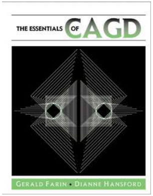
The Essentials of CAGD

Chapter 7: Working with Polynomial Patches

Gerald Farin & Dianne Hansford

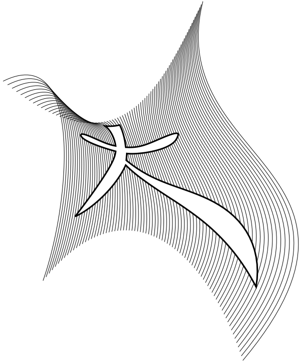
CRC Press, Taylor & Francis Group, An A K Peters Book
www.farinhansford.com/books/essentials-cagd



Outline

Introduction to Working with Polynomial Patches

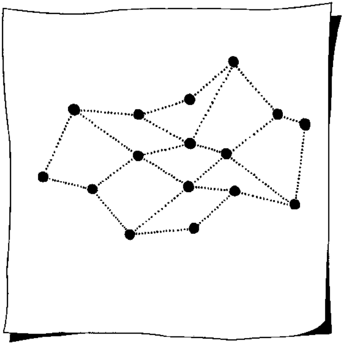
Basic surface theory ⇒ several applications



A Bézier surface trimmed by a ConS (Curve on a Surface)

Bicubic Interpolation

Given: 16 points $\mathbf{p}_{i,j}$ and associated parameter values (u_i, v_j)



Find: Interpolating bicubic patch $\mathbf{x}(u, v)$ such that

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_{0,0} & \mathbf{p}_{0,1} & \mathbf{p}_{0,2} & \mathbf{p}_{0,3} \\ \mathbf{p}_{1,0} & \mathbf{p}_{1,1} & \mathbf{p}_{1,2} & \mathbf{p}_{1,3} \\ \mathbf{p}_{2,0} & \mathbf{p}_{2,1} & \mathbf{p}_{2,2} & \mathbf{p}_{2,3} \\ \mathbf{p}_{3,0} & \mathbf{p}_{3,1} & \mathbf{p}_{3,2} & \mathbf{p}_{3,3} \end{bmatrix} = \begin{bmatrix} \mathbf{x}(u_0, v_0) & \mathbf{x}(u_0, v_1) & \mathbf{x}(u_0, v_2) & \mathbf{x}(u_0, v_3) \\ \mathbf{x}(u_1, v_0) & \mathbf{x}(u_1, v_1) & \mathbf{x}(u_1, v_2) & \mathbf{x}(u_1, v_3) \\ \mathbf{x}(u_2, v_0) & \mathbf{x}(u_2, v_1) & \mathbf{x}(u_2, v_2) & \mathbf{x}(u_2, v_3) \\ \mathbf{x}(u_3, v_0) & \mathbf{x}(u_3, v_1) & \mathbf{x}(u_3, v_2) & \mathbf{x}(u_3, v_3) \end{bmatrix}$$

Bicubic Interpolation

Recall that

$$\mathbf{x}(u_1, v_2) = \begin{bmatrix} B_0^3(u_1) & B_1^3(u_1) & B_2^3(u_1) & B_3^3(u_1) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} & \mathbf{b}_{0,2} & \mathbf{b}_{0,3} \\ \mathbf{b}_{1,0} & \mathbf{b}_{1,1} & \mathbf{b}_{1,2} & \mathbf{b}_{1,3} \\ \mathbf{b}_{2,0} & \mathbf{b}_{2,1} & \mathbf{b}_{2,2} & \mathbf{b}_{2,3} \\ \mathbf{b}_{3,0} & \mathbf{b}_{3,1} & \mathbf{b}_{3,2} & \mathbf{b}_{3,3} \end{bmatrix} \begin{bmatrix} B_0^3(v_2) \\ B_1^3(v_2) \\ B_2^3(v_2) \\ B_3^3(v_2) \end{bmatrix}$$

