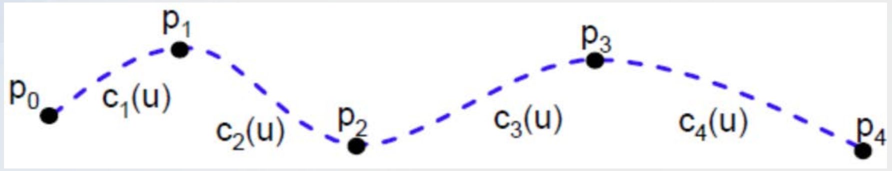


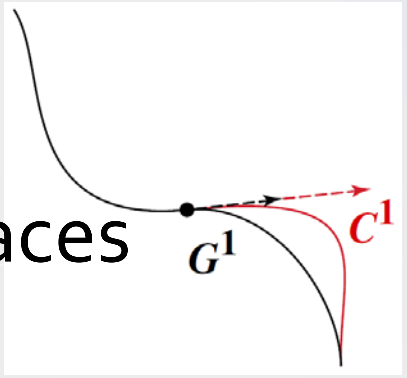
Left: Laser scan reconstruction with unwanted, pointed artifacts in the eye.
Middle: Defining the region to be smoothed with a (red) superellipsoid.
Right: Smoothing the surface within the superellipsoid. The surface is constrained to only move inwards.

Point-Set Attraction and Embossing



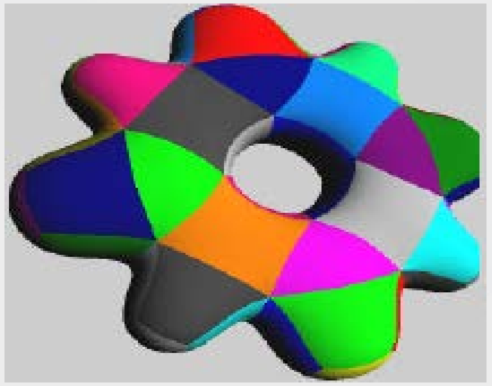
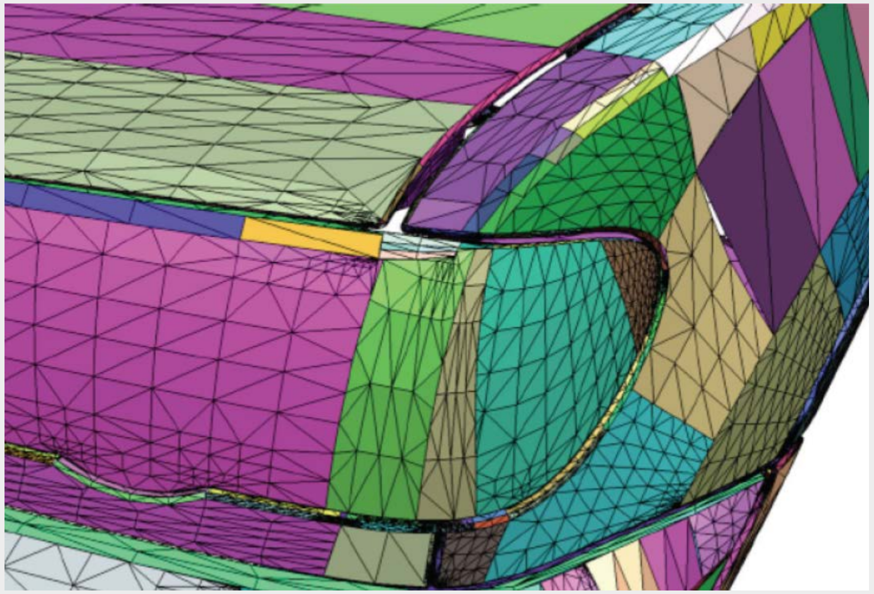
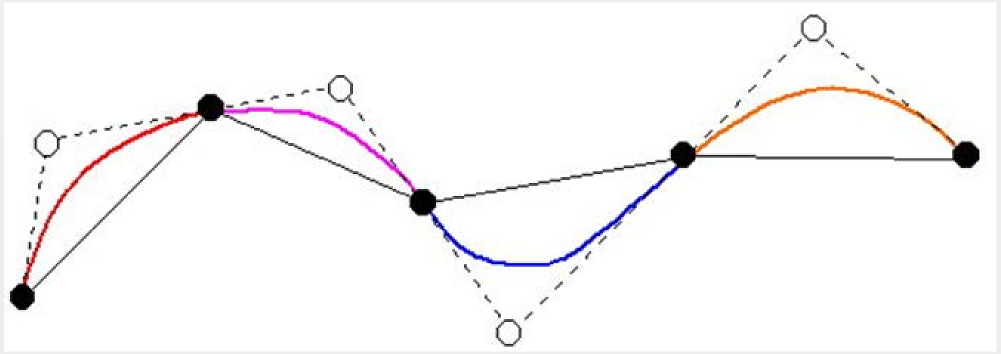
Left: Three types of single point attractions/repulsions using different ROI primitives and values.
Right: Utah teapot embossed with 7862 points sampling the ”SIGGRAPH 2002” logo.





Piecewise cubic curves and bicubic surfaces

- permit multiple values for a single x or y
- represent infinite slopes
- easier to manipulate interactively
- can either interpolate or approximate
- space efficient
- more complex algorithms
- little hardware support this

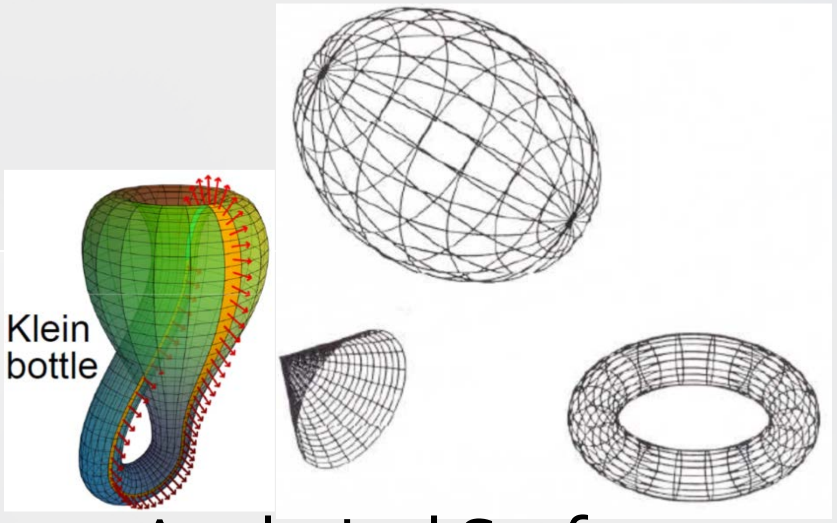




Sculptured Surface

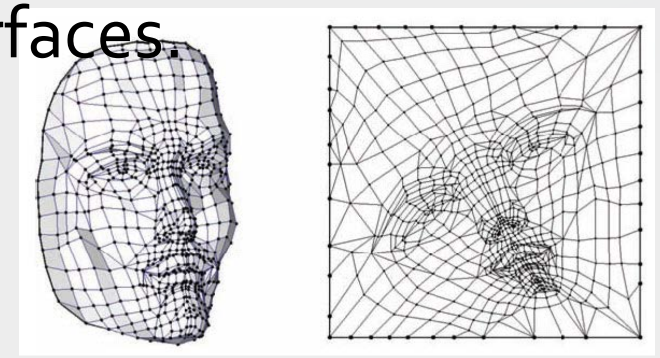
- General surface
- Composed of connected surface patches

Surface Modeling

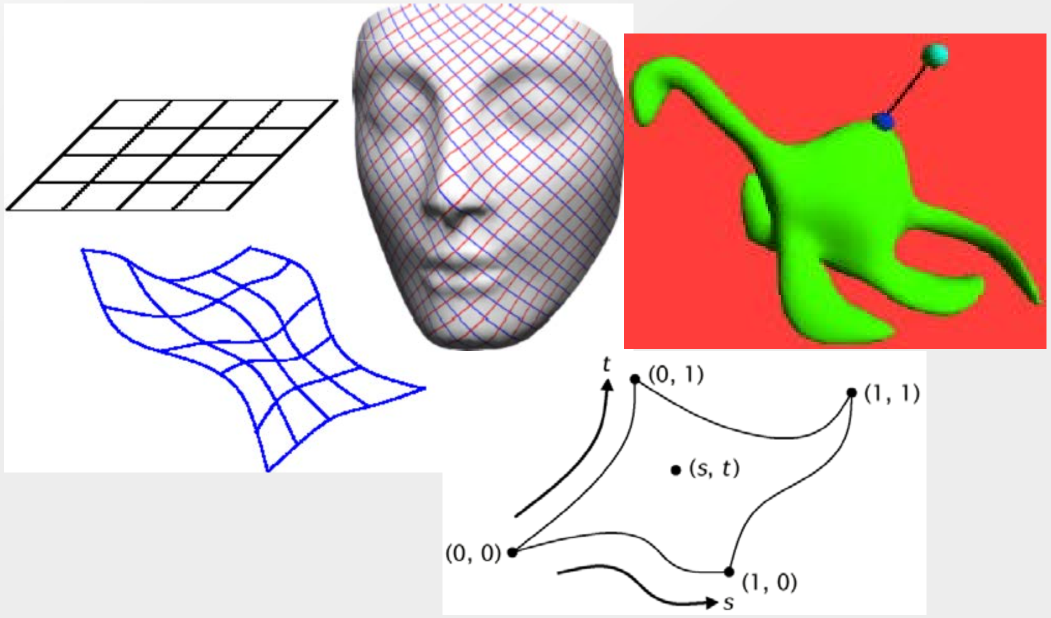
Surface models define only the geometry, no topology.
Shading is possible.
Free-form, Curved, Sculptured Surfaces.

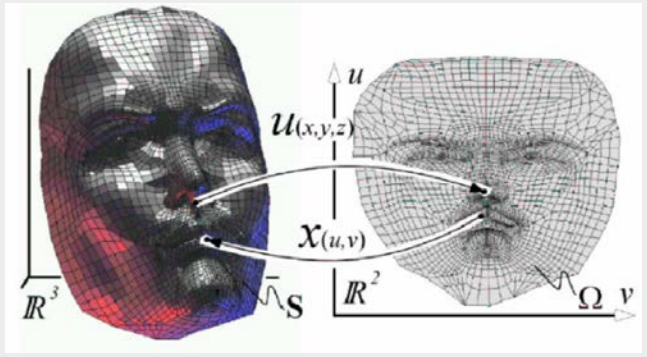


Analytical Surface



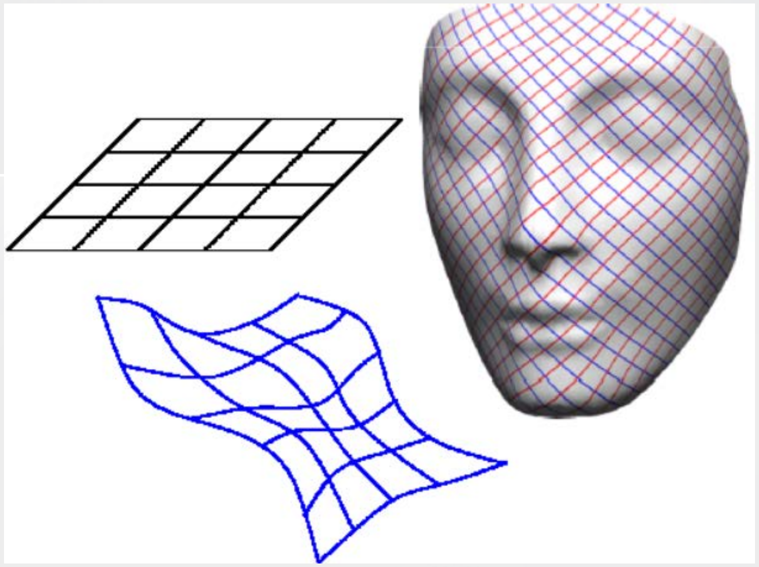
Parameterization



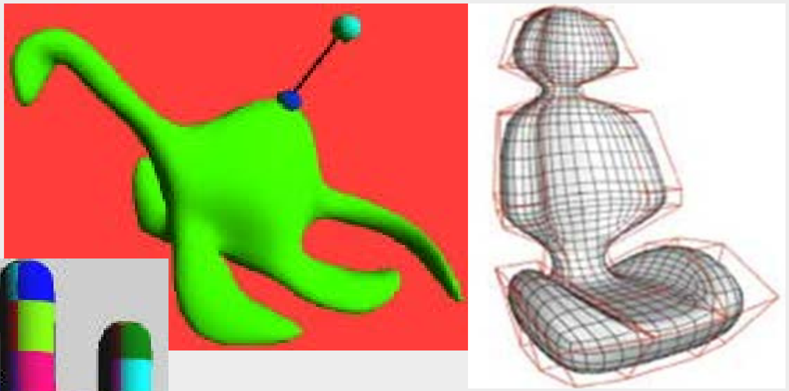


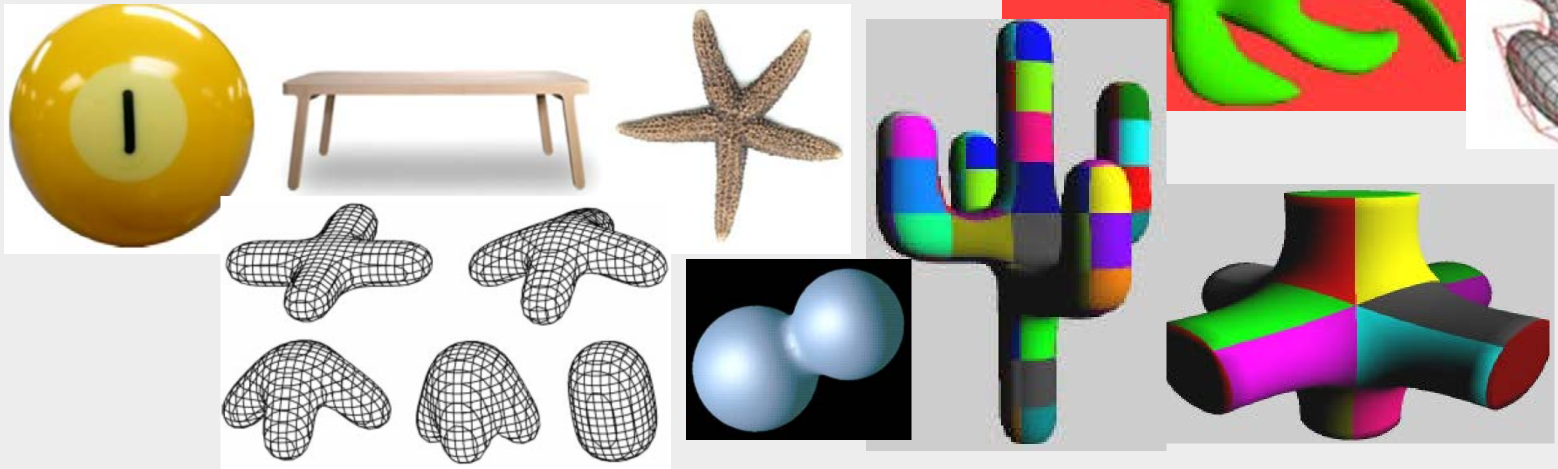
Homeomorphic surfaces

These surfaces are topologically-equivalent (homeomorphic) to a square.



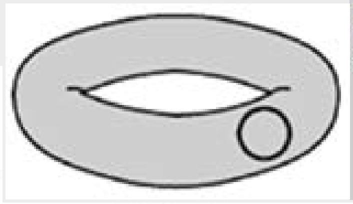
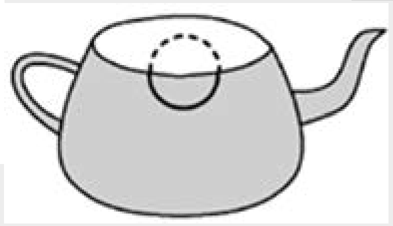
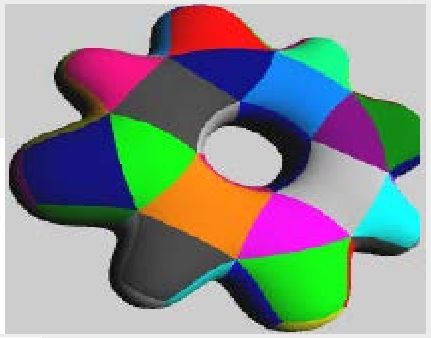
These surfaces are topologically-equivalent (homeomorphic) to a sphere.



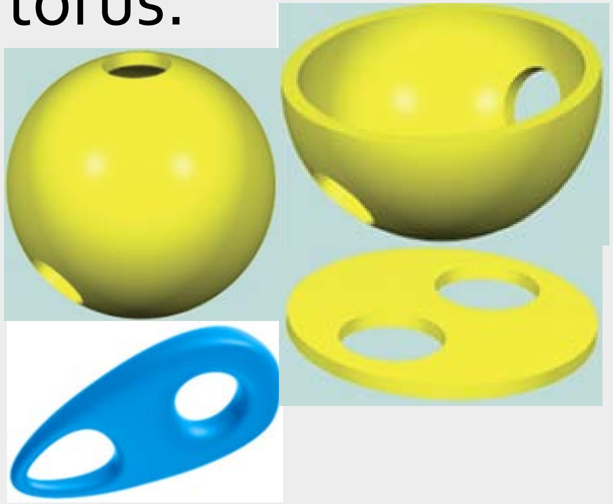
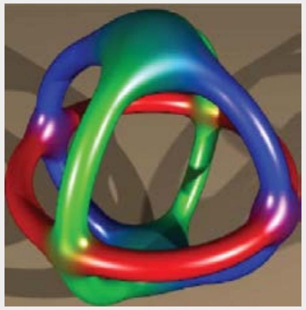


Homeomorphic surfaces

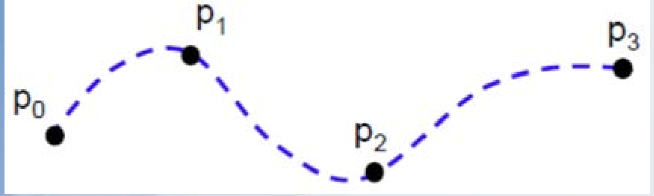
These surfaces are topologically-equivalent (homeomorphic) to a torus.

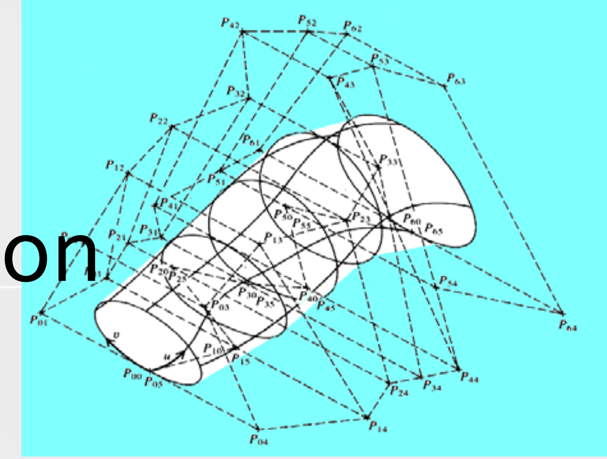




Homeomorphic to a double torus.

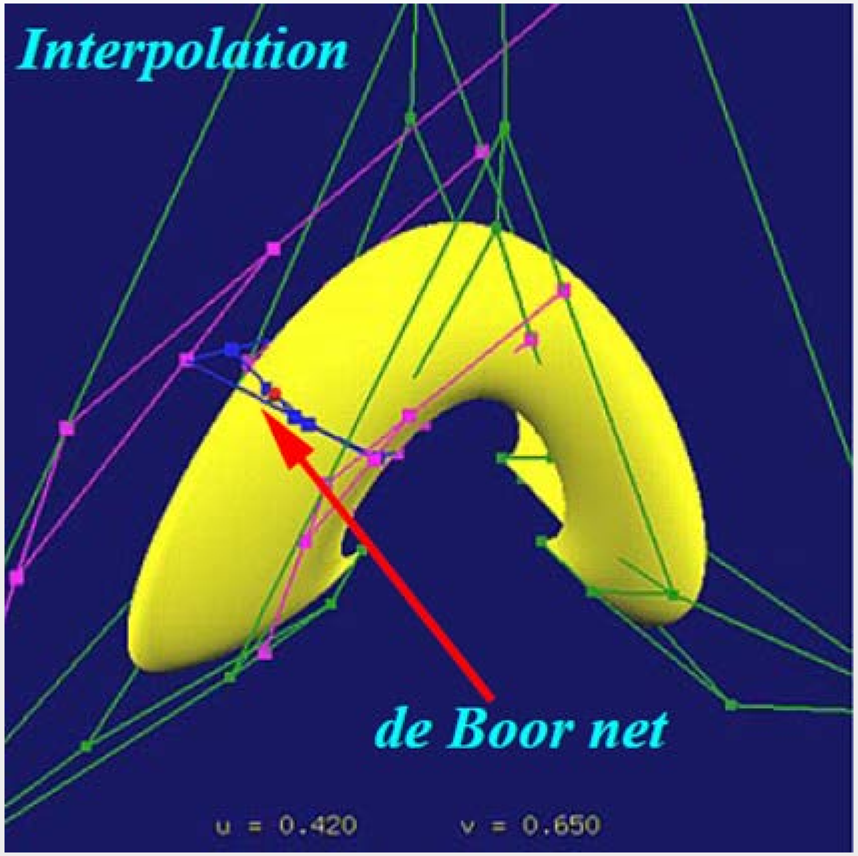
???

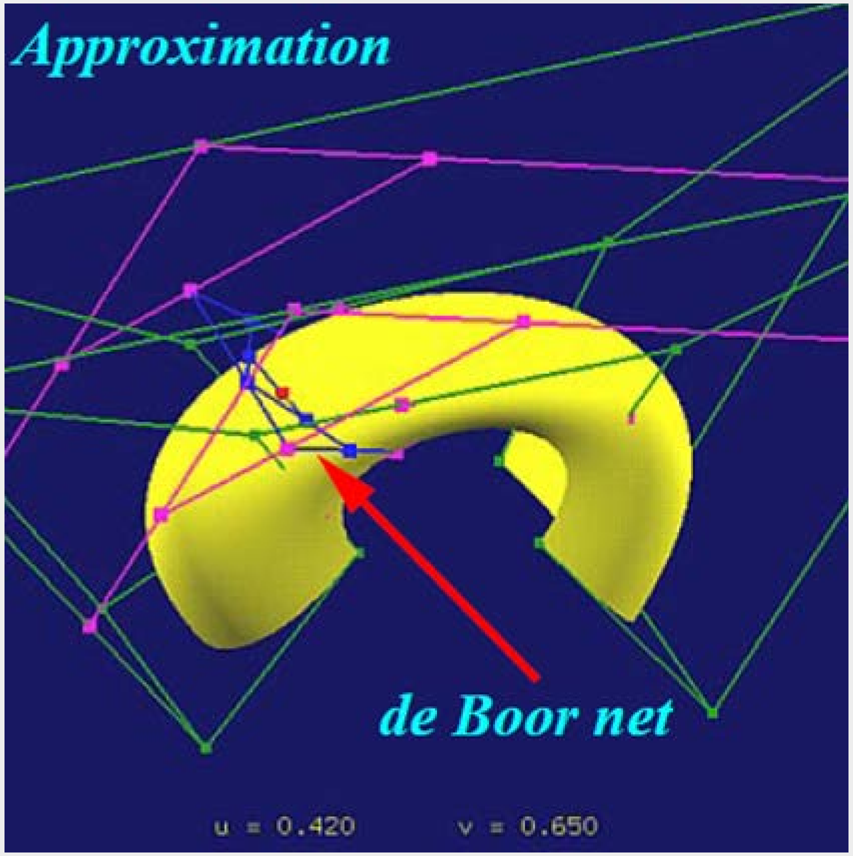




Interpolation and Approximation

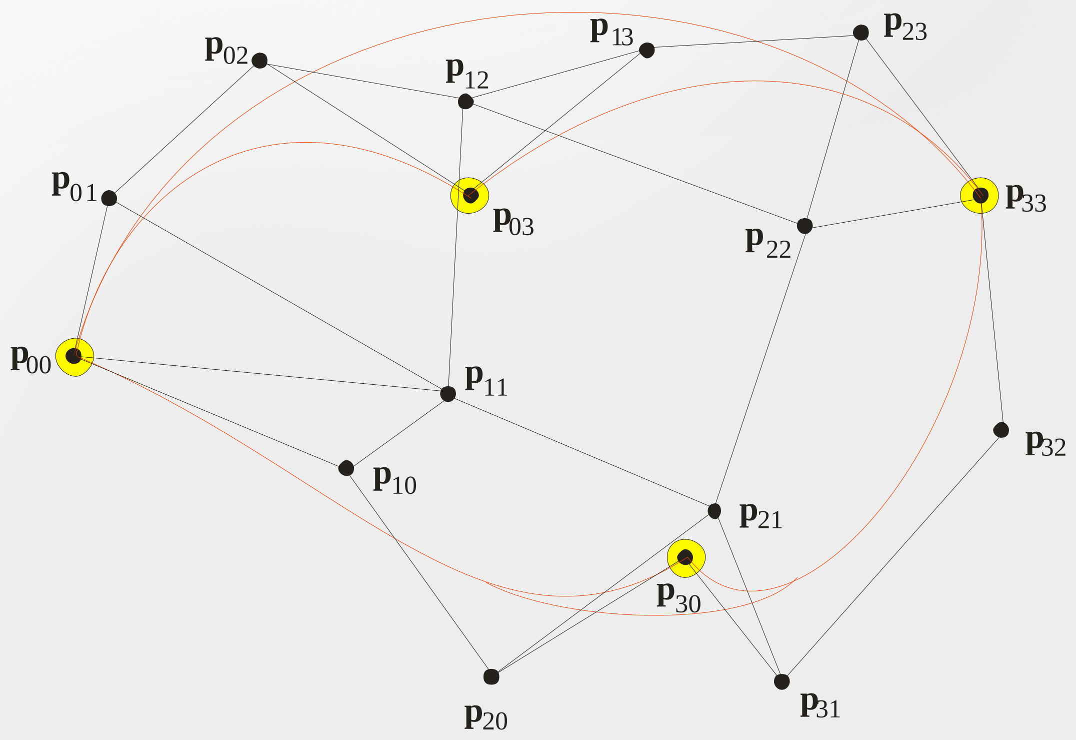
Surface fitting methods

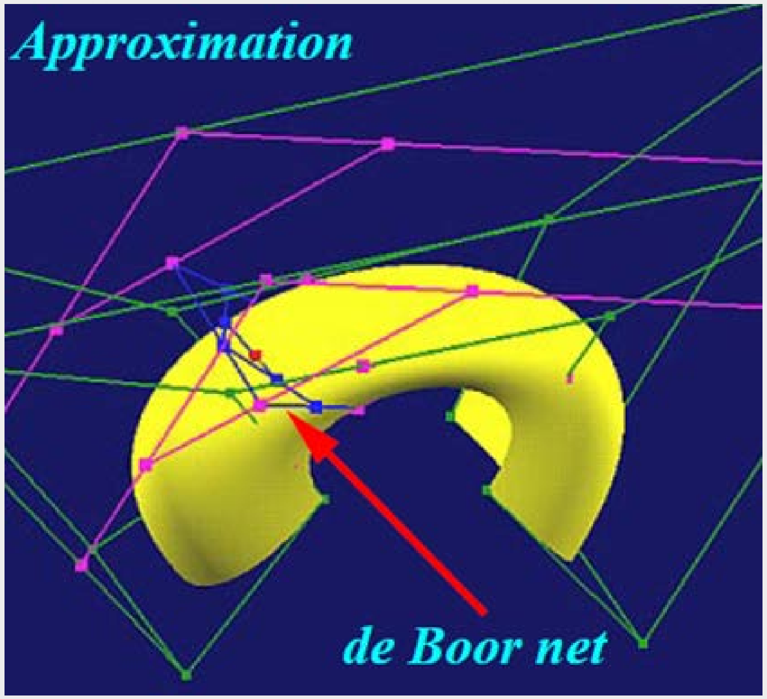




Surface modeling, approximation

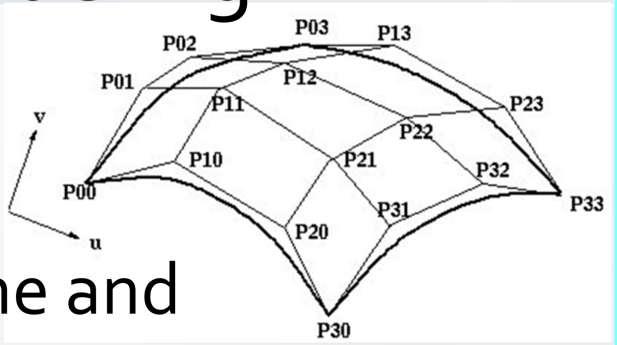
Only a limited number of control points lie on the Bézier surface, (e.g., (u,w)=[(0,0), (0,1), (1,0) and (1,1)]).

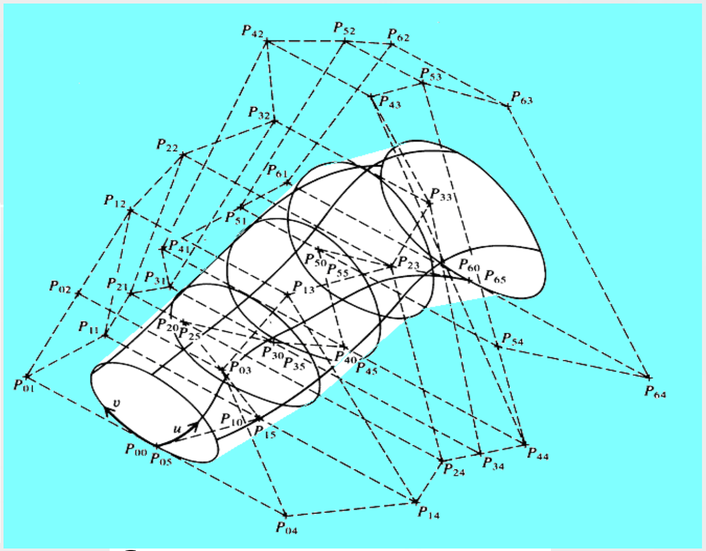




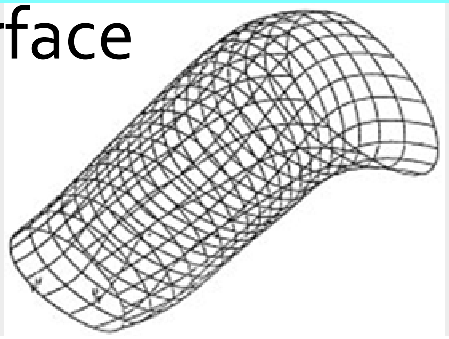
Surface Modeling

Control points





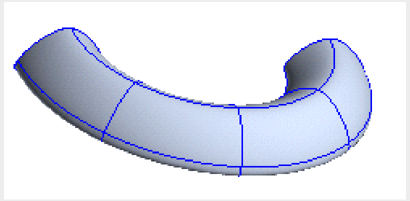
Bezier, B–spline and
NURBS surface is a tensor product surface
and is the product of two curves.

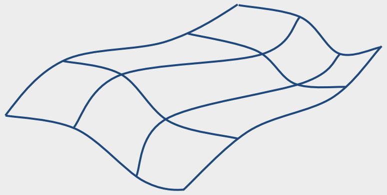


Surfaces are defined by a grid and
have two sets of parameters,
two sets of knots and so on.

$$\mathbf{S}(\mathrm{u}, \mathrm{v}) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u) B_{j,m}(v) \mathbf{P}_{\mathrm{ij}}$$

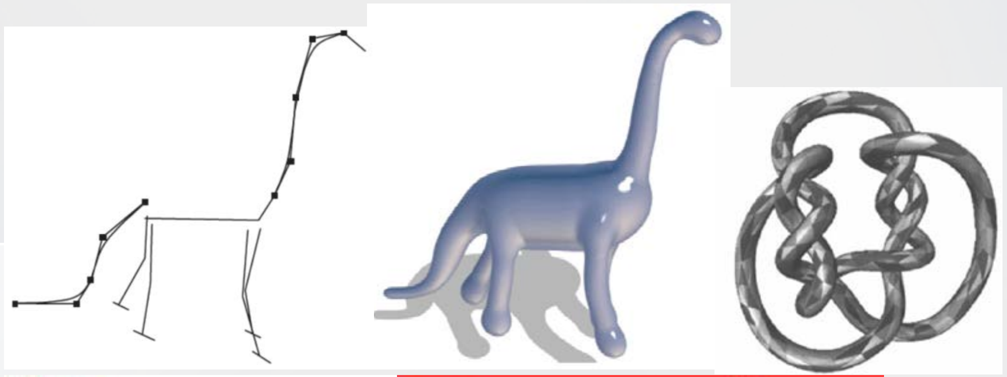
$$\mathbf{S}(\mathrm{u}, \mathrm{v}) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,k}(u) N_{j,l}(v) \mathbf{P}_{\mathrm{ij}}$$

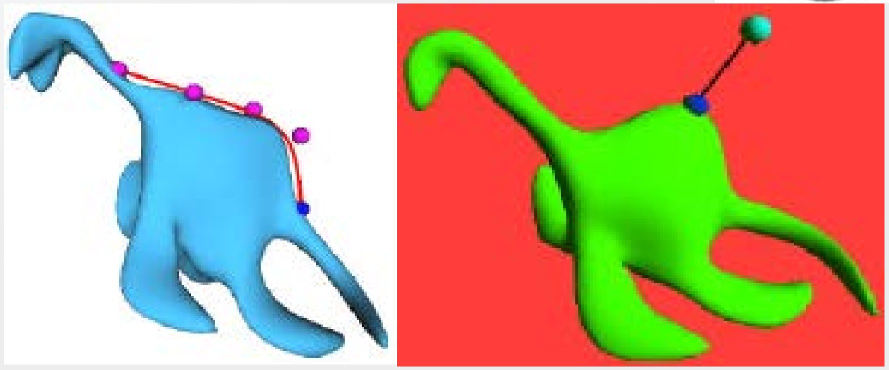




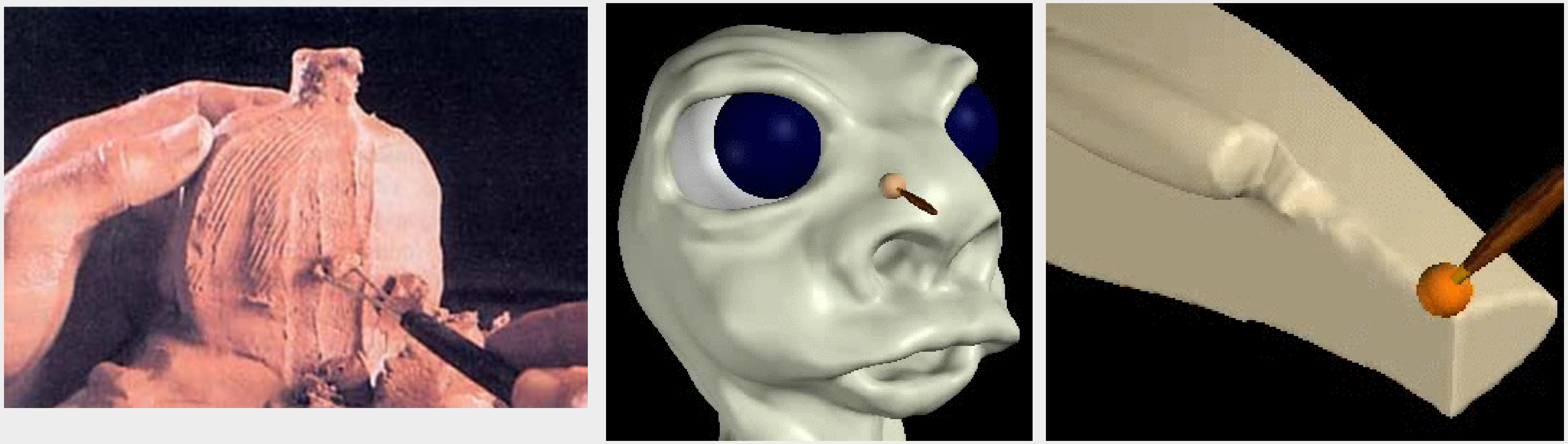
Surface Modeling

Curve-based design and
Surface deformation



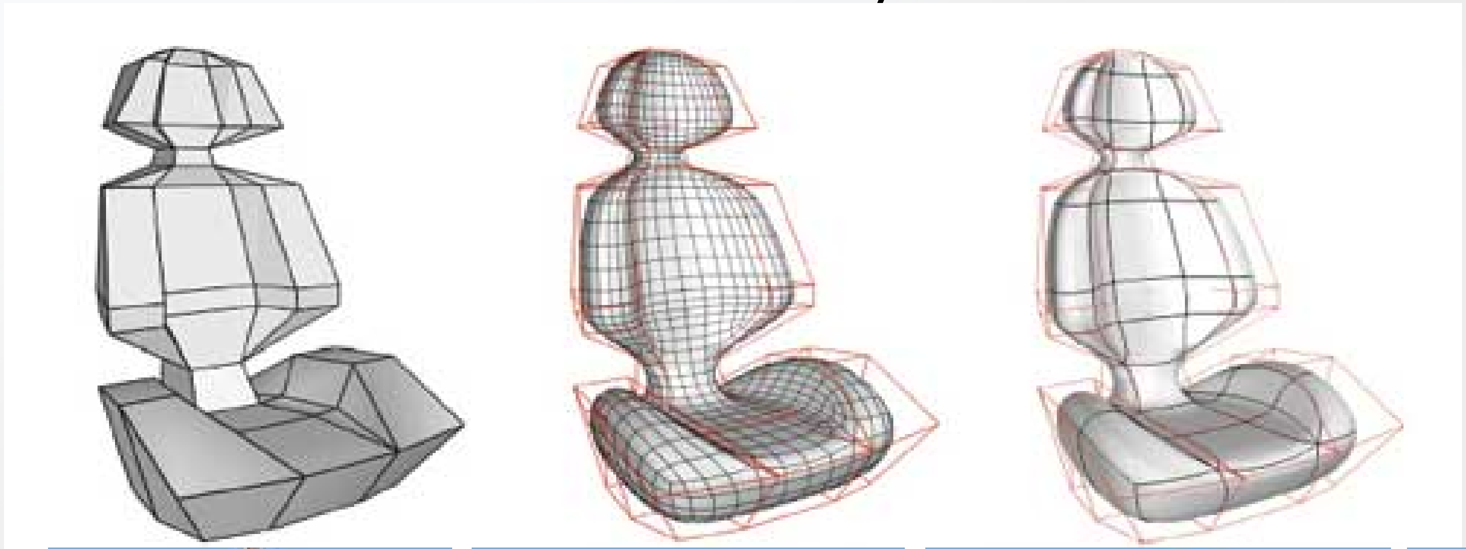


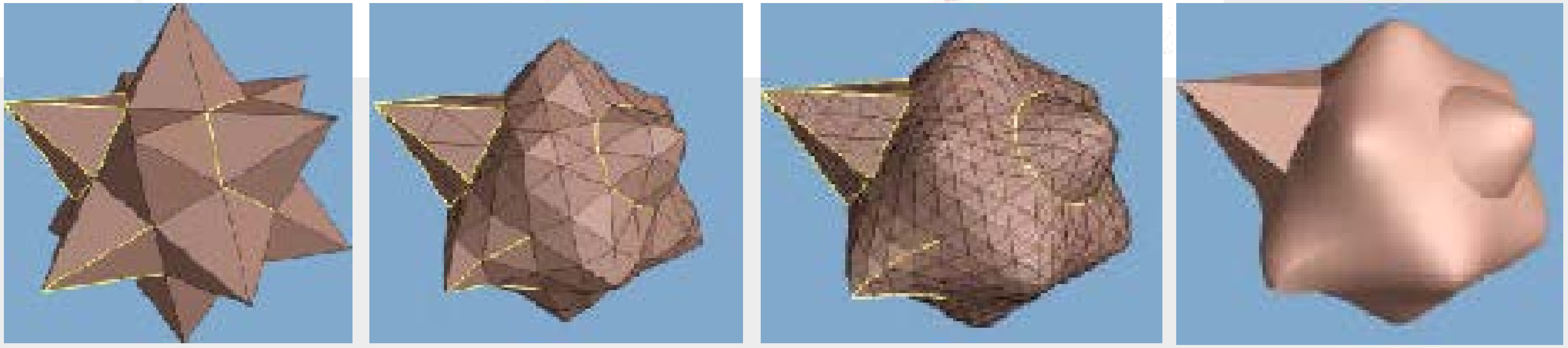
Freeform Phantom Arm
Designing by sculpturing



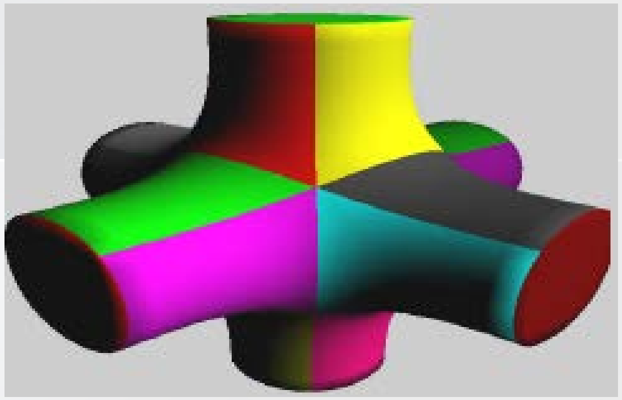
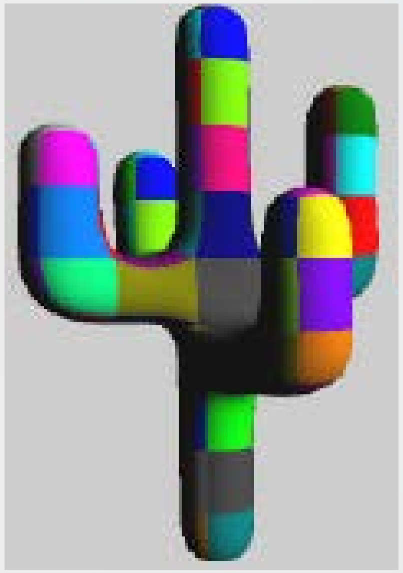
Surface Modeling

Surface Reconstruction by Recursive Subdivision

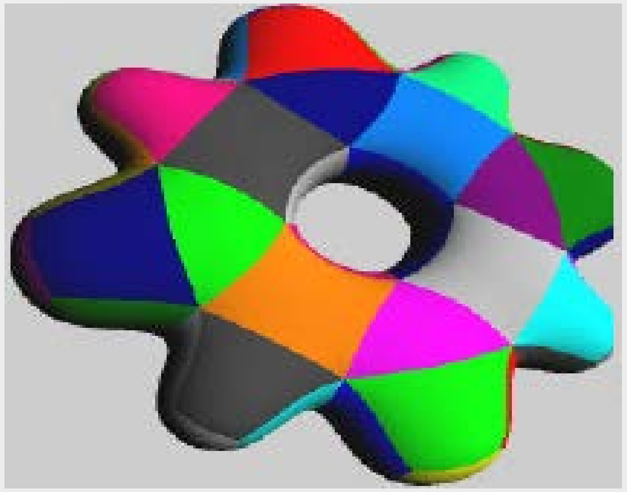
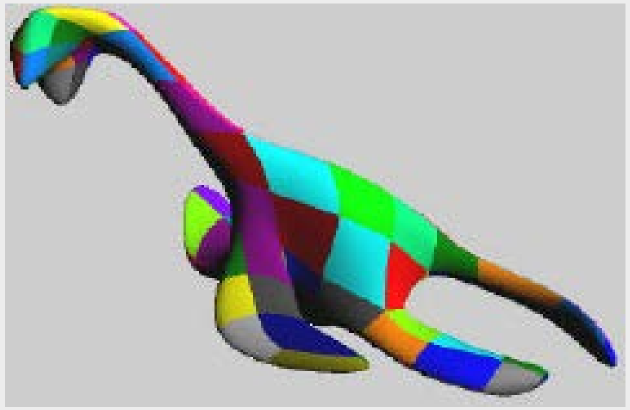
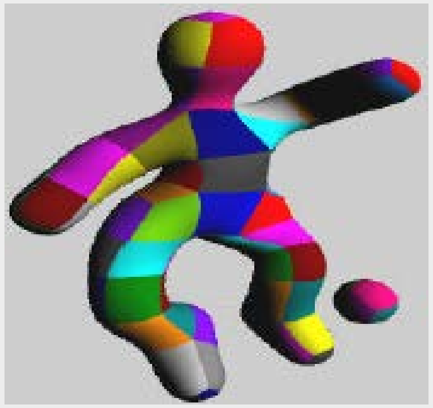




Surface Modeling

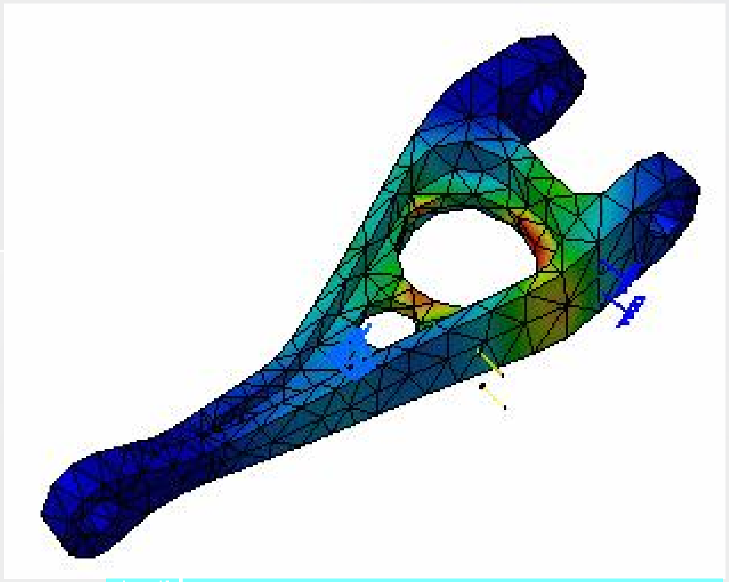



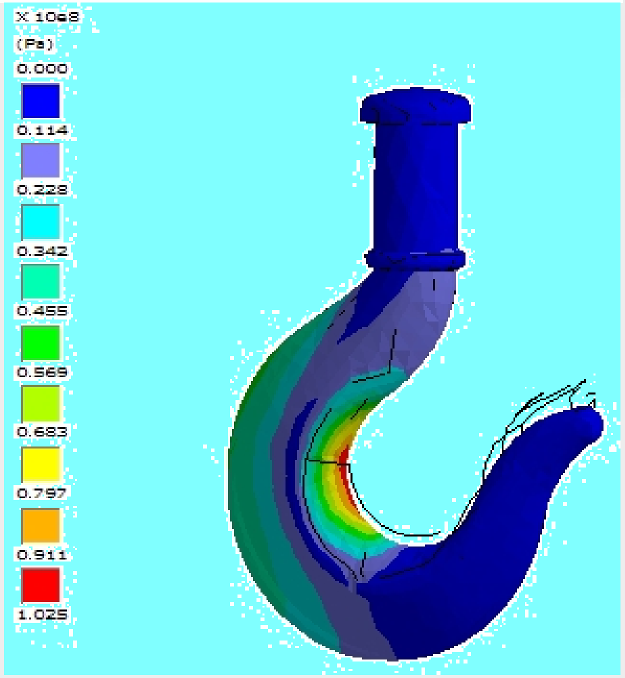
Surface subdivision
Subdivision models
The integrated way to look at the shape:
Object can be considered as a set of faces,
each face can be decomposed into a set of edges,
each edge can be decomposed into vertices

Surface Modeling

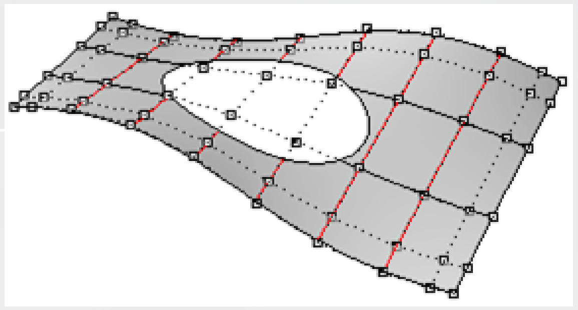
Finite Element Simulation
Subdivision to finite mesh

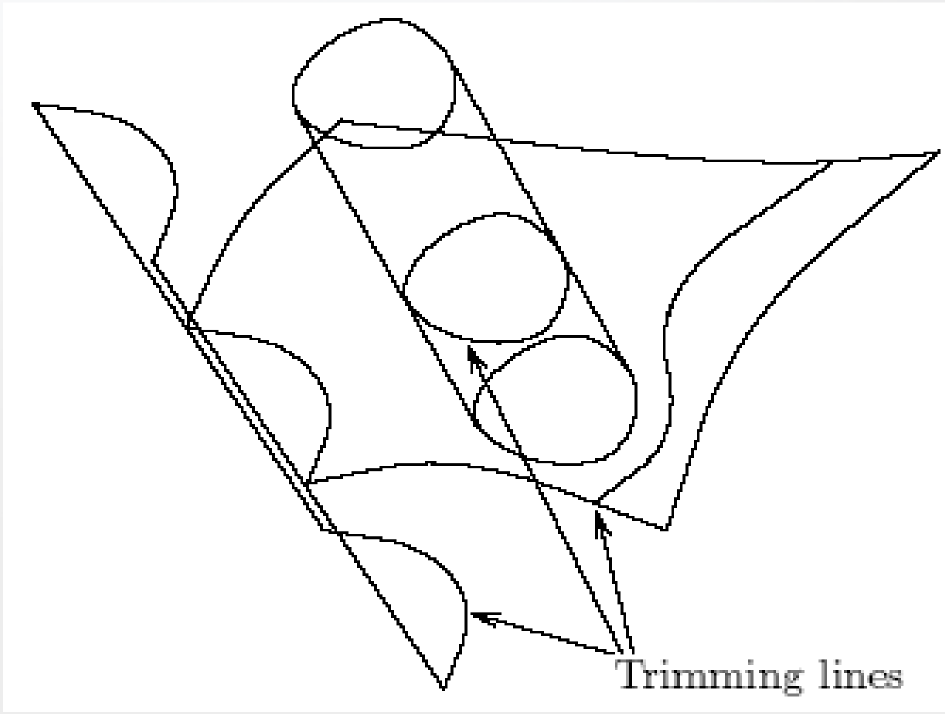


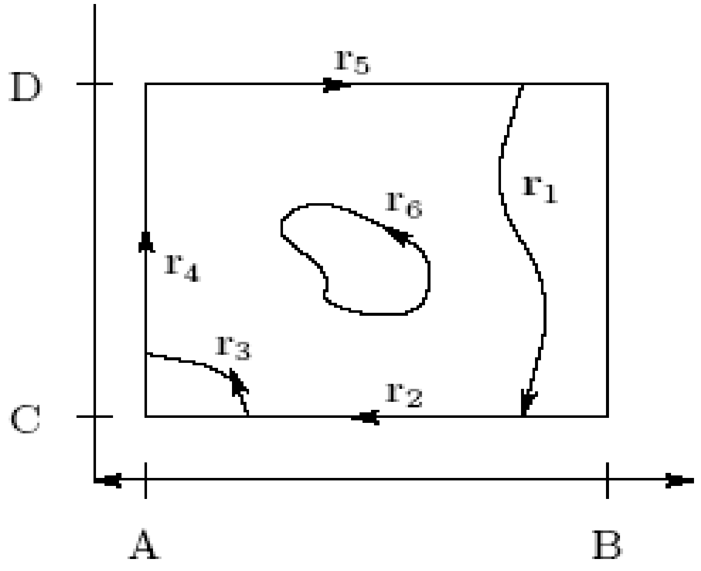


Surface Modeling

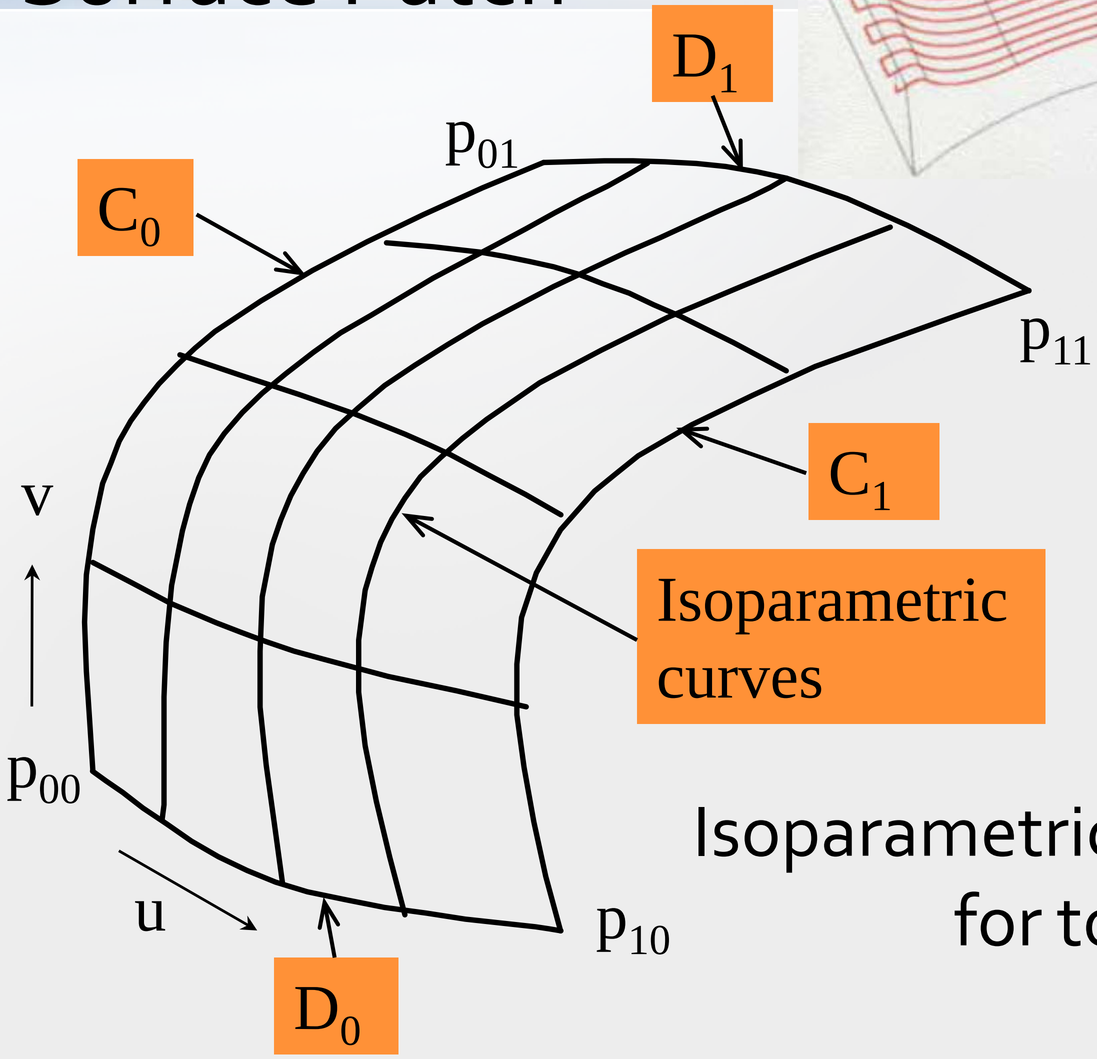
Defining a surface with hole

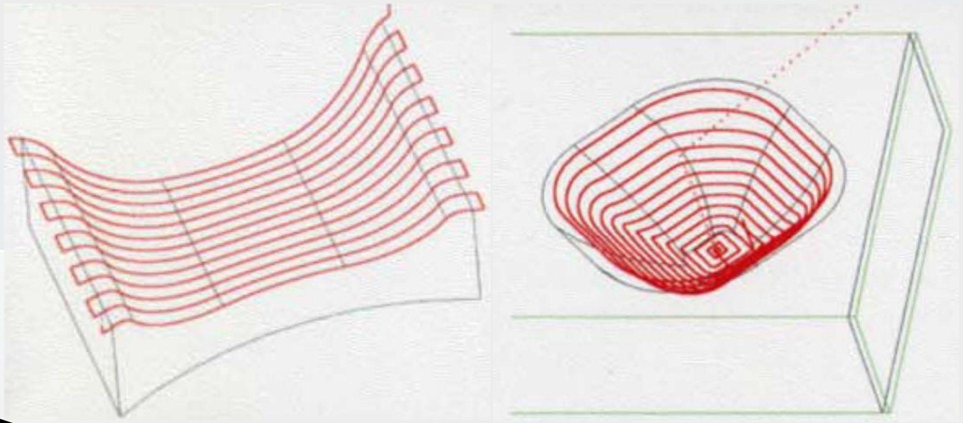


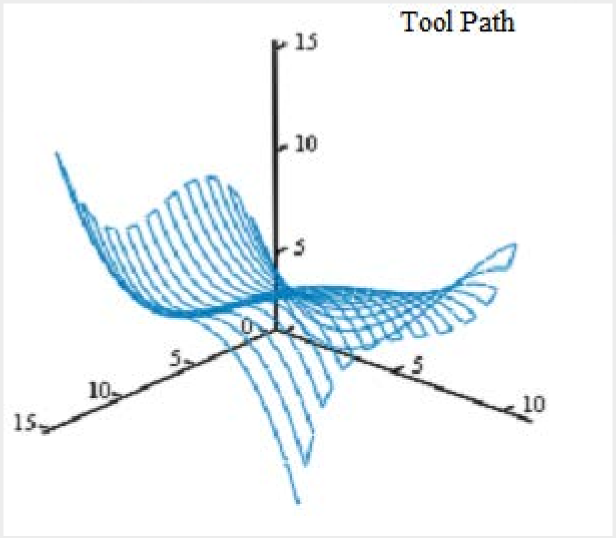




Surface Patch



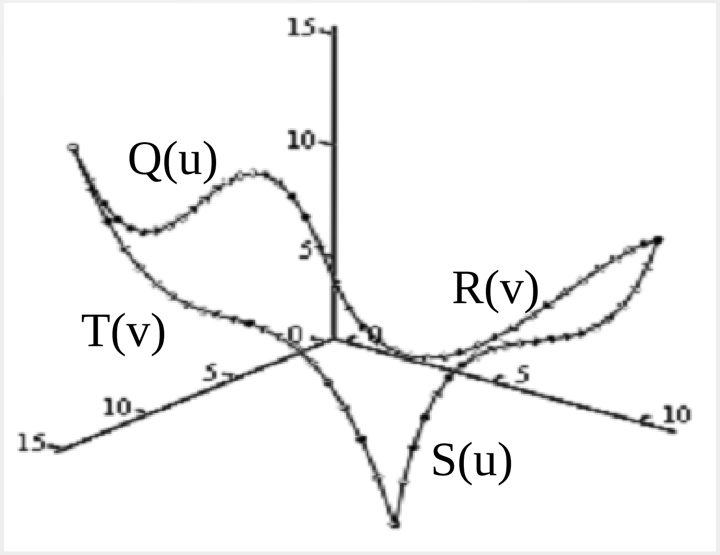


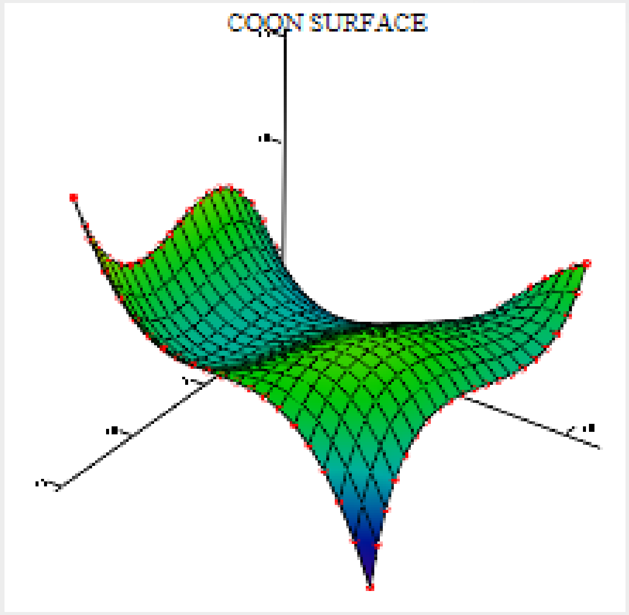


Isoparametric curves can be used for tool path generation.

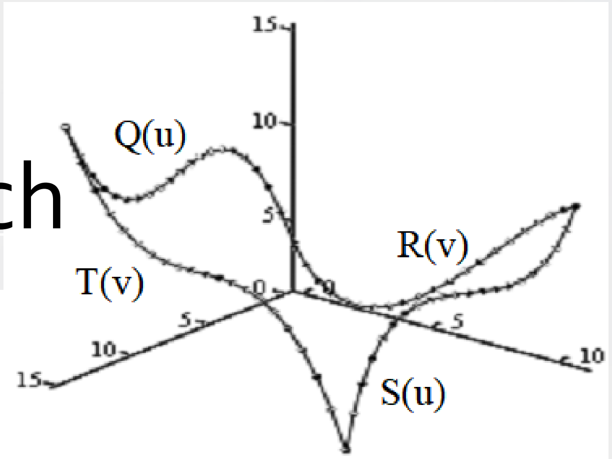
Linearly Blended Coons Patch

- Surface is defined by linearly interpolating between the boundary curves
- Simple, but doesn't allow adjacent patches to be joined smoothly





Linearly Blended Coons Patch



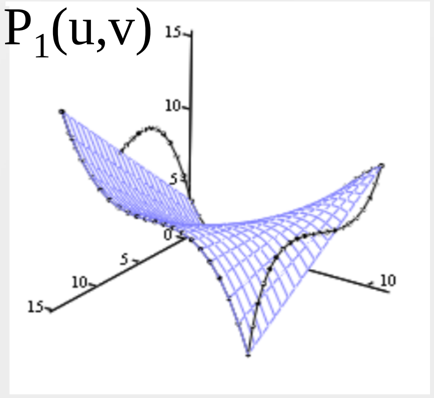
$$P1(u,v) := (1-u)\cdot T(v) + u\cdot R(v)$$

$$P2(u,v) := (1-v)\cdot Q(u) + v\cdot S(u)$$

$$P3(u,v) := (1-v)\cdot[(1-u)\cdot T(0) + u\cdot R(0)] + v\cdot[(1-u)\cdot T(1) + u\cdot R(1)]$$

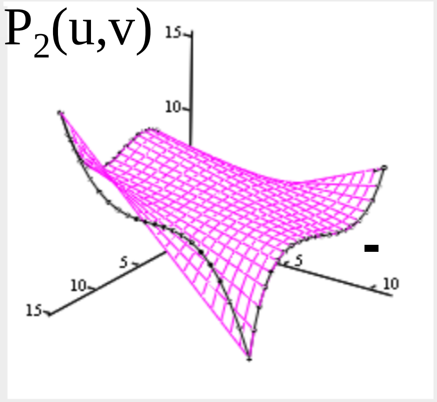
$$P(u,v) := P1(u,v) + P2(u,v) - P3(u,v)$$

$P_1(u,v)$



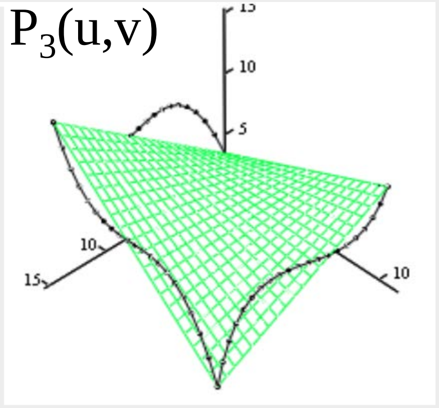
$(E^{\langle 0\rangle}, E^{\langle 1\rangle}, E^{\langle 2\rangle}), (X1, Y1, Z1)$

$P_2(u,v)$



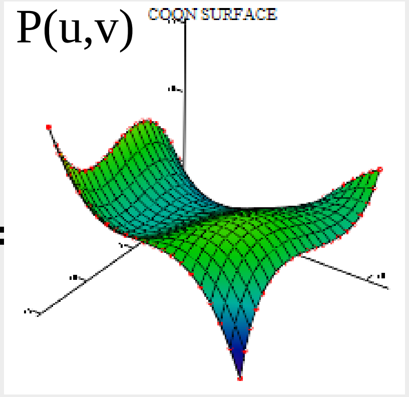
$(E^{\langle 0\rangle}, E^{\langle 1\rangle}, E^{\langle 2\rangle}), (X2, Y2, Z2)$

$P_3(u,v)$



$(E^{\langle 0\rangle}, E^{\langle 1\rangle}, E^{\langle 2\rangle}), (X3, Y3, Z3)$

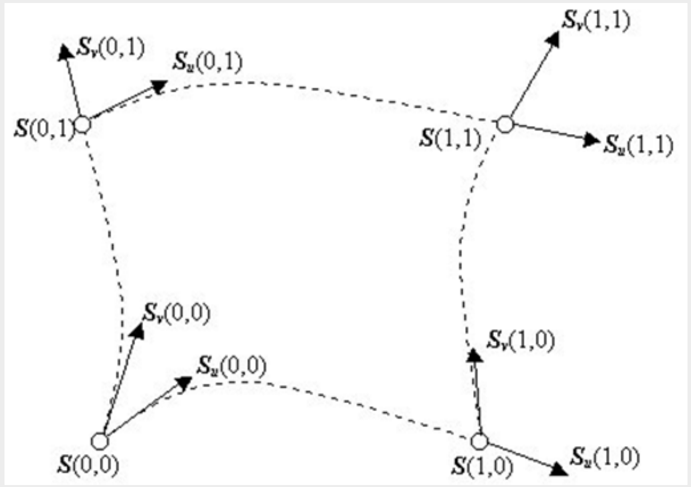
$P(u,v)$

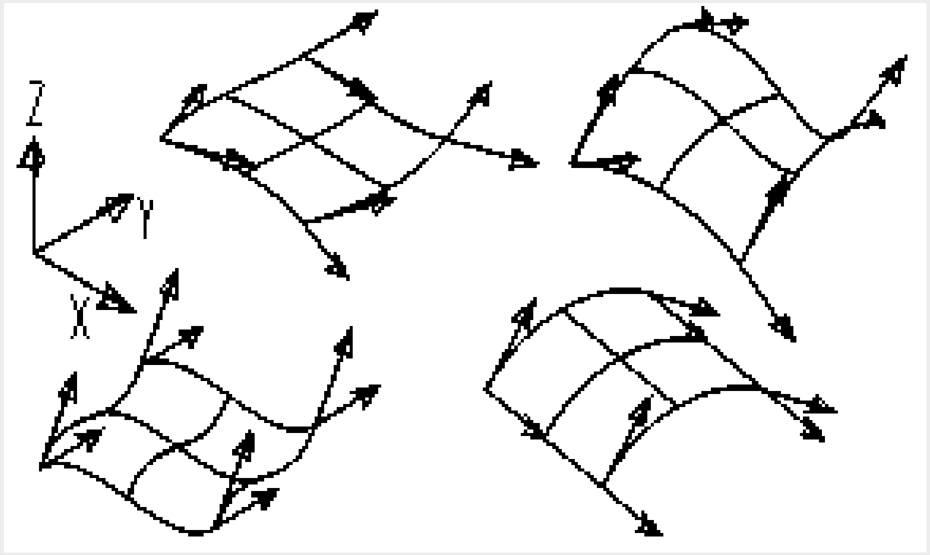


Bicubic Patch

Extension of cubic curve
16 unknown coefficients – 16 boundary conditions
Tangents and “twists” at corners of patch can be used
Like Lagrange and Hermite curves, difficult to work with

$$\mathbf{P}(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{k}_{i,j}u^i v^j$$





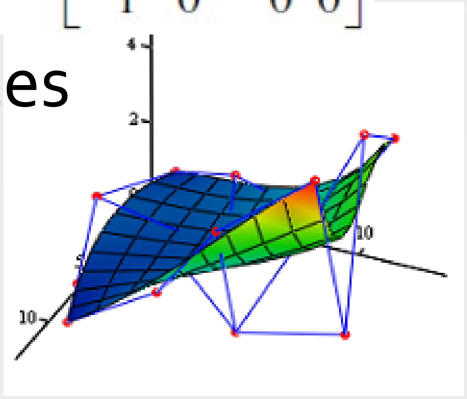
$$p(u,w) = U^T[M_B][P][M_B]^T W$$

$$U^T = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

$$[M_B] = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

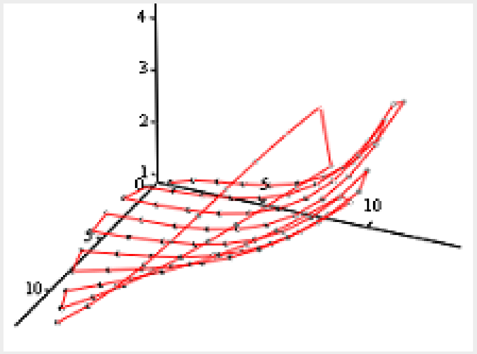
Bezier Surfaces

- Bezier curves can be extended to surfaces
- Same problems as for Bezier curves:
  - no local modification possible
  - smooth transition between adjacent patches difficult to achieve



$$C_{n,i} := \frac{n!}{i!\cdot n-i!} \qquad B(u,i) := C_{n,i}\cdot u^i\cdot(1-u)^{n-i}$$

$$P(u,v,r) := \sum_{i=0}^{m} \sum_{j=0}^{n} (p_{i,j})_r \cdot B(u,i)\cdot B(v,j)$$



Isoparametric curves used for tool path generation

B-Spline Surfaces

$$p(u,w)=U^T[M_B][P][M_B]^T W$$

$$P(u,v)=[N_{0,k}(u)\ \ N_{1,k}(u)\ \cdots\ N_{n,k}(u)]\begin{bmatrix} P_{00} & P_{01} & \cdots & P_{0m} \\ P_{10} & P_{11} & \cdots & P_{1m} \\ \vdots & \vdots & & \vdots \\ P_{n0} & P_{n1} & \cdots & P_{nm} \end{bmatrix}\begin{bmatrix} N_{0,l}(v) \\ N_{1,l}(v) \\ \vdots \\ N_{m,l}(v) \end{bmatrix}$$

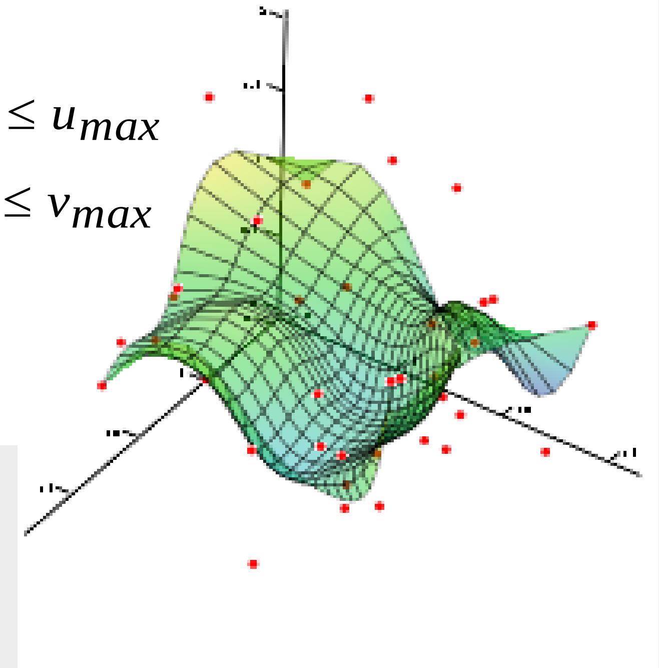
$$U^T=[u^3\ \ u^2\ \ u\ \ 1]$$

$$M_{\text{Bspline}}=\frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 3 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

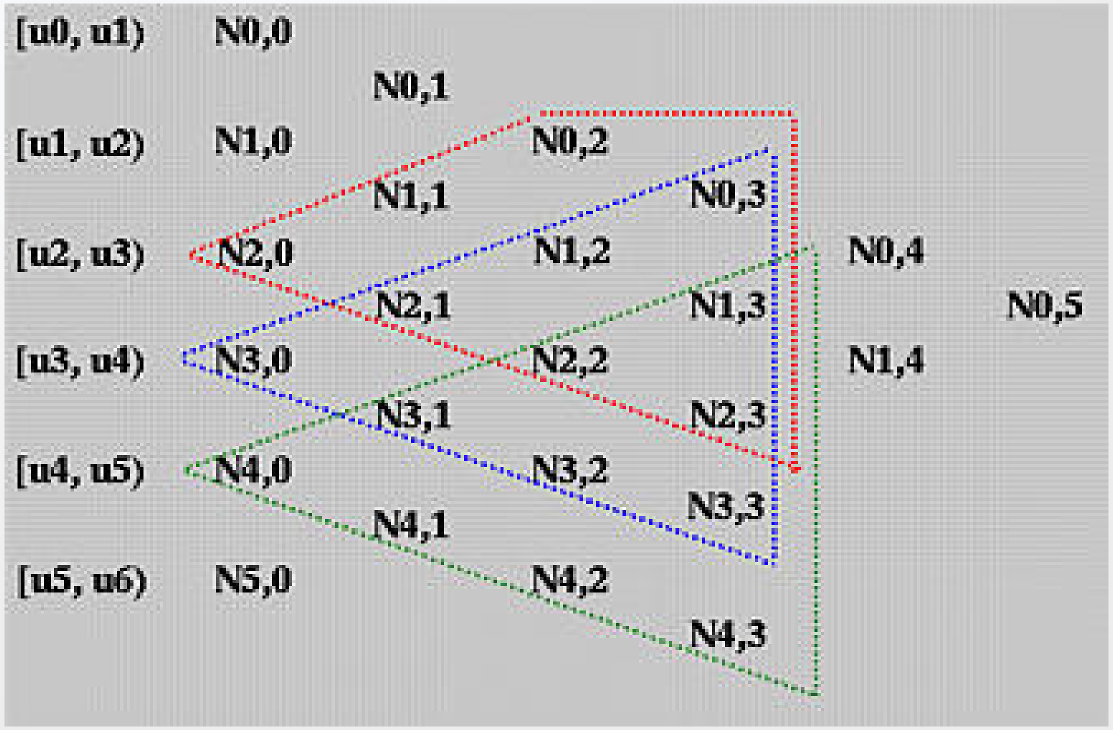
- As with curves, B-spline surfaces are a generalization of Bezier surfaces
- The surface approximates a **control polygon**
- Open and closed surfaces can be represented

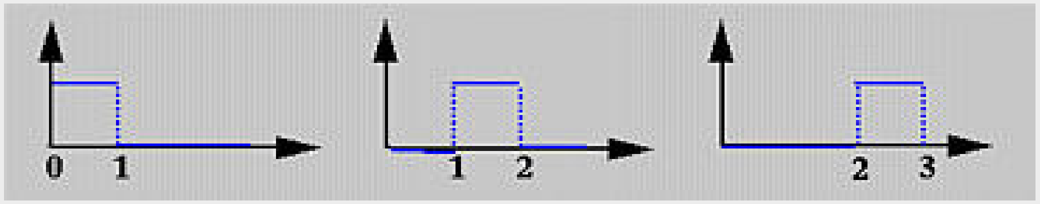
$$P(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} P_{ij}\cdot N_{i,k}(u)\cdot N_{j,l}(v) \qquad 0\le u\le u_{max} \quad 0\le v\le v_{max}$$

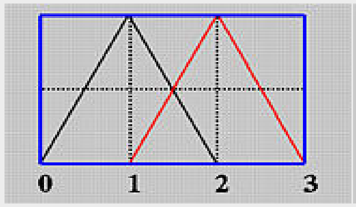
$$N_{i,k}(u)=(u-u_i)\cdot\frac{N_{i,k-1}}{u_{i+k-1}-u_i}+(u_{i+k}-u)\cdot\frac{N_{i+1,k-1}}{u_{i+k}-u_{i+1}}$$

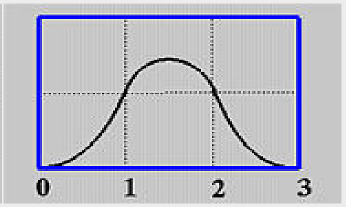

B-Spline Surface

B-Spline Surfaces



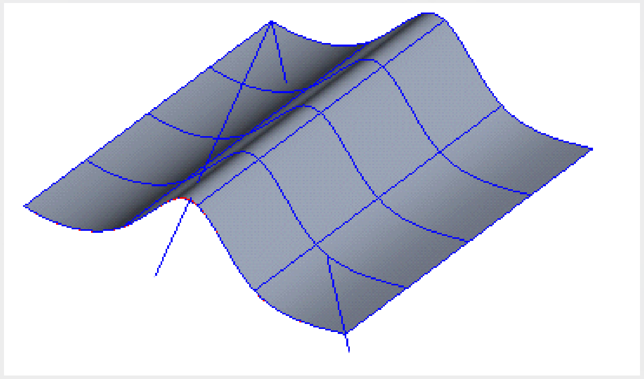


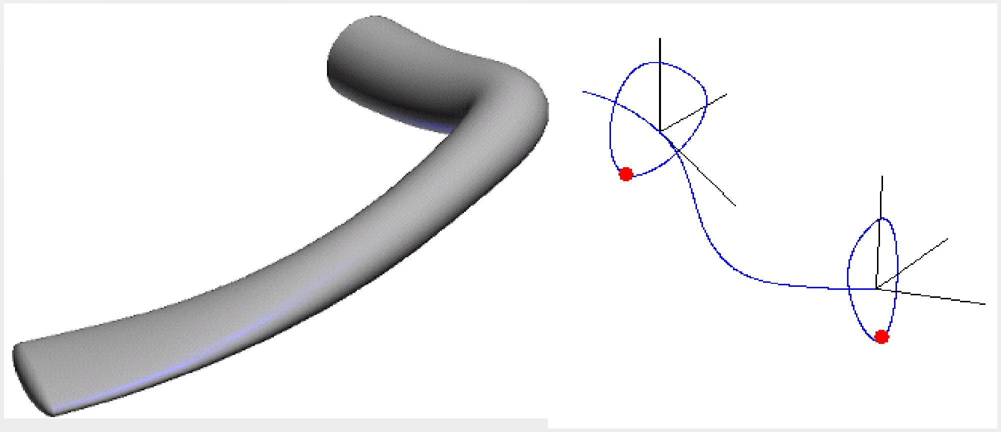


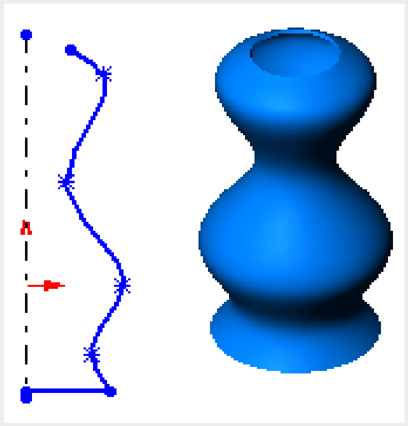


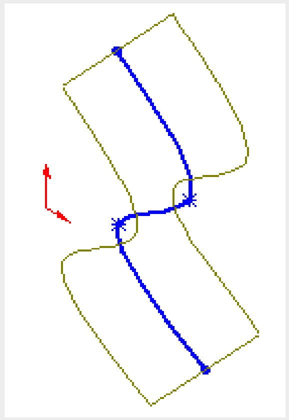
Surfaces from Curves

- Tabulated cylinder (extrusion)
- Ruled surface (lofting or spined)
- Surface of revolution
- Swept surface
- Sculptured surface



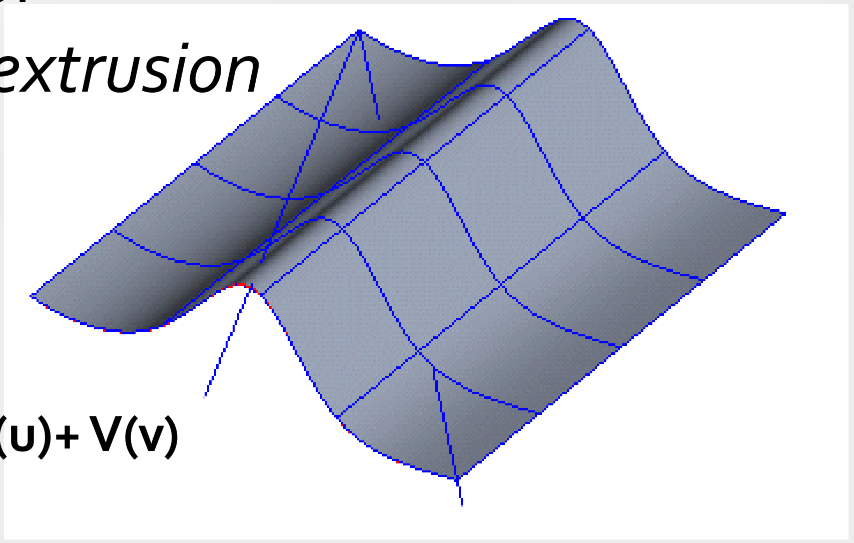


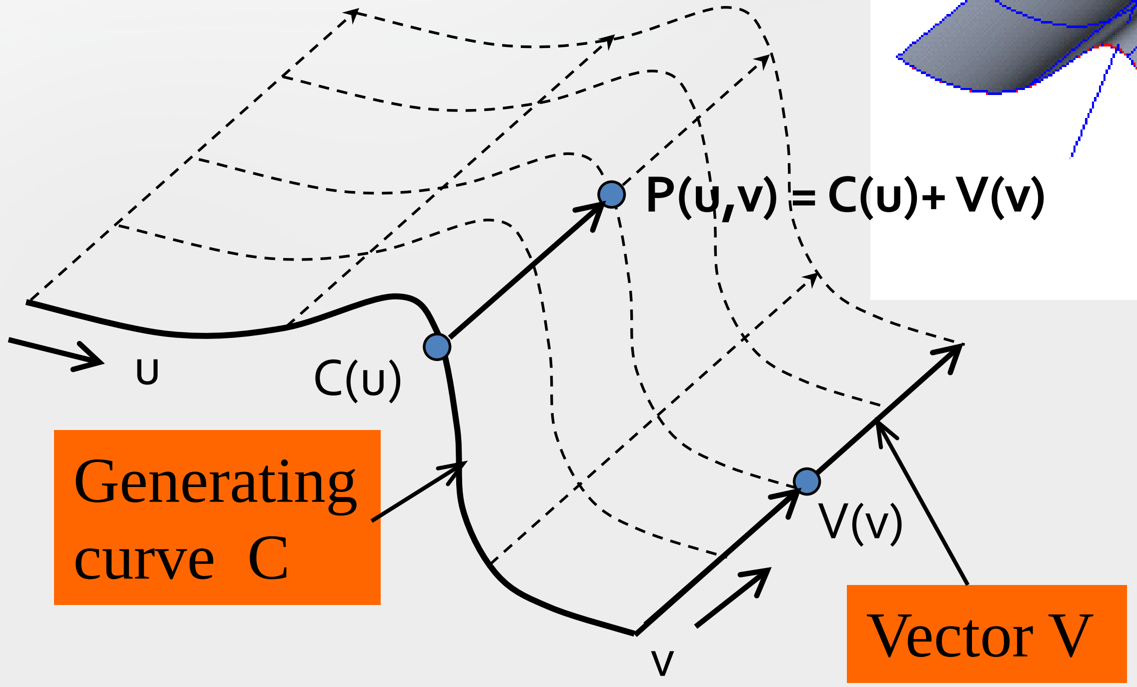




Tabulated Cylinder

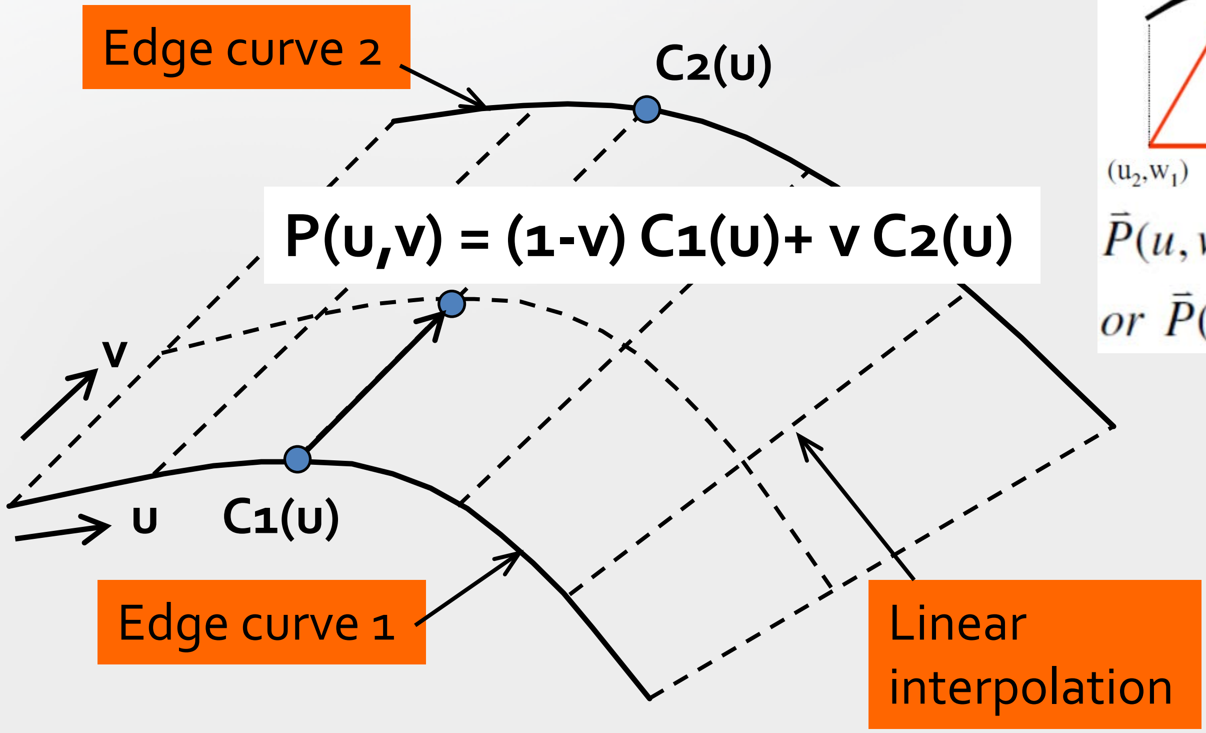
- Project curve along a vector
- In SolidWorks, created by *extrusion*



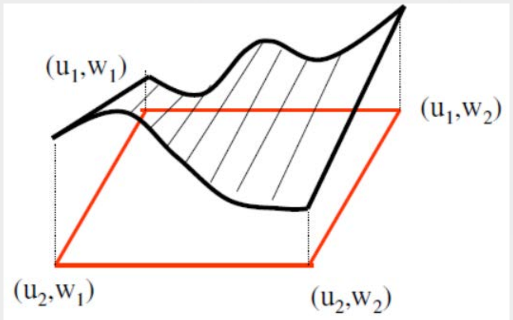


Ruled Surface

- Linear interpolation between two edge curves
- Created by **lofting** through cross sections



P(u,v) = (1-v) C1(u)+ v C2(u)

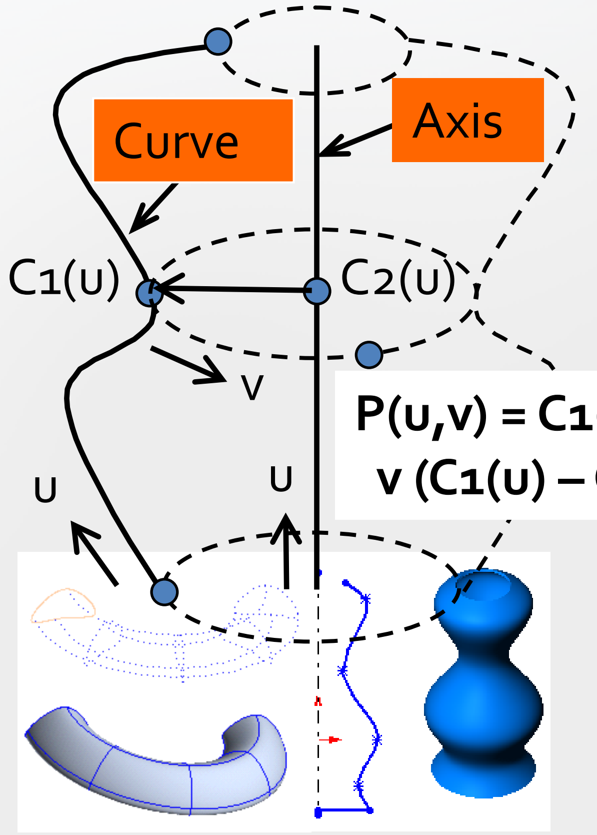


$$\vec{P}(u,w) = \vec{p}(u,0)(1-w) + \vec{p}(u,1)w$$

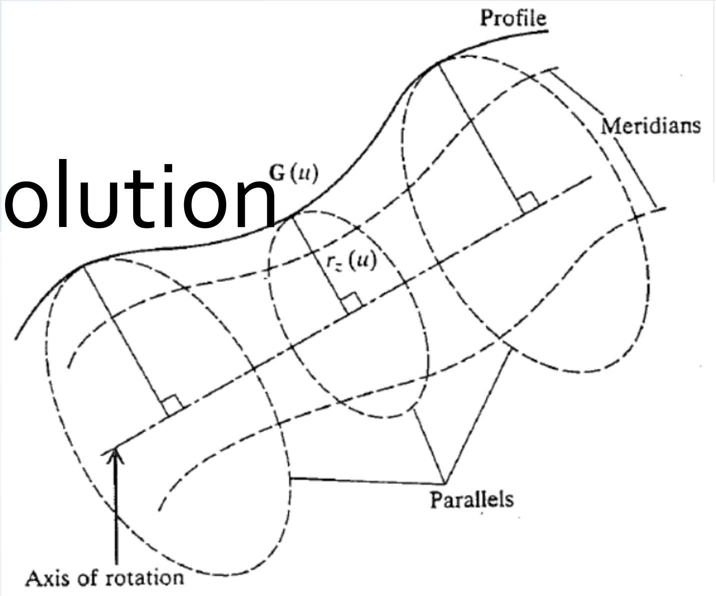
$$or\ \ \vec{P}(u,w) = \vec{p}(0,w)(1-u) + \vec{p}(1,w)u$$

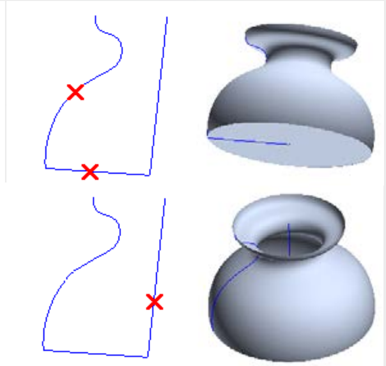
Surface of Revolution

Revolve curve about an axis

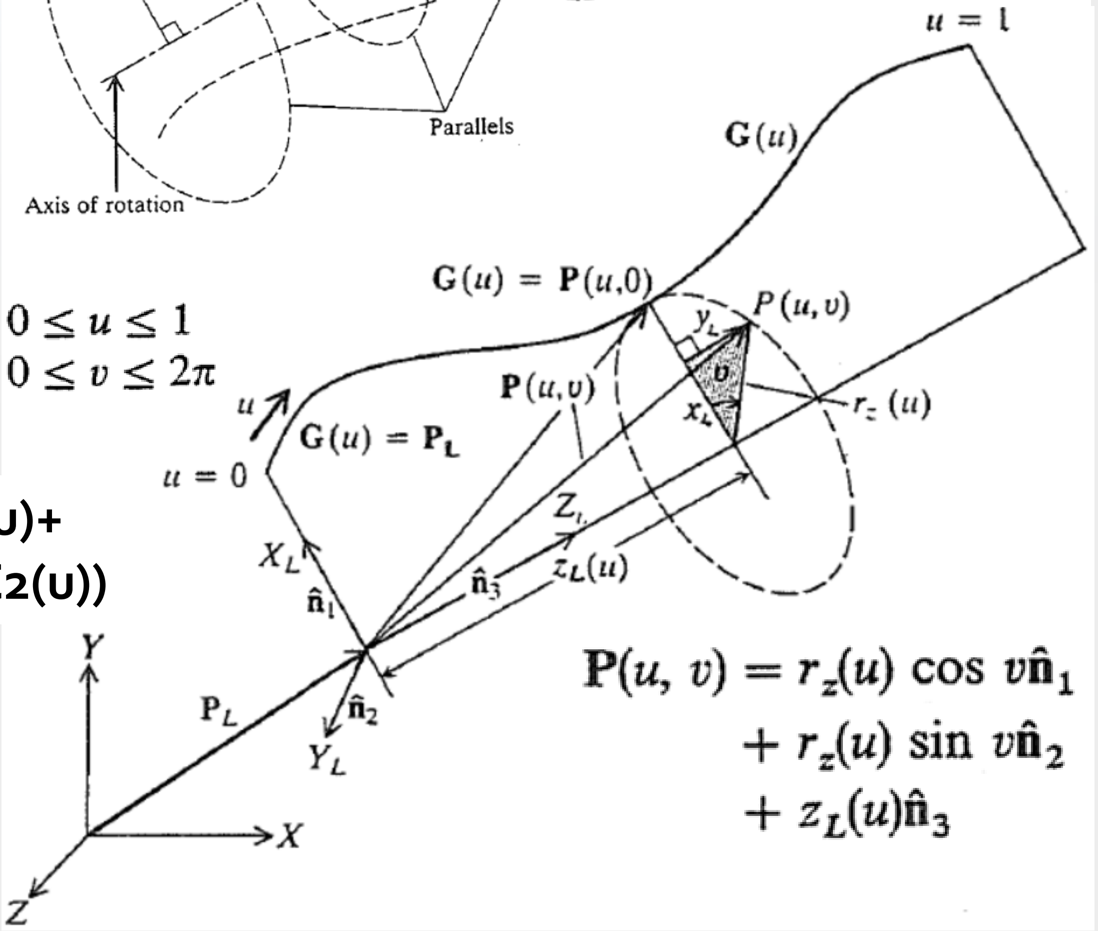


P(u,v) = C1(u)+
v (C1(u) – C2(u))





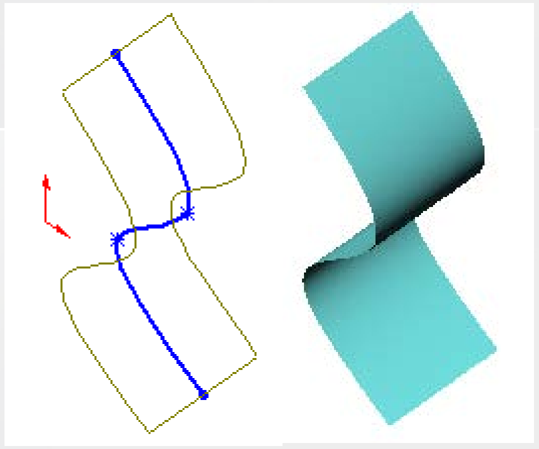
$0 \le u \le 1$
$0 \le v \le 2\pi$

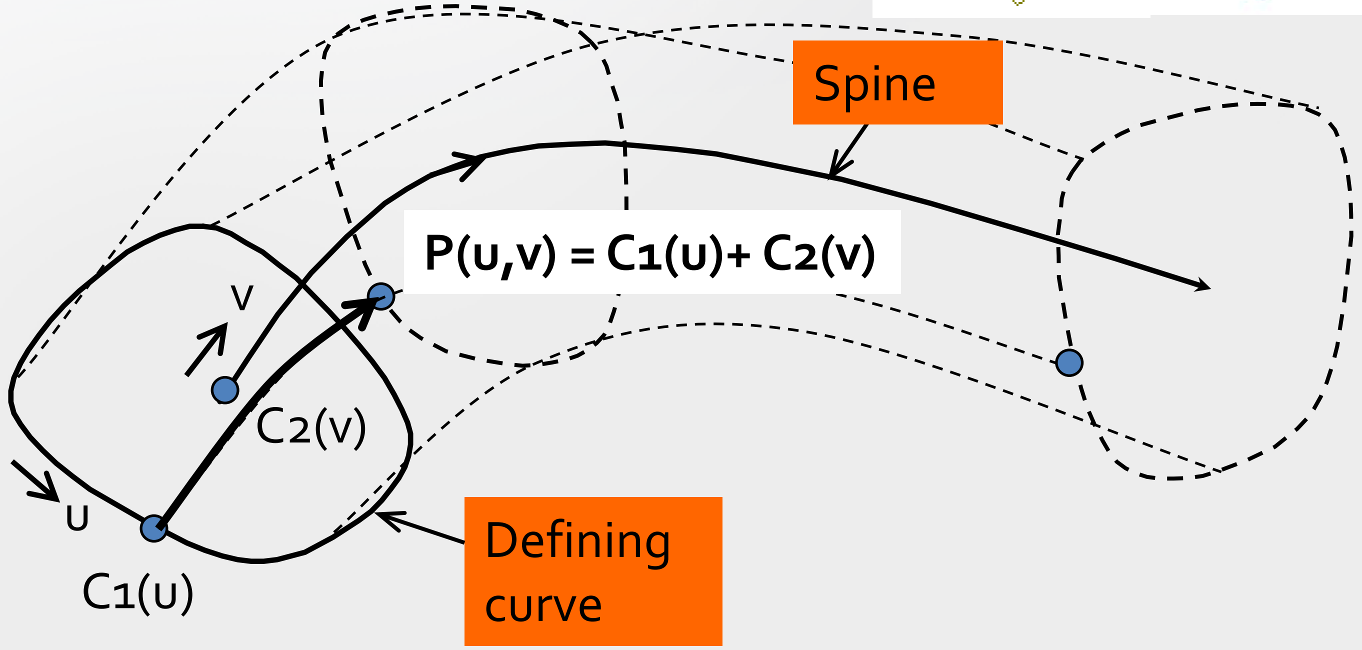


$$\begin{aligned} \mathbf{P}(u, v) &= r_z(u) \cos v\hat{\mathbf{n}}_1 \\ &+ r_z(u) \sin v\hat{\mathbf{n}}_2 \\ &+ z_L(u)\hat{\mathbf{n}}_3 \end{aligned}$$