Bicubic Interpolation

Interpolation problem written as

$$\mathbf{P} = M^{\mathrm{T}} \mathbf{B} N$$

$$M^{\mathrm{T}} = \begin{bmatrix} B_0^3(u_0) & B_1^3(u_0) & B_2^3(u_0) & B_3^3(u_0) \\ B_0^3(u_1) & B_1^3(u_1) & B_2^3(u_1) & B_3^3(u_1) \\ B_0^3(u_2) & B_1^3(u_2) & B_2^3(u_2) & B_3^3(u_2) \\ B_0^3(u_3) & B_1^3(u_3) & B_2^3(u_3) & B_3^3(u_3) \end{bmatrix}$$

$$N = \begin{bmatrix} B_0^3(v_0) & B_0^3(v_1) & B_0^3(v_2) & B_0^3(v_3) \\ B_1^3(v_0) & B_1^3(v_1) & B_1^3(v_2) & B_1^3(v_3) \\ B_2^3(v_0) & B_2^3(v_1) & B_2^3(v_2) & B_2^3(v_3) \\ B_3^3(v_0) & B_3^3(v_1) & B_3^3(v_2) & B_3^3(v_3) \end{bmatrix}$$

Bicubic Interpolation

$$\mathbf{P} = M^{\mathrm{T}}\mathbf{B}N$$

Tensor Product Approach: Decompose into a sequence of linear systems

$$\mathbf{P} = \mathbf{C}N \quad \text{then} \quad \mathbf{C} = M^{\mathrm{T}}\mathbf{B}$$

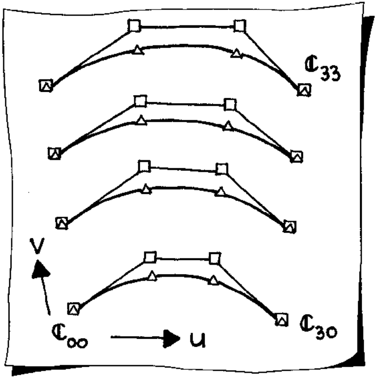
Solve two sets of four systems – Example from each:

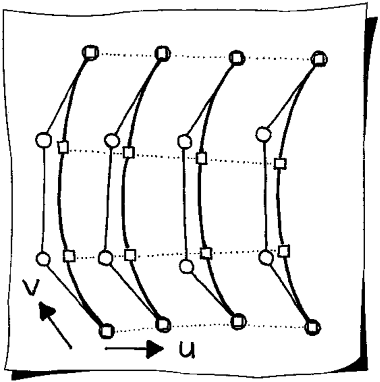
$$\begin{bmatrix}\mathbf{p}_{1,0} & \mathbf{p}_{1,1} & \mathbf{p}_{1,2} & \mathbf{p}_{1,3}\end{bmatrix} = \begin{bmatrix}\mathbf{c}_{1,0} & \mathbf{c}_{1,1} & \mathbf{c}_{1,2} & \mathbf{c}_{1,3}\end{bmatrix} N$$

$$\begin{bmatrix}\mathbf{c}_{0,1}\\ \mathbf{c}_{1,1}\\ \mathbf{c}_{2,1}\\ \mathbf{c}_{3,1}\end{bmatrix} = M^{\mathrm{T}} \begin{bmatrix}\mathbf{b}_{0,1}\\ \mathbf{b}_{1,1}\\ \mathbf{b}_{2,1}\\ \mathbf{b}_{3,1}\end{bmatrix}$$

Bicubic Interpolation

Given $\mathbf{p}_{i,j}$ depicted as triangles





Step 1: $\mathbf{P} = \mathbf{C}N$

Step 2: $\mathbf{C} = M^{\mathrm{T}}\mathbf{B}$

Sketch error: the u- and v- parameter directions need to be reversed.

Bicubic Interpolation

Direct approach:

$$\mathbf{B} = (M^{\mathrm{T}})^{-1}\mathbf{P}N^{-1}$$

The *tensor product* approach is more efficient
– Important for larger problems

Bicubic Interpolation

Standard parameter selection:

$$(u_0, u_1, u_2, u_3) = (v_0, v_1, v_2, v_3) = (0, 1/3, 2/3, 1)$$

Different values *might* improve the result
– Requires effort
– Must define *improve*

Interpolation using Higher Degrees

Given: array of points with associated parameter values (u_i, v_j)

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_{0,0} & \cdots & \mathbf{p}_{0,n} \\ \vdots & & \vdots \\ \mathbf{p}_{m,0} & \cdots & \mathbf{p}_{m,n} \end{bmatrix}$$

Find: a Bézier patch

$$\mathbf{x}(u, v) = M^{\mathrm{T}} \mathbf{B} N \quad \text{such that} \quad \mathbf{x}(u_i, v_j) = \mathbf{p}_{i,j}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_{0,0} & \cdots & \mathbf{b}_{0,n} \\ \vdots & & \vdots \\ \mathbf{b}_{m,0} & \cdots & \mathbf{b}_{m,n} \end{bmatrix}$$