Swept Surface

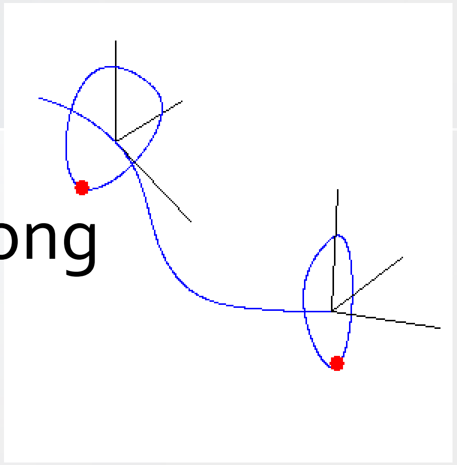
Defining curve swept along an arbitrary spine curve



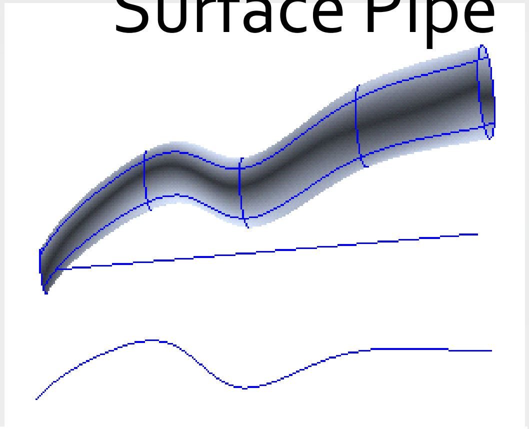


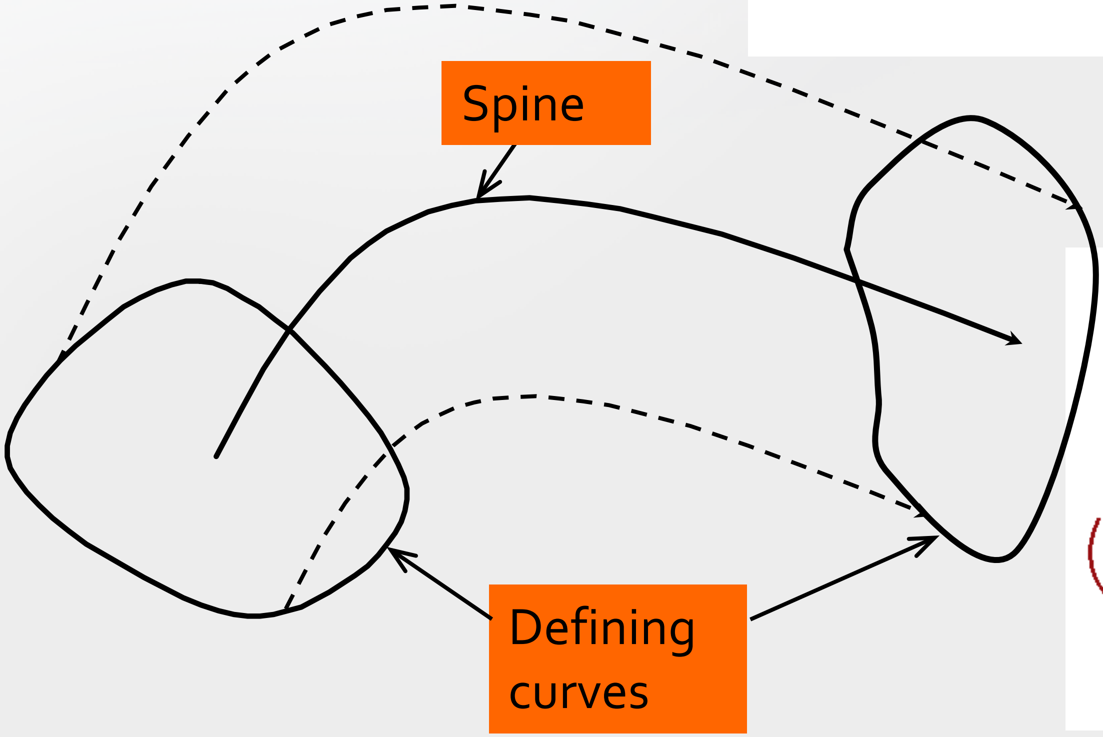
Lofted Surface

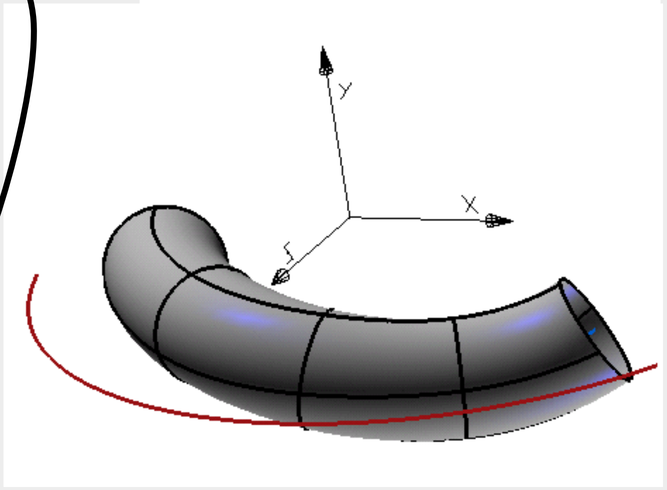
Defining curve swept along an arbitrary spine curve



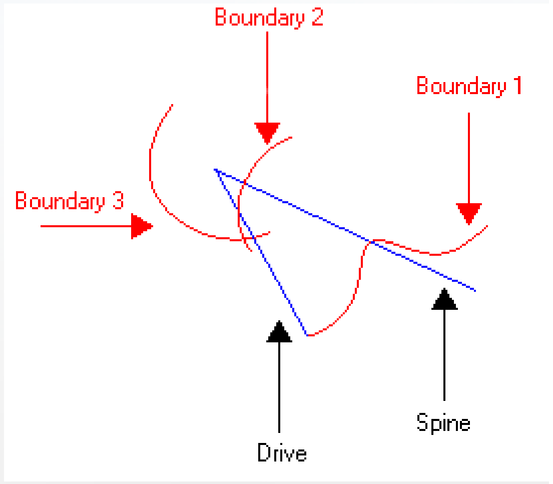
Surface Pipe

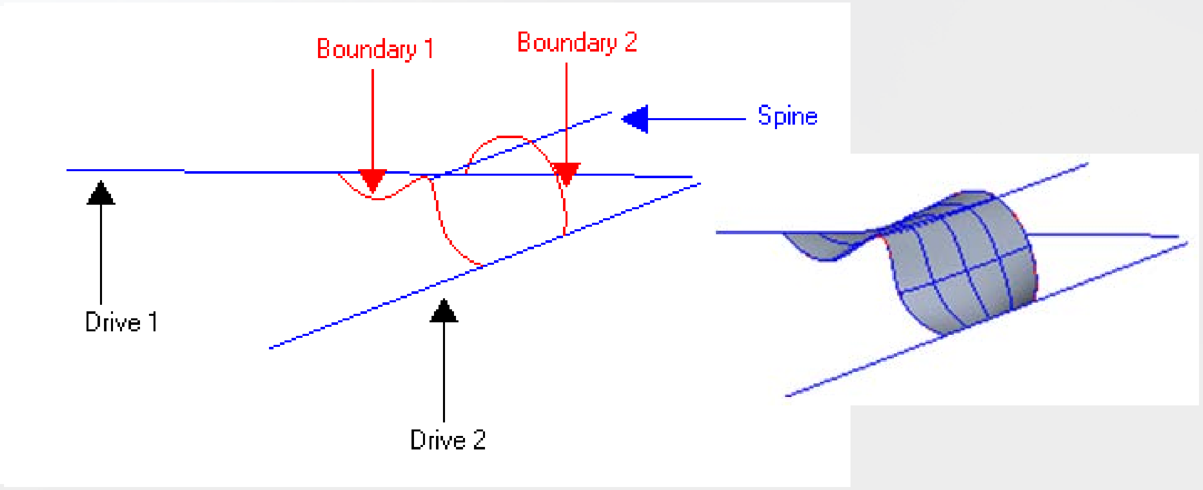


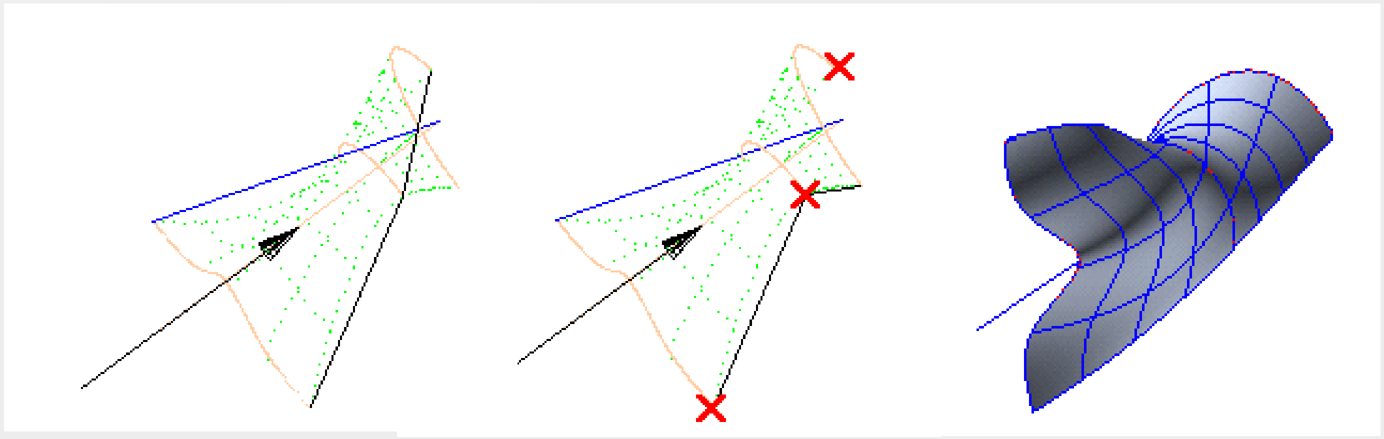




Spined Surface

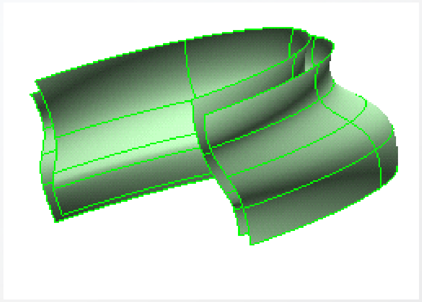
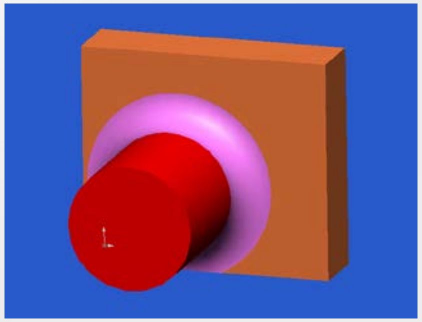
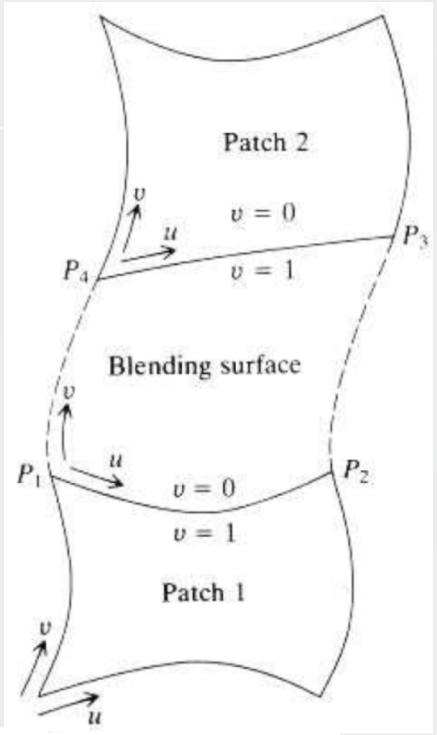


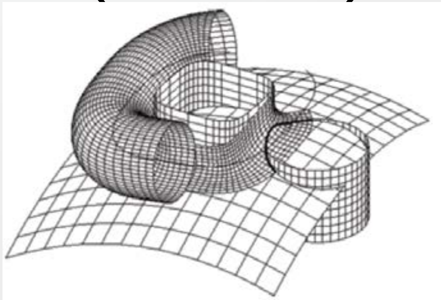
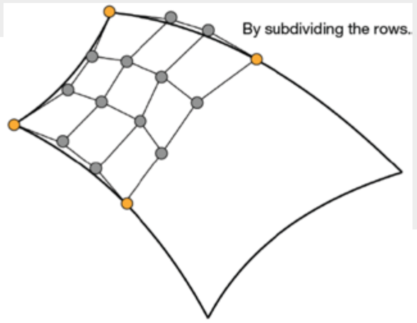


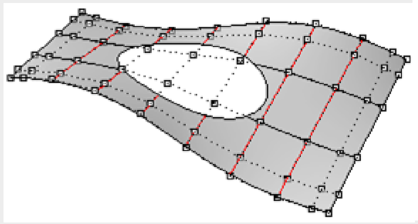
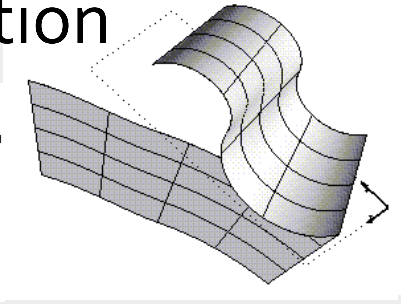
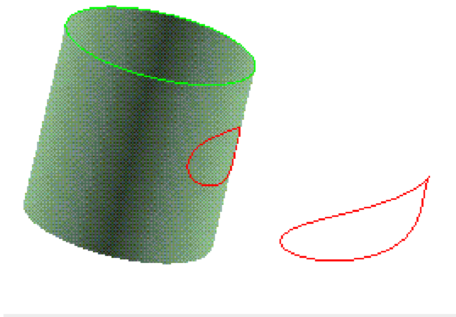
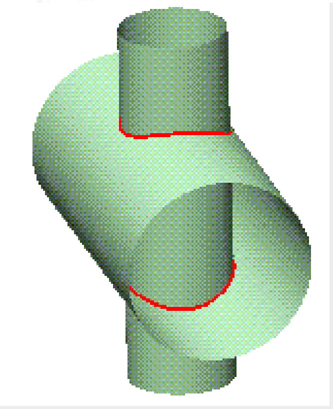


Surface Manipulations

- **Offset**
- **Blend**
- Display
- Segmentation (division)
- Trimming
- Intersection
- Projection
- Transformation

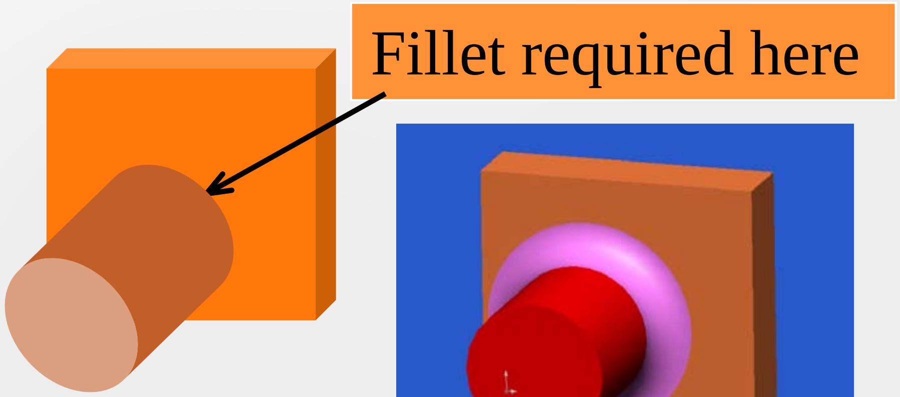



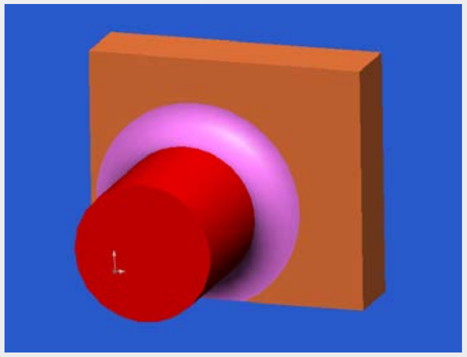





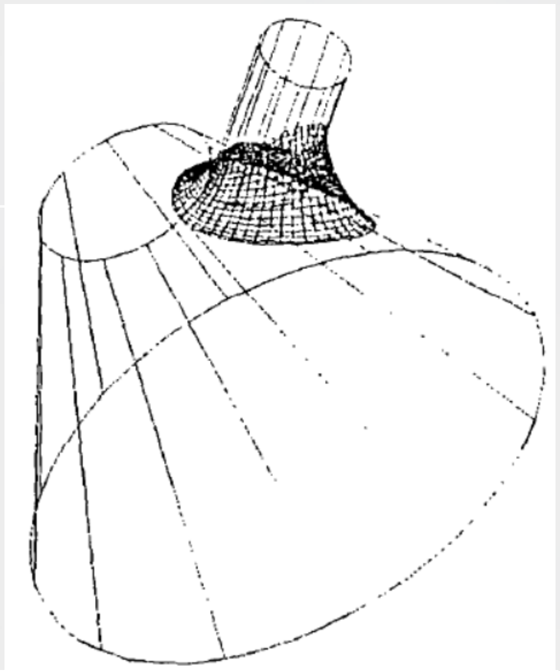
Fillets and Blends

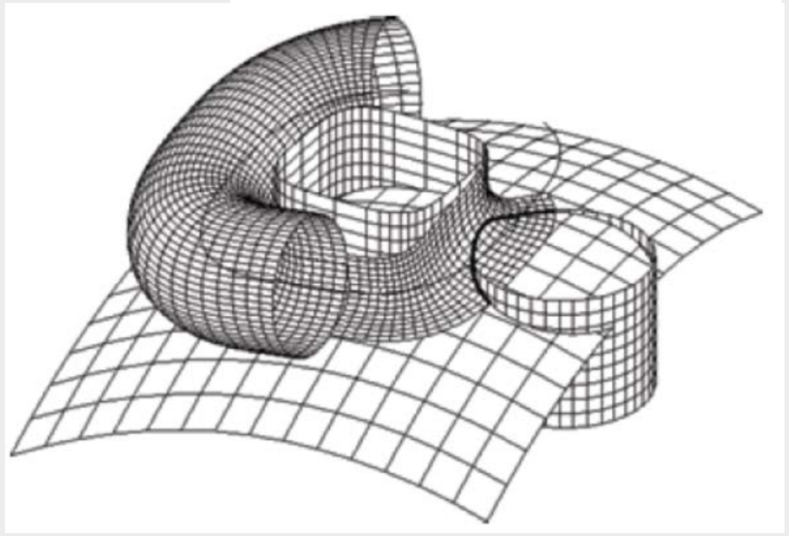
Often necessary to create a blend between intersecting surfaces.

Most common application is a **fillet**



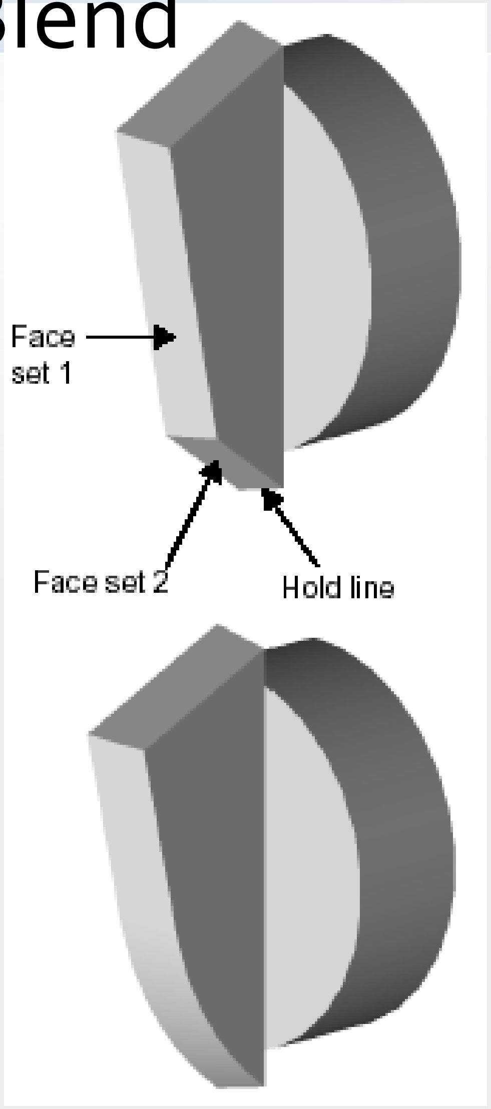


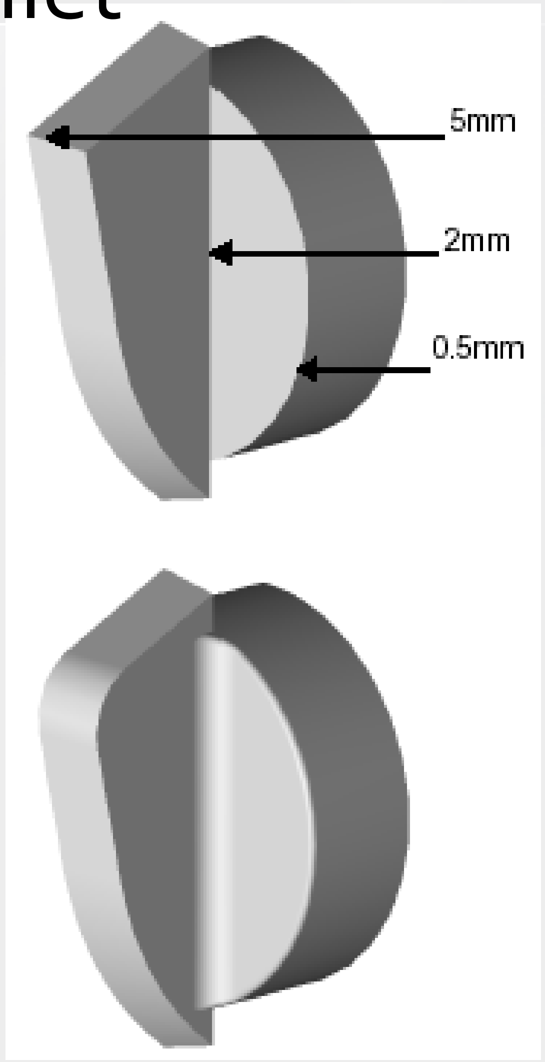




Face Blend

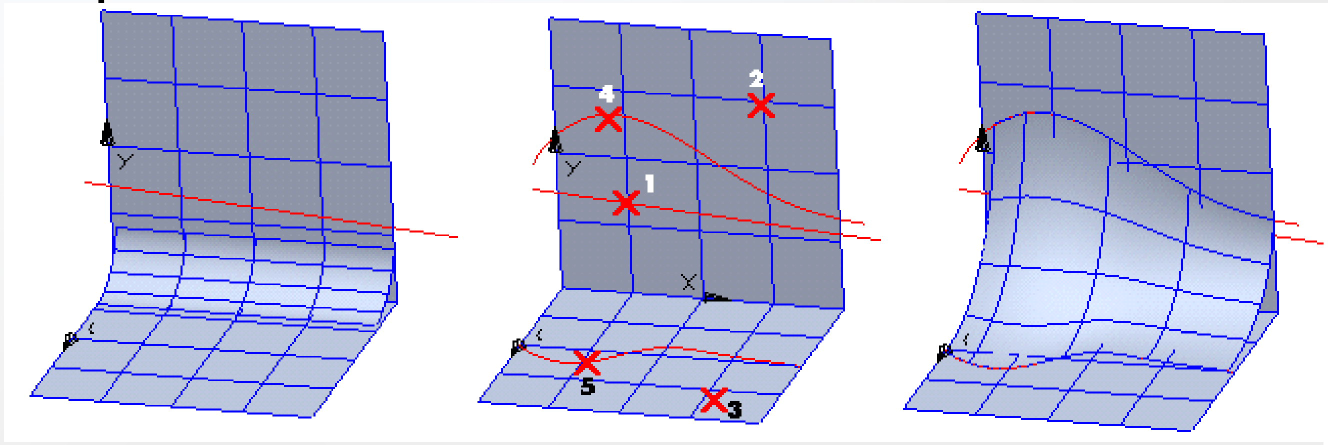
Fillet

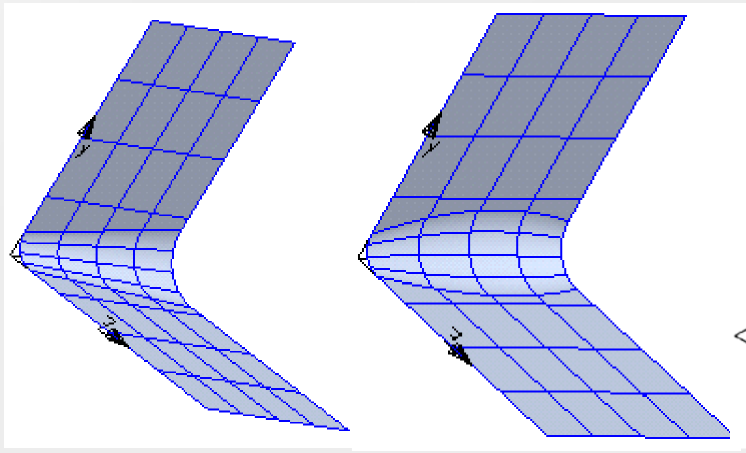


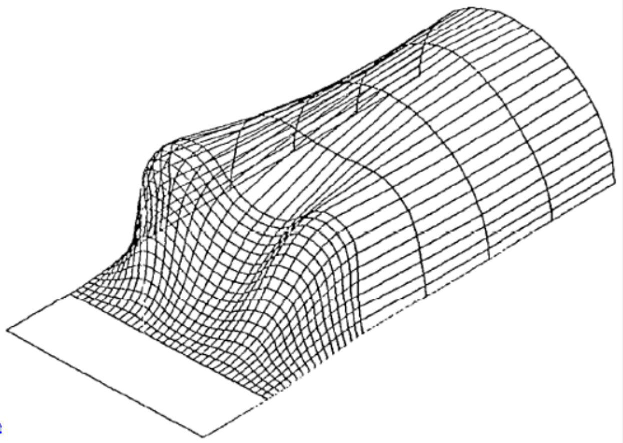


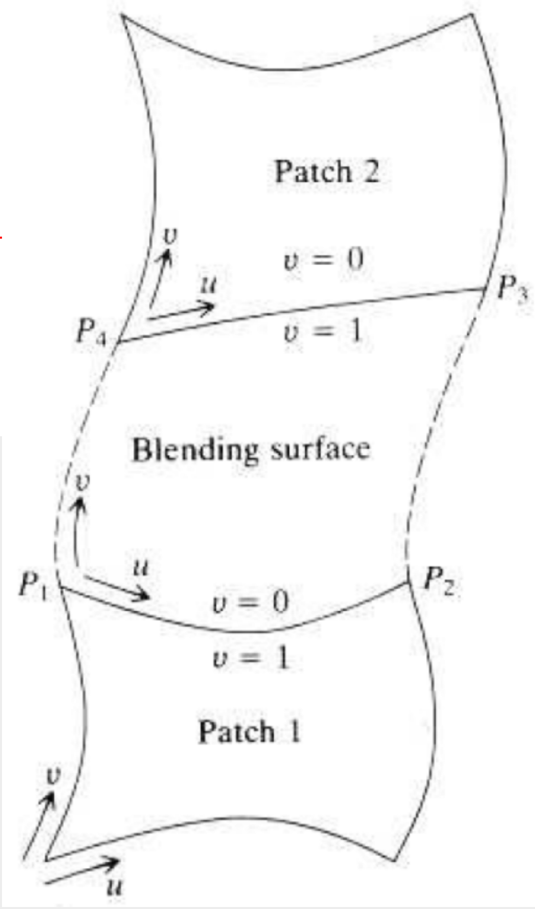
Fillet between two selected surfaces

Spined fillet with drive curve



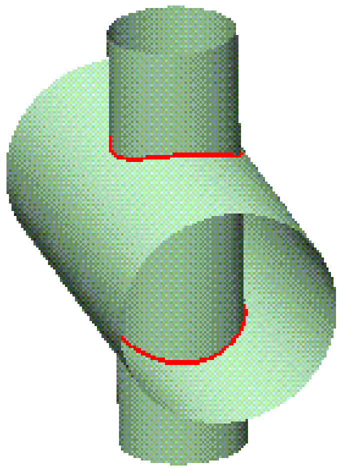
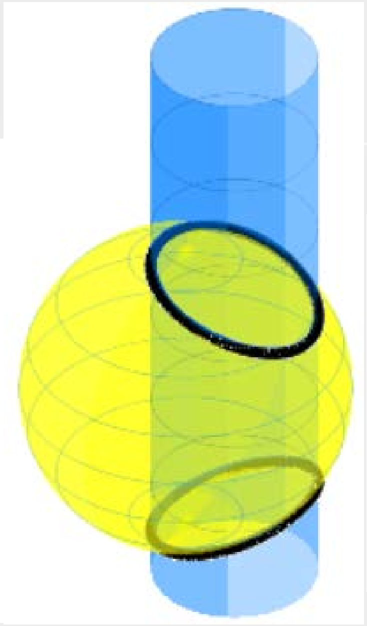
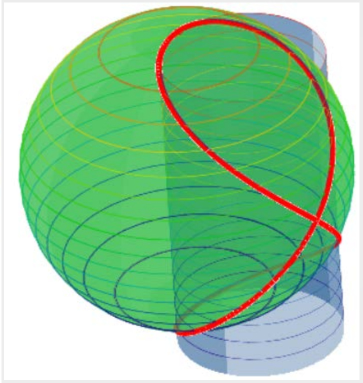




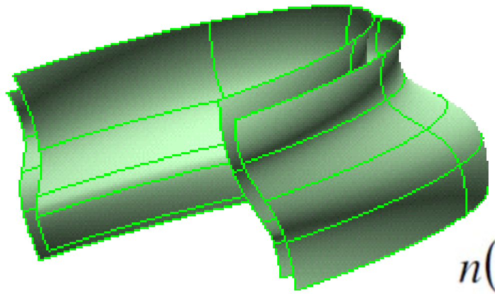


Surface functions

- Curve in surface intersection

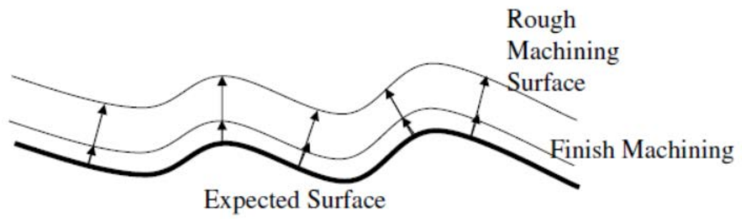




- Offset Surface

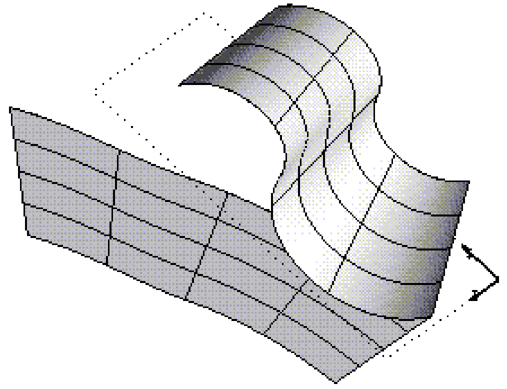


$$n(u,w) = p'_u(u,w) \times p'_w(u,w)$$

$$\overline{p}(u,w)_{offset} = \overline{p}(u,w) + \overline{n}(u,w) * d$$

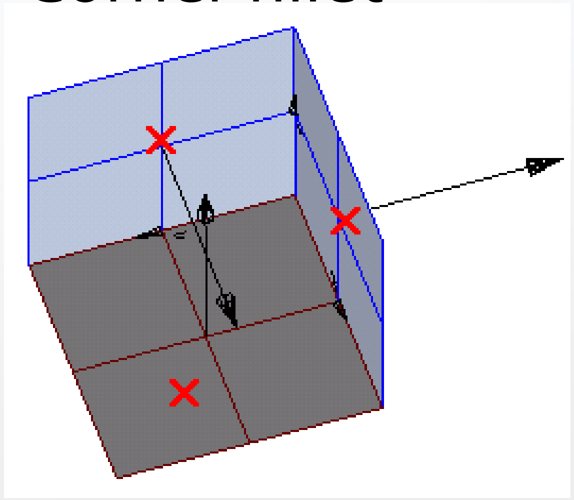


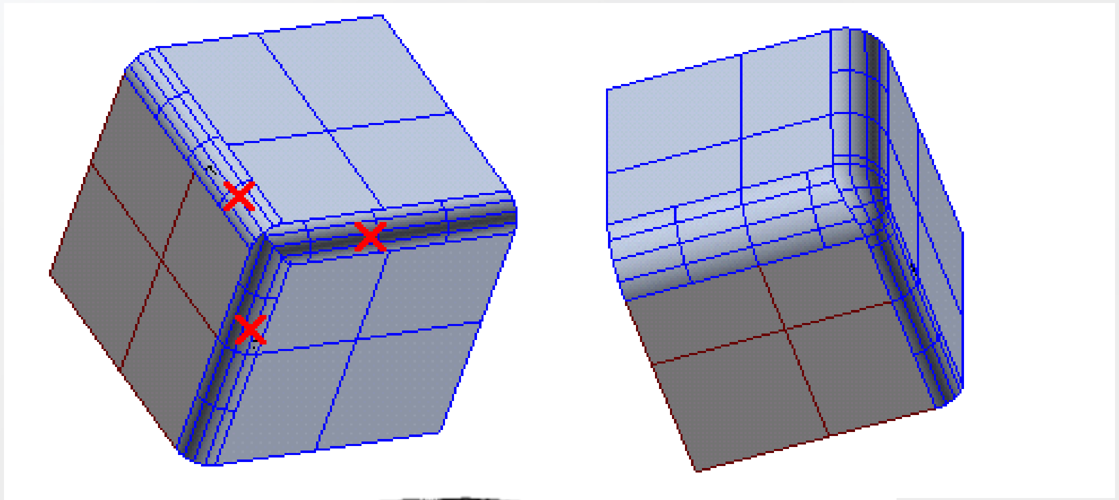
- Planar development of surfaces

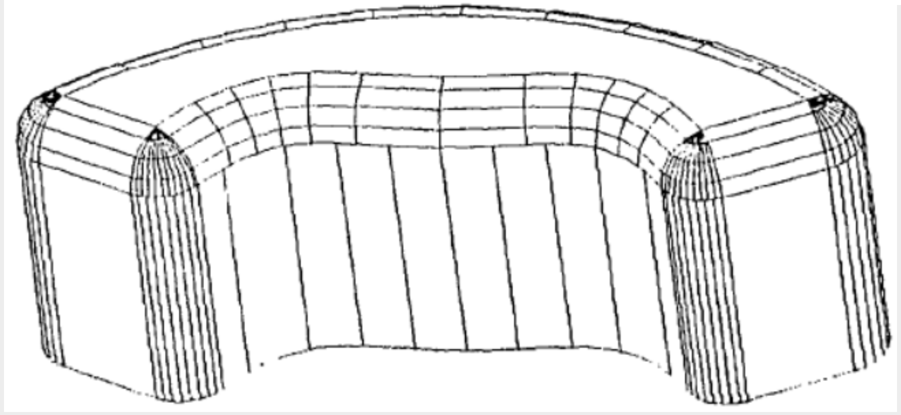


Fillet between three surfaces

Corner fillet



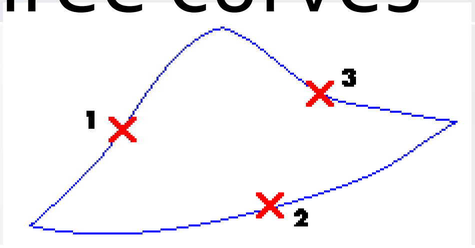


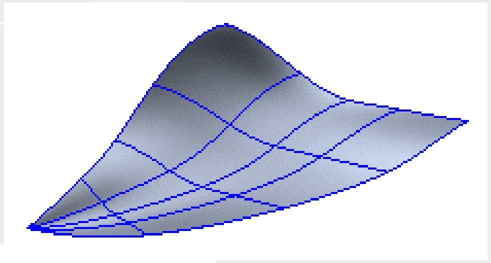


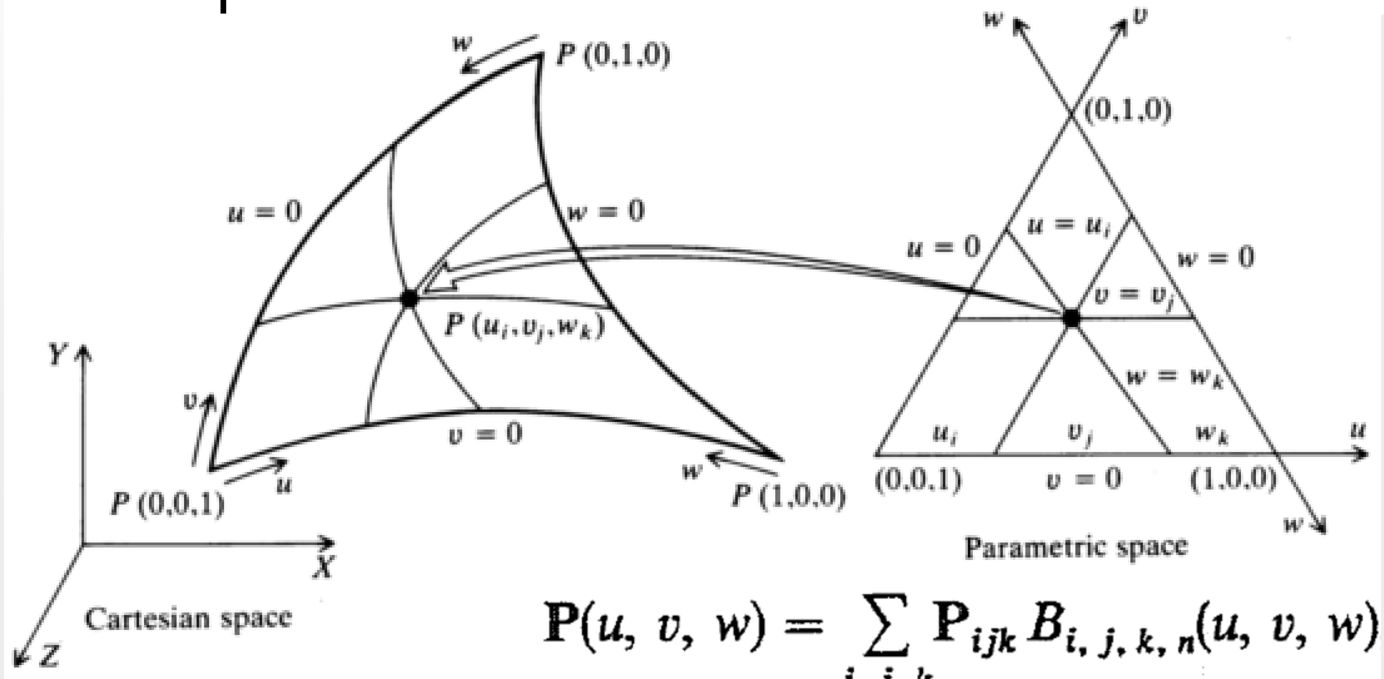
Fillet between three curves

Triangular Surface

Bezier patch







$$\mathbf{P}(u, v, w) = \sum_{i,j,k} \mathbf{P}_{ijk} B_{i,j,k,n}(u, v, w)$$

Bernstein polynomials of degree n:

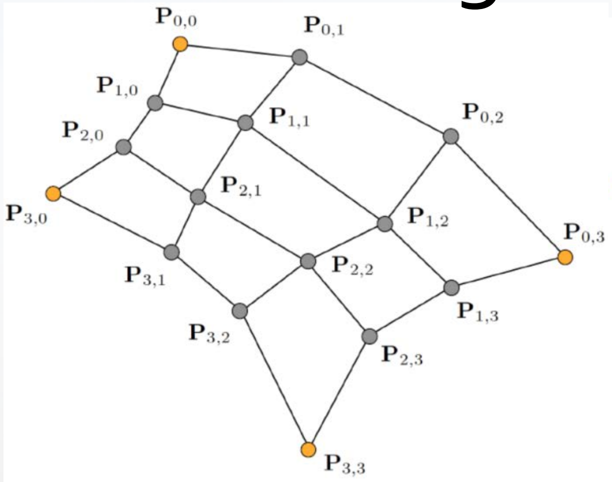
$$B_{i,j,k,n} = \frac{n!}{i!\,j!\,k!} u^i v^j w^k$$

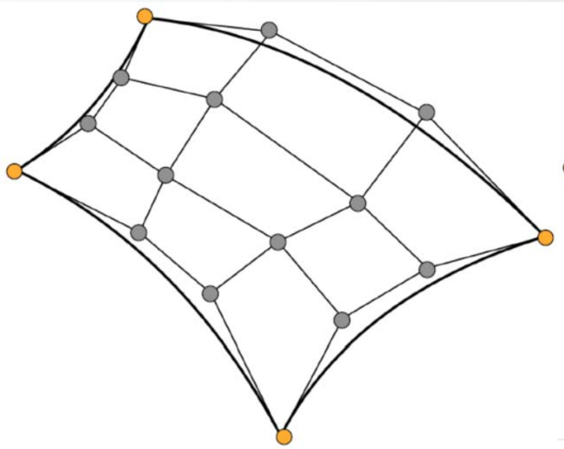
$0 \le u \le 1,\ 0 \le v \le 1,\ 0 \le w \le 1$

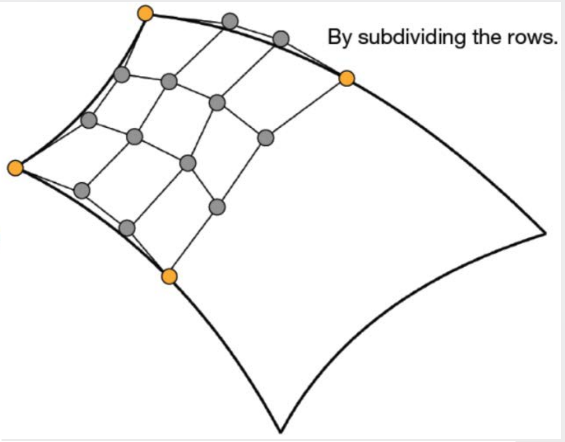
where $i, j, k \ge 0,\ i + j + k = n$

Subdividing a Bézier Patch

$$P = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \mathbf{P}_{0,3} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \mathbf{P}_{1,3} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \mathbf{P}_{2,3} \\ \mathbf{P}_{3,0} & \mathbf{P}_{3,1} & \mathbf{P}_{3,2} & \mathbf{P}_{3,3} \end{bmatrix}$$







$$\mathbf{P}(\frac{u}{2}, v) = \begin{bmatrix} 1 & (\frac{u}{2}) & (\frac{u}{2})^2 & (\frac{u}{2})^3 \end{bmatrix} M \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \mathbf{P}_{0,3} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \mathbf{P}_{1,3} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \mathbf{P}_{2,3} \\ \mathbf{P}_{3,0} & \mathbf{P}_{3,1} & \mathbf{P}_{3,2} & \mathbf{P}_{3,3} \end{bmatrix} M^T \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

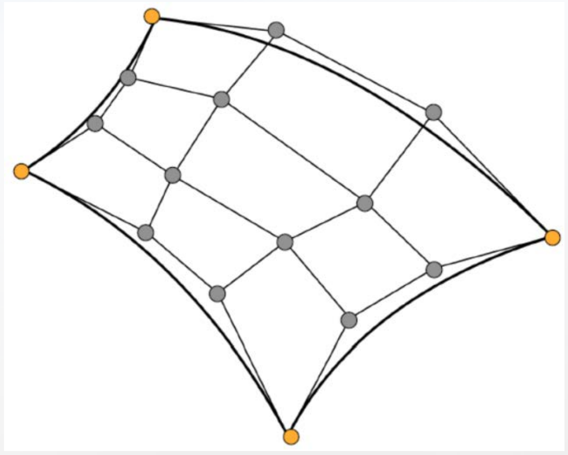
$$= \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix} M \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \mathbf{P}_{0,3} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \mathbf{P}_{1,3} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \mathbf{P}_{2,3} \\ \mathbf{P}_{3,0} & \mathbf{P}_{3,1} & \mathbf{P}_{3,2} & \mathbf{P}_{3,3} \end{bmatrix} M^T \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

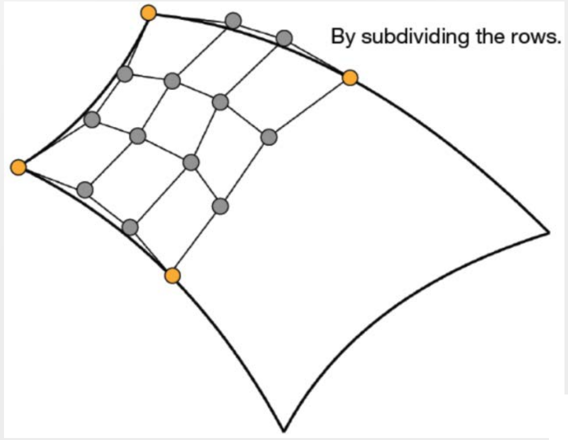
$$= \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} M S_L \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \mathbf{P}_{0,3} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \mathbf{P}_{1,3} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \mathbf{P}_{2,3} \\ \mathbf{P}_{3,0} & \mathbf{P}_{3,1} & \mathbf{P}_{3,2} & \mathbf{P}_{3,3} \end{bmatrix} M^T \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

$$\mathbf{P}(\frac{u}{2}, v) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} M S_L \; P \; M^T \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

$$P = \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \mathbf{P}_{0,3} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \mathbf{P}_{1,3} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \mathbf{P}_{2,3} \\ \mathbf{P}_{3,0} & \mathbf{P}_{3,1} & \mathbf{P}_{3,2} & \mathbf{P}_{3,3} \end{bmatrix}$$

Subdivided Patch





$$S_L = M^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix} M$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{3} & 0 & 0 \\ 1 & \frac{2}{3} & \frac{1}{3} & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{8} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{6} & 0 & 0 \\ 1 & \frac{1}{3} & \frac{1}{12} & 0 \\ 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

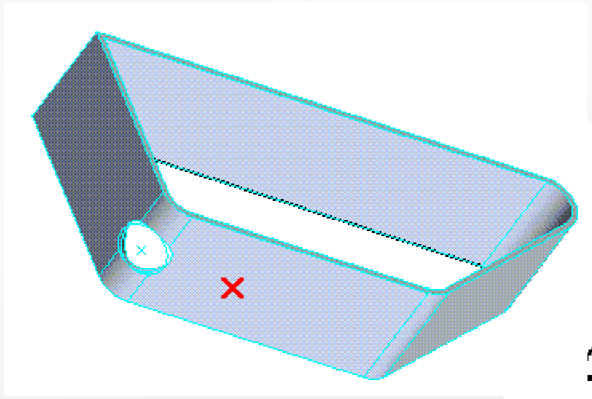
$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{bmatrix}$$

the control points of the four sub-patches can be written as

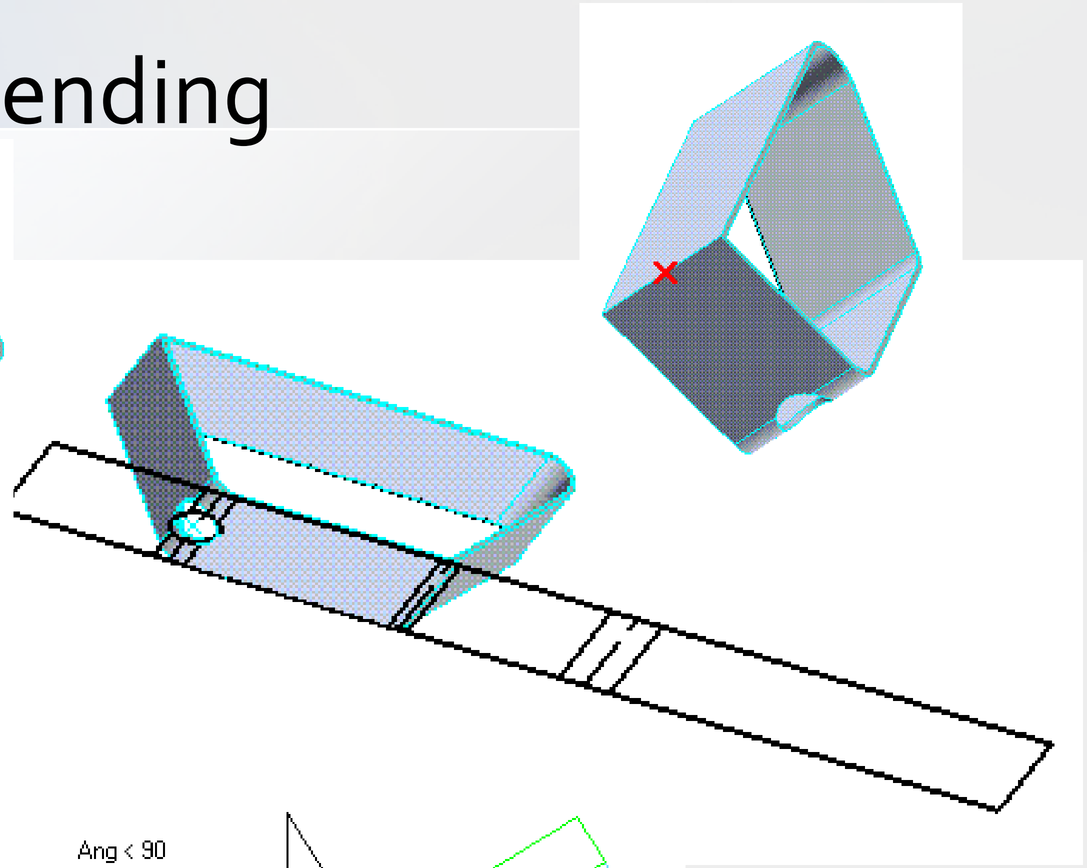
$$S_L P S_L^T$$
$$S_R P S_L^T$$
$$S_L P S_R^T$$
$$S_R P S_R^T$$

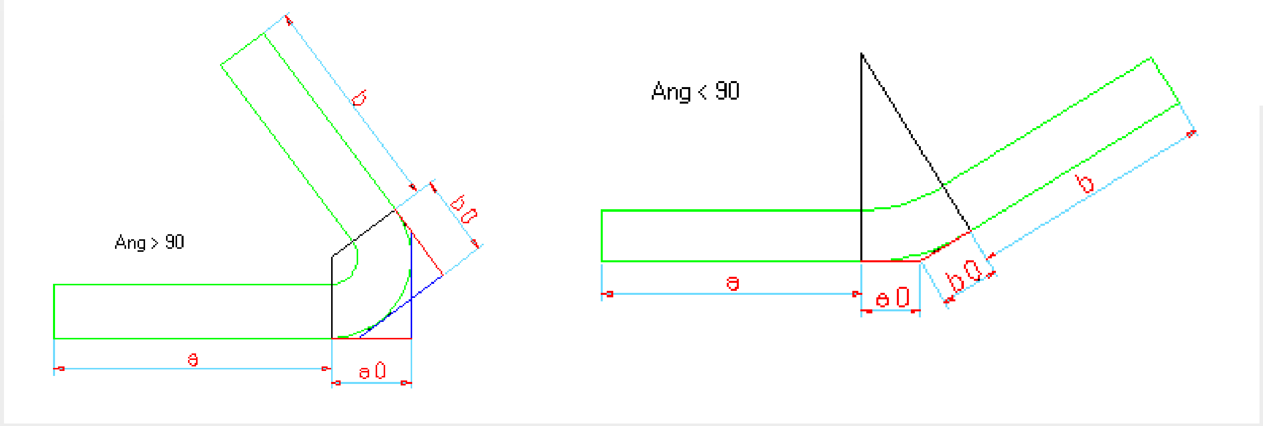
$$S_L P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \mathbf{P}_{0,2} & \mathbf{P}_{0,3} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \mathbf{P}_{1,3} \\ \mathbf{P}_{2,0} & \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \mathbf{P}_{2,3} \\ \mathbf{P}_{3,0} & \mathbf{P}_{3,1} & \mathbf{P}_{3,2} & \mathbf{P}_{3,3} \end{bmatrix}$$

Sheet metal bending

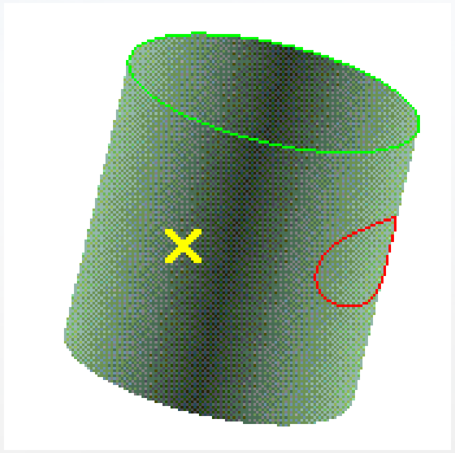
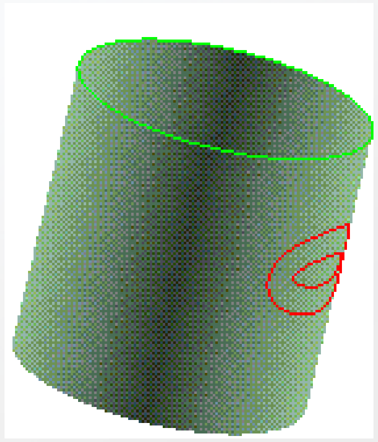
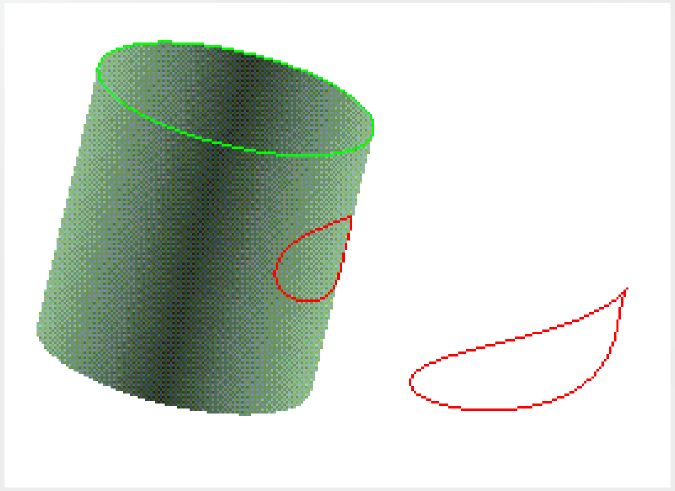


- Unfolding

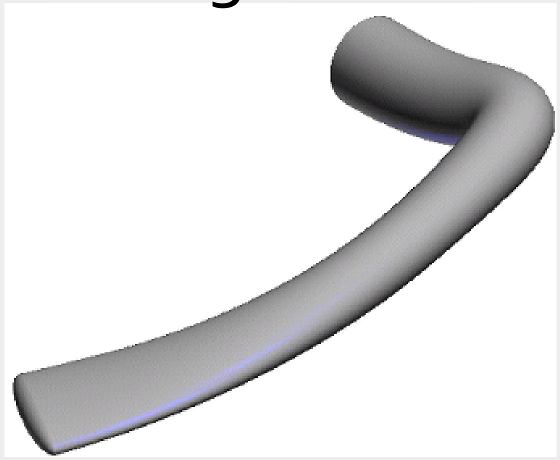
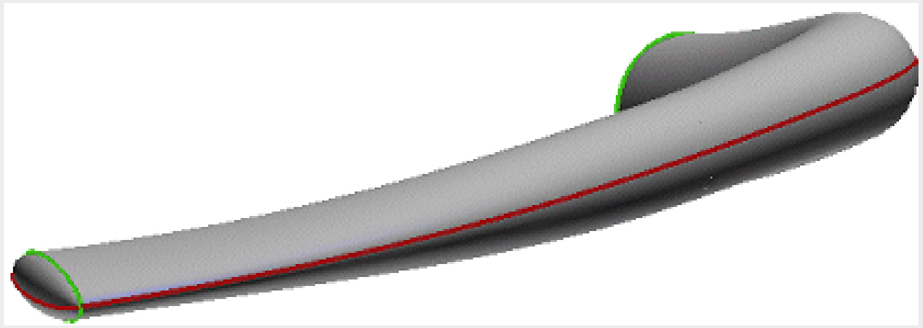




Offset curves on Surface

Molding and silhouettes

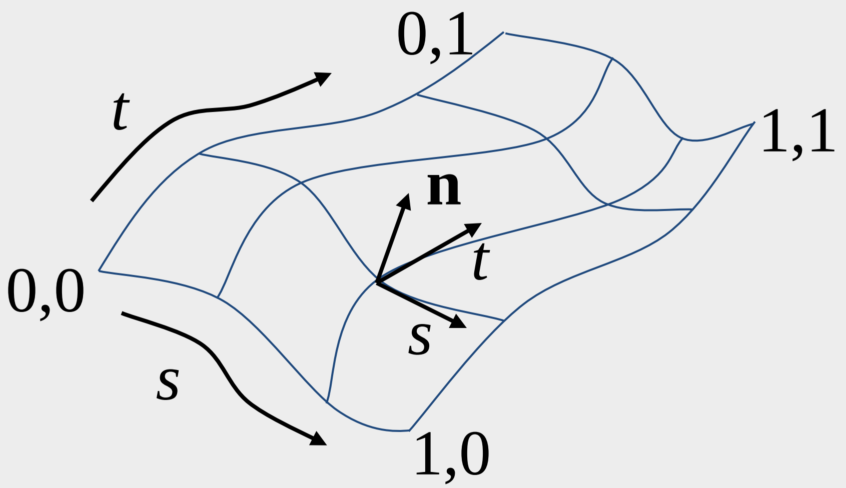



Curved Surfaces

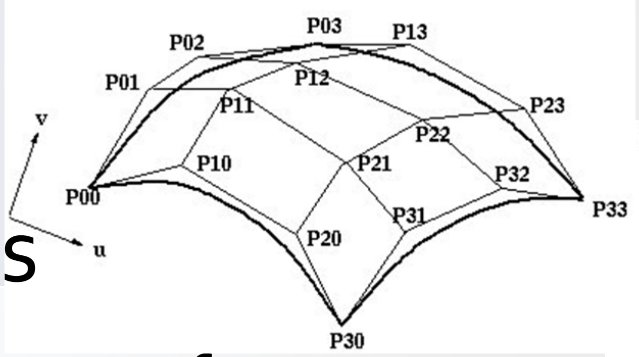
CSE167: Computer Graphics
Instructor: Steve Rotenberg
UCSD, Fall 2006

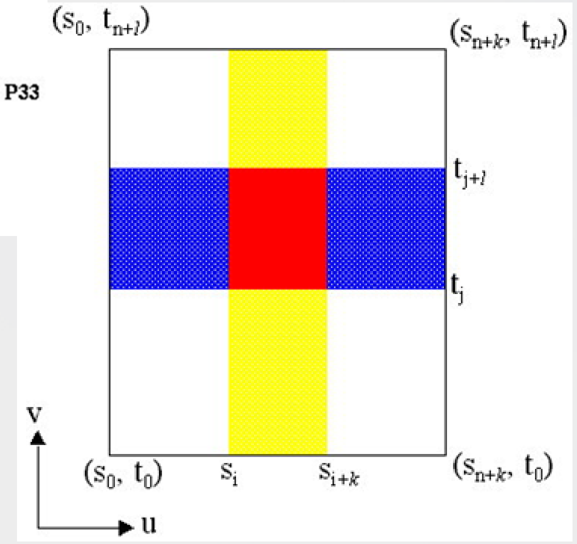
Bezier Surfaces

- Bezier surfaces are an extension to Bezier curves.
- Instead of the curve defined by a single parameter variable *t*, we use two variables, ***s*** and ***t*** for surface.
- By definition, we choose to have *s* and *t* range from **0 to 1** and we say that an ***s*-tangent crossed with** a ***t*-tangent** will represent the **normal n** for the front of the surface.



Curved Surfaces





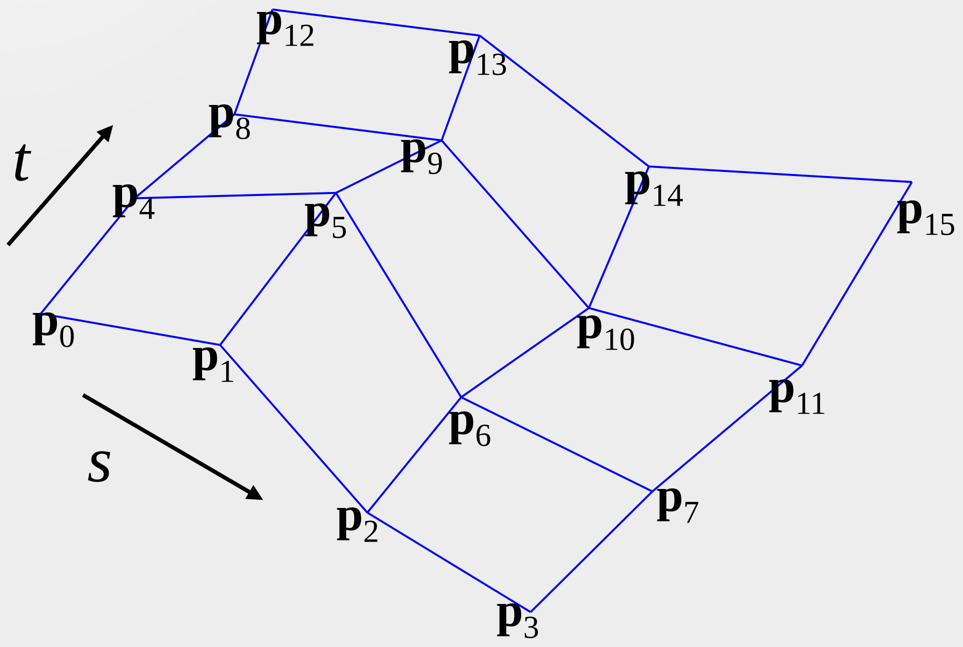
The Bezier parametric surface
is a surface that can be parameterized
by **two variables, s and t** (u and v).

Parametric surfaces have a rectangular topology.
In computer graphics, parametric surfaces are called *patches*, *curved surfaces*, or just *surfaces*.

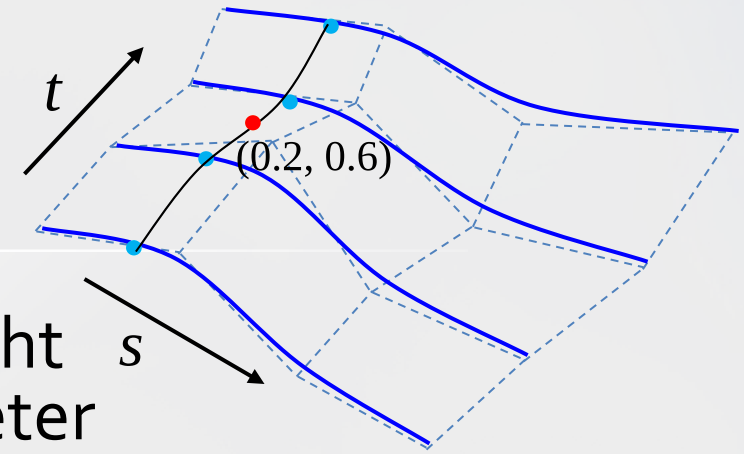
There are also some non-parametric surfaces used in computer graphics, but we won’t consider those now.

Control Mesh

Consider a *bicubic* Bezier surface (bicubic means that it is a cubic function in both the *s* and *t* parameters).
A cubic curve has 4 control points, and a bicubic surface has a grid of 4x4 control points, $\mathbf{p}_0$ through $\mathbf{p}_{15}$.



Surface Evaluation



The bicubic surface can be thought of as 4 curves along the *s* parameter (or alternately as 4 curves along the *t* parameter).

To compute the location of the surface for some (*s*,*t*) pair, we can first solve each of the 4 *s*-curves for the specified value of *s* .

Those 4 points now make up a new curve which we evaluate at *t* . Alternately, we first may solve the 4 t-curves by evaluating at *s* .

This gives a simple way to implement smooth surfaces with little more than what is needed to implement curves.

Matrix Form

- We saw the matrix form for a 3D Bezier curve is

$$\mathbf{x} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_{0x} & p_{0y} & p_{0z} \\ p_{1x} & p_{1y} & p_{1z} \\ p_{2x} & p_{2y} & p_{2z} \\ p_{3x} & p_{3y} & p_{3z} \end{bmatrix}$$

$$\mathbf{x} = \mathbf{t} \cdot \mathbf{B}_{Bez} \cdot \mathbf{G}_{Bez}$$

$$\mathbf{x} = \mathbf{t} \cdot \mathbf{C}$$

Matrix Form

- To simplify notation for surfaces, we will define a matrix equation for each of the *x*, *y*, and *z* components, instead of combining them into a single equation as for curves
- For example, to evaluate the *x* component of a Bezier curve, we can use:

$$x = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_{0x} \\ p_{1x} \\ p_{2x} \\ p_{3x} \end{bmatrix}$$

$$x = \mathbf{t} \cdot \mathbf{B}_{Bez} \cdot \mathbf{g}_x$$

$$x = \mathbf{t} \cdot \mathbf{c}_x$$

Matrix Form

To evaluate the *x* component of 4 curves simultaneously, we can combine 4 curves into a 4x4 matrix.
To evaluate a **surface**, we evaluate the 4 curves, and use them to make a new curve which is then evaluated.

This can be written in a compact matrix form:

$$\boxed{x(s,t) = \mathbf{s} \cdot \mathbf{B}_{Bez} \cdot \mathbf{G}_x \cdot \mathbf{B}_{Bez}^T \cdot \mathbf{t}^T}$$

$$\mathbf{s} = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix}$$
$$\mathbf{t} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$$

$$x(s,t) = \mathbf{s} \cdot \mathbf{B}_{Bez} \cdot \mathbf{G}_x \cdot \mathbf{B}_{Bez}^T \cdot \mathbf{t}^T$$

Matrix Form

$$\mathbf{s} = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix}$$
$$\mathbf{t} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$$

$$\boxed{\mathbf{x}(s,t) = \begin{bmatrix} \mathbf{s} \cdot \mathbf{C}_x \cdot \mathbf{t}^T \\ \mathbf{s} \cdot \mathbf{C}_y \cdot \mathbf{t}^T \\ \mathbf{s} \cdot \mathbf{C}_z \cdot \mathbf{t}^T \end{bmatrix}}$$

$$\mathbf{B}_{Bez} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = \mathbf{B}_{Bez}^T$$

$$\mathbf{C}_x = \mathbf{B}_{Bez} \cdot \mathbf{G}_x \cdot \mathbf{B}_{Bez}^T$$
$$\mathbf{s} = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix}$$
$$\mathbf{t} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$$

$$\mathbf{G}_x = \begin{bmatrix} p_{0x} & p_{4x} & p_{8x} & p_{12x} \\ p_{1x} & p_{5x} & p_{9x} & p_{13x} \\ p_{2x} & p_{6x} & p_{10x} & p_{14x} \\ p_{3x} & p_{7x} & p_{11x} & p_{15x} \end{bmatrix}$$

$$x(s,t) = \mathbf{s} \cdot \mathbf{B}_{Bez} \cdot \mathbf{G}_x \cdot \mathbf{B}_{Bez}^T \cdot \mathbf{t}^T$$

Matrix Form

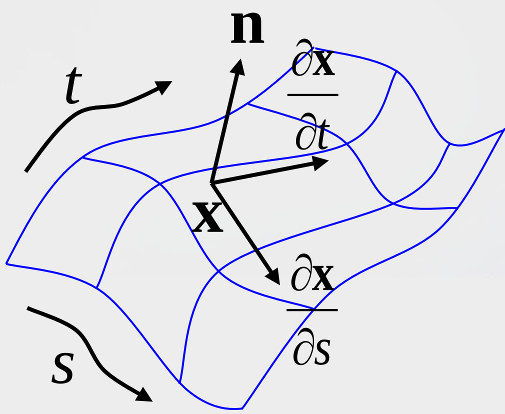
$$\mathbf{s} = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix}$$
$$\mathbf{t} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$$

$\mathbf{C}_x$ stores the coefficients of the bicubic equation for *x*
$\mathbf{G}_x$ stores the geometry (*x* components of the control points)
$\mathbf{B}_{Bez}$ is the basis matrix (Bezier basis)
s and **t** are the vectors formed from the exponents of *s* and *t*

- The matrix form is a nice and compact notation and leads to an efficient method of computation
- It can also take advantage of 4x4 matrix support which is built into modern graphics hardware

Tangents

- To compute the *s* and *t* tangent vectors at some (*s*,*t*) location, we can use:



$\mathbf{n}$
$\frac{\partial \mathbf{x}}{\partial t}$
t
$\mathbf{x}$
$\frac{\partial \mathbf{x}}{\partial s}$
s

$$\frac{\partial \mathbf{x}}{\partial s} = \begin{bmatrix} d\mathbf{s} \cdot \mathbf{C}_x \cdot \mathbf{t}^T \\ d\mathbf{s} \cdot \mathbf{C}_y \cdot \mathbf{t}^T \\ d\mathbf{s} \cdot \mathbf{C}_z \cdot \mathbf{t}^T \end{bmatrix}$$

$$\mathbf{s} = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix}$$

$$\mathbf{t} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$$

$$\frac{\partial \mathbf{x}}{\partial t} = \begin{bmatrix} \mathbf{s} \cdot \mathbf{C}_x \cdot d\mathbf{t}^T \\ \mathbf{s} \cdot \mathbf{C}_y \cdot d\mathbf{t}^T \\ \mathbf{s} \cdot \mathbf{C}_z \cdot d\mathbf{t}^T \end{bmatrix}$$

$$d\mathbf{s} = \begin{bmatrix} 3s^2 & 2s & 1 & 0 \end{bmatrix}$$

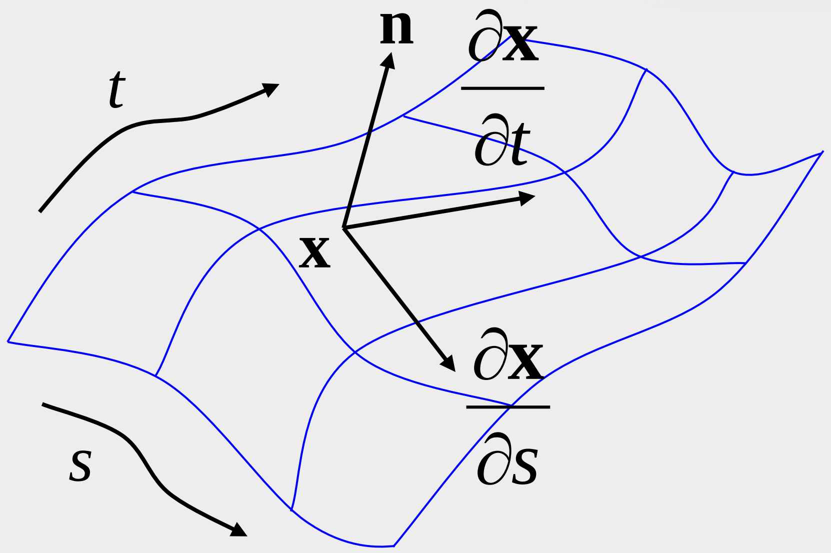
$$d\mathbf{t} = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix}$$

Normals

- To compute the normal of the surface at some location (*s*,*t*), we compute the two tangents at that location and then take their cross product
- Usually, it is normalized as well

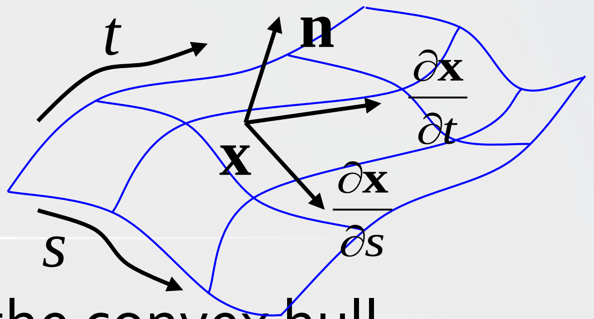
$$\mathbf{n}^* = \frac{\partial \mathbf{x}}{\partial s} \times \frac{\partial \mathbf{x}}{\partial t}$$

$$\mathbf{n} = \frac{\mathbf{n}^*}{\left|\mathbf{n}^*\right|}$$



$\mathbf{n}$
$\frac{\partial \mathbf{x}}{\partial t}$
t
$\mathbf{x}$
$\frac{\partial \mathbf{x}}{\partial s}$
s

Bezier Surface Properties



t
$\mathbf{n}$
$\frac{\partial \mathbf{x}}{\partial t}$
$\mathbf{x}$
$\frac{\partial \mathbf{x}}{\partial s}$
s

- Like Bezier curves, Bezier surfaces retain the convex hull property, so that any point on the actual surface will fall **within the convex hull** of the control points

- With Bezier curves, the curve will **interpolate (pass through) the first and last control points**, but will only approximate the other control points

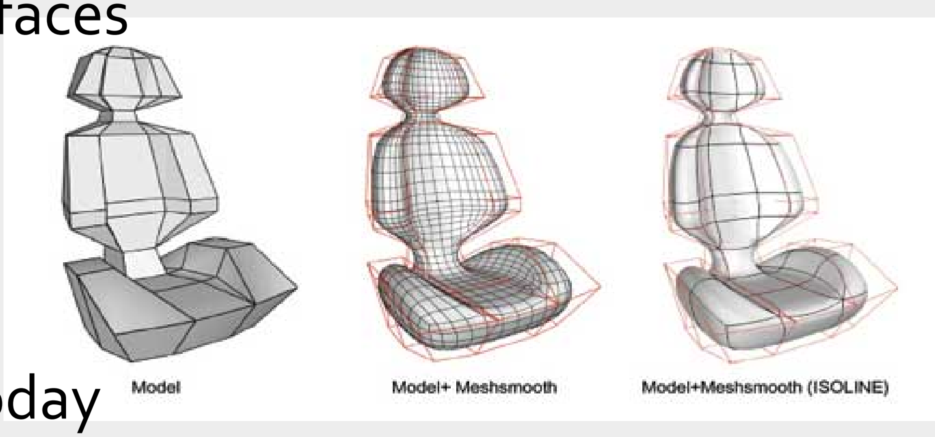
- With Bezier surfaces, the 4 corners will interpolate, and the other 12 points in the control mesh are **only approximated**

- The **4 boundaries of the Bezier surface** are just Bezier curves defined by the points on the edges of the surface

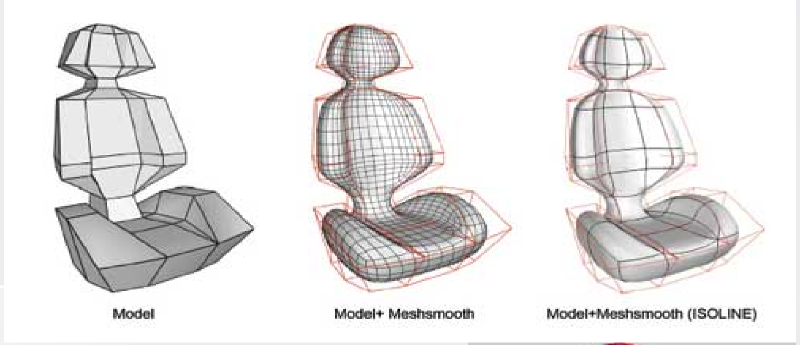
- By **matching these** points, two Bezier surfaces can be **connected precisely**

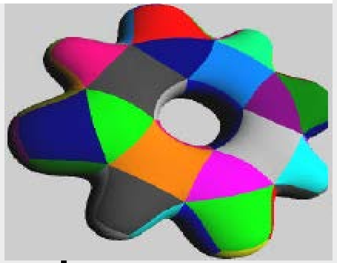
Tessellation

- Tessellation is the process of taking a complex surface (like a bicubic patch) and **approximating** it with a set of simpler surfaces (**like triangles**)
- In computer graphics, there are a lot of different types of complex surfaces one might want to tessellate, such as:
  - Parametric surfaces (such as Bezier surfaces)
  - Displacement mapped surfaces
  - Subdivision surfaces
  - Fractals
  - Procedural models
  - Implicit surfaces
- We will look at the first two today



Uniform Tessellation





- The most straightforward way to tessellate a parametric surface is *uniform tessellation*
- With this method, we simply choose some resolution in *s* and *t* and uniformly divide up the surface like a grid
- This method is very efficient to compute, as the cost of evaluating the surface reduces to approximately the same cost as evaluating a curve
- However, as the generated mesh is uniform, it may have more triangles than it needs in flatter areas and fewer than it needs in highly curved areas

Adaptive Tessellation

- Very often, the goal of a tessellation is to provide the fewest triangles necessary to accurately represent the original surface
- For a curved surface, this means that we want more triangles in areas where the curvature is high, and fewer triangles in areas where the curvature is low
- We may also want more triangles in areas that are closer to the camera, and fewer farther away
- *Adaptive tessellation* schemes are designed to address these requirements

Mixed Tessellation

- Some practical renderers use a mixed tessellation scheme
- First, the original surface patch is adaptively subdivided into several *subpatches*, each approximately the same size (say around 10 pixels on a side)
- Then, each of the subpatches (which is just a rectangular *s*,*t* range within the larger 0,1 rectangle) is uniformly tessellated to some size (say 10 x 10)
- The result is that the curved surface is tessellated into triangles roughly the size of a single pixel
- The bulk of the cost of the algorithm is in the uniform tessellation, which can be implemented in a very efficient way

Displacement Mapping

- To add additional geometric detail to a tessellated surface, we can use *displacement mapping*
- With this technique, a displacement map is stored, which is much like a texture map that stores a height value per texel instead of a color
- As with texture mapping, we can assign a texture coordinate to each corner of the patch that allows us to position the displacement map onto the surface
- This coordinate gets interpolated when we evaluate the position and normal of the patch for some (s,t) value
- We can displace the position by the height value. The displacement is usually done along the computed patch normal
- Once we’ve displaced our tessellated triangle mesh, we will need to recompute accurate normals, as they will change based on the displacements
- To avoid geometry aliasing, we should really perform some sort of filtering on the height value (such as mipmapping)

Scan Conversion

- The serial scan conversion technique we looked at earlier in the quarter requires expensive set-up computations in order to make the per-pixel cost very low, thus making it efficient for large triangles
- Some surface renderers generate triangles smaller than a single pixel, or the size of a few subpixels in an antialiased rendering
- For triangles this small, it is usually better to use different approaches in the scan conversion process
- Also, as these *micropolygons* are so usually generated from uniform tessellations, other optimizations can be made to account for all of the shared vertices and edges between them

Other Curve Types

- Curves
  - Hermite curves
  - Catmull-Rom curves
  - B-Splines
  - NURBS
- Surfaces
  - B-Spline / NURBS
  - Trim curves
  - Subdivision surfaces
  - Implicit surfaces

CAD/CAE/CAM

Part 5: Representation and Manipulation of Surfaces

What Is It?

The ways and means to define and manipulate 3D surfaces, used by geometric modeling systems to store surfaces such as faces in a B-Rep.

Types of Surface Equations

- Implicit
  Describe a surface by equations relating to the X, Y, Z coordinates.
  Advantages:
  Compact; Easy to check if a point belongs to the surface.
  Disadvantages:
  - Difficult for surface evaluation.
  - Difficult for partial surface definition (1/4 of a sphere).

- Parametric
  Represent the X, Y, Z coords as a function of ***two*** parameter.
  Advantages:
  - Easy for surface evaluation.
  - Convenient for partial surface definition.
  - Many others such as easy for manipulation.

Parametric Surfaces

$S(u,v) = [x(u,v), y(u,v), z(u,v)]^T$

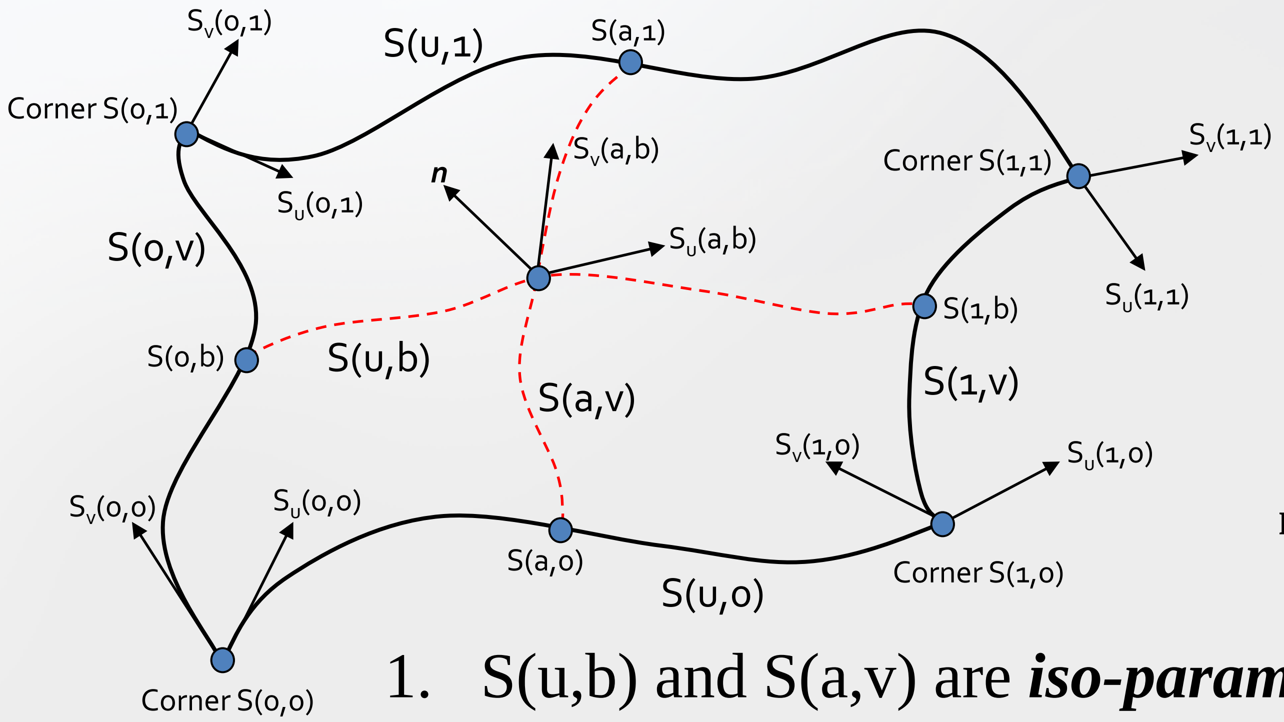
with $u_{\text{min}} < u < u_{\text{max}}$ and $v_{\text{min}} < v < v_{\text{max}}$

In most surfaces, the intervals for u and v are [0,1].
Surfaces can be modeled by a group of surface patches.
A surface patch has the following boundary conditions:

- 4 corner vectors – $\mathbf{S}(0,0)$, $\mathbf{S}(0,1)$, $\mathbf{S}(1,0)$, $\mathbf{S}(1,1)$
- 8 tangent vectors – 2 at each corner, $\mathbf{S}_u(u,v)$, $\mathbf{S}_v(u,v)$
- 4 twist vectors at the corners - $\mathbf{S}_{uv}(u,v)$
- 4 boundary curves – $u = 0$, $u = 1$, $v = 0$, $v = 1$.

Parametric Surfaces

Basic Terminologies of Parametric Surface S(u,v)



$$S_u(u,v) = \left.\frac{\partial S}{\partial u}\right|_{(u,v)}$$

$$S_v(u,v) = \left.\frac{\partial S}{\partial u}\right|_{(u,v)}$$

$$\mathbf{n} = \frac{S_u(a,b)\times S_v(a,b)}{\left\|S_u(a,b)\times S_v(a,b)\right\|}$$

1. S(u,b) and S(a,v) are ***iso-parametric*** curves at v=b and u=a respectively.
2. **n** is the unit normal vector at (u,v) = (a,b).

Classification of Surfaces

- Bi-linear Patch
- Ruled Patch
- Coons Patch
- Bicubic Patch
- Hermite Patch
- Coons Patch with tangents
- Bezier Patch
- B-Spline Patch
- Non-Uniform Rational B-Splines (NURBS)

Bi-linear Patch

The simplest surface defined by 4 points in space

- Input

Four points $P_{0,0}$, $P_{0,1}$, $P_{1,0}$, $P_{1,1}$

- Output

A surface S(u,v) with four corners
S(0,0), S(0,1), S(1,0), and S(1,1) at the four given points

- Definition
- Each of the X, Y, and Z components is a bi-linear function of u and v

- Example

Ruled Surface

The simplest surface defined by two curves

- Input

Two curves $Q_0(u)$ and $Q_1(u)$ $(0 \leq u \leq 1)$

- Output

A surface $S(u,v)$ with its two boundary curves $S(u,0)$ and $S(u,1)$ identical to $Q_0(u)$ and $Q_1(u)$ respectively.

- Definition

- Example

Coons Patch

The simplest surface defined by four curves $Q_0(u)$, $Q_1(u)$, $P_0(v)$, $P_1(v)$

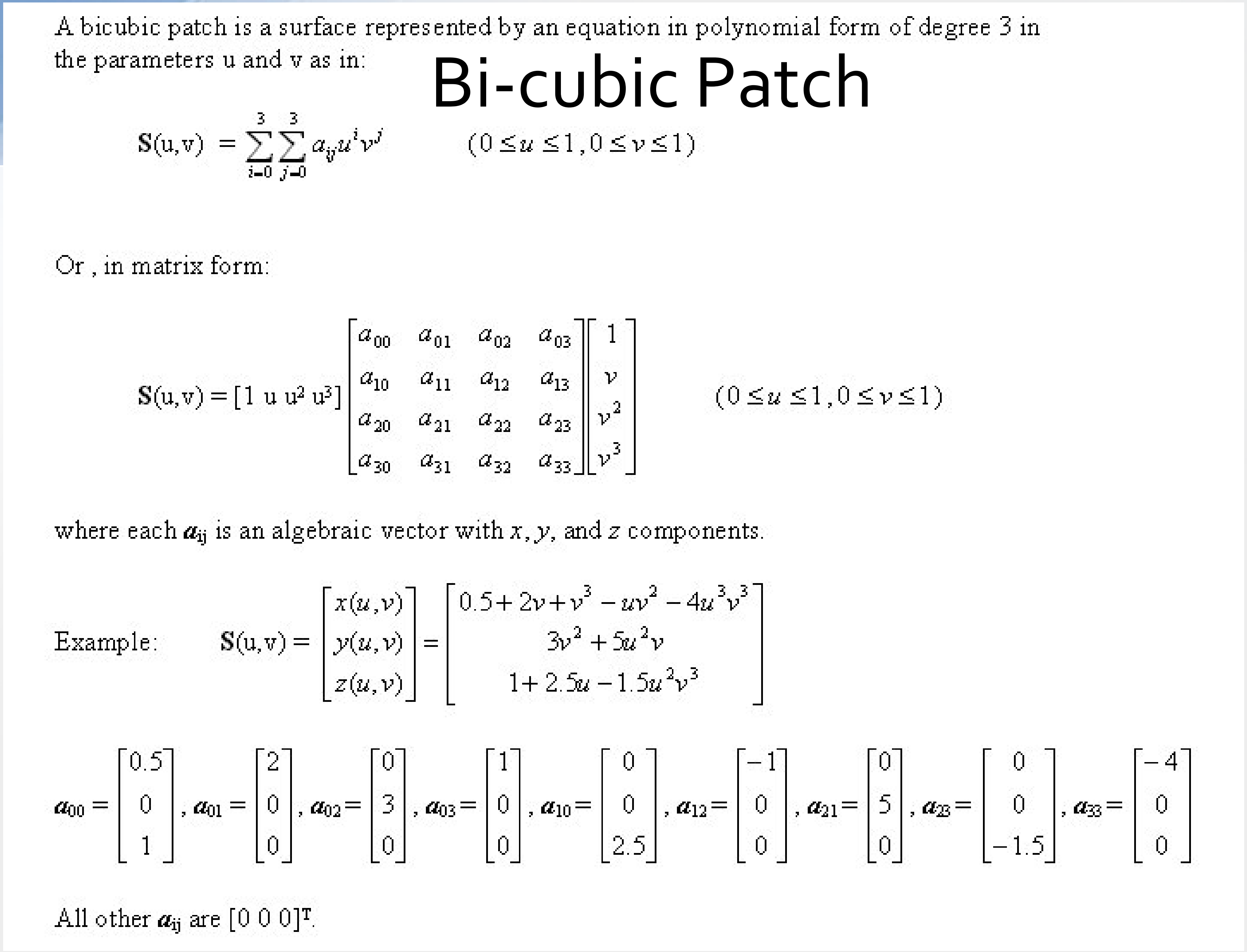
- Input and Output

- Definition
  Step 1. Define a ruled surface $S_1(u,v)$ on two opposite curves
  Step 2. Define a ruled surface $S_2(u,v)$ on the other two opposite curves
  Step 3. Add two together and find the compensating patch

$$S(u,v) = \begin{bmatrix} (1-u) & u \end{bmatrix} \begin{bmatrix} P_0(v) \\ P_1(v) \end{bmatrix} + \begin{bmatrix} Q_0(u) & Q_1(u) \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix} - \begin{bmatrix} (1-u) & u \end{bmatrix} \begin{bmatrix} Q_0(0) & Q_1(0) \\ Q_0(1) & Q_1(1) \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}$$

- Generalization
- Examples

Bi-cubic Patch



A bicubic patch is a surface represented by an equation in polynomial form of degree 3 in the parameters u and v as in:

$$\mathbf{S}(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}u^i v^j \qquad (0 \le u \le 1, 0 \le v \le 1)$$

Or , in matrix form:

$$\mathbf{S}(u,v) = [1\ u\ u^2\ u^3]\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix} \qquad (0 \le u \le 1, 0 \le v \le 1)$$

where each $\boldsymbol{a}_{ij}$ is an algebraic vector with x, y, and z components.

Example: $$\mathbf{S}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} = \begin{bmatrix} 0.5 + 2v + v^3 - uv^2 - 4u^3v^3 \\ 3v^2 + 5u^2v \\ 1 + 2.5u - 1.5u^2v^3 \end{bmatrix}$$

$$\boldsymbol{a}_{00} = \begin{bmatrix} 0.5 \\ 0 \\ 1 \end{bmatrix}, \boldsymbol{a}_{01} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{a}_{02} = \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix}, \boldsymbol{a}_{03} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{a}_{10} = \begin{bmatrix} 0 \\ 0 \\ 2.5 \end{bmatrix}, \boldsymbol{a}_{12} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{a}_{21} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}, \boldsymbol{a}_{23} = \begin{bmatrix} 0 \\ 0 \\ -1.5 \end{bmatrix}, \boldsymbol{a}_{33} = \begin{bmatrix} -4 \\ 0 \\ 0 \end{bmatrix}$$

All other $\boldsymbol{a}_{ij}$ are $[0\ 0\ 0]^T$.

Hermite Patch

Similar to Hermite curve, non-intuitive algebraic coefficients $\boldsymbol{a}_{ij}$ need to be replaced by geometric coefficients like corner points and tangents. There are 16 unknowns $\boldsymbol{a}_{ij}$, so we need 16 boundary vectors in order to find them.

- 12 intuitive vectors
- 4 more boundary vectors (twist vectors)
  (usually set to zero vectors if difficult to decide)
- Compute the Hermite form
  1. Compute the derivatives of the bicubic surface
  2. Plug in the 16 boundary vectors and solve the linear equations for $\boldsymbol{a}_{ij}$
  3. Rearrange into the Hermite Form
- Major properties
  – Boundary curves are Hermite
  – Iso-parametric curves are Hermite
- Examples

Drawbacks of Hermite Patch

It is not easy and not intuitive to predict surface shape according to changes in magnitude of the tangents (partial derivatives) at the four corners.

In addition, the four cross-derivatives
$\mathbf{S}_{uv}(0, 0)$, $\mathbf{S}_{uv}(0, 1)$ and $\mathbf{S}_{uv}(1, 0)$
most of time are not known.

Bezier Surface

- Definition

$$\mathbf{S}(u, v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u) B_{j,m}(v) \mathbf{P}_{ij} \qquad (0 \le u \le 1, 0 \le v \le 1)$$

$\mathbf{P}_{ij}$: the control points that form a (n+1) by (m+1) control mesh

n: the degree of the surface in u direction

m: the degree of the surface in v direction

- Examples

  a 5x6 patch; a closed patch

- Properties
- How to evaluate

Drawbacks of Bezier Surface

- High degree
  The degree is determined by the number of control points which tend to be large for complicated surfaces. This causes oscillation as well as increases the computation burden.

- Non-local modification control
  When modifying a control point, the designer wants to see the shape change locally around the moved control point. In Bezier patch case, moving a control point affects the shape of the entire surface, and thus the portions on the surface not intended to change.

- Intractable linear equations
  If we are interested in interpolation rather than just approximating a shape, we will have to compute control points from points on the surface. This leads to systems of linear equations, and solving such systems can be impractical when the *degree* of the surface is large.

B-Spline Surface

- Definition

$$\mathbf{S}(\mathrm{u}, \mathrm{v}) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,k}(u) N_{j,l}(v)\, \mathbf{P}_{\mathrm{ij}} \qquad (s_{k-1} \le u \le s_{n+1},\ \ t_{l-1} \le v \le t_{m+1})$$

  P$_{\mathrm{ij}}$: the control points that form a (n+1) by (m+1) control mesh

  k: the order of the basis functions in u direction

  l: the order of the basis functions in v direction

  U = $\{s_0, \ldots, s_{n+k}\}$ is the knots vector in u-direction

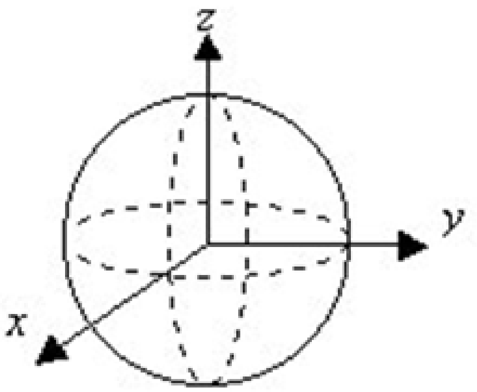
  V = $\{t_0, \ldots, t_{n+l}\}$ is the knots vector in v-direction

- Examples

  a 6 by 6 patch, a B-Spline Editor

- Properties
  - Local-modification property
  - Degrees *k* and *l* independent of n and m

Implicit Surface Representations



Implicit representation

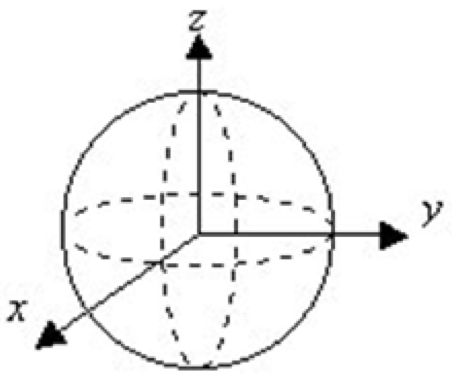
$x^2 + y^2 + z^2 - R^2 = 0$

Problems:

- Surface evaluation. Say, to display this sohere on the screen, we need to approximate it by 1000 triangles, equally sized. Hard to do.