Interpolation using Higher Degrees

$$\mathbf{P} = M^{\mathrm{T}} \mathbf{B} N$$

Tensor product approach:

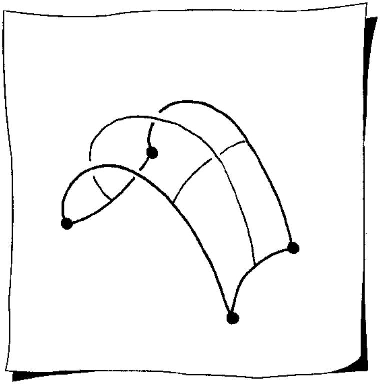
$$\mathbf{P} = \mathbf{C} N \quad \Rightarrow \quad \mathbf{C} = M^{\mathrm{T}} \mathbf{B}$$

– $m+1$ linear systems for the rows of **C**
– $n+1$ linear systems for the columns of **B**

Could use polynomials other than the Bernstein polynomials
– Obtain the same interpolating surface

High degree polynomials tend to oscillate
– Just as in the curve case

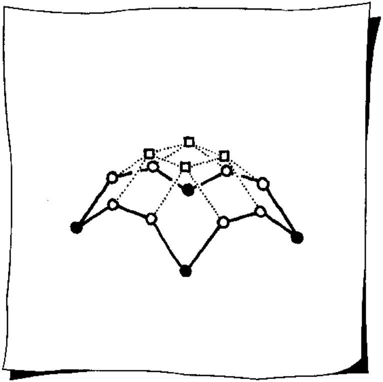
Coons Patches



A common practical situation:
– Four boundary curves of a surface designed
– Whole surface must be constructed

S. Coons developed most widely used technique in the 1960s for Ford
– Here: Boundary curves are Bézier curves

Coons Patches



Given: Four boundary polygons
As an array of points

$$\mathbf{b}_{i,j} \quad i = 0 \ldots m, \quad j = 0 \ldots n$$

Example: $m = n = 3$

$$\begin{array}{cccc} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} & \mathbf{b}_{0,2} & \mathbf{b}_{0,3} \\ \mathbf{b}_{1,0} & & & \mathbf{b}_{1,3} \\ \mathbf{b}_{2,0} & & & \mathbf{b}_{2,3} \\ \mathbf{b}_{3,0} & \mathbf{b}_{3,1} & \mathbf{b}_{3,2} & \mathbf{b}_{3,3} \end{array}$$

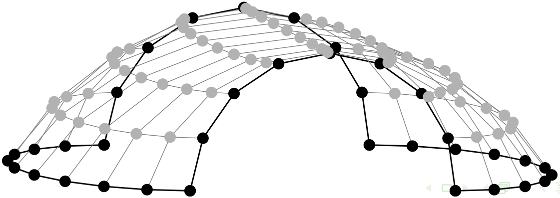
Find: Missing (four) interior points
–Depicted by squares

Coons Patches

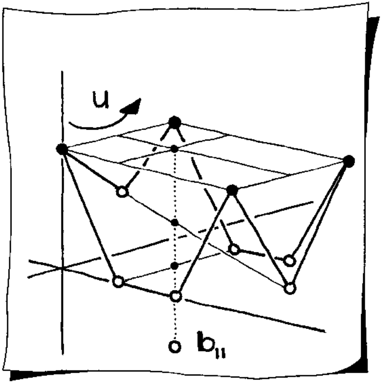
General Coons formula:

Blend of two linear interpolations and one bilinear interpolation:

$$\begin{aligned}\mathbf{b}_{i,j} &= (1-\frac{i}{m})\mathbf{b}_{0,j} + \frac{i}{m}\mathbf{b}_{m,j} \\ &+ (1-\frac{j}{n})\mathbf{b}_{i,0} + \frac{j}{n}\mathbf{b}_{i,n} \\ &- \begin{bmatrix} 1-\frac{i}{m} & \frac{i}{m} \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{0,n} \\ \mathbf{b}_{m,0} & \mathbf{b}_{m,n} \end{bmatrix} \begin{bmatrix} 1-\frac{j}{n} \\ \frac{j}{n} \end{bmatrix} \\ &\text{for} \quad i = 1 \ldots m-1 \quad \text{and} \quad j = 1 \ldots n-1\end{aligned}$$