- Partial surface. To define an octant of the sphere, not the entire sphere which is closed. Difficult.

Parametric Surface Representations



Parametric representation

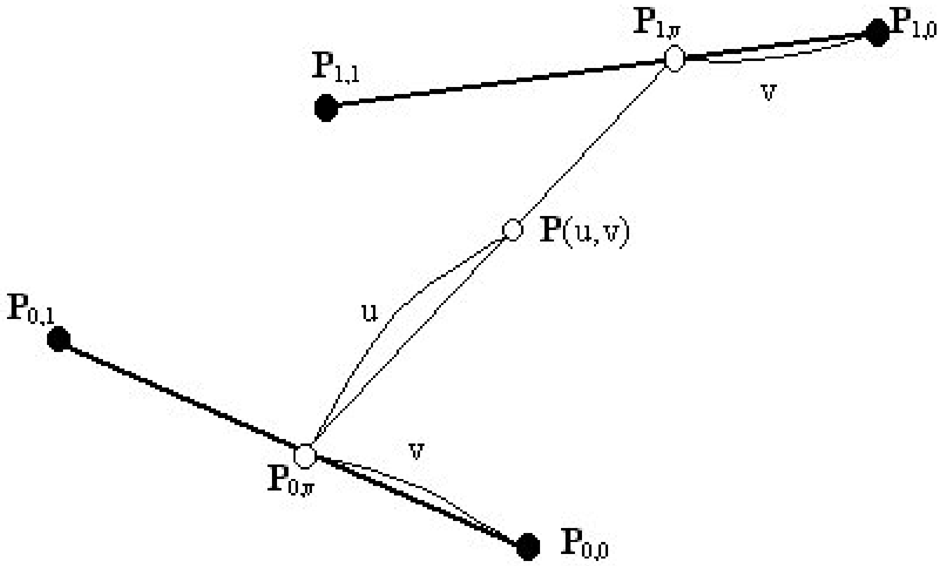
$x = R\cos\theta\sin\phi, \quad y = R\sin\theta\sin\phi, \quad z = R\cos\phi \qquad (0 \le \theta \le 2\pi)$ and $(0 \le \phi \le \pi)$.

- To compute 10000 equally spaced points on the sphere? Just evaluate the above equations at $(\theta_i, \phi_j) = ((i/1000)*2\pi, (j/1000)*2\pi)$, $i = 0, 1, 2, \ldots, 99$, $j = 1, 2, \ldots, 99$.
- To represent the sphere in the first octant? Just limit the range of (θ, ϕ) to $(0 \le \theta \le 0.5\pi, 0 \le \phi \le 0.5\pi)$.

Bi-Linear Surface

(The simplest surface defined by 4 points)

1. Define two linear boundary curves along u = 0 and u = 1



$P_{0,\bar{v}} = (1-v)\ P_{0,0} + vP_{0,1}$

$P_{1,\bar{v}} = (1-v)\ P_{1,0} + vP_{1,1}$

2. Define a linear curve along u direction between $P_{0,\bar{v}}$ and $P_{1,\bar{v}}$.

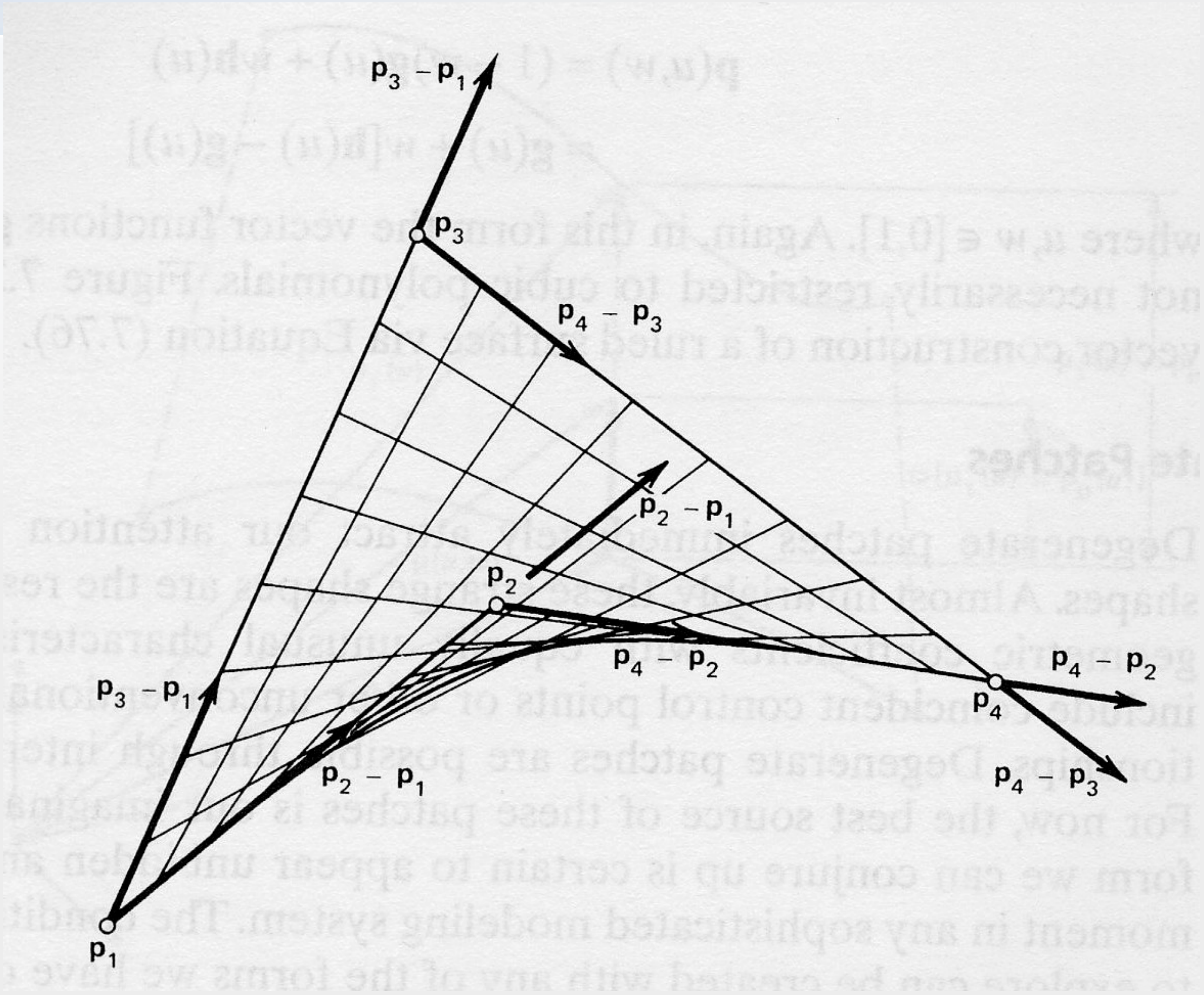
$$P(u, v) = (1-u)\ P_{0,\bar{v}} + uP_{1,\bar{v}}$$

3. Merge 1 and 2 together.

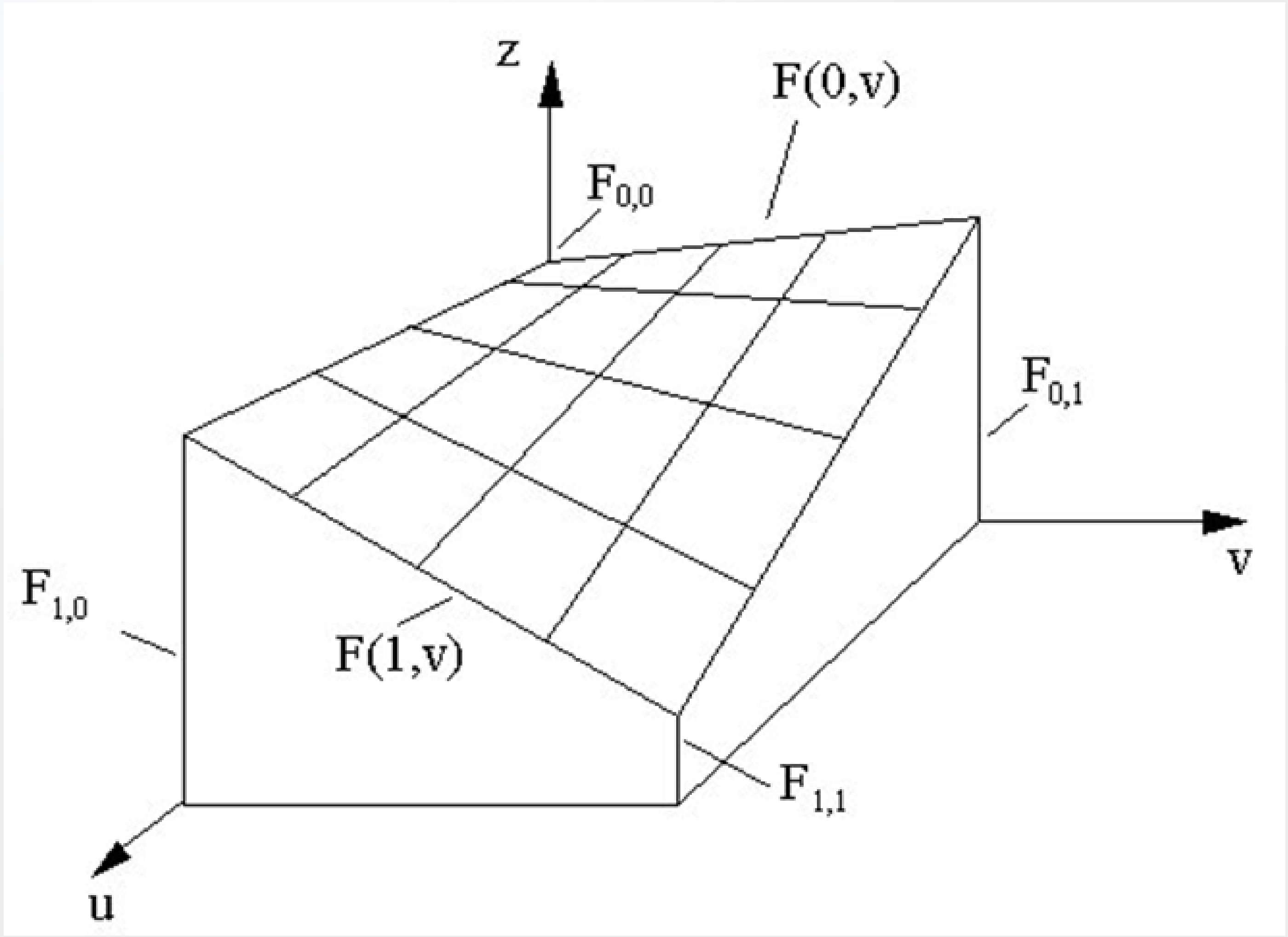
$$P(u, v) = (1-u)(1-v)P_{0,0} + u(1-v)P_{1,0} + (1-u)vP_{0,1} + uvP_{1,1}$$

$$= \boldsymbol{BL}(u,v)\ (P_{0,0}, P_{1,0}, P_{0,1}, P_{1,1})$$

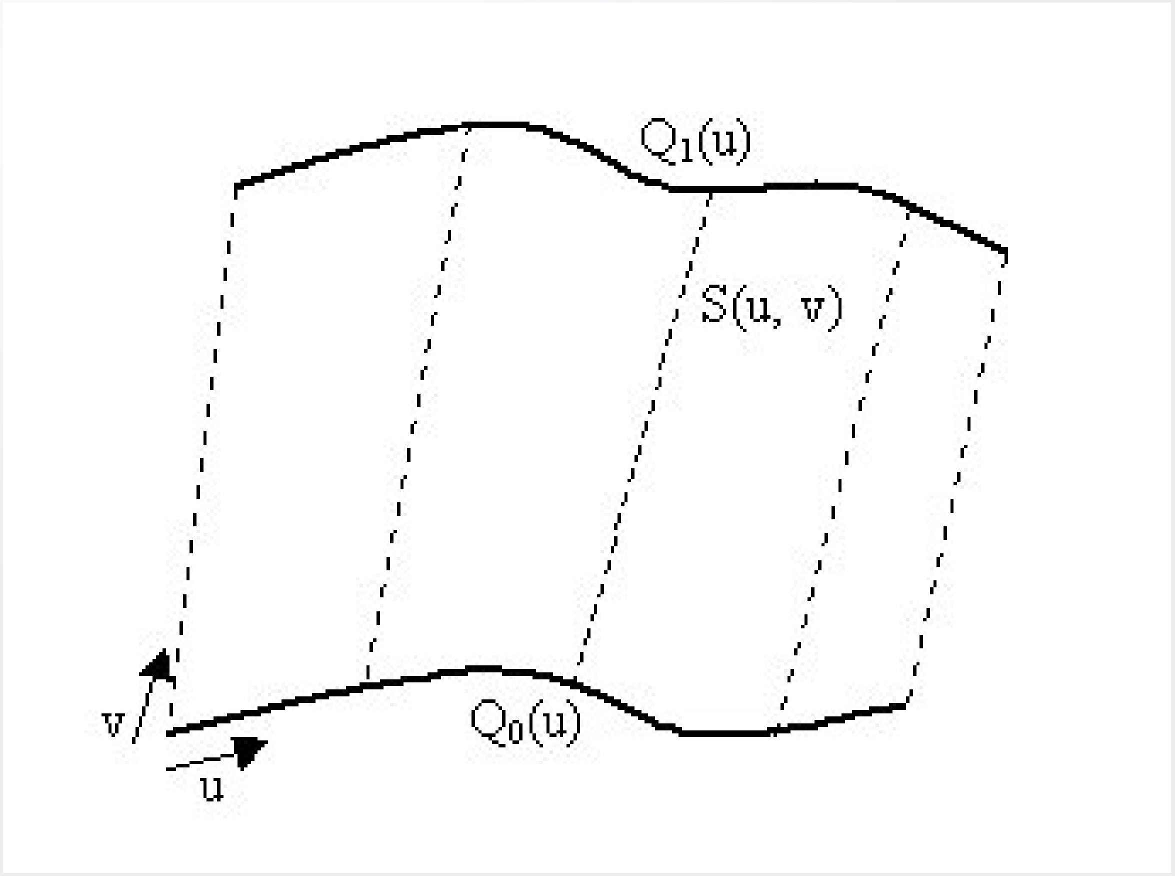
Example of a Bi-linear Surface



Bi-linear Function

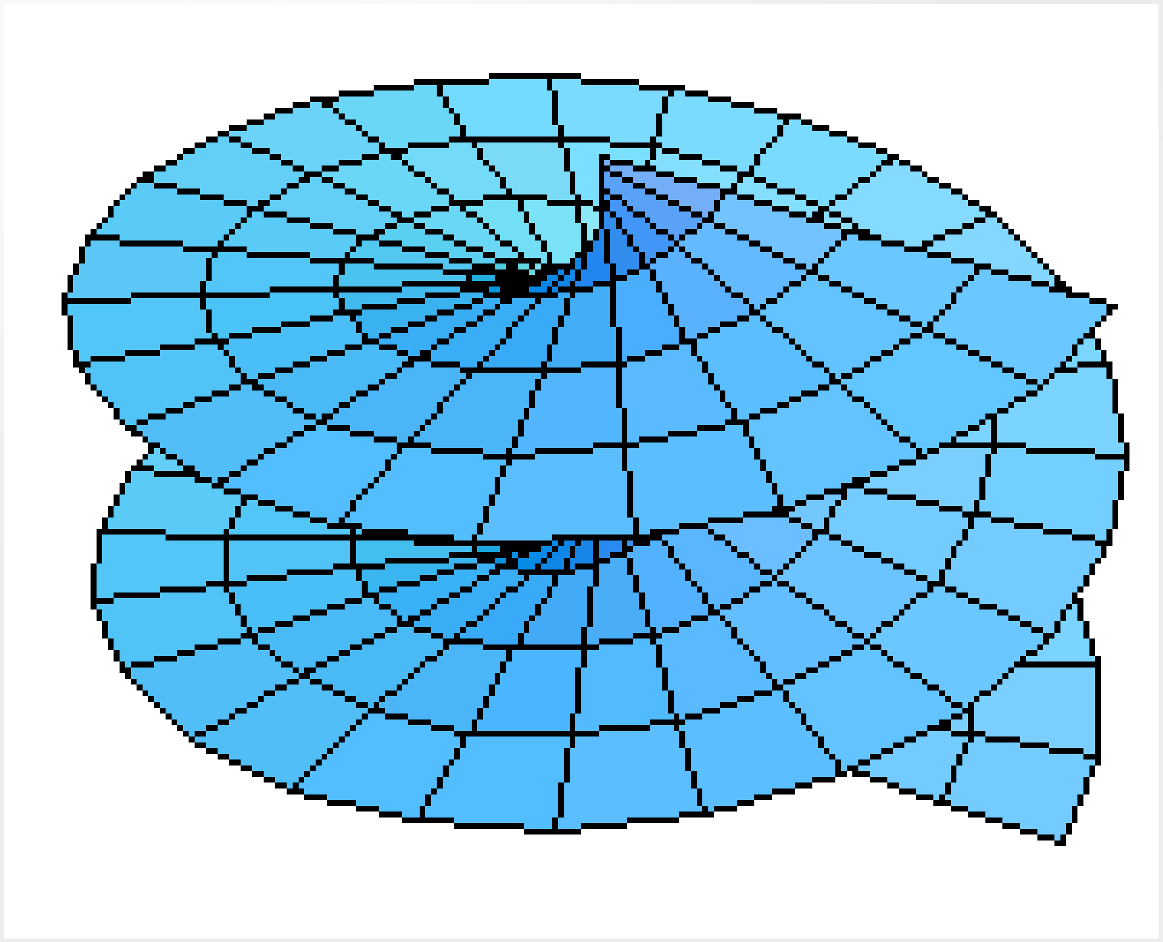


Ruled Surface

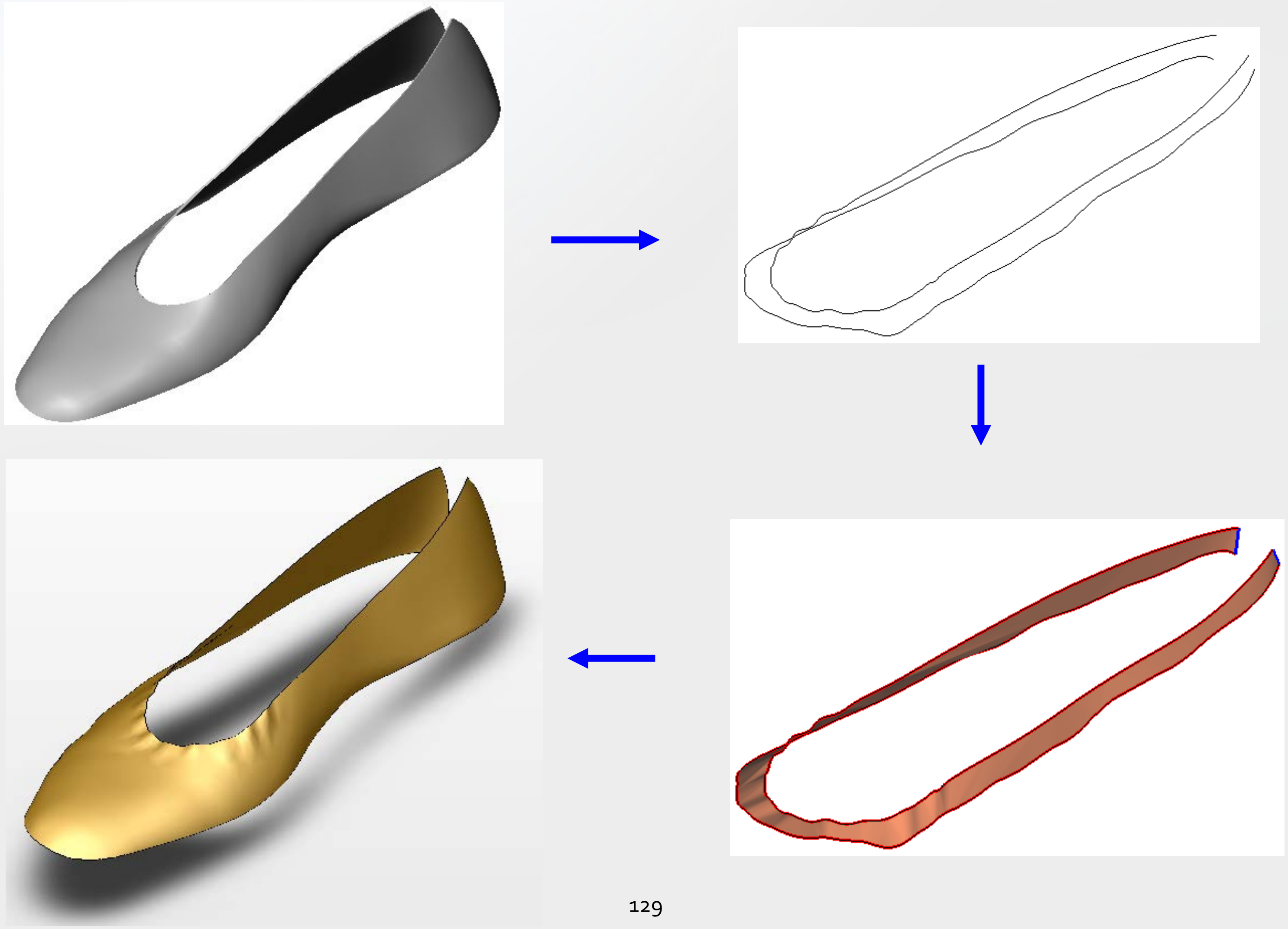


$$S(u,v) = (1-v)\bullet Q_0(u) + v\bullet Q_1(u)$$

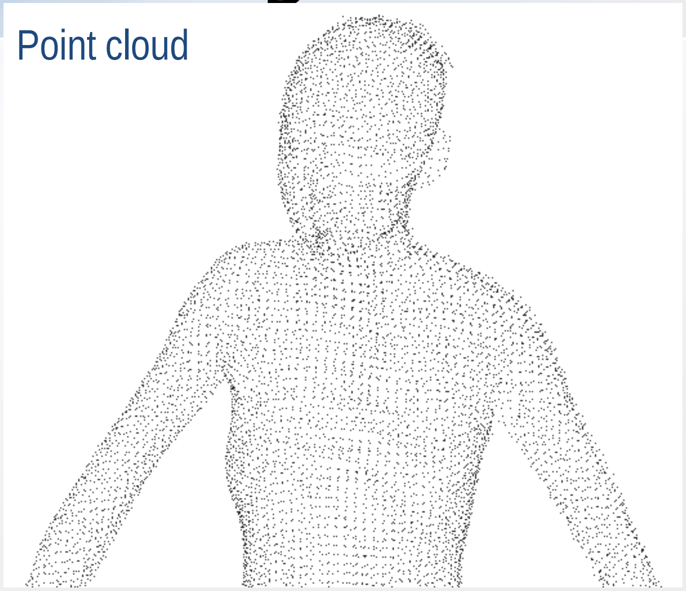
Example of a Ruled Surface (Helicoid)

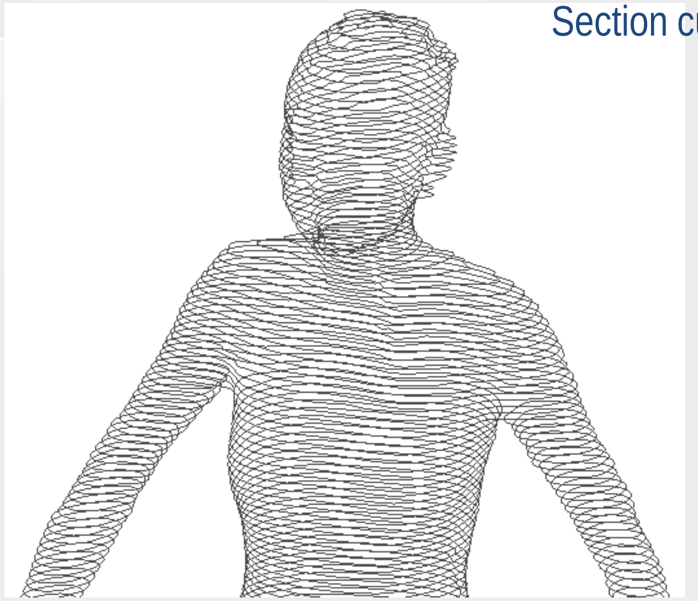


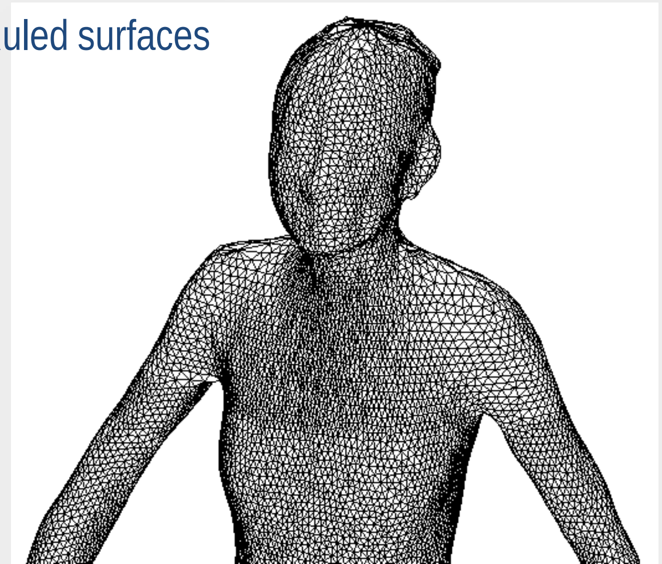
Example of a Ruled Surface (shoe design)

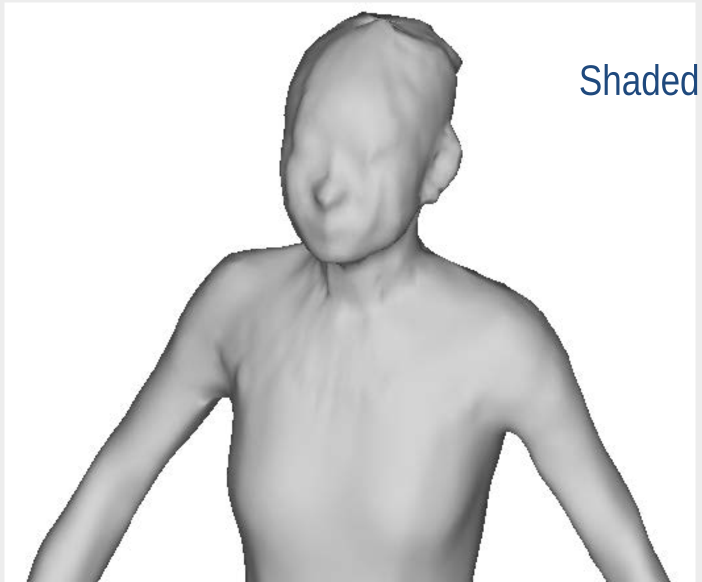


Example of a Ruled Surface (geometric reconstruction)





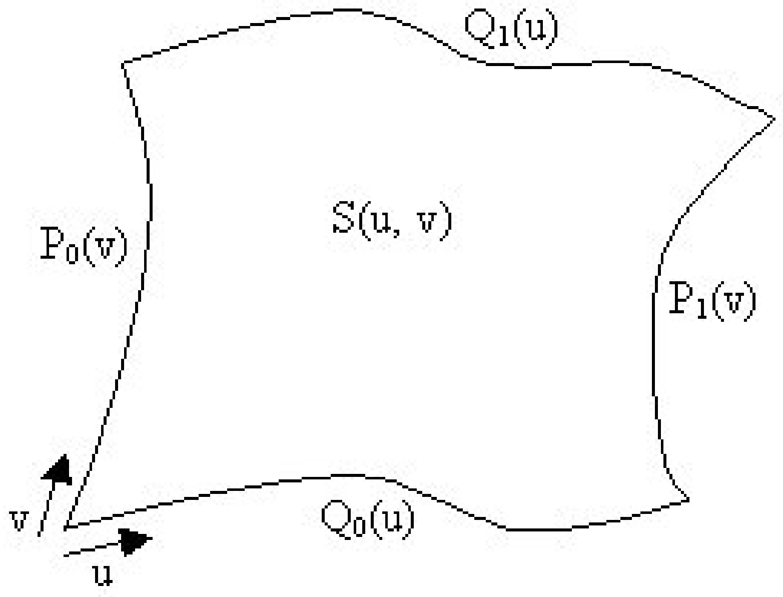




Coons Patch

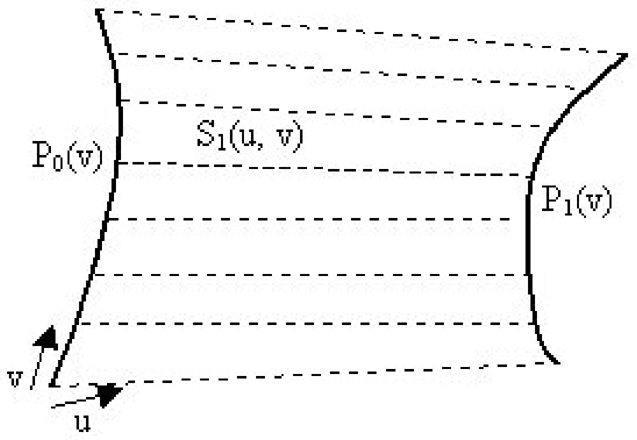
(The simplest surface defined by 4 curves)

Given four boundary curves $\mathbf{Q}_0(u)$, $\mathbf{Q}_1(u)$, $\mathbf{P}_0(v)$ and $\mathbf{P}_1(v)$ on which a patch $S(u, v)$ needs to be defined, such that the four boundary curves $S(u,0)$, $S(u,1)$, $S(0,v)$, and $S(1,v)$ are identical to the four curves $\mathbf{Q}_0(u)$, $\mathbf{Q}_1(u)$, $\mathbf{P}_0(v)$ and $\mathbf{P}_1(v)$ respectively. Bi-linear surface can be viewed as a special case of this in which the four input curves are all linear.



Defining a Coons Patch: Step 1 and Step 2

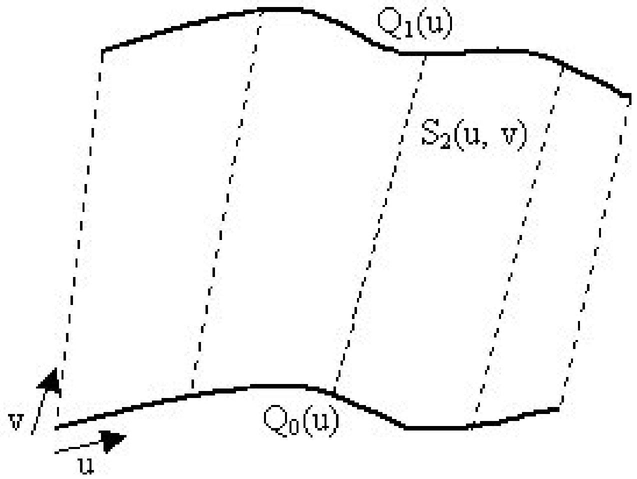
1. Define a ruled surface along u direction between $\mathbf{P}_0(v)$ and $\mathbf{P}_1(v)$.



$\mathbf{S}_1(u,v) = (1-u)\mathbf{P}_0(v) + u\mathbf{P}_1(v)$

This surface is bounded by $\mathbf{P}_0(v)$ at $u = 0$ and $\mathbf{P}_1(v)$ at $u = 1$, as desired. However, the other pairs of boundary curves will be straight line segments, not the desired $\mathbf{Q}_0(u)$ and $\mathbf{Q}_1(u)$.

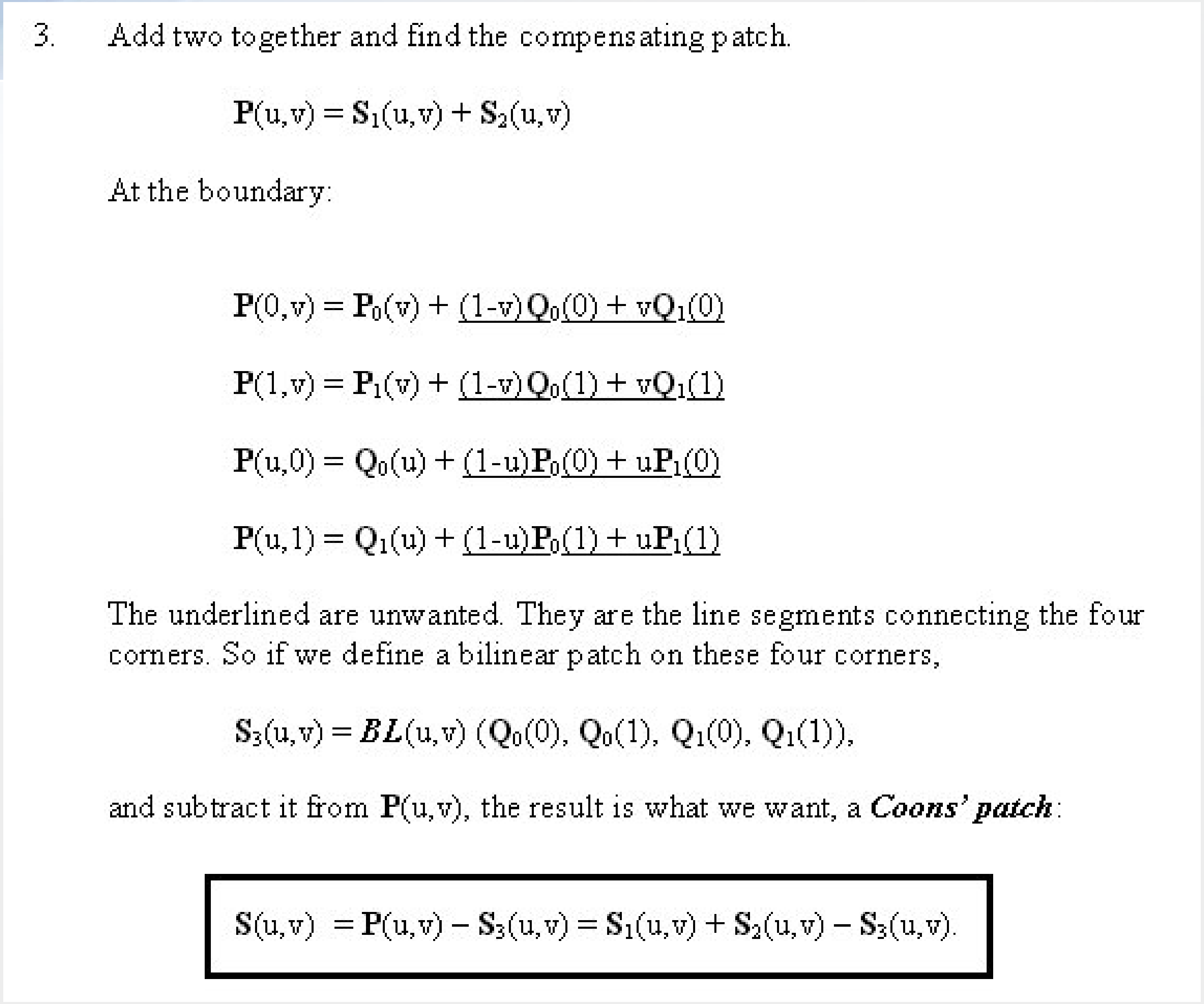
2. Define a ruled surface along v direction between $\mathbf{Q}_0(u)$ and $\mathbf{Q}_1(u)$.



$\mathbf{S}_2(u,v) = (1-v)\mathbf{Q}_0(u) + v\mathbf{Q}_1(u)$

This surface is bounded by $\mathbf{Q}_0(u)$ at $v = 0$ and $\mathbf{Q}_1(u)$ at $v = 1$, as desired. However, the other pairs of boundary curves will be straight line segments, not the desired $\mathbf{P}_0(v)$ and $\mathbf{P}_1(v)$.

Defining a Coons Patch: Step 3



3. Add two together and find the compensating patch.

$$\mathbf{P}(u,v) = \mathbf{S}_1(u,v) + \mathbf{S}_2(u,v)$$

At the boundary:

$$\mathbf{P}(0,v) = \mathbf{P}_0(v) + \underline{(1-v)\mathbf{Q}_0(0) + v\mathbf{Q}_1(0)}$$

$$\mathbf{P}(1,v) = \mathbf{P}_1(v) + \underline{(1-v)\mathbf{Q}_0(1) + v\mathbf{Q}_1(1)}$$

$$\mathbf{P}(u,0) = \mathbf{Q}_0(u) + \underline{(1-u)\mathbf{P}_0(0) + u\mathbf{P}_1(0)}$$

$$\mathbf{P}(u,1) = \mathbf{Q}_1(u) + \underline{(1-u)\mathbf{P}_0(1) + u\mathbf{P}_1(1)}$$

The underlined are unwanted. They are the line segments connecting the four corners. So if we define a bilinear patch on these four corners,

$$\mathbf{S}_3(u,v) = \boldsymbol{BL}(u,v)\ (\mathbf{Q}_0(0),\ \mathbf{Q}_0(1),\ \mathbf{Q}_1(0),\ \mathbf{Q}_1(1)),$$

and subtract it from $\mathbf{P}(u,v)$, the result is what we want, a ***Coons' patch***:

$$\boxed{\mathbf{S}(u,v) \ = \mathbf{P}(u,v) - \mathbf{S}_3(u,v) = \mathbf{S}_1(u,v) + \mathbf{S}_2(u,v) - \mathbf{S}_3(u,v).}$$

Generalized Coons Patch

The terms (1-u) and u (similarly for v and (1-v)) in the Coons patch are linear blending functions, as a direct result of the bi-linear construction in Step 1 and Step 2. If we replace them by a pair $\{\alpha_0(u), 1-\alpha_0(u)\}$ (similarly for v), where $a_0(u)$ can be any continuous function, the resulting surface still meets the boundary condition:
$S(0,v) = P_0(v)$, $S(1,v) = P_1(v)$, $S(u,0) = Q_0(u)$, and $S(u,1) = Q_1(u)$.

$$S(u,v) = \begin{bmatrix}(1-u) & u\end{bmatrix}\begin{bmatrix}P_0(v)\\ P_1(v)\end{bmatrix} + \begin{bmatrix}Q_0(u) & Q_1(u)\end{bmatrix}\begin{bmatrix}1-v\\ v\end{bmatrix} - \begin{bmatrix}(1-u) & u\end{bmatrix}\begin{bmatrix}Q_0(0) & Q_1(0)\\ Q_0(1) & Q_1(1)\end{bmatrix}\begin{bmatrix}1-v\\ v\end{bmatrix}$$

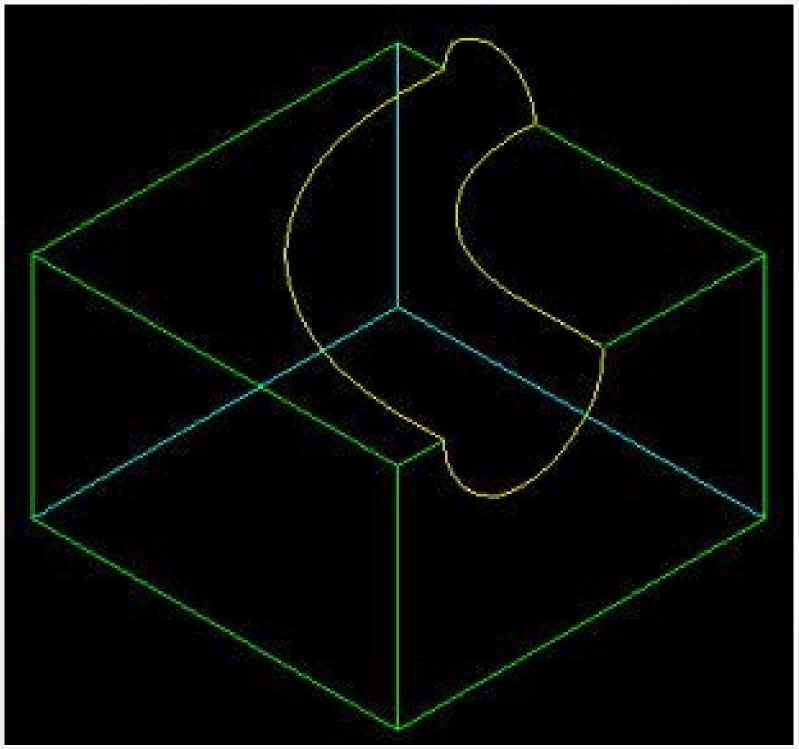
$$\begin{aligned}(1\text{-u}) &\rightarrow \alpha_0(\text{u})\\ \text{u} &\rightarrow 1-\alpha_0(\text{u})\end{aligned}$$

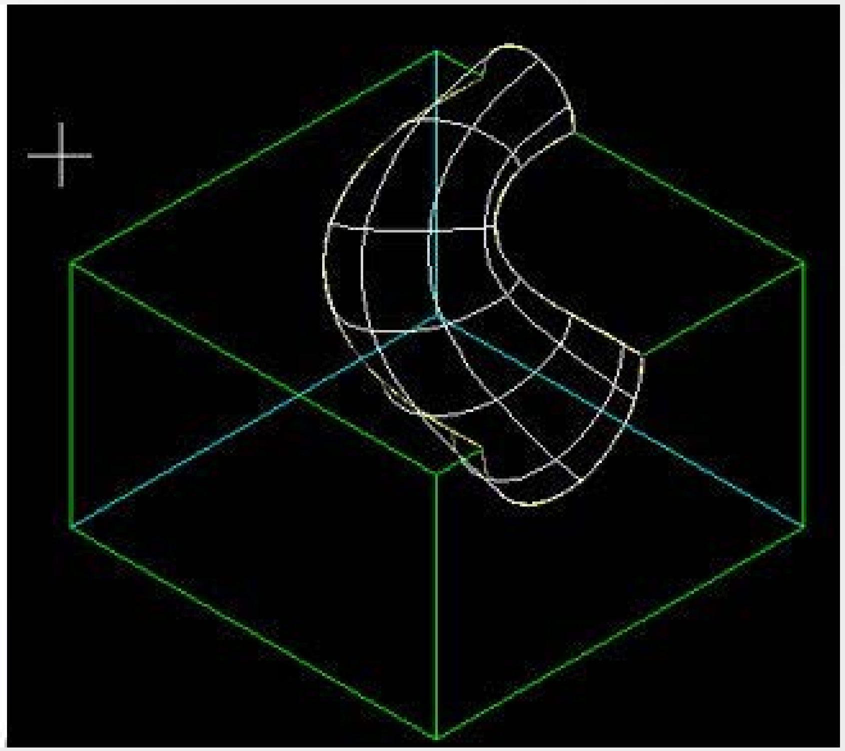
$$\Big\downarrow$$

$$S(u,v) = \begin{bmatrix}\alpha_0(u) & 1-\alpha_0(u)\end{bmatrix}\begin{bmatrix}P_0(v)\\ P_1(v)\end{bmatrix} + \begin{bmatrix}Q_0(u) & Q_1(u)\end{bmatrix}\begin{bmatrix}\alpha_0(v)\\ 1-\alpha_0(v)\end{bmatrix} - \begin{bmatrix}\alpha_0(u) & 1-\alpha_0(u)\end{bmatrix}\begin{bmatrix}Q_0(0) & Q_1(0)\\ Q_0(1) & Q_1(1)\end{bmatrix}\begin{bmatrix}\alpha_0(v)\\ 1-\alpha_0(v)\end{bmatrix}$$

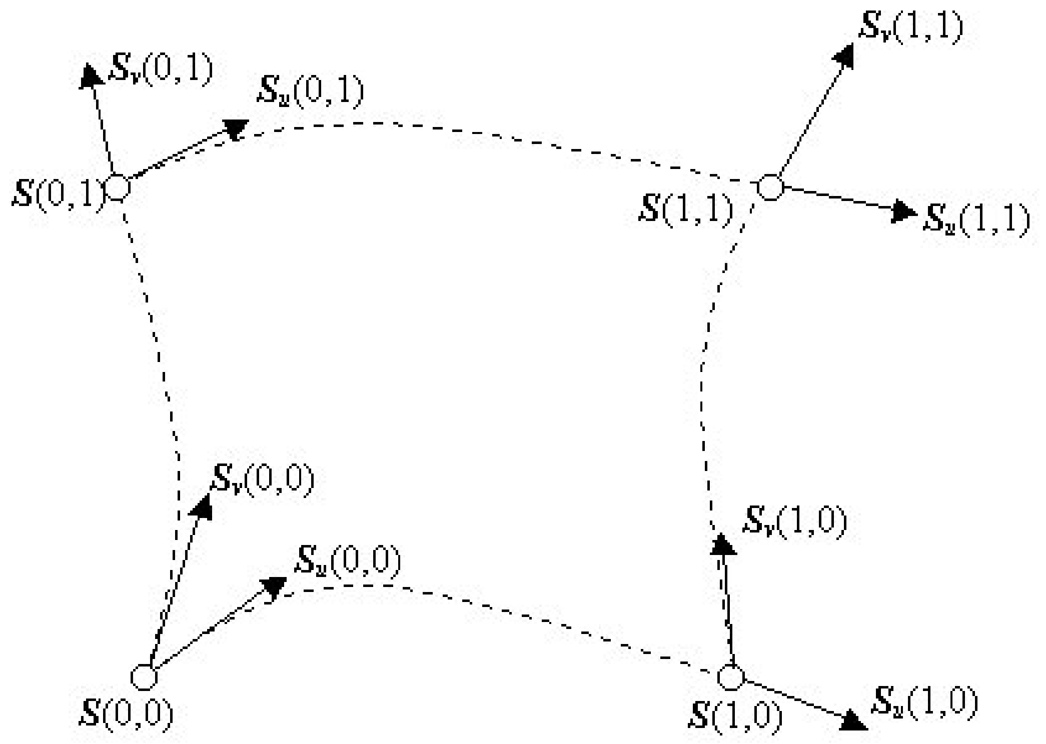
Example: $\alpha_0(\text{u}) = 1 - 3\text{u}^2 + 2\text{u}^3$

Example of Coons Patch



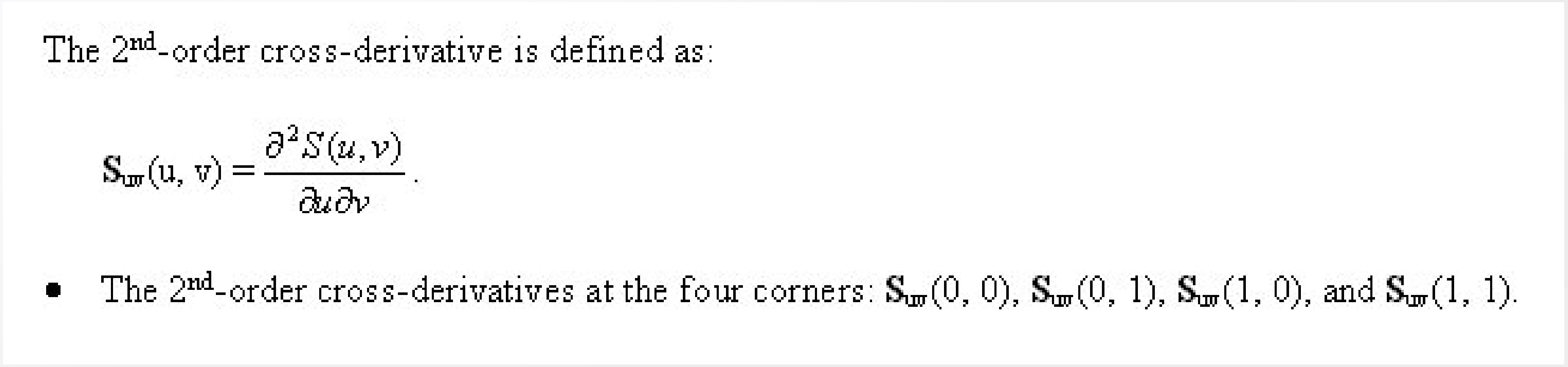


First 12 Boundary Conditions in a Patch



- The four corner points $\mathbf{S}(0, 0)$, $\mathbf{S}(0, 1)$, $\mathbf{S}(1, 0)$, and $\mathbf{S}(1, 1)$.
- The four tangent vectors along u direction at the four corners: $\mathbf{S}_u(0, 0)$, $\mathbf{S}_u(0, 1)$, $\mathbf{S}_u(1, 0)$, and $\mathbf{S}_u(1, 1)$.
- The four tangent vectors along v direction at the four corners: $\mathbf{S}_v(0, 0)$, $\mathbf{S}_v(0, 1)$, $\mathbf{S}_v(1, 0)$, and $\mathbf{S}_v(1, 1)$.

Four more boundary conditions in a patch



The 2nd-order cross-derivative is defined as:

$$\mathbf{S}_{uv}(\mathrm{u},\ \mathrm{v}) = \frac{\partial^2 S(u,v)}{\partial u \partial v}\ .$$

- The 2nd-order cross-derivatives at the four corners: $\mathbf{S}_{uv}(0,\ 0)$, $\mathbf{S}_{uv}(0,\ 1)$, $\mathbf{S}_{uv}(1,\ 0)$, and $\mathbf{S}_{uv}(1,\ 1)$.

Derivatives of Bi-cubic Patch

$$\mathbf{S}(u,v) = [1\ u\ u^2\ u^3]\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

$$\mathbf{S}_u(u,v) = \left([1\ u\ u^2\ u^3]\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}\right)_u = [0\ 1\ 2u\ 3u^2]\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$

$$\mathbf{S}_v(u,v) = \left([1\ u\ u^2\ u^3]\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}\right)_v = [1\ u\ u^2\ u^3]\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 2v \\ 3v^2 \end{bmatrix}$$

$$\mathbf{S}_{uv}(u,v) = \left([0\ 1\ 2u\ 3u^2]\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}\right)_v = [0\ 1\ 2u\ 3u^2]\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 2v \\ 3v^2 \end{bmatrix}$$

Definition of Hermite Patch

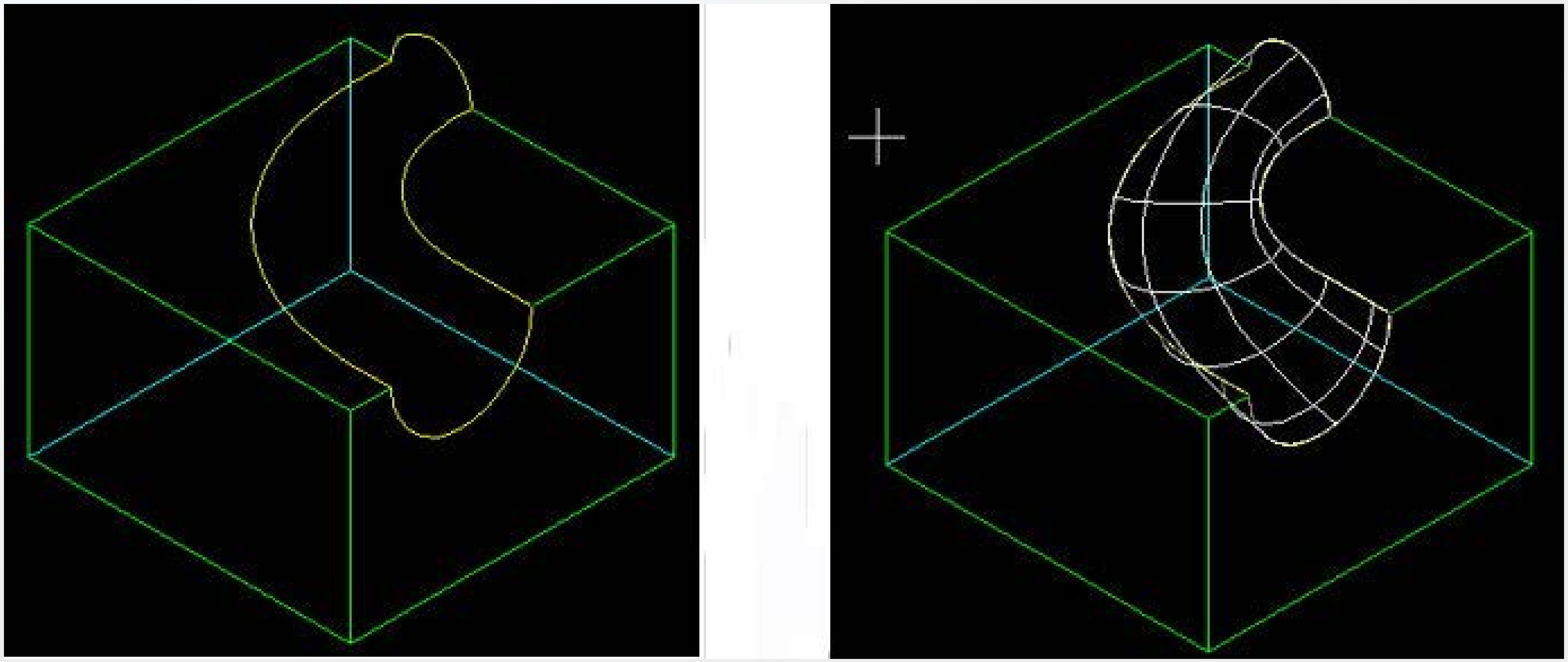
$$\mathbf{S}(u,v) = [f_0(u)\ \ f_1(u)\ \ f_2(u)\ \ f_3(u)] \begin{bmatrix} S(0,0) & S(0,1) & S_v(0,0) & S_v(0,1) \\ S(1,0) & S(1,1) & S_v(1,0) & S_v(1,1) \\ S_u(0,0) & S_u(0,1) & S_{uv}(0,0) & S_{uv}(0,1) \\ S_u(1,0) & S_u(1,1) & S_{uv}(1,0) & S_{uv}(1,1) \end{bmatrix} \begin{bmatrix} f_0(v) \\ f_1(v) \\ f_2(v) \\ f_3(v) \end{bmatrix}$$

$$(0 \le u \le 1, 0 \le v \le 1)$$

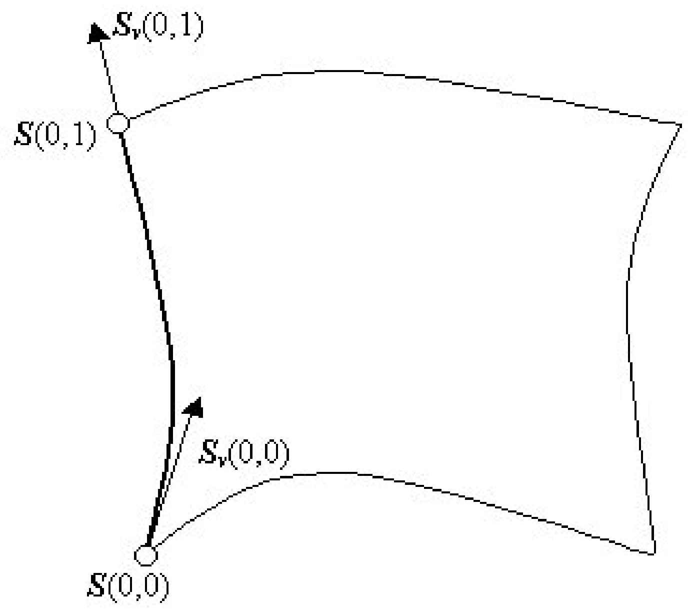
where the blending functions $f_0(u)$, $f_1(u)$, $f_2(u)$, and $f_3(u)$ are Hermite functions:

$$f_0(u) = 1 - 3u^2 + 2u^3$$
$$f_1(u) = 3u^2 - 2u^3$$
$$f_2(u) = u - 2u^2 + u^3$$
$$f_3(u) = -u^2 + u^3.$$

Example of Hermite Patch



Boundary Curves of a Hermite Patch are Hermite Curves



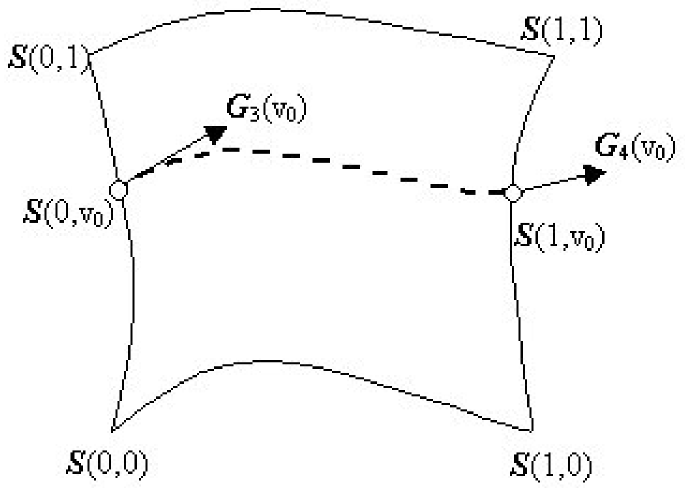
Each of the four boundary curves is a Hermite curve.

Example: $\mathbf{S}(0,v)$. After substituting 0 for u into S(u,v):

$$\mathbf{S}(0,v) = [f_0(v)\ \ f_1(v)\ \ f_2(v)\ \ f_3(v)] \begin{bmatrix} S(0,0) \\ S(0,1) \\ S_v(0,0) \\ S_v(0,1) \end{bmatrix}.$$

$\mathbf{S}(0, 0)$ and $\mathbf{S}(0, 1)$ are the two corners and $\mathbf{S}_v(0, 0)$ and $\mathbf{S}_v(0, 1)$ are the tangent vectors of the patch along the v direction. Notice that $\frac{\partial S(0,v)}{\partial v}\Big|_{v=0} = \mathbf{S}_v(0, 0)$, and $\frac{\partial S(0,v)}{\partial v}\Big|_{v=1} = \mathbf{S}_v(0, 1)$.

Iso-parametric Curves of a Hermite Patch are Hermite Curves



Any iso-parametric curve $\mathbf{S}(u, v_0)$ or $\mathbf{S}(u_0, v)$is a Hermite.

Example: $\mathbf{S}(u, v_0)$. Substituting $v = v_0$ leads to:

$$\mathbf{S}(u, v_0) = [f_0(u) \;\; f_1(u) \;\; f_2(u) \;\; f_3(u)] \begin{bmatrix} S(0,v_0) \\ S(1,v_0) \\ G_3(v_0) \\ G_4(v_0) \end{bmatrix}.$$

where:

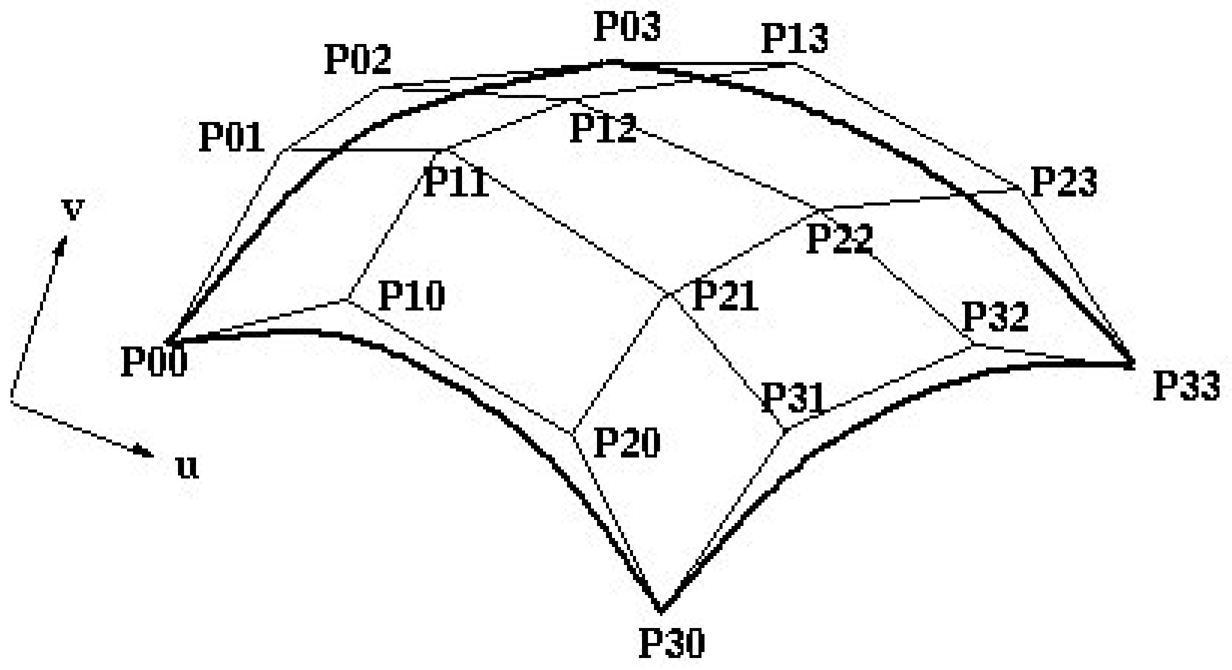
$$G_3(v_0) = \mathbf{S}_u(0,0)f_0(v_0) + \mathbf{S}_u(0,1)f_1(v_0) + \mathbf{S}_{uv}(0,0)f_2(v_0) + \mathbf{S}_{uv}(0,1)f_3(v_0)$$

$$G_4(v_0) = \mathbf{S}_u(1,0)f_0(v_0) + \mathbf{S}_u(1,1)f_1(v_0) + \mathbf{S}_{uv}(1,0)f_2(v_0) + \mathbf{S}_{uv}(1,1)f_3(v_0).$$

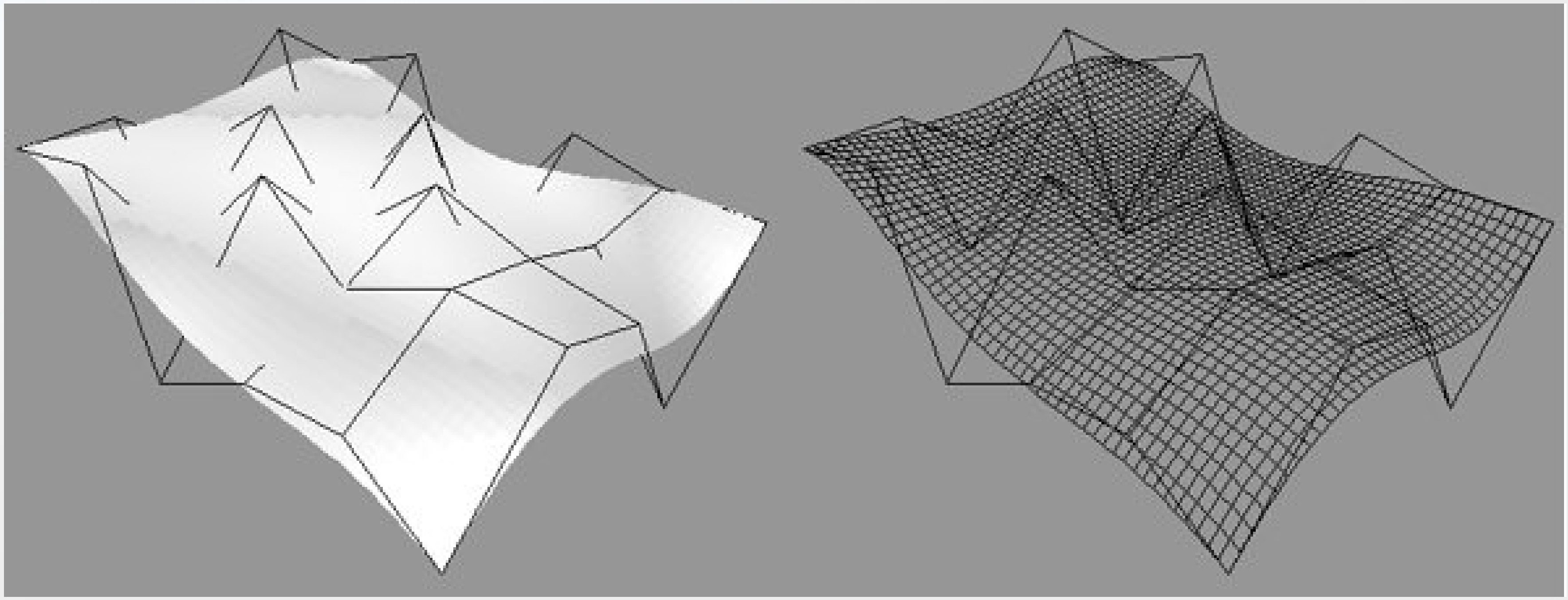
Note that:

$$\frac{\partial S(u,v)}{\partial u}\bigg|_{(0,v_0)} = G_3(v_0) \qquad \frac{\partial S(u,v)}{\partial u}\bigg|_{(1,v_0)} = G_4(v_0).$$

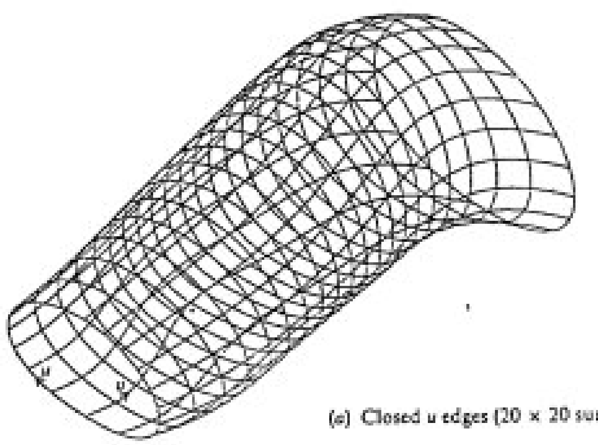
Control Points of a Bezier Surface



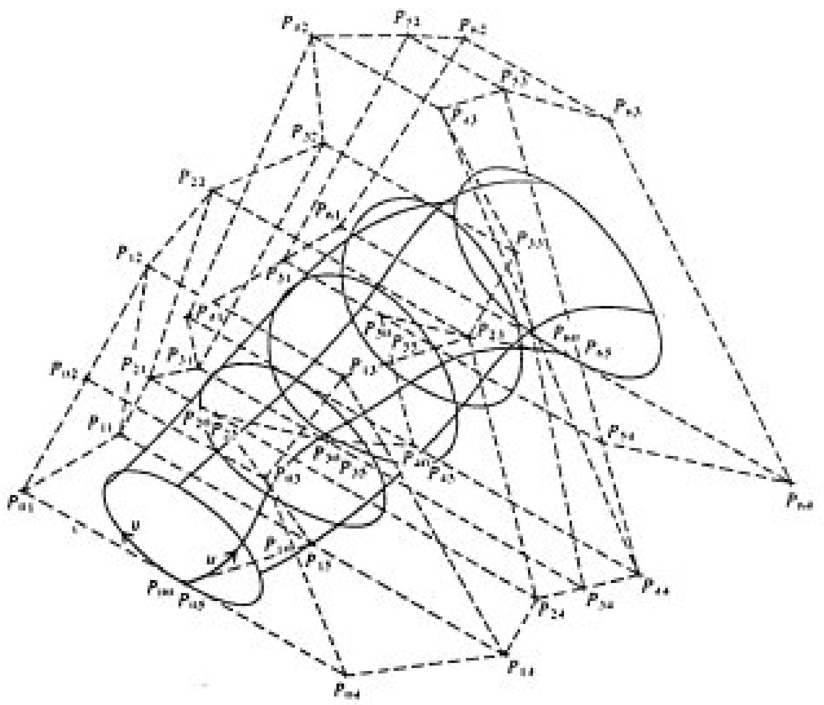
A 5x6 Bezier Surface



A Closed Bezier Surface



(a) Closed u edges (20 × 20 surface mesh)



(b) Closed u edges (polygon and 4 × 4 surface mesh)

Closed Bezier surface.

Properties of Bezier Surface

- The four control points $\mathbf{P}_{00}$, $\mathbf{P}_{0m}$, $\mathbf{P}_{n0}$, and $\mathbf{P}_{nm}$ lie on the surface and are its four corners $\mathbf{S}(0,0)$, $\mathbf{S}(0,1)$, $\mathbf{S}(1,0)$, and $\mathbf{S}(1,1)$ respectively.

- Any iso-parametric curve is a Bezier curve.

  ($\mathbf{S}(u_0, v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u_0)B_{j,m}(v)\,\mathbf{P}_{ij}$ is a Bezier curve, so is $\mathbf{S}(u, v_0)$, for any constant u_0 or v_0.)

- Convex Hull Property – The Bezier patch is inside the convex hull of its control points.

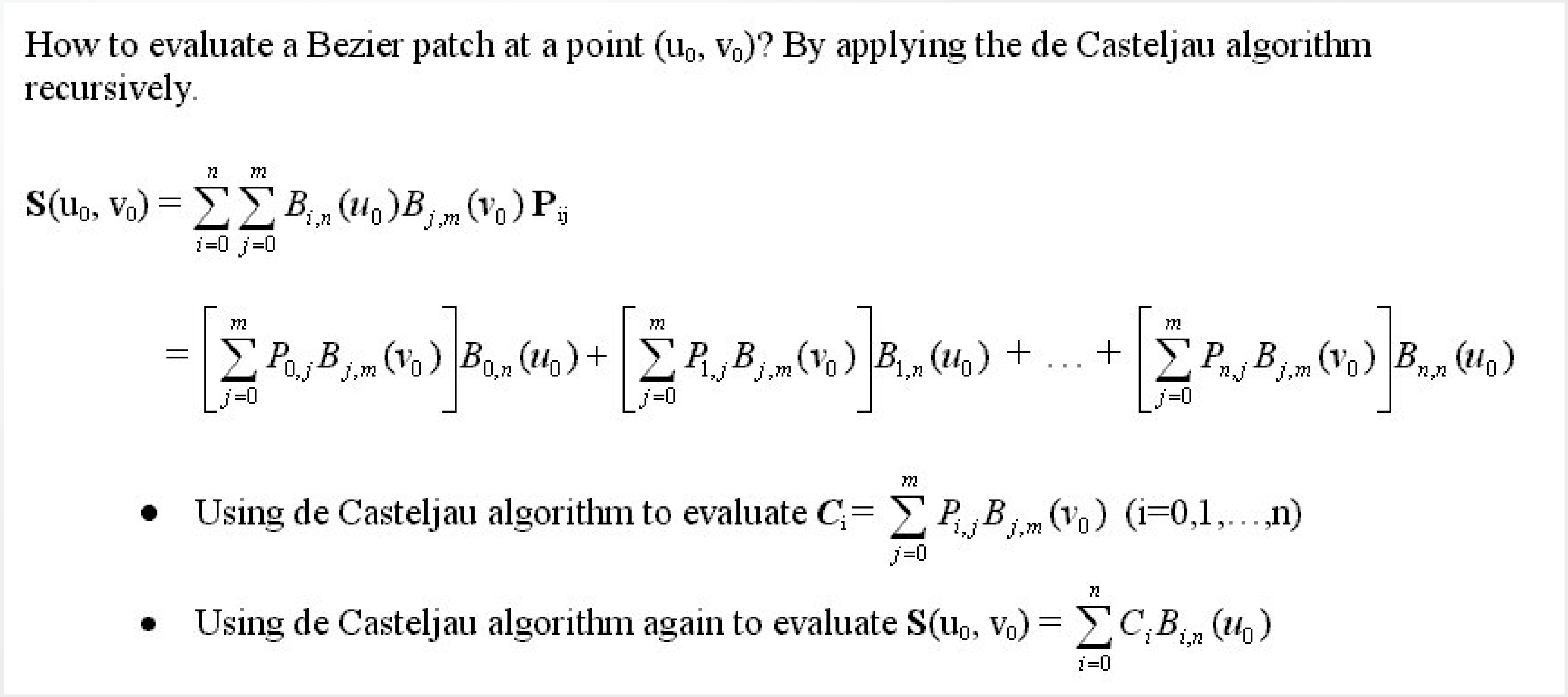
  (To prove: show that $\sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v) = 1$ for any u and v.)

- The partial derivative $\frac{\partial S(u,v)}{\partial u}\Big|_{(0,0)}$ is parallel to $(\mathbf{P}_{10} - \mathbf{P}_{00})$, and the partial derivative $\frac{\partial S(u,v)}{\partial v}\Big|_{(0,0)}$ is parallel to $(\mathbf{P}_{01} - \mathbf{P}_{00})$.

  (Similarly for the other three corners)

- The partial derivatives $\frac{\partial S(u,v)}{\partial u}$ and $\frac{\partial S(u,v)}{\partial v}$ are also Bezier surfaces themselves.

Evaluation of Bezier Surface



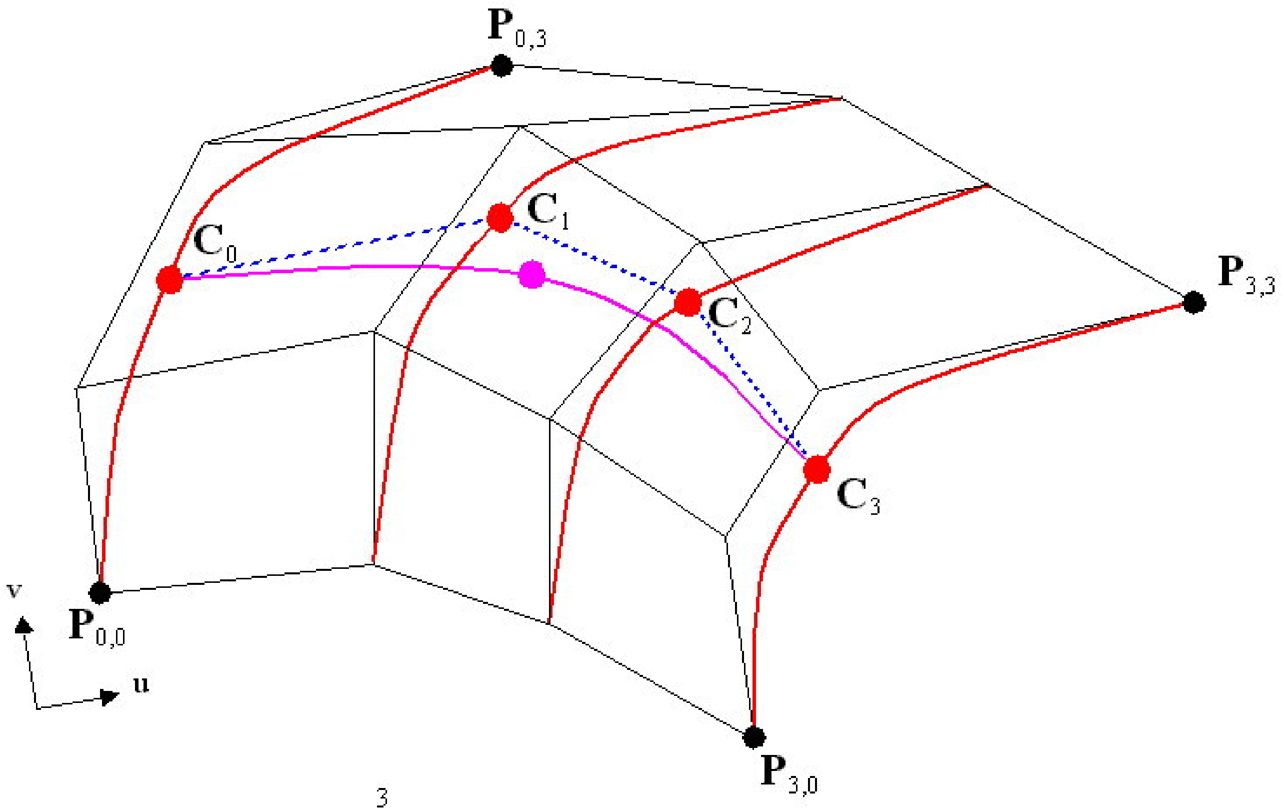
How to evaluate a Bezier patch at a point (u_0, v_0)? By applying the de Casteljau algorithm recursively.

$$\mathbf{S}(u_0, v_0) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u_0) B_{j,m}(v_0)\, \mathbf{P}_{ij}$$

$$= \left[\sum_{j=0}^{m} P_{0,j} B_{j,m}(v_0)\right] B_{0,n}(u_0) + \left[\sum_{j=0}^{m} P_{1,j} B_{j,m}(v_0)\right] B_{1,n}(u_0) + \ldots + \left[\sum_{j=0}^{m} P_{n,j} B_{j,m}(v_0)\right] B_{n,n}(u_0)$$

- Using de Casteljau algorithm to evaluate $C_i = \sum_{j=0}^{m} P_{i,j} B_{j,m}(v_0)$ (i=0,1,…,n)
- Using de Casteljau algorithm again to evaluate $\mathbf{S}(u_0, v_0) = \sum_{i=0}^{n} C_i B_{i,n}(u_0)$

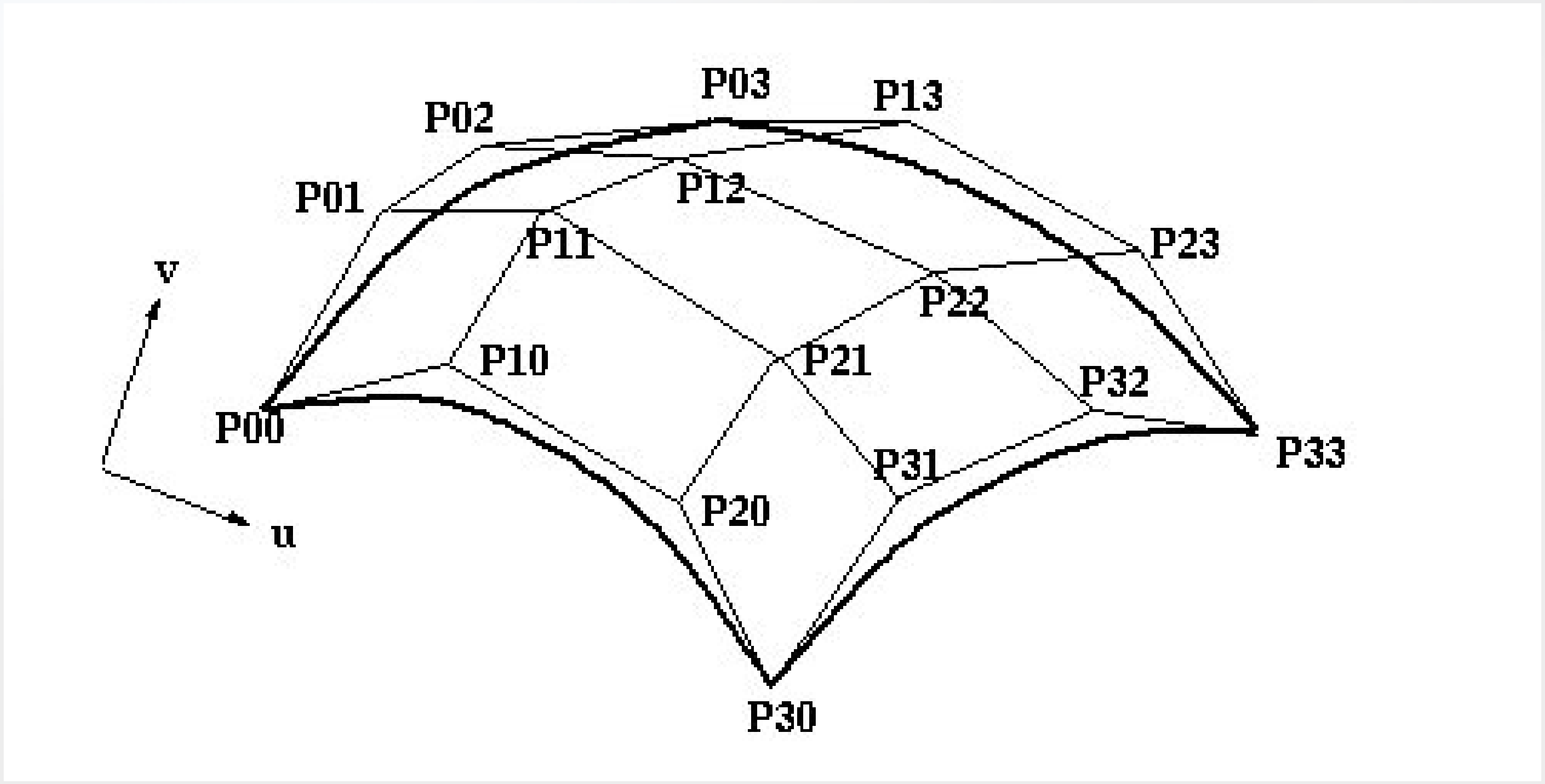
Pictorial Illustration of Evaluating a Cubic Bezier Surface



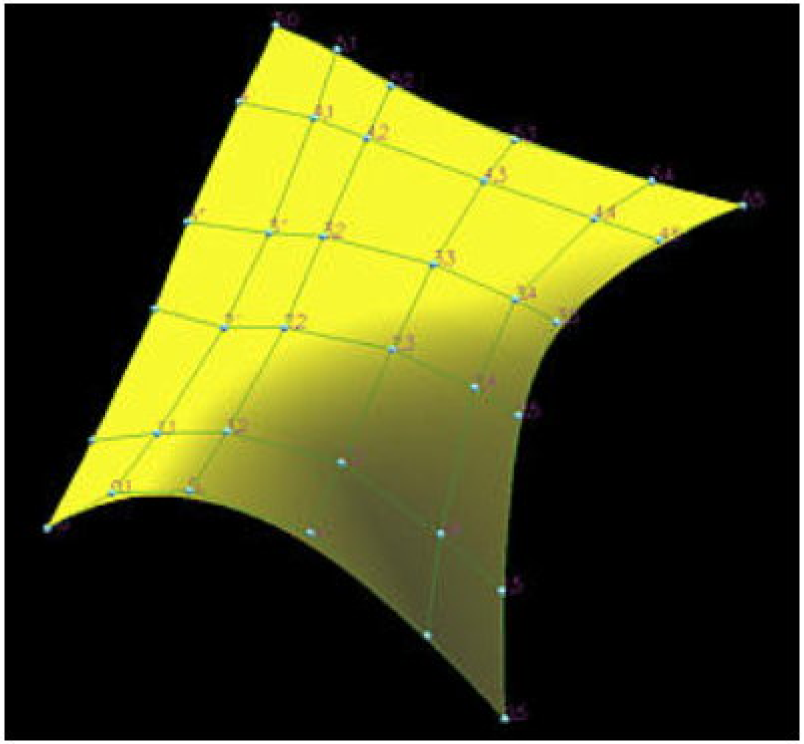
● $\mathbf{C}_i = \sum_{j=0}^{3} B_{j,3} P_{i,j}(v_0)$

● $\mathbf{S}(u_0, v_0) = \sum_{i=0}^{3} B_{i,3} C_i(u_0)$

Control Mesh of a B-Spline Surface



A 6x6 B-Spline Surface



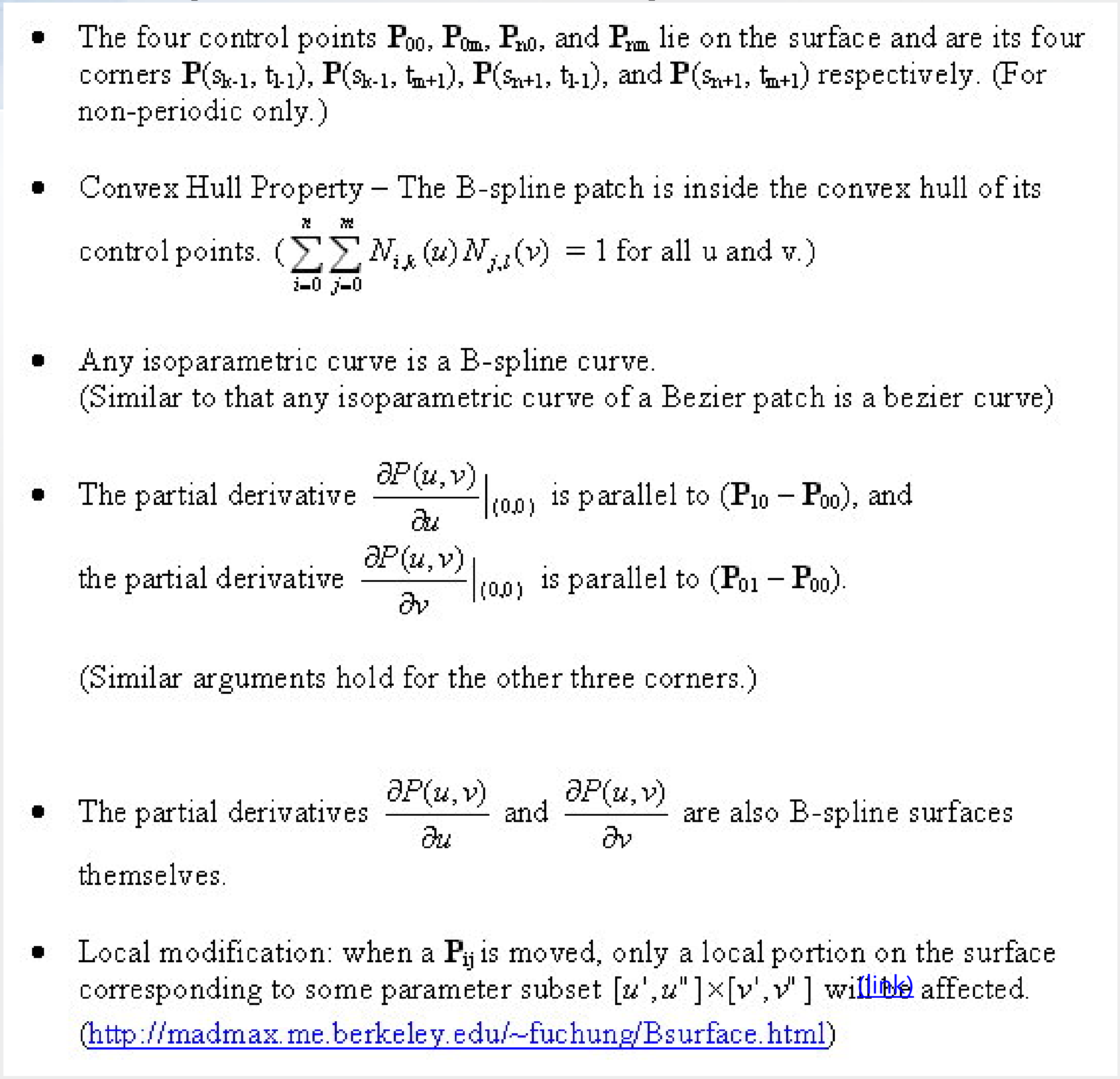
n = m = 5

k = 3, knot vector (u-direction) U = {0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1}

l = 4, knot vector (v-direction) V = {0, 0, 0, 0, 0.33, 0.66, 1, 1, 1, 1}

Both U and V are non-periodic and uniform (the patch passes all the four corners).

Properties of B-Spline Surface



- The four control points $\mathbf{P}_{00}$, $\mathbf{P}_{0m}$, $\mathbf{P}_{n0}$, and $\mathbf{P}_{nm}$ lie on the surface and are its four corners $\mathbf{P}(s_{k-1}, t_{l-1})$, $\mathbf{P}(s_{k-1}, t_{m+1})$, $\mathbf{P}(s_{n+1}, t_{l-1})$, and $\mathbf{P}(s_{n+1}, t_{m+1})$ respectively. (For non-periodic only.)

- Convex Hull Property – The B-spline patch is inside the convex hull of its control points. ($\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,k}(u)N_{j,l}(v) = 1$ for all u and v.)

- Any isoparametric curve is a B-spline curve.
  (Similar to that any isoparametric curve of a Bezier patch is a bezier curve)

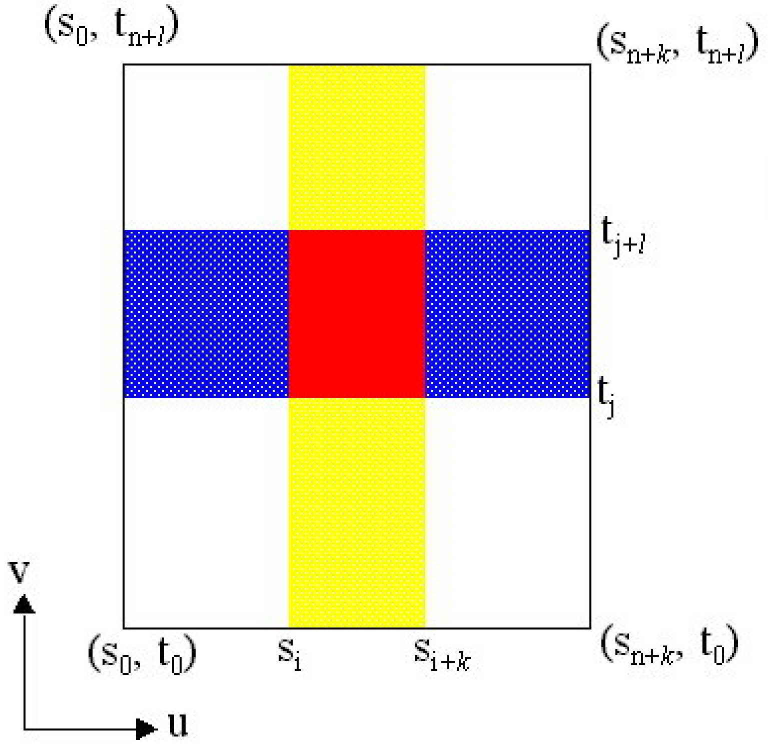
- The partial derivative $\left.\frac{\partial P(u,v)}{\partial u}\right|_{(0,0)}$ is parallel to $(\mathbf{P}_{10} - \mathbf{P}_{00})$, and the partial derivative $\left.\frac{\partial P(u,v)}{\partial v}\right|_{(0,0)}$ is parallel to $(\mathbf{P}_{01} - \mathbf{P}_{00})$.

  (Similar arguments hold for the other three corners.)

- The partial derivatives $\frac{\partial P(u,v)}{\partial u}$ and $\frac{\partial P(u,v)}{\partial v}$ are also B-spline surfaces themselves.

- Local modification: when a $\mathbf{P}_{ij}$ is moved, only a local portion on the surface corresponding to some parameter subset $[u', u''] \times [v', v'']$ will be affected. (link)
  (http://madmax.me.berkeley.edu/~fuchung/Bsurface.html)

Local-modification Control of B-Spline Surface



🟦🟥🟦	Non-zero interval of $N_{i,k}(u)$
🟨🟥🟨	Non-zero interval of $N_{j,l}(v)$
🟥	Non-zero interval of $N_{i,k}(u)N_{j,l}(v)$

$N_{i,k}(u)N_{j,l}(v)$ is the blending function before control point $\mathbf{P}_{ij}$. So when $\mathbf{P}_{ij}$ moves, it will only effect the portion of the shape of the original B-spline surface corresponding to the non-zero interval 🟥 .