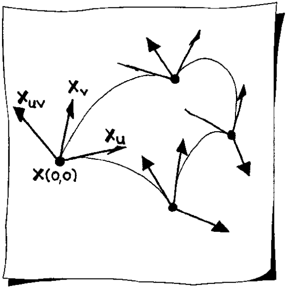
Coons Patches



Three building blocks for a Coons patch

Bicubic Hermite Interpolation

Given: points, partials, and mixed partials



Note: Some partial directions should be reversed.

$$\begin{bmatrix} \mathbf{x}(0,0) & \mathbf{x}_v(0,0) & \mathbf{x}_v(0,1) & \mathbf{x}(0,1) \\ \mathbf{x}_u(0,0) & \mathbf{x}_{uv}(0,0) & \mathbf{x}_{uv}(0,1) & \mathbf{x}_u(0,1) \\ \mathbf{x}_u(1,0) & \mathbf{x}_{uv}(1,0) & \mathbf{x}_{uv}(1,1) & \mathbf{x}_u(1,1) \\ \mathbf{x}(1,0) & \mathbf{x}_v(1,0) & \mathbf{x}_v(1,1) & \mathbf{x}(1,1) \end{bmatrix}$$

Find: Interpolating cubic Bézier patch

Bicubic Hermite Interpolation

4 patch boundaries ⇒ 4 cubic Hermite *curve* interpolation problems

$$\mathbf{b}_{0,0} = \mathbf{x}(0,0) \qquad \mathbf{b}_{3,0} = \mathbf{x}(1,0)$$
$$\mathbf{b}_{0,1} = \mathbf{b}_{0,0} + \frac{1}{3}\mathbf{x}_v(0,0) \qquad \mathbf{b}_{3,1} = \mathbf{b}_{3,0} + \frac{1}{3}\mathbf{x}_v(1,0)$$
$$\mathbf{b}_{1,0} = \mathbf{b}_{0,0} + \frac{1}{3}\mathbf{x}_u(0,0) \qquad \mathbf{b}_{2,0} = \mathbf{b}_{3,0} - \frac{1}{3}\mathbf{x}_u(1,0)$$

$$\mathbf{b}_{0,3} = \mathbf{x}(0,1) \qquad \mathbf{b}_{3,3} = \mathbf{x}(1,1)$$
$$\mathbf{b}_{0,2} = \mathbf{b}_{0,3} - \frac{1}{3}\mathbf{x}_v(0,1) \qquad \mathbf{b}_{3,2} = \mathbf{b}_{3,3} - \frac{1}{3}\mathbf{x}_v(1,1)$$
$$\mathbf{b}_{1,3} = \mathbf{b}_{0,3} + \frac{1}{3}\mathbf{x}_u(0,1) \qquad \mathbf{b}_{2,3} = \mathbf{b}_{3,3} - \frac{1}{3}\mathbf{x}_u(1,1)$$

Bicubic Hermite Interpolation

Interior control points found using the *twist* vector data

$$\mathbf{x}_{uv}(0,0) = 9[\mathbf{b}_{1,1} - \mathbf{b}_{1,0} - \mathbf{b}_{0,1} + \mathbf{b}_{0,0}]$$

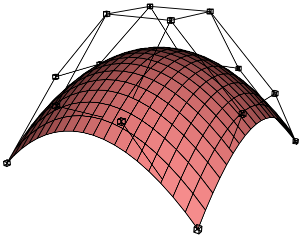
Solve for $\mathbf{b}_{1,1}$:

$$\mathbf{b}_{1,1} = \frac{1}{9}\mathbf{x}_{uv}(0,0) + \mathbf{b}_{0,1} + \mathbf{b}_{1,0} - \mathbf{b}_{0,0}$$

At other corners:

$$\mathbf{b}_{2,1} = -\frac{1}{9}\mathbf{x}_{uv}(1,0) + \mathbf{b}_{3,1} - \mathbf{b}_{3,0} + \mathbf{b}_{2,0}$$
$$\mathbf{b}_{1,2} = -\frac{1}{9}\mathbf{x}_{uv}(0,1) + \mathbf{b}_{1,3} - \mathbf{b}_{0,3} + \mathbf{b}_{0,2}$$
$$\mathbf{b}_{2,2} = \frac{1}{9}\mathbf{x}_{uv}(1,1) - \mathbf{b}_{3,3} + \mathbf{b}_{2,3} + \mathbf{b}_{3,2}$$