Subdivision Surface

Catmull-Clark surface (1978)

(a) Initial control mesh

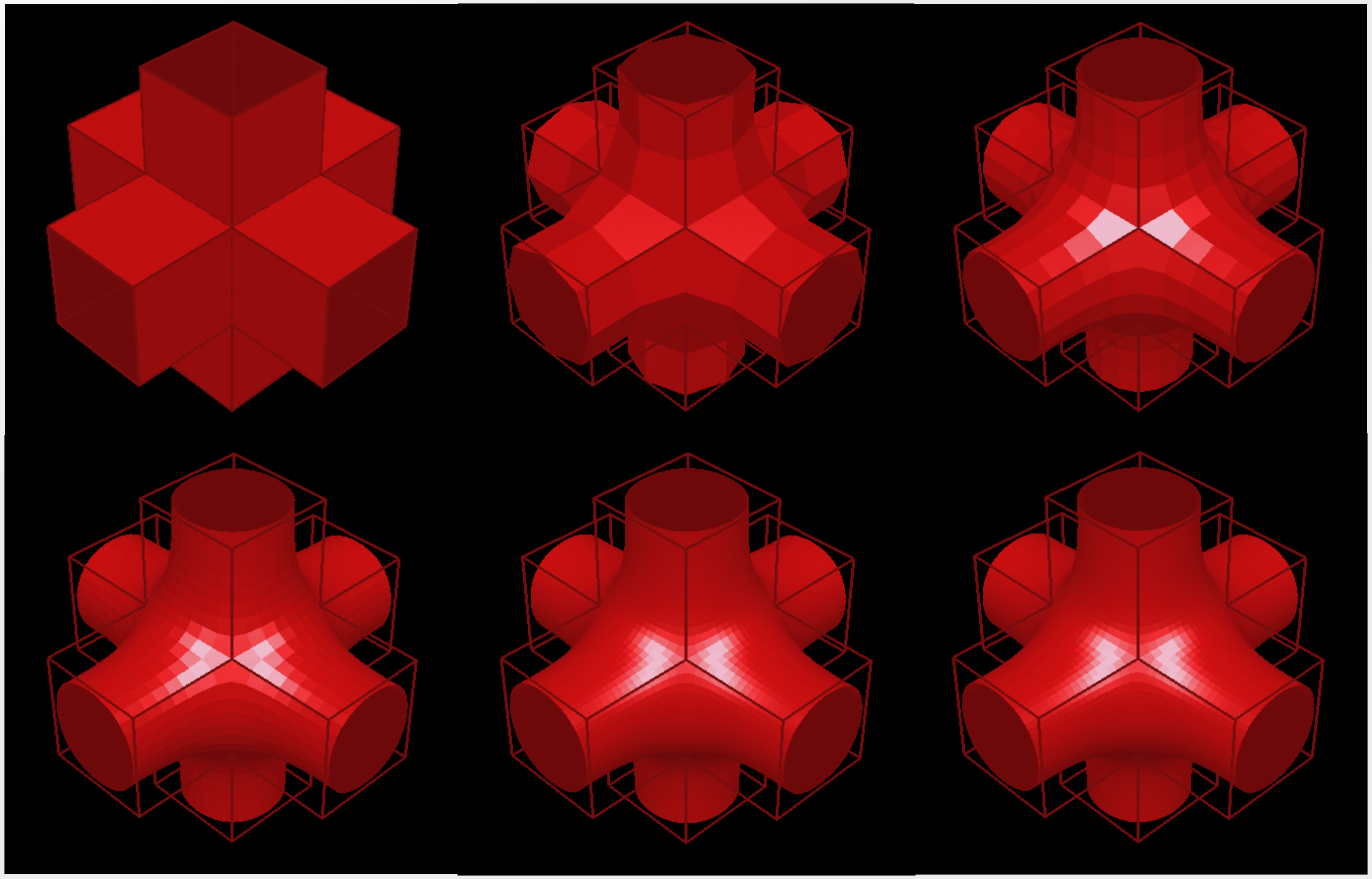
(b) 1st level subdivision

(c) 2nd level subdivision

(d) 3rd level subdivision

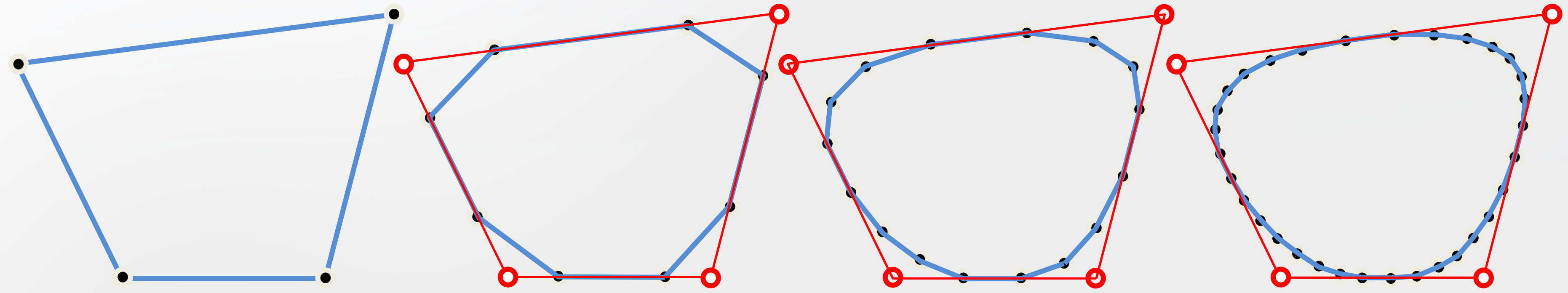
(e) 4th level subdivision

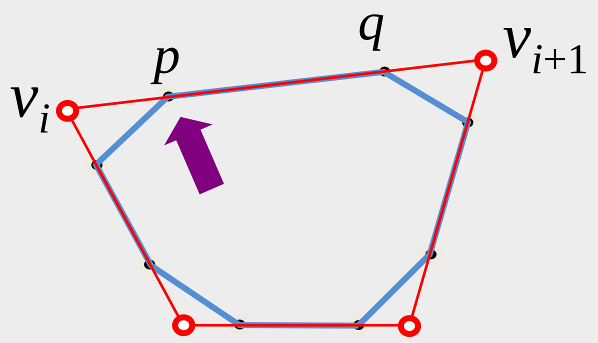
(f) Limit surface



The Basic Idea of Subdivision

- Chaikin's algorithm (1974)
  - equivalent to a uniform quadratic B-spline curve





$$p = \frac{3}{4} v_i + \frac{1}{4} v_{i+1}$$

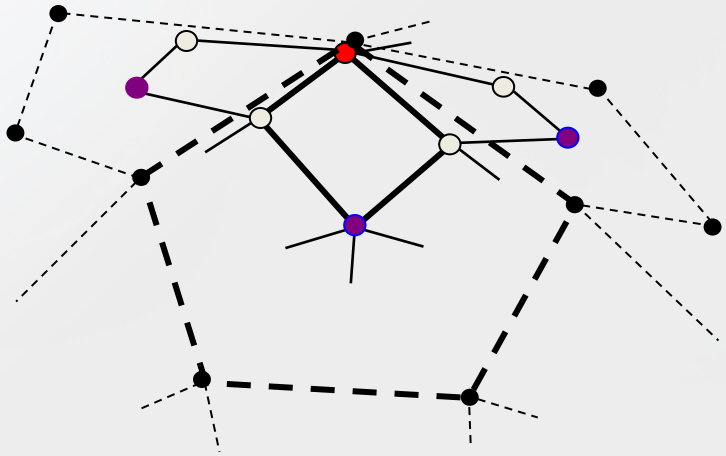
$$q = \frac{1}{4} v_i + \frac{3}{4} v_{i+1}$$

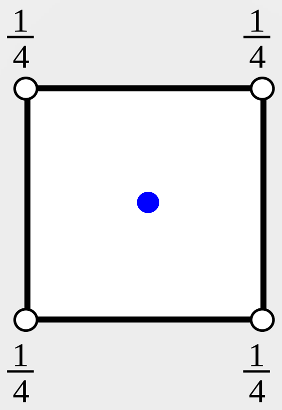
Catmull-Clark Surface Subdivision

● Newly inserted face vertices

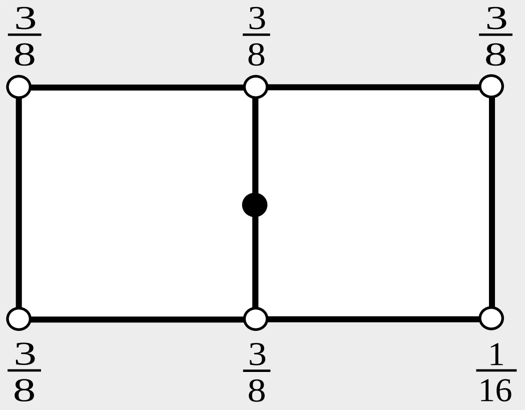
○ Newly inserted edge vertices

● Updated old vertices

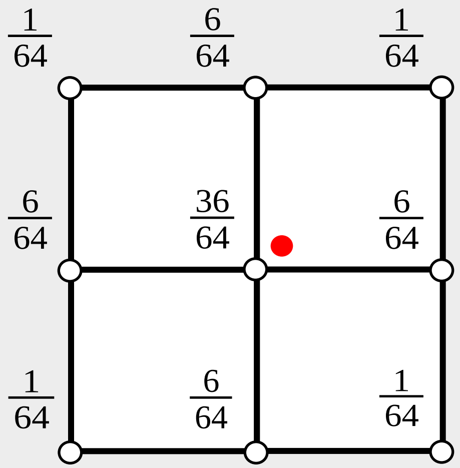




(a) New face vertices

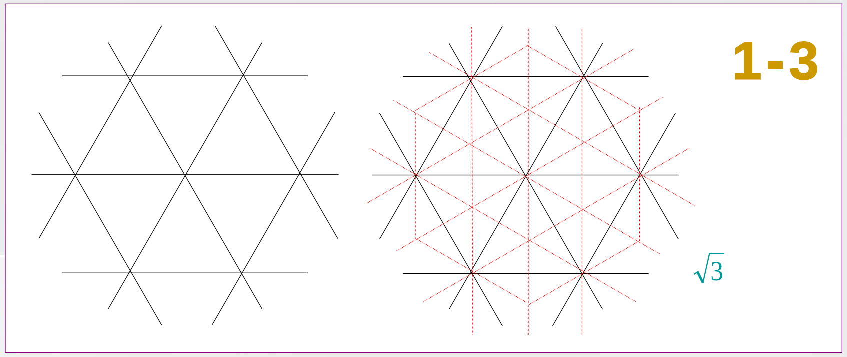


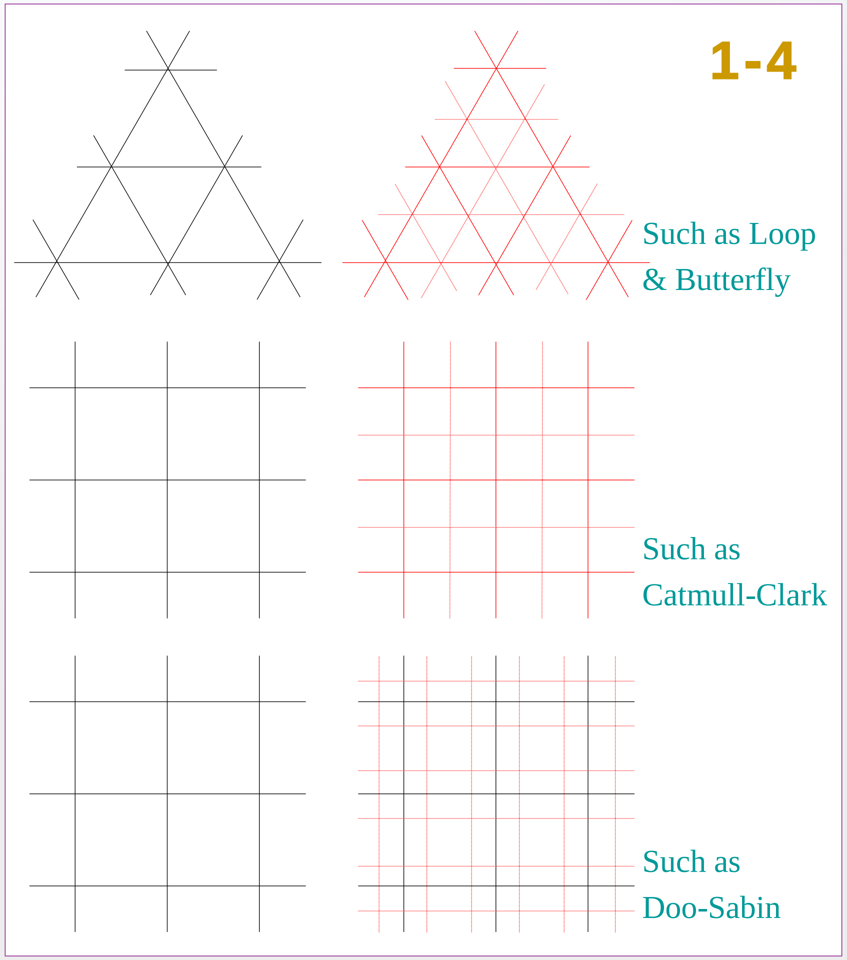
(b) New edge vertices

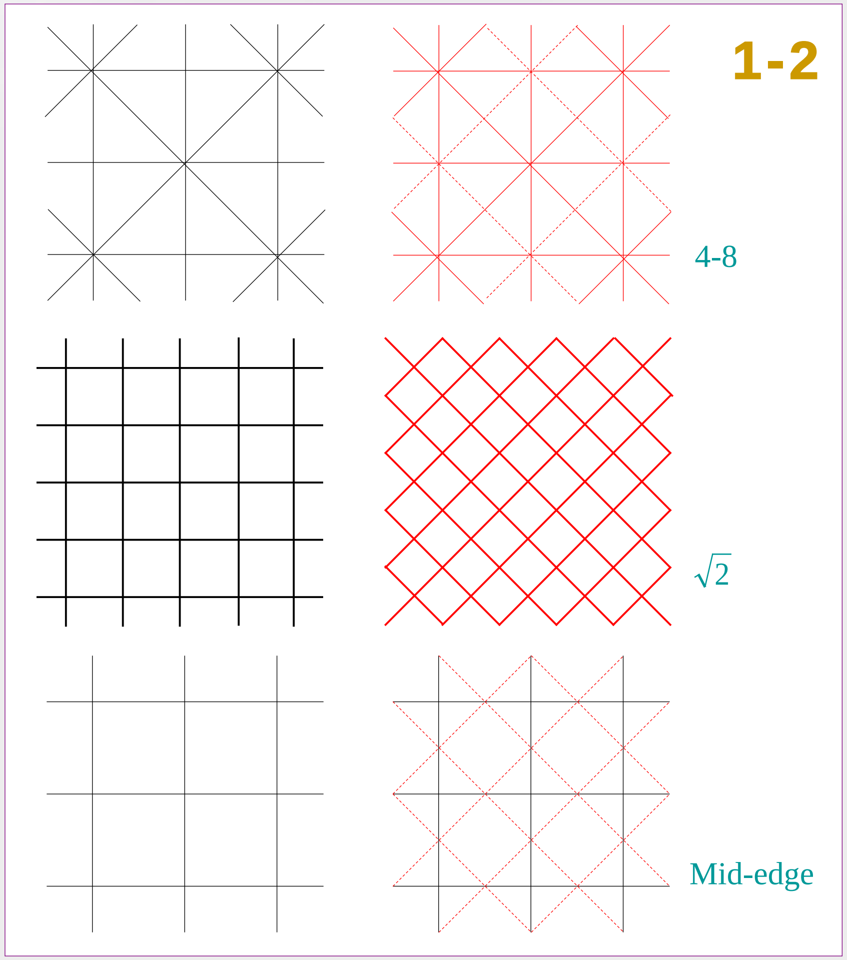


(c) Updated old vertices

Various Split Schemes

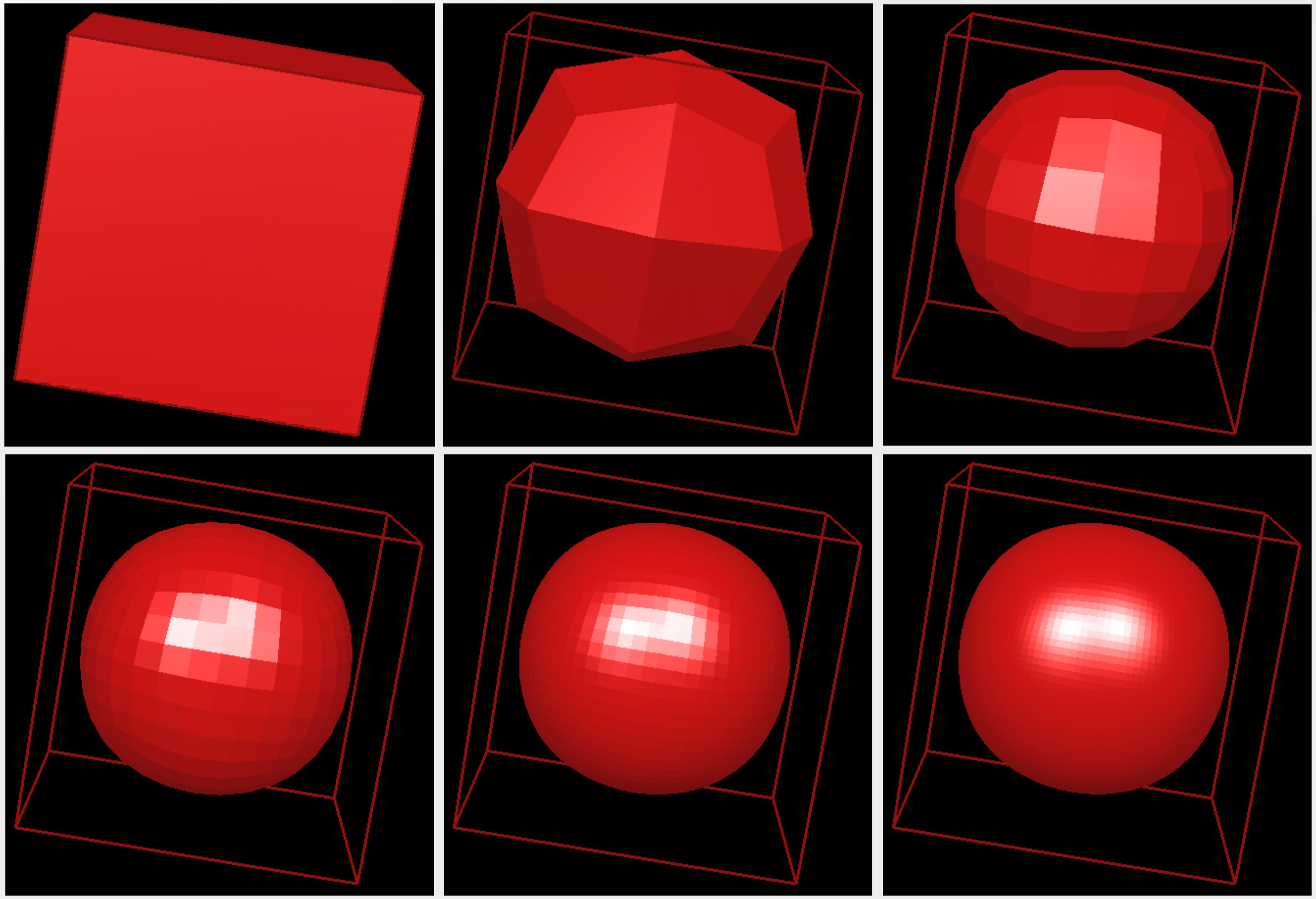




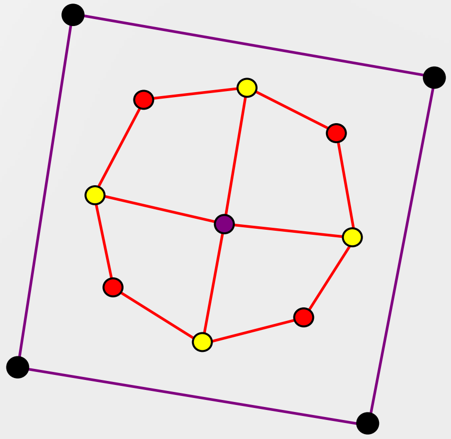


Catmull-Clark Surfaces

Extension of bi-cubic B-spline surfaces (Catmull and Clark 1978)



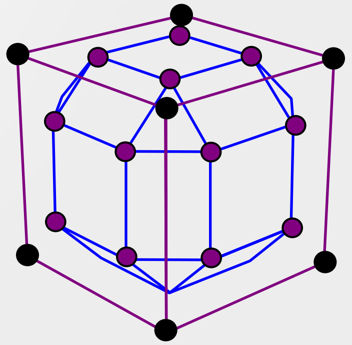
One-to-four splitting:

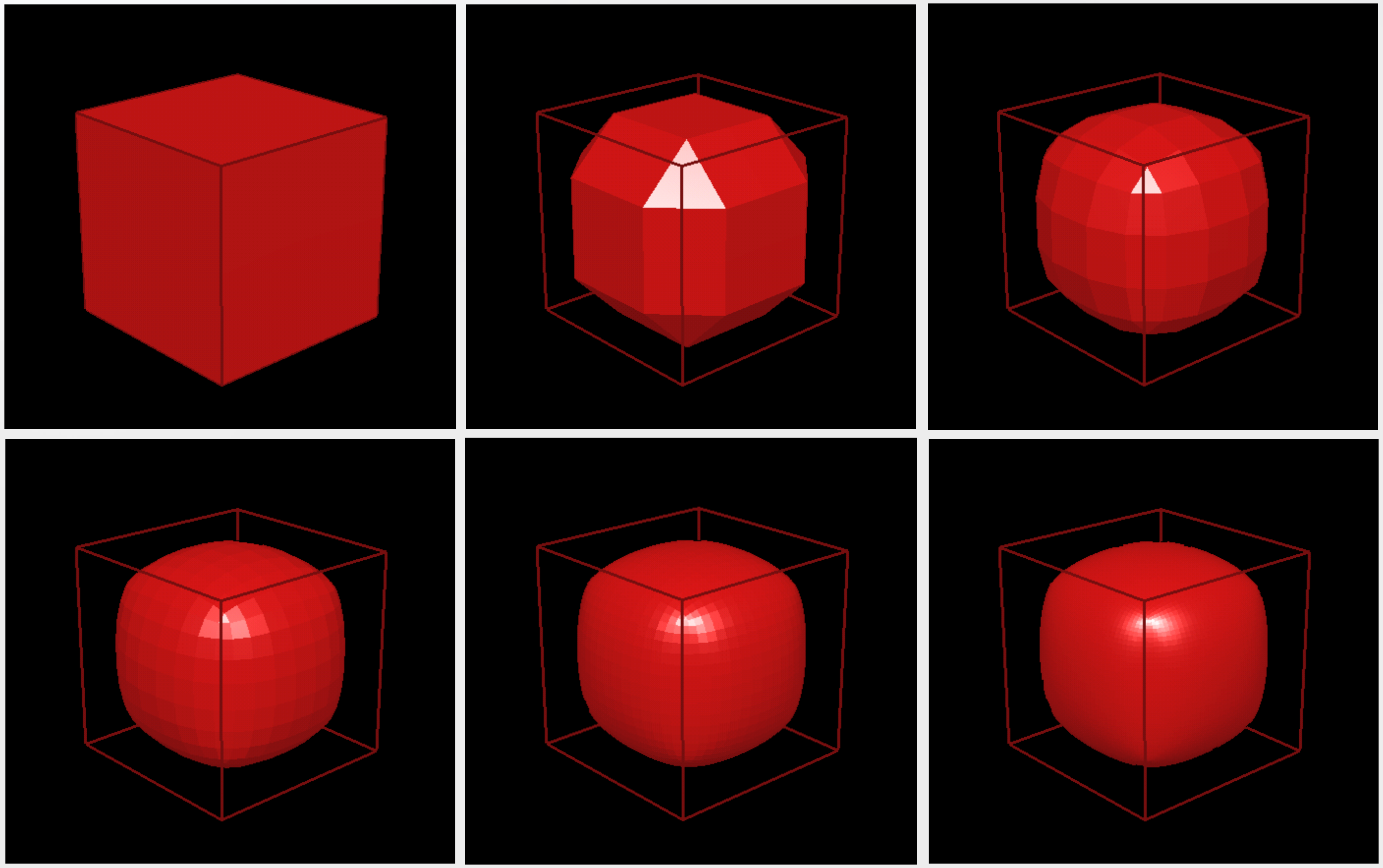


Doo-Sabin Surfaces

Extension of bi-quadratic B-spline surfaces (Doo and Sabin 1978)

Corner cutting:

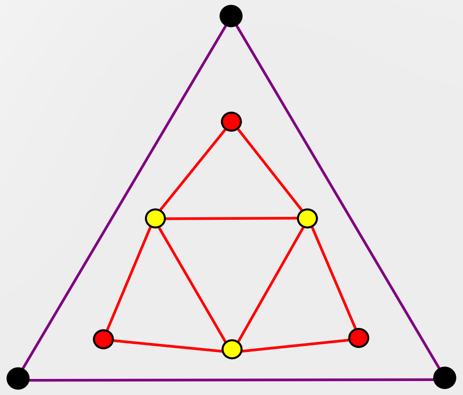


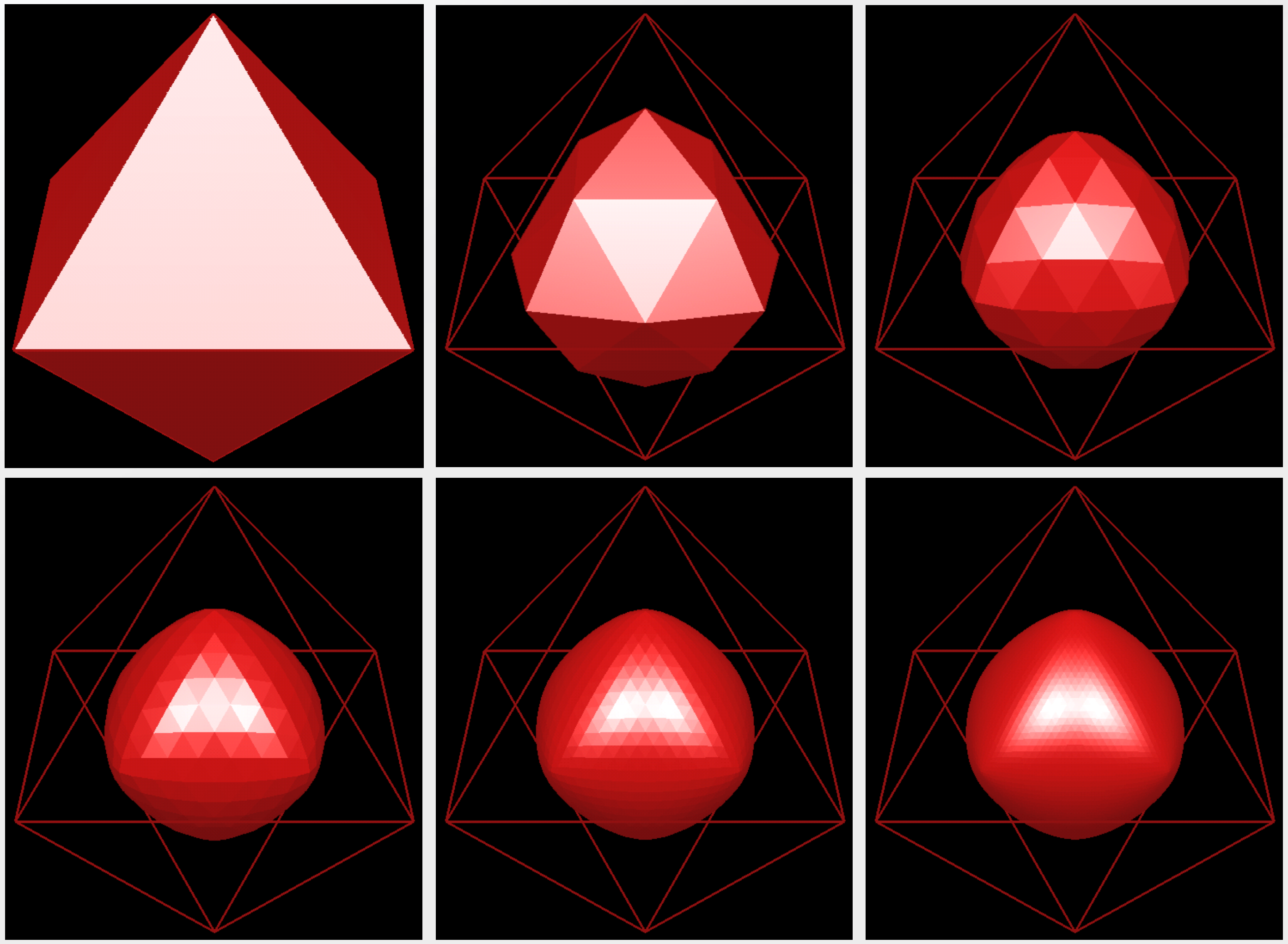


Loop Subdivision

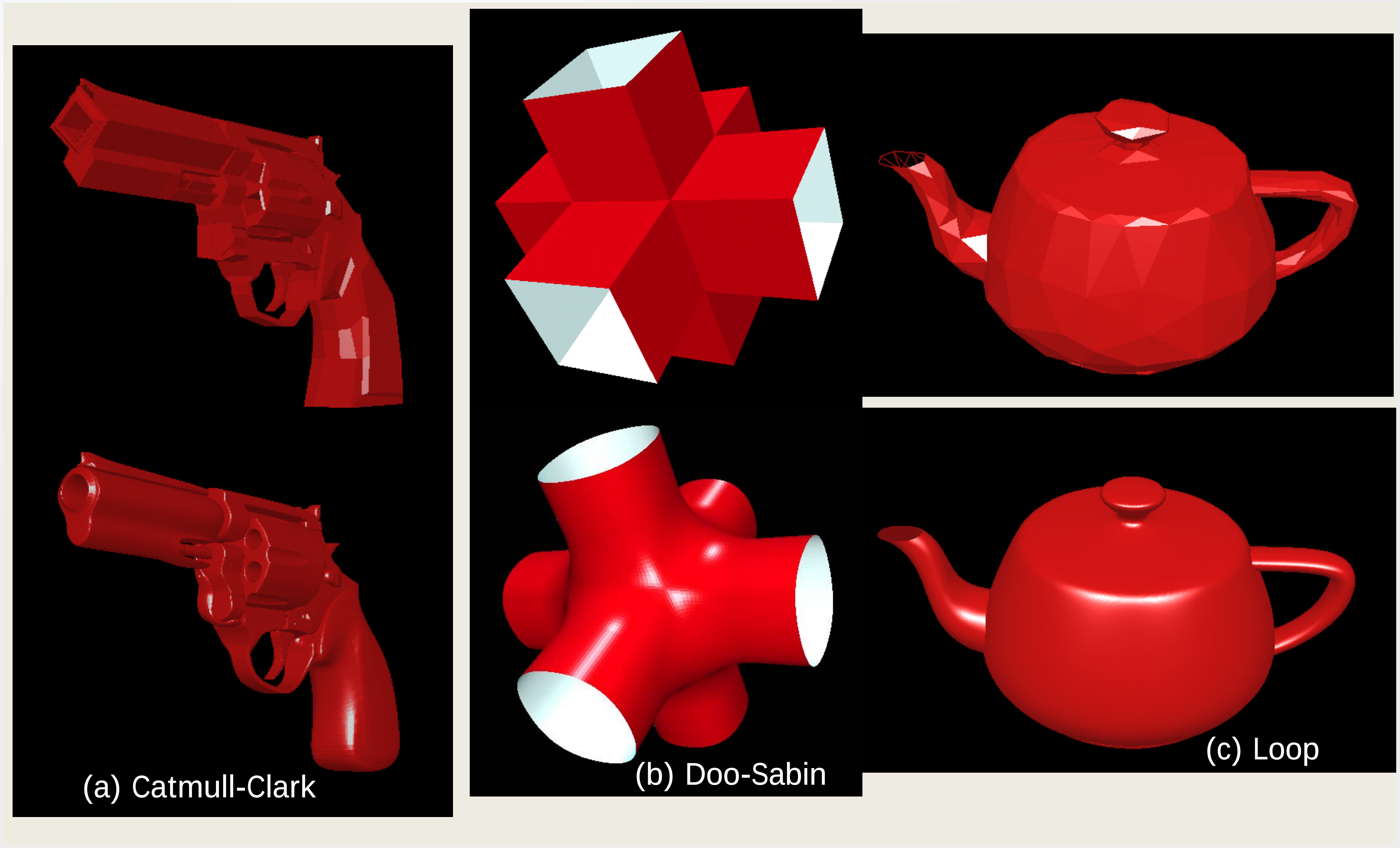
Loop subdivision (Loop 1987) extension of 3D box-splines

One-to-four splitting:



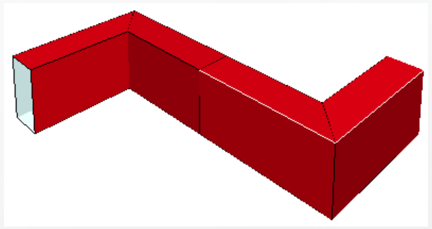


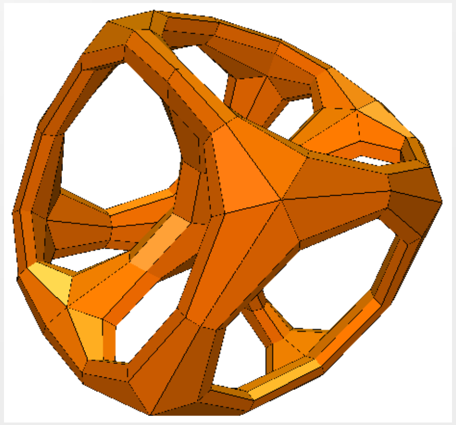
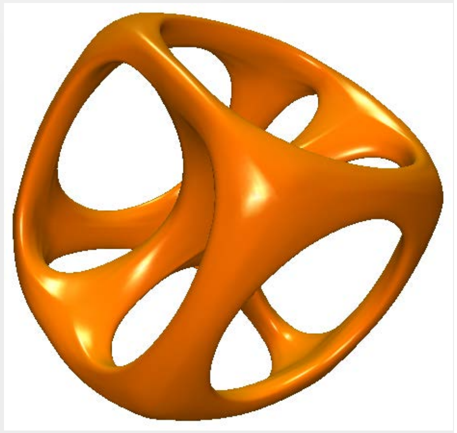
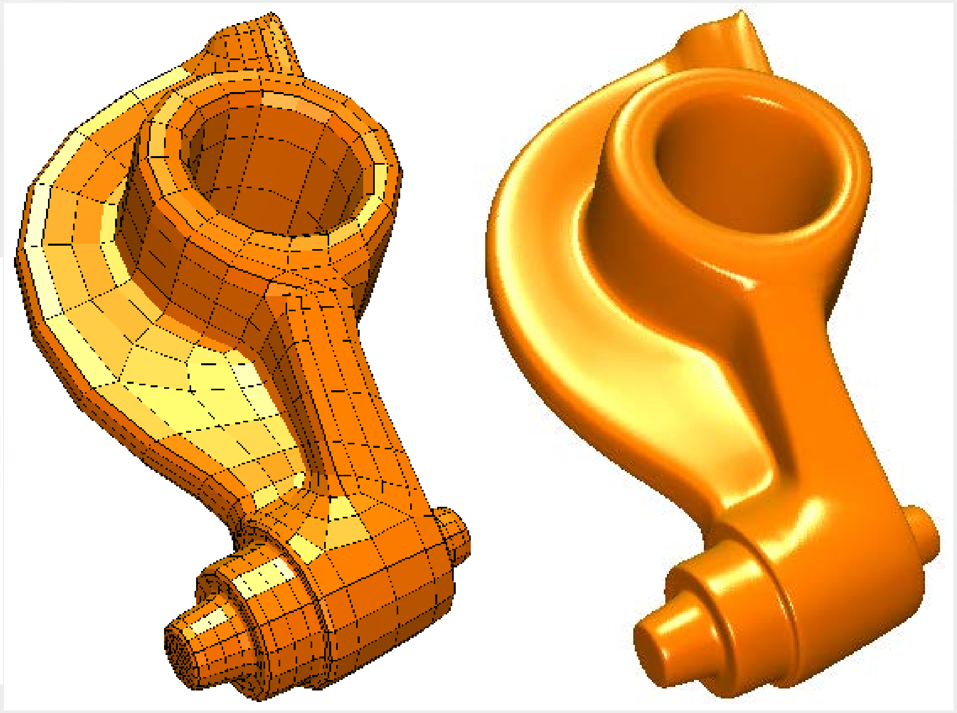
A Few More Examples



Interpolatory Subdivision Surface

- All or specified points on the initial mesh are kept in the final limit surface-



A Few Differential Geometry Topics Related to Continuity

Local Curve Topics

- Principal Vectors
  - Tangent
  - Normal
  - Binormal
- Osculating Plane and Circle
- Frenet Frame
- Curvature
- Torsion
- Revisiting the Definition of Geometric Continuity

source: Ch 12 Mortenson

Intrinsic Definition

- No reliance on external frame of reference
- Requires 2 equations as functions of arc length* s:

1) Curvature: $\frac{1}{\rho} = f(s)$

2) Torsion: $\tau = g(s)$

**length measured along the curve*

- For plane curves, alternatively:

Torsion (in 3D) measures how much curve deviates from a plane curve.

$$\frac{1}{\rho} = \frac{d\theta}{ds}$$

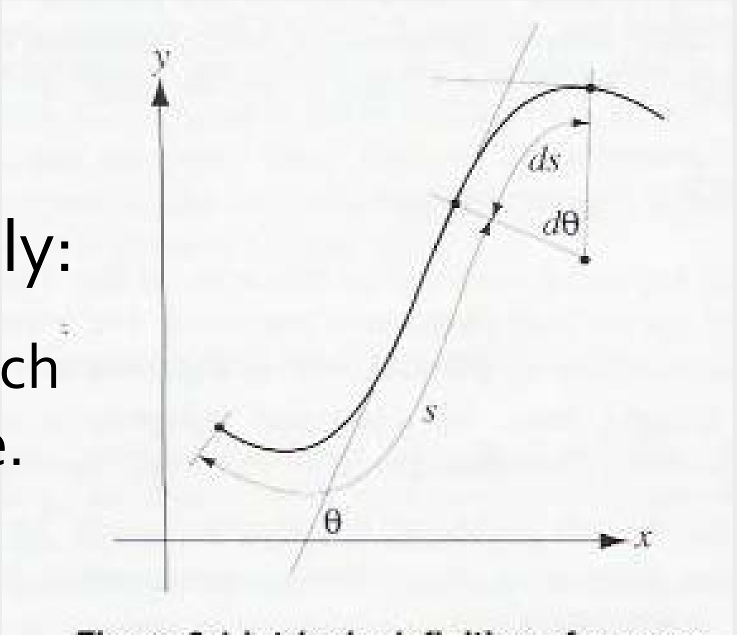


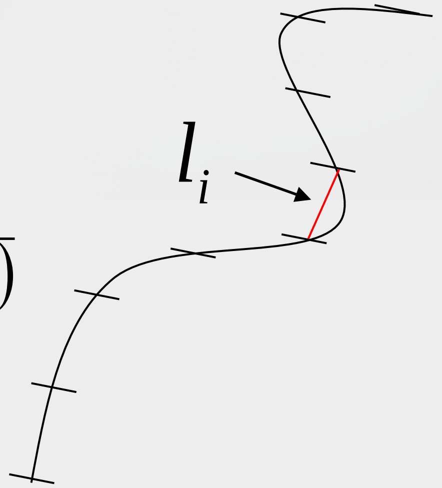
Figure 2.1 Intrinsic definition of a curve.

Treated in more detail in Chapter 12 of Mortenson and Chapter 10 of Farin.

source: Mortenson

Calculating Arc Length

Approximation: For parametric interval u_1 to u_2, subdivide curve segment into n equal pieces.



$$L = \sum_{i=1}^{n} l_i \quad \text{where} \quad l_i = \sqrt{(\mathbf{p}_i - \mathbf{p}_{i-1}) \bullet (\mathbf{p}_i - \mathbf{p}_{i-1})}$$

using $\mathbf{p} \bullet \mathbf{p} = |\mathbf{p}|^2$

$$L = \int_{u_1}^{u_2} \sqrt{\mathbf{p}^u \bullet \mathbf{p}^u}\, du$$ is more accurate.

source: Mortenson, p. 401

Tangent

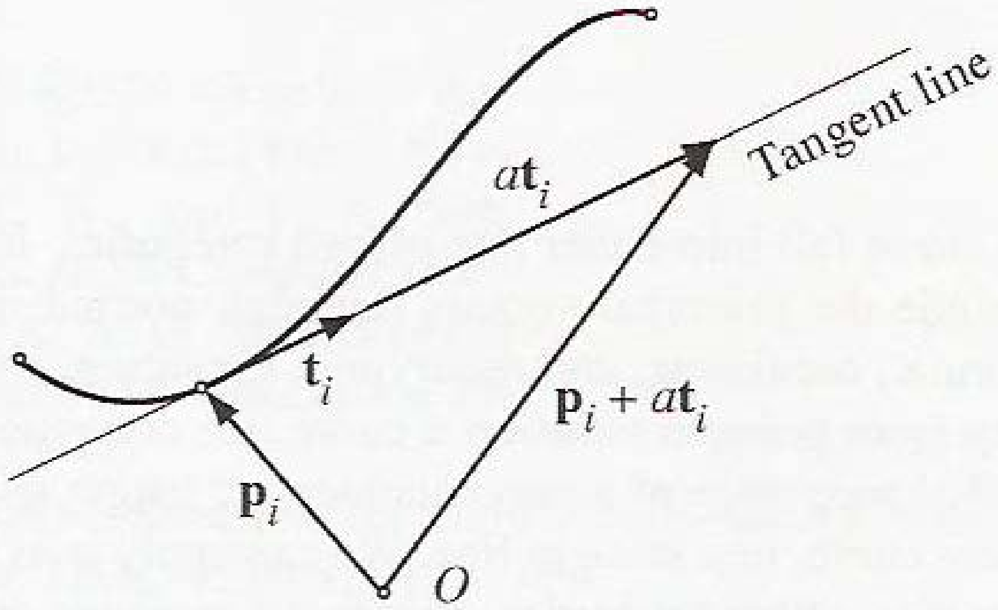


Figure 12.1 Tangent vector and line.

unit tangent vector: $\mathbf{t}_i = \frac{\mathbf{p}_i^u}{\left|\mathbf{p}_i^u\right|}$

source: Mortenson, p. 388

Normal Plane

Plane through $\mathbf{p}_i$ perpendicular to $\mathbf{t}_i$

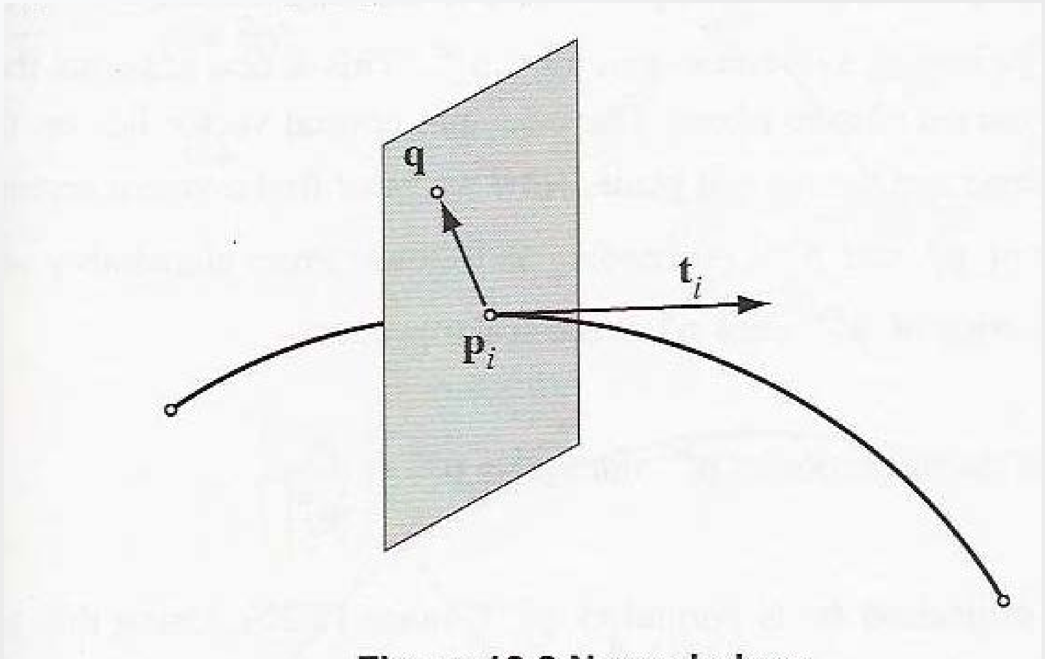


Figure 12.2 Normal plane.

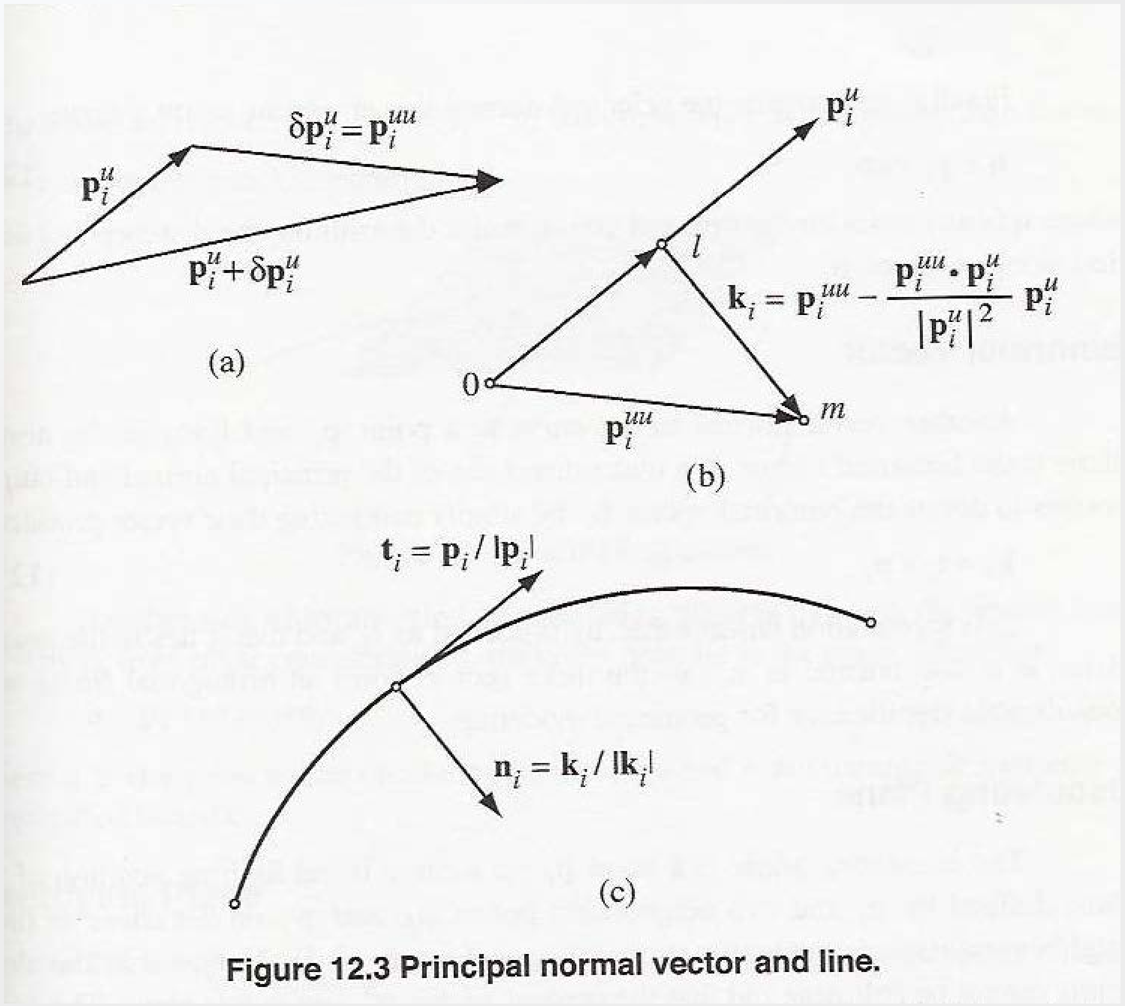
$$q = (x, y, z)$$

$$x_i^u x + y_i^u y + z_i^u z - (x_i x_i^u + y_i y_i^u + z_i z_i^u) = 0$$

source: Mortenson, p. 388-389

Principal Normal Vector and Line

Moving slightly along curve in neighborhood of $\mathbf{p}_i$ causes tangent vector to move in direction specified by: $\mathbf{p}_i^{uu}$



$\delta\mathbf{p}_i^u = \mathbf{p}_i^{uu}$
$\mathbf{p}_i^u$
$\mathbf{p}_i^u + \delta\mathbf{p}_i^u$
(a)
$\mathbf{p}_i^u$
l
$\mathbf{k}_i = \mathbf{p}_i^{uu} - \frac{\mathbf{p}_i^{uu} \cdot \mathbf{p}_i^u}{|\mathbf{p}_i^u|^2}\mathbf{p}_i^u$
0
m
$\mathbf{p}_i^{uu}$
(b)
$\mathbf{t}_i = \mathbf{p}_i / |\mathbf{p}_i|$
$\mathbf{n}_i = \mathbf{k}_i / |\mathbf{k}_i|$
(c)

Figure 12.3 Principal normal vector and line.

Use dot product to find projection of $\mathbf{p}_i^u$ onto $\mathbf{p}_i^{uu}$

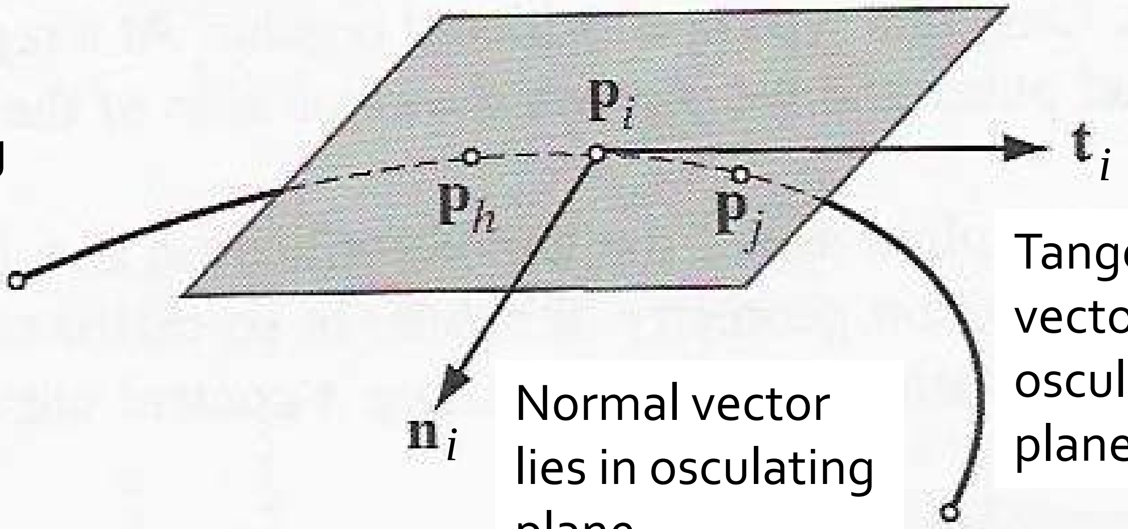
Principal normal vector is on intersection of normal plane with (osculating) plane shown in (a).

Binormal vector $\mathbf{b}_i = \mathbf{t}_i \times \mathbf{n}_i$ lies in normal plane.

source: Mortenson, p. 389-391

Osculating Plane

Limiting position of plane defined by $\mathbf{p}_i$ and two neighboring points $\mathbf{p}_j$ and $\mathbf{p}_h$ on the curve as these neighboring points independently approach $\mathbf{p}_i$.



Normal vector lies in osculating plane.

Tangent vector lies in osculating plane.

Figure 12.4 Osculating plane.

Note: $\mathbf{p}_i$, $\mathbf{p}_j$ and $\mathbf{p}_h$ cannot be collinear.

$$\begin{vmatrix} x - x_i & x_i^u & x_i^{uu} \\ y - y_i & y_i^u & y_i^{uu} \\ z - z_i & z_i^u & z_i^{uu} \end{vmatrix} = 0$$

source: Mortenson, p. 392-393

Frenet Frame

Rectifying plane at $\mathbf{p}_i$ is the plane through $\mathbf{p}_i$ and perpendicular to the principal normal $\mathbf{n}_i$:

$$(\mathbf{q}-\mathbf{p}_i)\bullet\mathbf{n}_i = 0$$

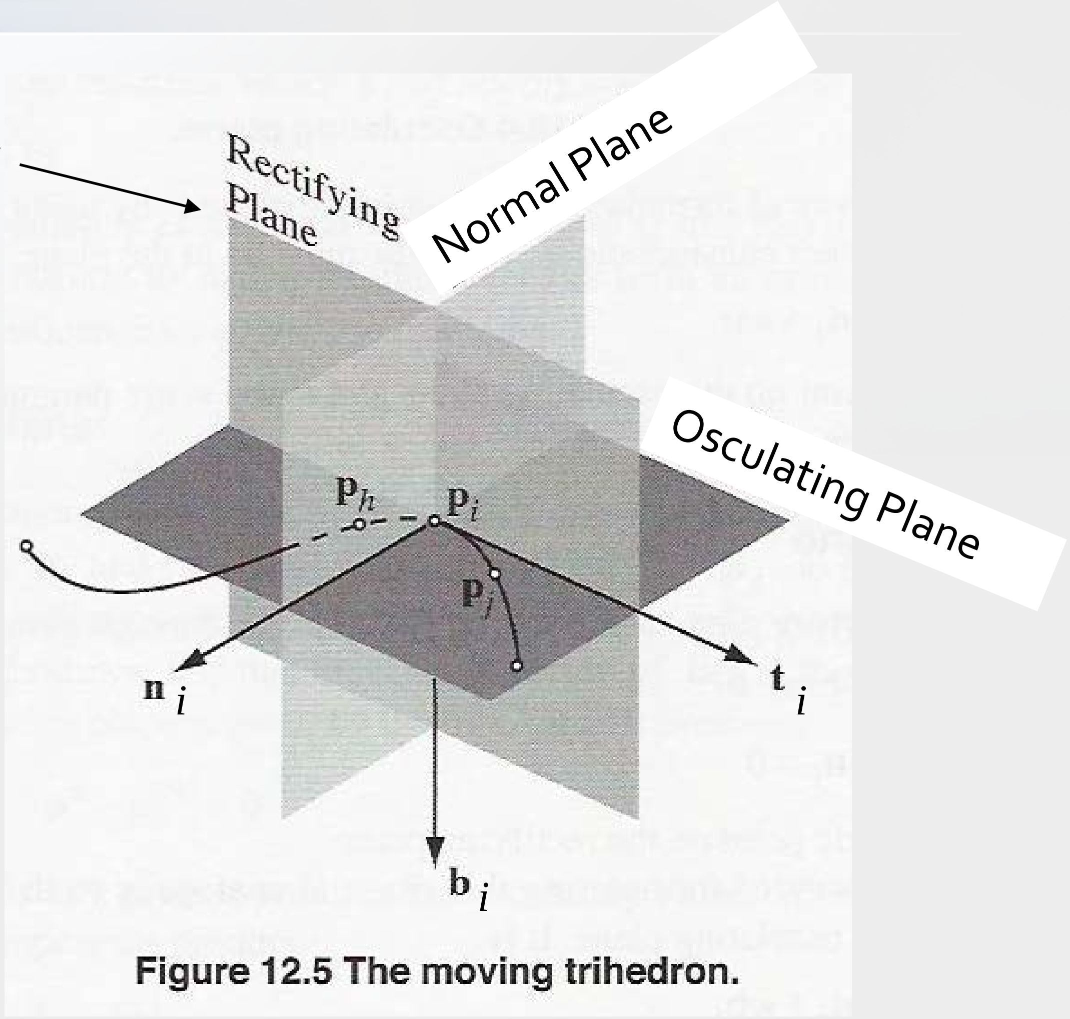


Figure 12.5 The moving trihedron.

Note changes to Mortenson's figure 12.5.

source: Mortenson, p. 393-394

Curvature

Radius of curvature is ρ_i and curvature at point $\mathbf{p}_i$ on a curve is:

$$\kappa_i = \frac{1}{\rho_i} = \frac{\left|\mathbf{p}_i^u \times \mathbf{p}_i^{uu}\right|}{\left|\mathbf{p}_i^u\right|^3}$$

Recall that vector $\mathbf{p}_i^{uu}$ lies in the osculating plane.

Curvature of a planar curve in x, y plane:

$$\frac{1}{\rho} = \frac{d^2y/dx^2}{\left[1+(dy/dx)^2\right]^{3/2}}$$

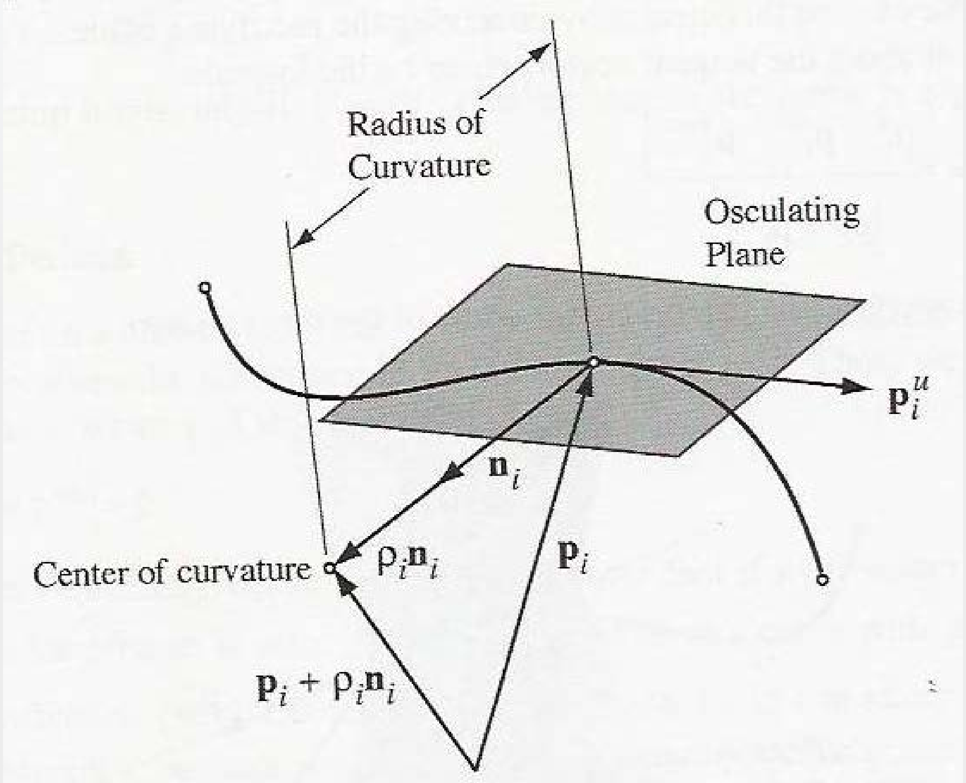


Figure 12.6 Curvature.

Curvature is *intrinsic* and does not change with a change of parameterization.

source: Mortenson, p. 394-397

Torsion

Torsion at $\mathbf{p}_i$ is limit of ratio of angle between binormal at $\mathbf{p}_i$ and binormal at neighboring point $\mathbf{p}_h$ to arc-length of curve between $\mathbf{p}_h$ and $\mathbf{p}_i$, as $\mathbf{p}_h$ approaches $\mathbf{p}_i$ along the curve.

$$\tau_i = \frac{\left[\mathbf{p}_i^u \quad \mathbf{p}_i^{uu} \quad \mathbf{p}_i^{uuu}\right]}{\left|\mathbf{p}_i^u \times \mathbf{p}_i^{uu}\right|^2} = \frac{\mathbf{p}_i^u \bullet \left(\mathbf{p}_i^{uu} \times \mathbf{p}_i^{uuu}\right)}{\left|\mathbf{p}_i^u \times \mathbf{p}_i^{uu}\right|^2}$$

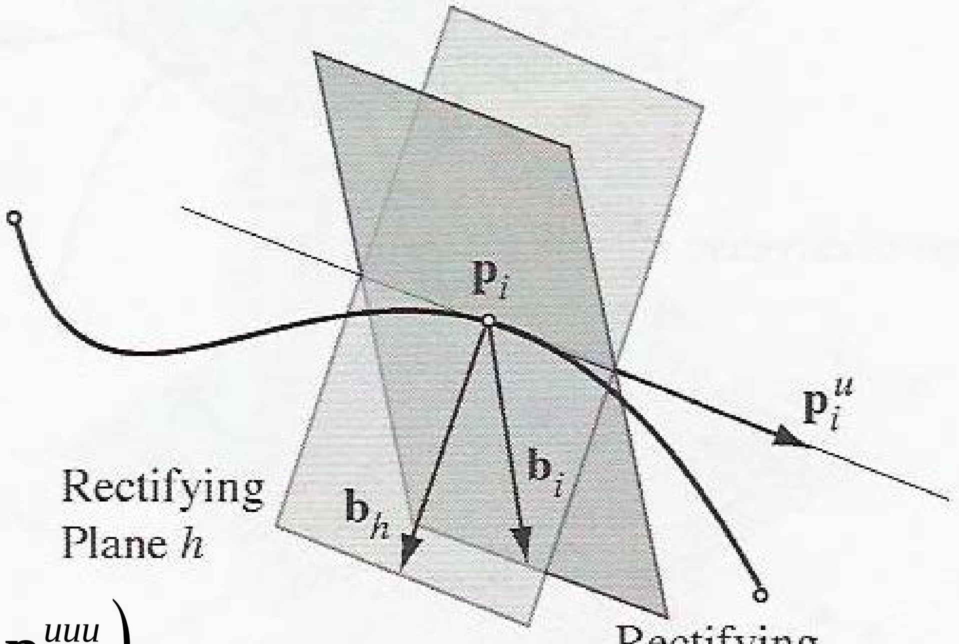


Figure 12.7 Torsion.

Torsion is *intrinsic* and does not change with a change of parameterization.

source: Mortenson, p. 394-397

Reparameterization Relationship

- Curve has G^r continuity if an arc-length reparameterization exists after which it has C^r continuity.
- This is equivalent to these 2 conditions:
  - C^{r-2} continuity of curvature
  - C^{r-3} continuity of torsion

Local properties **torsion** and **curvature** are *intrinsic* and *uniquely* determine a curve.

source: Farin, Ch 10, p.189 & Ch 11, p. 200

Local Surface Topics

- Fundamental Forms
- Tangent Plane
- Principal Curvature
- Osculating Paraboloid

source: Ch 12 Mortenson

Local Properties of a Surface

Fundamental Forms

- Given parametric surface **p**(u,w)
- Form I:

$$d\mathbf{p}\bullet d\mathbf{p} = Edu^2 + 2Fdudw + Gdw^2$$

$$E = \mathbf{p}^u \bullet \mathbf{p}^u \qquad F = \mathbf{p}^u \bullet \mathbf{p}^w \qquad G = \mathbf{p}^w \bullet \mathbf{p}^w$$

- Form II:

$$-d\mathbf{p}(u,w)\bullet d\mathbf{n}(u,w) = Ldu^2 + 2Mdudw + Ndw^2$$

$$L = \mathbf{p}^{uu} \bullet \mathbf{n} \qquad M = \mathbf{p}^{uw} \bullet \mathbf{n} \qquad N = \mathbf{p}^{ww} \bullet \mathbf{n} \qquad \mathbf{n} = \frac{\mathbf{p}^u \times \mathbf{p}^w}{\left|\mathbf{p}^u \times \mathbf{p}^w\right|}$$

- Useful for calculating arc length of a curve on a surface, surface area, curvature, etc.

Local properties first and second fundamental forms are *intrinsic* and *uniquely* determine a surface.

source: Mortenson, p. 404-405

Local Properties of a Surface
Tangent Plane

$$\mathbf{p}^u = \partial \mathbf{p}(u,w) / \partial u$$

$$\mathbf{p}^w = \partial \mathbf{p}(u,w) / \partial w$$

$$(\mathbf{q}-\mathbf{p}) \bullet (\mathbf{p}^u \times \mathbf{p}^w) = 0$$

$$\begin{vmatrix} x - x_i & x_i^u & x_i^w \\ y - y_i & y_i^u & y_i^w \\ z - z_i & z_i^u & z_i^w \end{vmatrix} = 0$$

q $\quad$ $\mathbf{p}(u_i,w_i)$ $\quad$ components of parametric tangent vectors $\mathbf{p}^u(u_i,w_i)$ and $\mathbf{p}^w(u_i,w_i)$

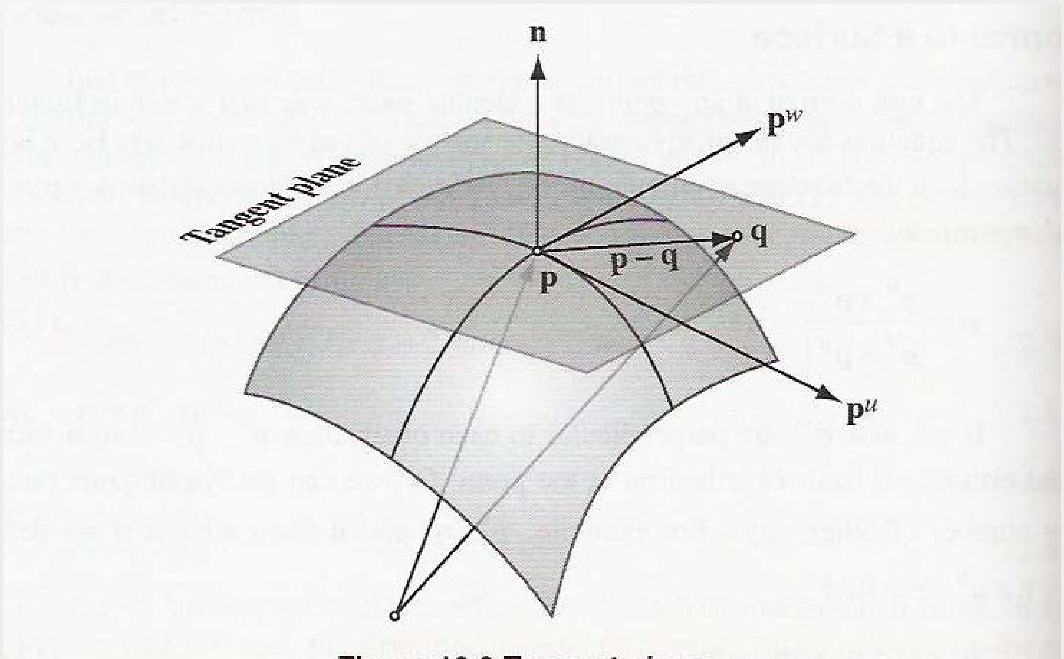


Figure 12.9 Tangent plane.

source: Mortenson, p. 406

Local Properties of a Surface

Principal Curvature

Derive **curvature** of all parametric curves ***C*** on parametric surface ***S*** passing through point **p** with same tangent line ***l*** at **p**.

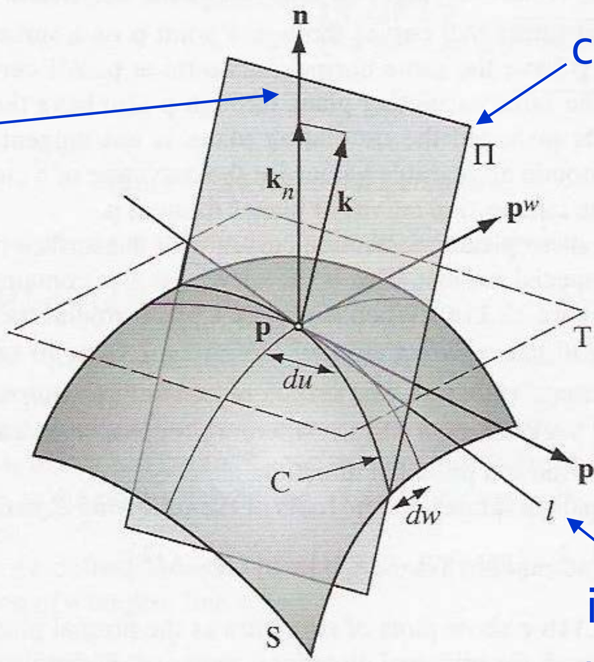


Figure 12.10 Normal curvature.

contains l

normal curvature vector $\mathbf{k}_n$ = projection of curvature vector **k** onto **n** at **p**

in tangent plane with parametric direction *dw/du*

$$\mathbf{k}_n = (\mathbf{k} \bullet \mathbf{n})\mathbf{n}$$

normal curvature: $\kappa_n = \mathbf{k} \bullet \mathbf{n}$

$$\kappa_n = \frac{L(du/dt)^2 + 2M(du/dt)(dw/dt) + N(dw/dt)^2}{E(du/dt)^2 + 2F(du/dt)(dw/dt) + G(dw/dt)^2}$$

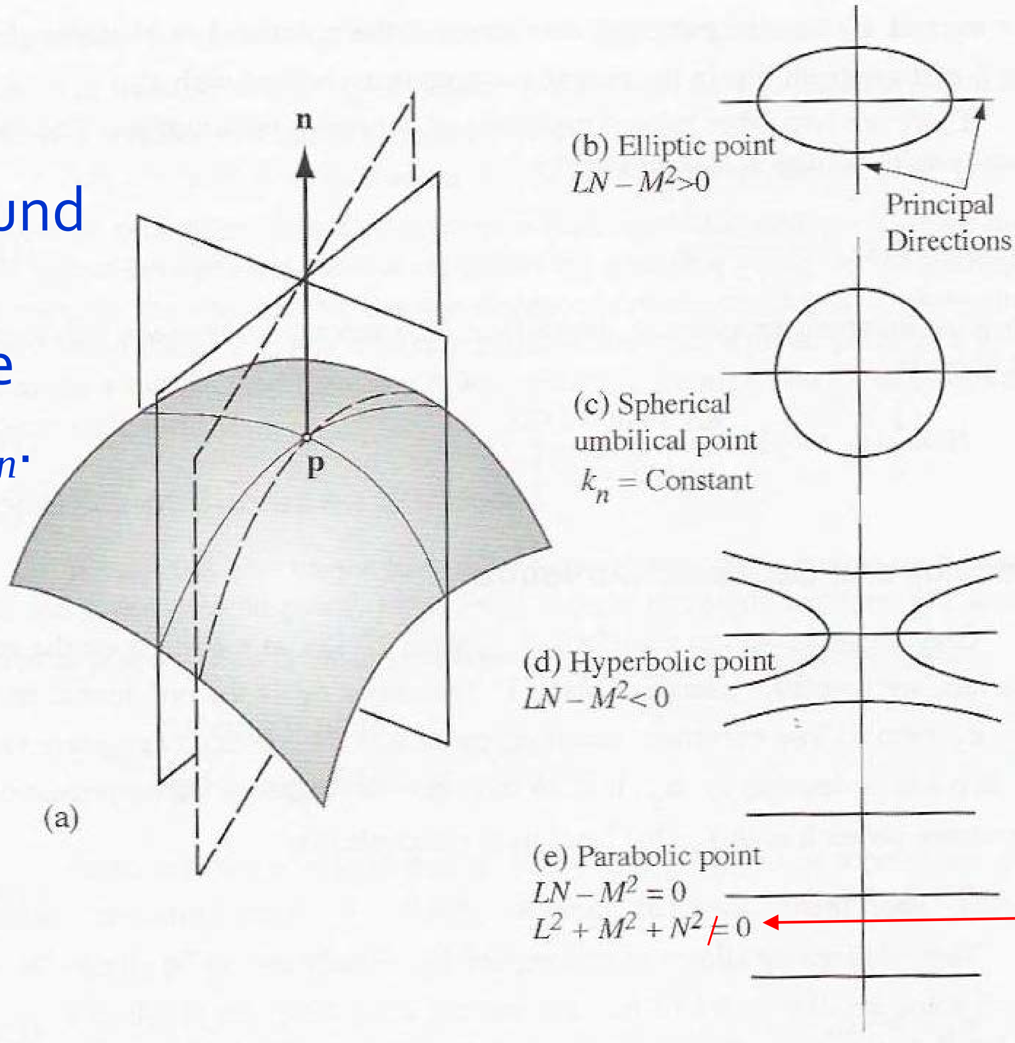
source: Mortenson, p. 407-410

Local Properties of a Surface

Principal Curvature (continued)

Rotating a plane around the normal changes the curvature κ_n.

curvature extrema: *principal normal curvatures*



typographical error?

Figure 12.11 Principal curvature.

source: Mortenson, p. 407-410

Local Properties of a Surface Osculating Paraboloid

Second fundamental form helps to measure distance of surface from tangent plane.

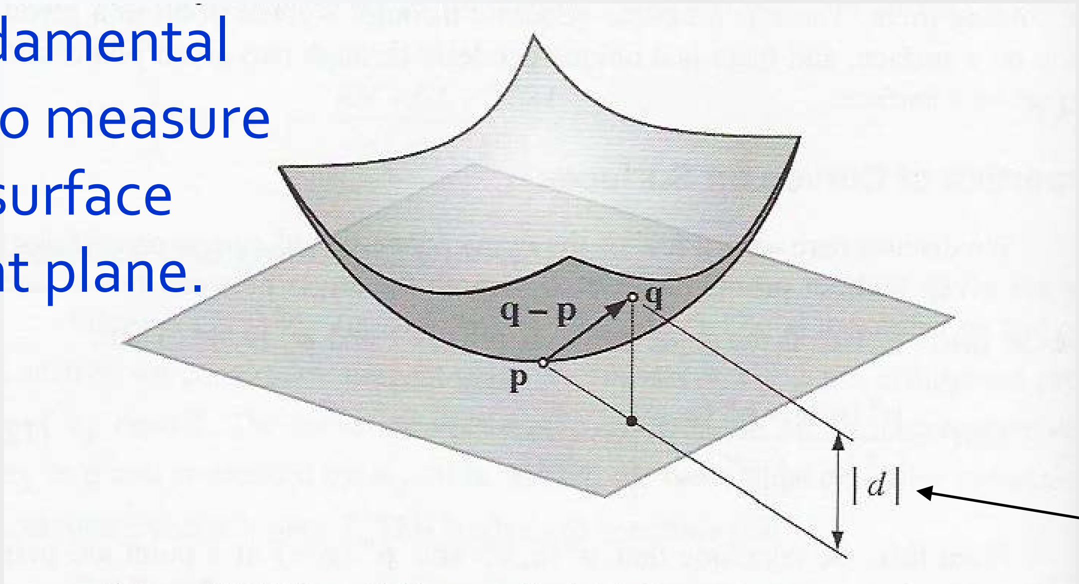


Figure 12.13 Osculating paraboloid at a point on a surface.

$|d| = (\mathbf{q} - \mathbf{p}) \bullet \mathbf{n}$

As **q** approaches **p**: $d = \underbrace{f\left[\frac{1}{2}\left(L du^2 + 2M du dw + N dw^2\right)\right]}_{\text{Osculating Paraboloid}}$

source: Mortenson, p. 412

Local Properties of a Surface

Local Surface Characterization

$$a)\, LN - M^2 > 0$$

Elliptic Point: locally convex

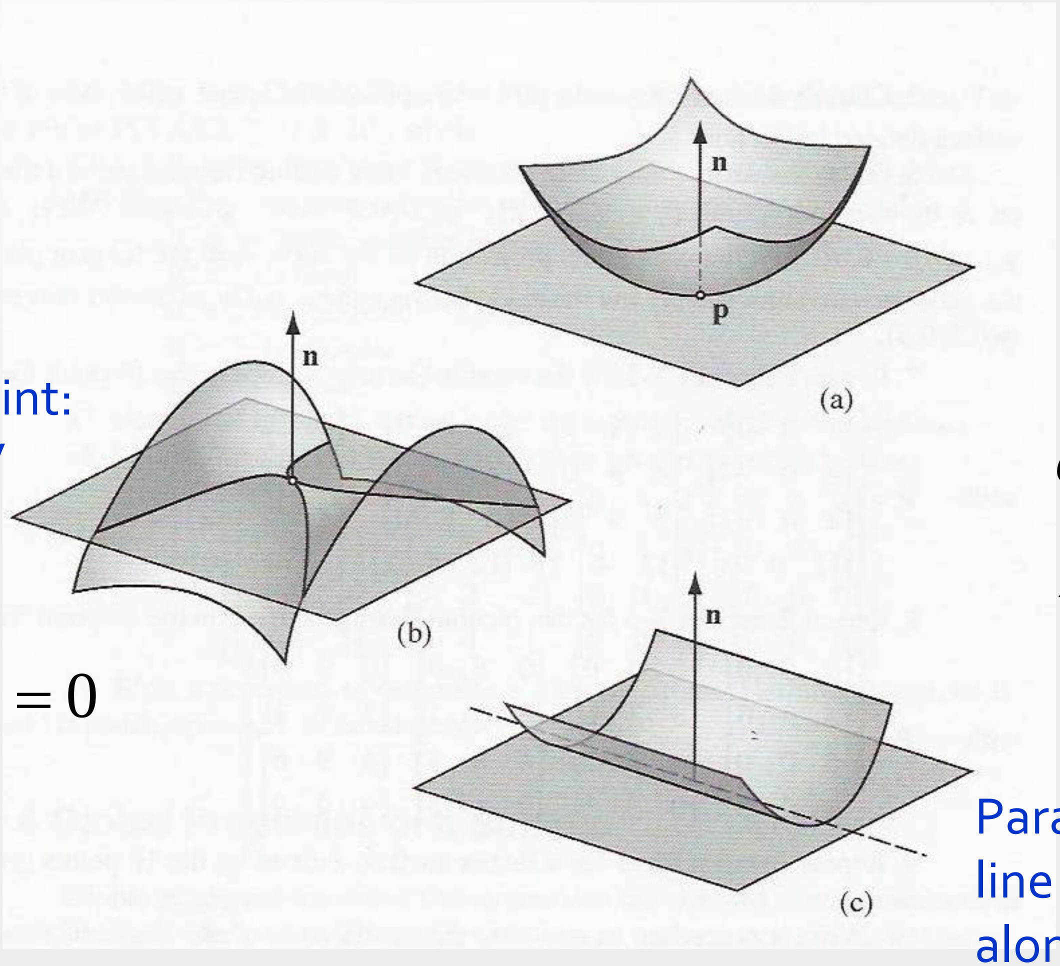
$$b)\, LN - M^2$$

Hyperbolic Point: "saddle point"

$$c)\, LN - M^2 = 0$$

$$L^2 + M^2 + N^2 \neq 0$$

typographical error?



$$L = M = N = 0$$

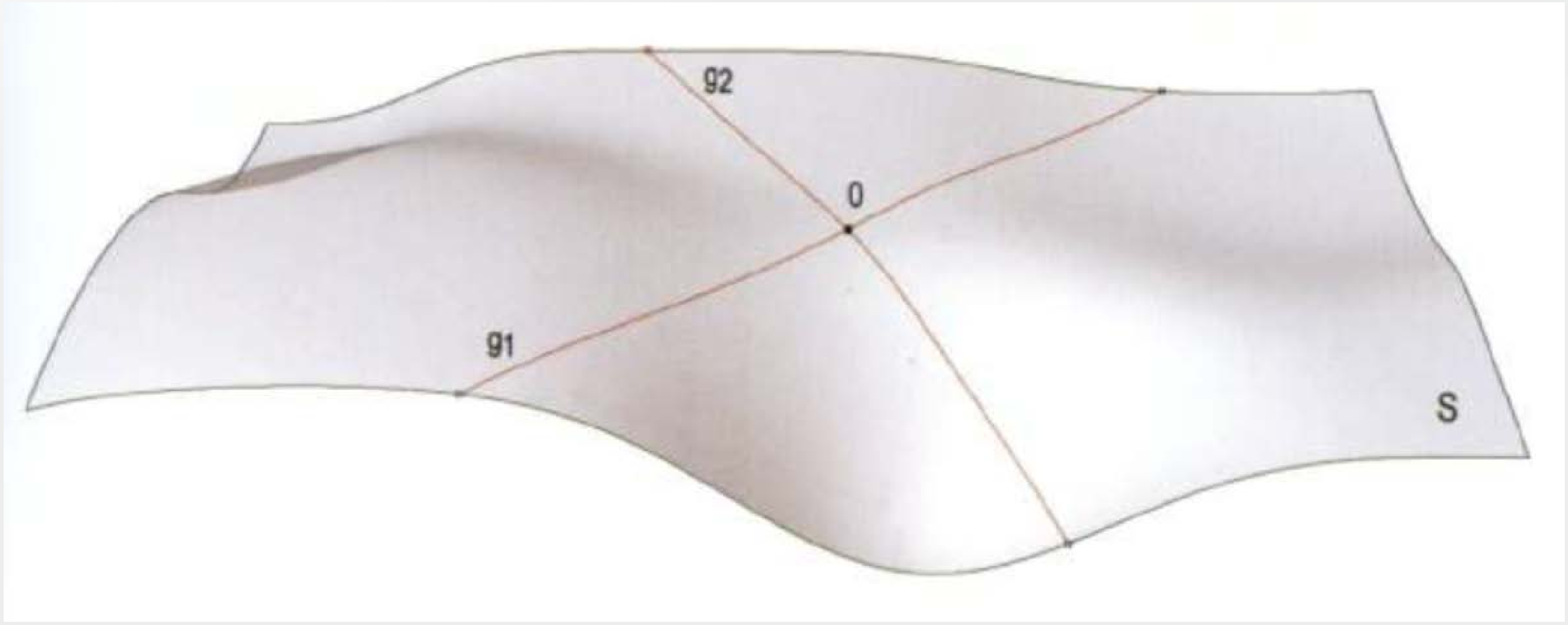
Planar Point (not shown)

Parabolic Point: single line in tangent plane along which d =o

source: Mortenson, p. 412-413

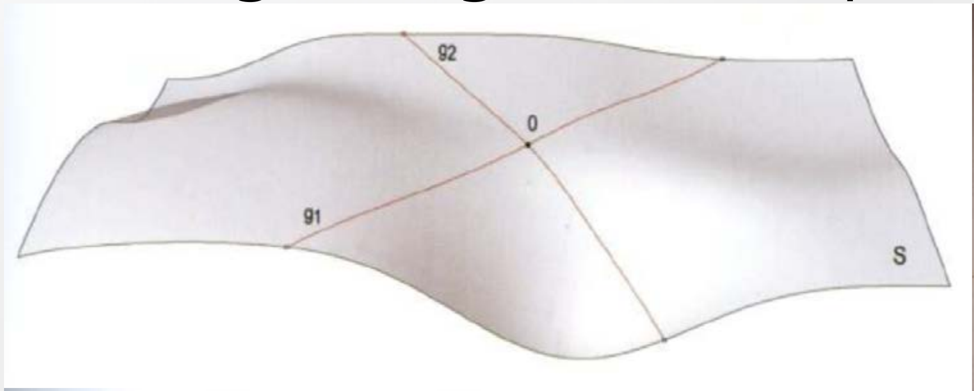
Chebyshev-net of equidistant points generated on surface. 3d Mapping on Surface

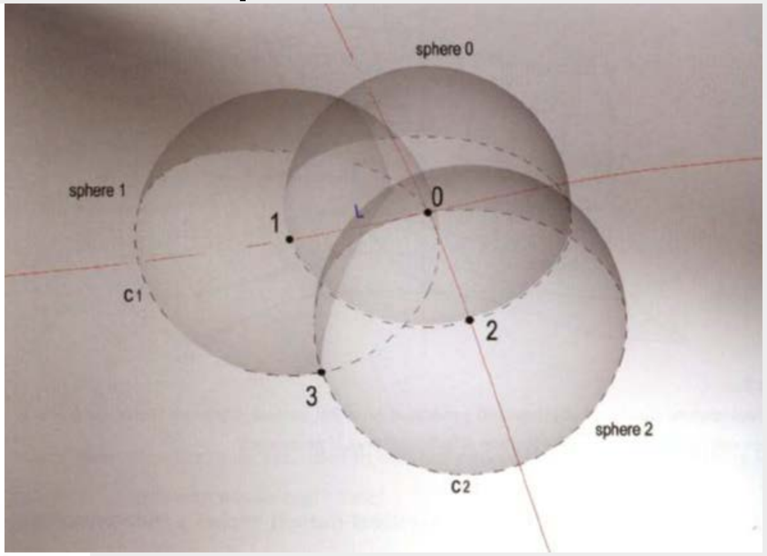
To construct a tridimensional net, two generic curves g1 and g2 are defined on a surface S and their intersection (point 0) is found.

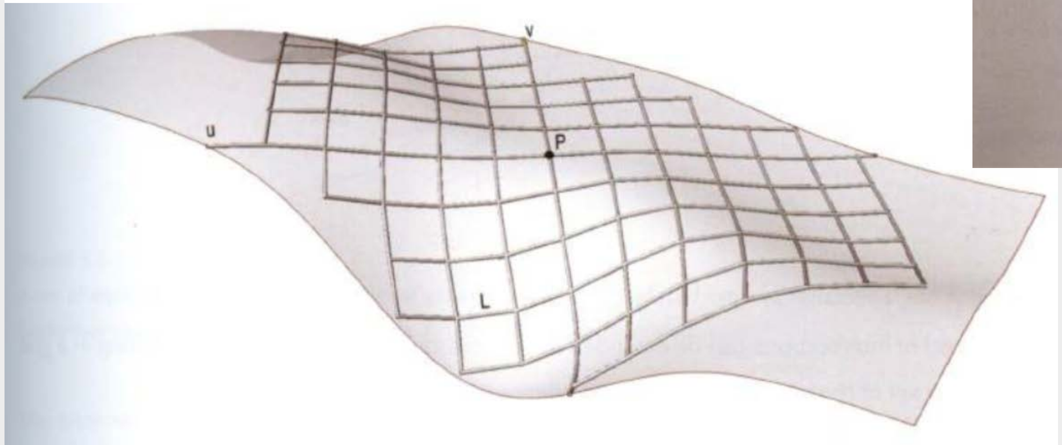


Chebyshev-net of equidistant points generated on surface. 3d Mapping on Surface

A sphere with radius (**L**) is drawn and intersected with curves **g1** and **g2** to define point **1** and point **2**.

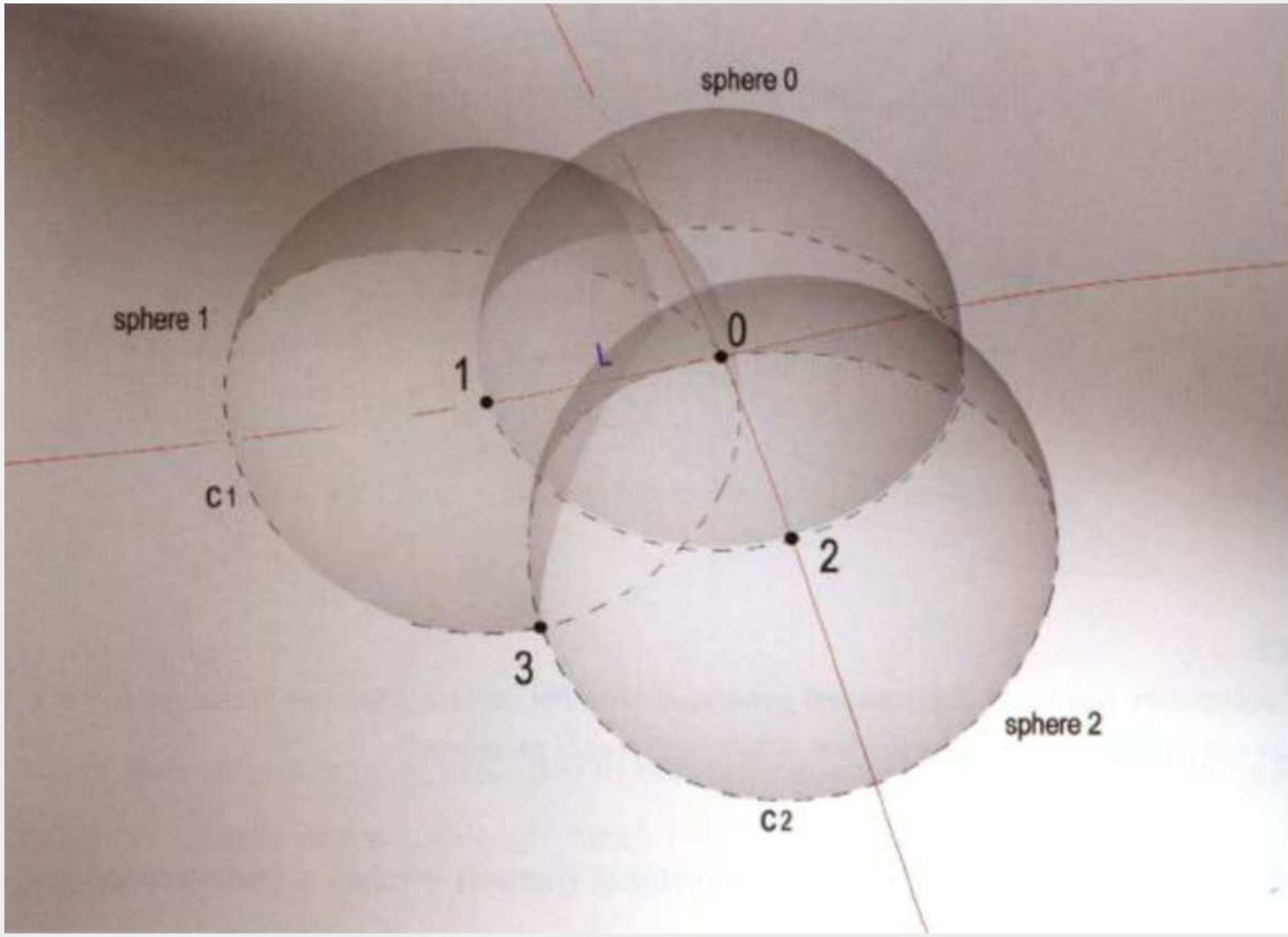




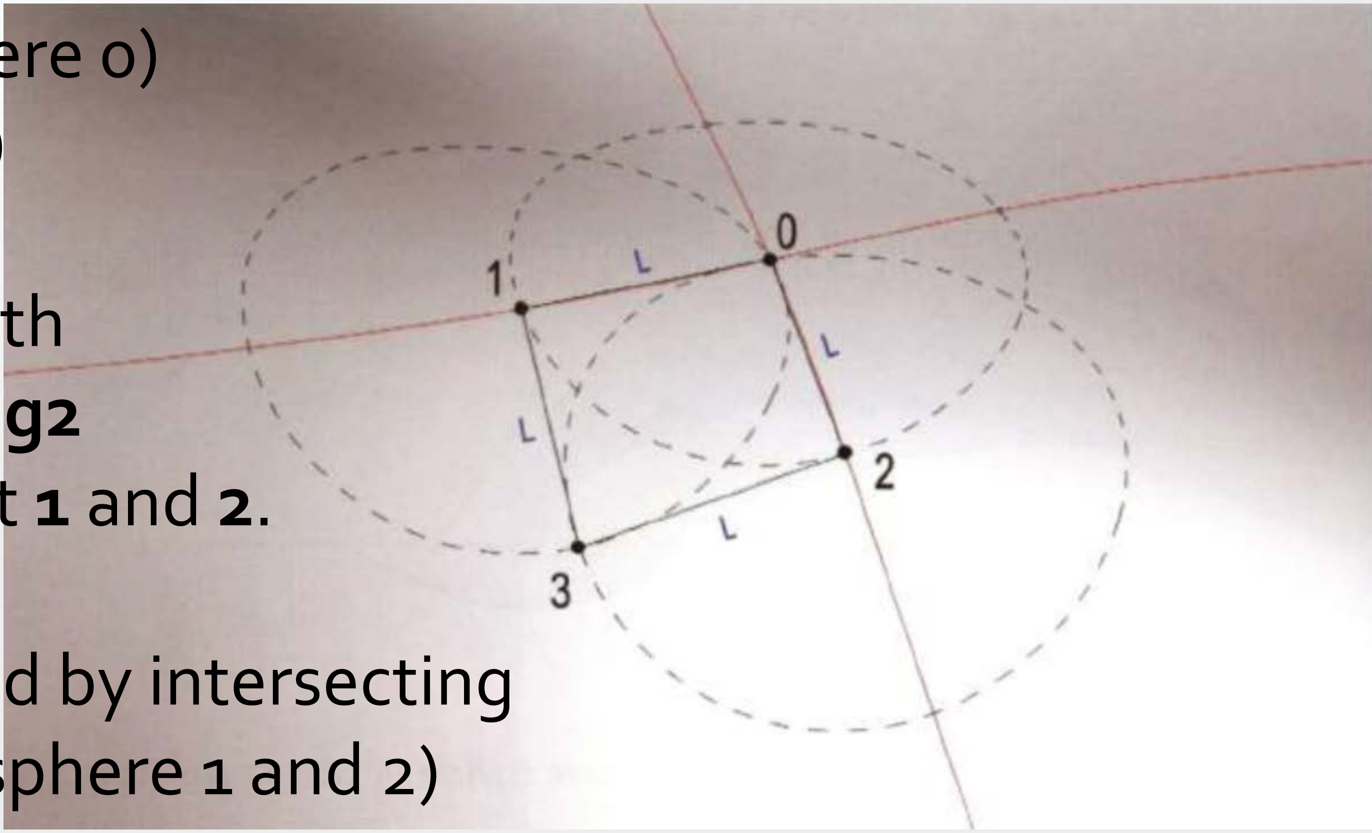


Chebyshev-net of equidistant points generated on surface. 3d Mapping on Surface

Tridimensional Geometric Construction uses a set of spheres to find equidistant points generated on surface.



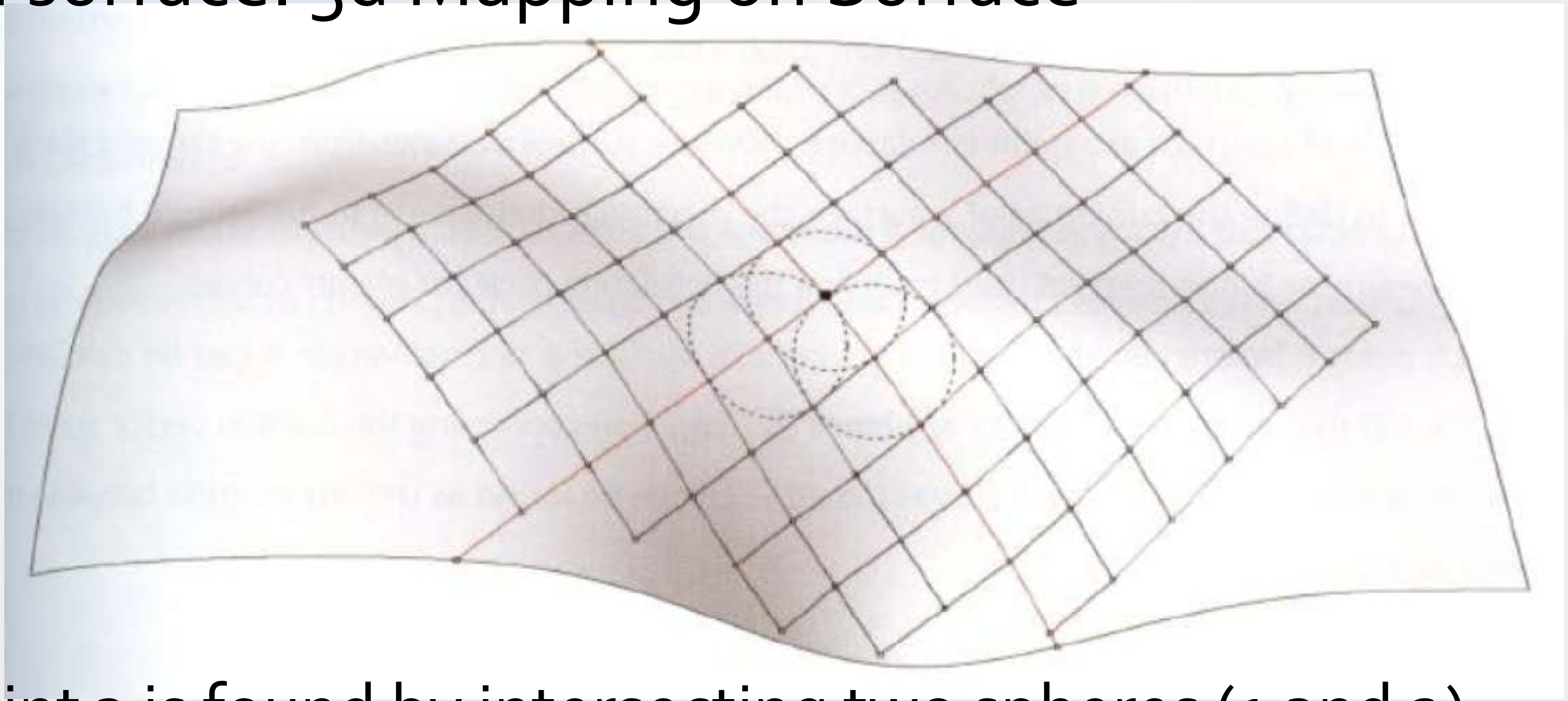
Chebyshev-net of equidistant points generated on surface. 3d Mapping on Surface



A sphere (sphere 0)
with radius (**L**)
is drawn and
intersected with
curves **g1** and **g2**
to define point **1** and **2**.

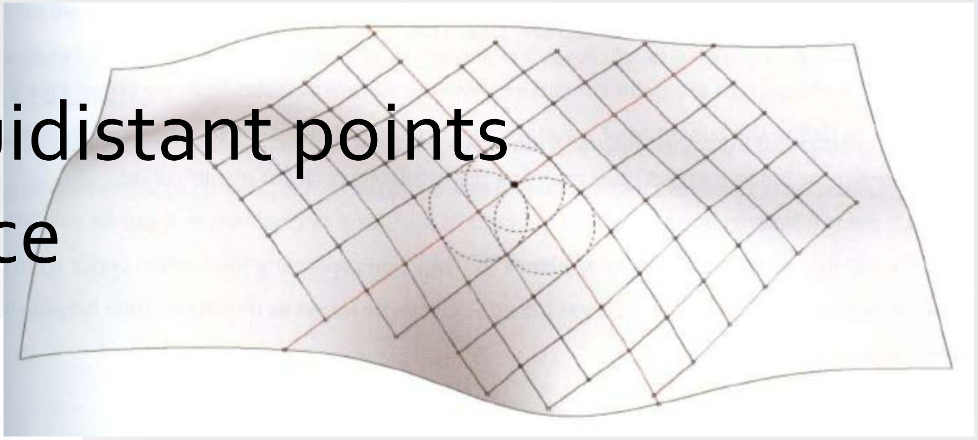
Point **3** is found by intersecting
two spheres (sphere 1 and 2)
with the surface.

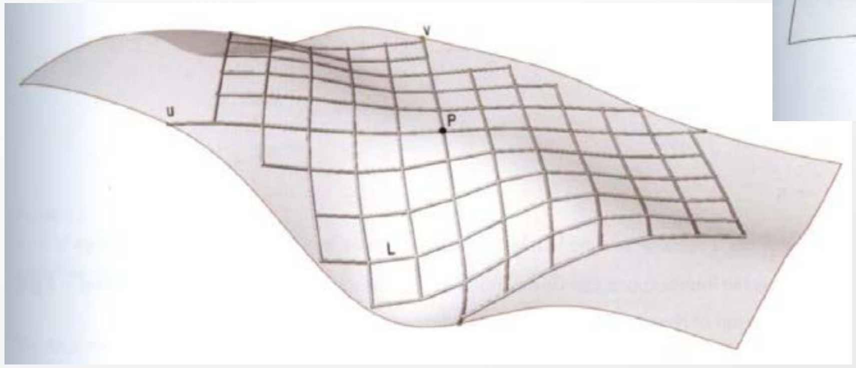
Chebyshev-net of equidistant points generated on surface. 3d Mapping on Surface

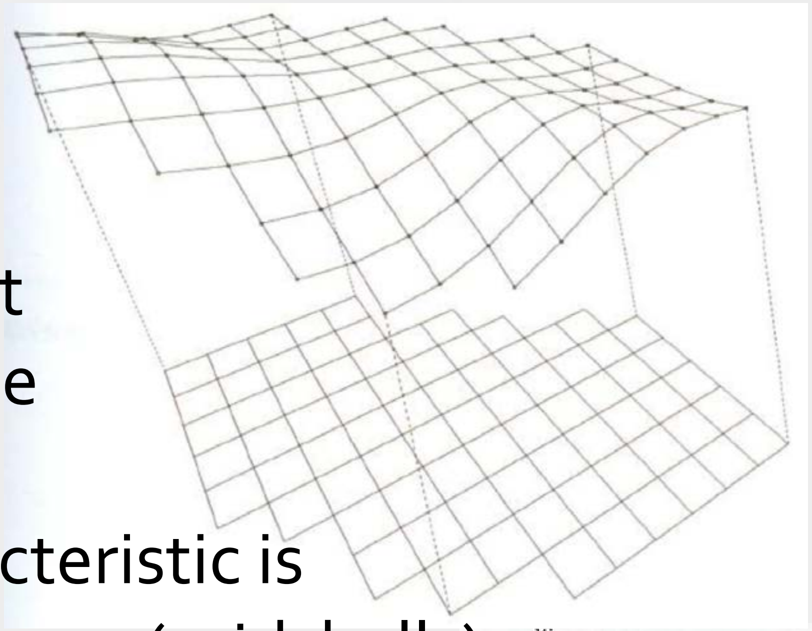


Point 3 is found by intersecting two spheres (1 and 2) with radius (L) with the surface, generating curves which are intersected to define point 3.

Chebyshev-net of equidistant points
3d Mapping on Surface







It is important to point out that an equidistant point grid can be flatten in a regular square grid as displayed below. This characteristic is crucial to form freeform structures (gridshells) starting from planar and deformable elements.