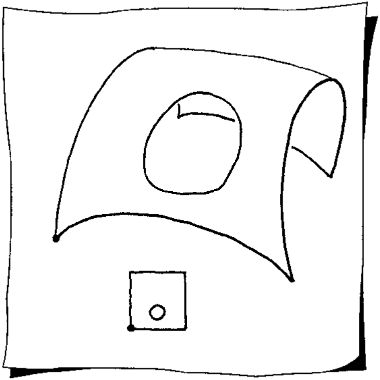
Bicubic Hermite Interpolation



A bicubic Bézier patch with zero twists

Twist data can be difficult to create
– Coons solution to given boundary data easier construction

Trimmed Patches



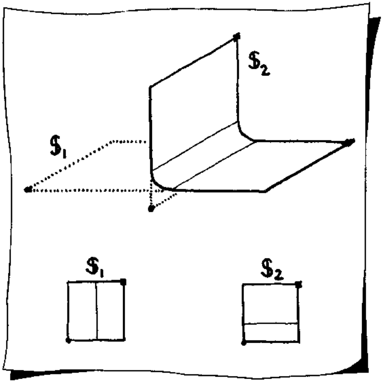
Parametric curve $(u(t), v(t))$
in the domain of surface $\mathbf{x}(u, v)$

Mapped to a
curve on the surface ConS

$$\mathbf{x}(u(t), v(t))$$

Trimmed Patches

ConS application: trimmed surfaces



Areas marked as
"invalid" or "invisible"

Example:

– Given two planes
– Blending surface between them
– Dashed parts of planes "invisible"

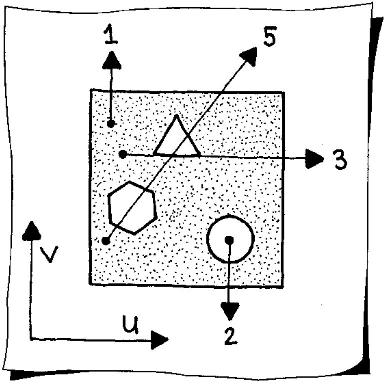
Trimmed Patches

Degree p domain curve mapped onto a degree $m \times n$ surface $\mathbf{x}(u, v)$
$\Rightarrow$ Degree $(m + n)p$ ConS (in general)

Isoparametric line in domain
$\Rightarrow$ Degree m or n isoparametric ConS

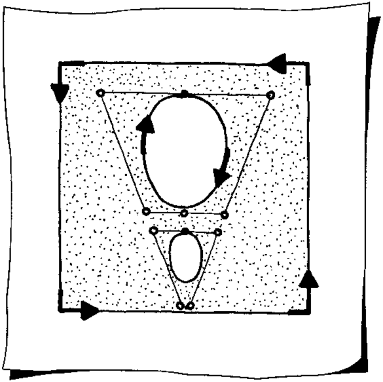
Trimmed Patches

Closed domain curve divides domain into two parts
Closed ConS divides the surface into two parts



Problem: Is domain point (u, v) inside the domain curve?
Solution:
– Construct arbitrary ray emanating from (u, v)
– Count number intersections with all domain curves and boundary
(Tangencies count as 2 intersections)
– Even: outside Odd : inside

Trimmed Patches



Orientation of trim curves

– Inside trim curves clockwise

– Outer-boundary is counterclockwise

Trimmed Patches

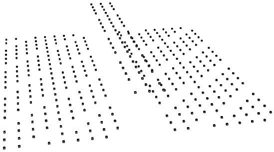
Trimmed surfaces are a bread-and-butter tool in all CAD/CAM systems

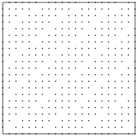
Arise in many applications

Most common: intersection between two surfaces

– Resulting intersection curve is a ConS on either of the two surfaces

Least Squares Approximation





Given: set of points
$\mathbf{p}_k \quad k = 0, \ldots K - 1$

– Not on a rectangular grid aligned with patch boundaries

Example: points from laser digitizer

– Number of points can be large

For each $\mathbf{p}_k$ need corresponding parameter pair (u_k, v_k)

Least Squares Approximation

Find a Bézier patch that fits the data as "good" as possible
– Control net coefficients $\mathbf{b}_{i,j}$ with $i = 0, \ldots, m$ and $j = 0, \ldots, n$

Use a *linearized* notation to solve the problem
– Traverse the control net row by row

$$\mathbf{x}(u,v) = \left[B_0^m(u)B_0^n(v), \ldots, B_m^m(u)B_n^n(v)\right] \begin{bmatrix} \mathbf{b}_{0,0} \\ \vdots \\ \mathbf{b}_{m,n} \end{bmatrix}$$

Best case: each data point lies on the approximating surface

$$\mathbf{p}_k = \mathbf{x}(u_k,v_k) = \left[B_0^m(u_k)B_0^n(v_k), \ldots, B_m^m(u_k)B_n^n(v_k)\right] \begin{bmatrix} \mathbf{b}_{0,0} \\ \vdots \\ \mathbf{b}_{m,n} \end{bmatrix}$$

Least Squares Approximation

Combining all K of these equations

$$\begin{bmatrix} \mathbf{p}_0 \\ \vdots \\ \vdots \\ \vdots \\ \mathbf{p}_K \end{bmatrix} = \begin{bmatrix} B_0^m(u_0)B_0^n(v_0) & \dots & B_m^m(u_0)B_n^n(v_0) \\ & \vdots & \\ & \vdots & \\ & \vdots & \\ B_0^m(u_K)B_0^n(v_K) & \dots & B_m^m(u_K)B_n^n(v_K) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} \\ \vdots \\ \mathbf{b}_{m,n} \end{bmatrix}$$

$$\mathbf{P} = M\mathbf{B}$$

K equations in $(m+1)(n+1)$ unknowns
Example: $m = n = 3$ bicubic case and $K =$ several hundred
$\Rightarrow$ 16 unknowns
$\Rightarrow$ Linear system is *overdetermined*

Least Squares Approximation

Overdetermined linear system

$$\mathbf{P} = M\mathbf{B}$$

In general no exact solution
Good approximation found by forming *normal equations*

$$M^{\mathrm{T}}\mathbf{P} = M^{\mathrm{T}}M\mathbf{B}$$

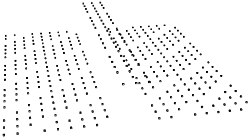
(Same procedure as for curve problem)

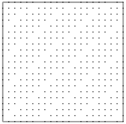
Example: bicubic case 16 equations in 16 unknowns

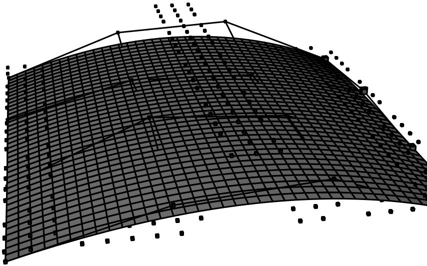
B is the least squares approximation to the given data in Bézier form

Least squares solution minimizes the sum of the squared distances of each data point to the resulting surface

Least Squares Approximation



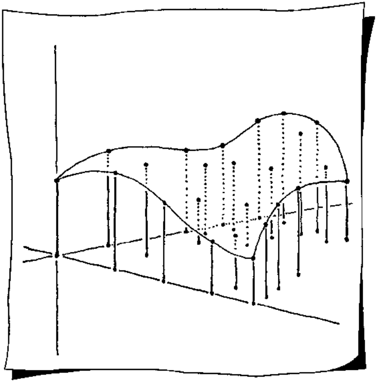




If $\#$ data points $= \#$ control points $\Rightarrow$ interpolation
(No need to form normal equations)

Least Squares Approximation

Finding parameter values



If data points can be projected into a plane:

– Example: project into $(x, y)-$plane

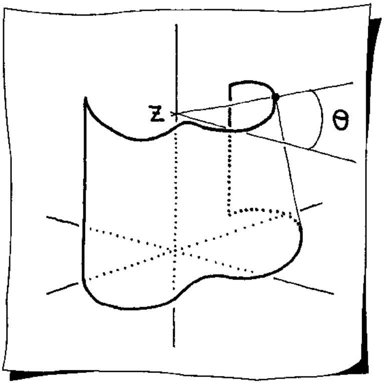
– Drop $z-$coordinate

$(u_k, v_k) = (x_k, y_k)$

– Scale to unit square

Least Squares Approximation

Finding parameter values con't



If data cannot be projected into a plane:

Look for a *basic surface*
with a known parametrization
that mimics the shape of the data

Example: a cylinder or a sphere

– Projected each point onto a cylinder
– Generates a (θ, z) parameter pair
– Scale parameters to unit square