Non-Uniform Rational B-Spline Curves and Surfaces

Hongxin Zhang and Jieqing Feng

2006-12-18

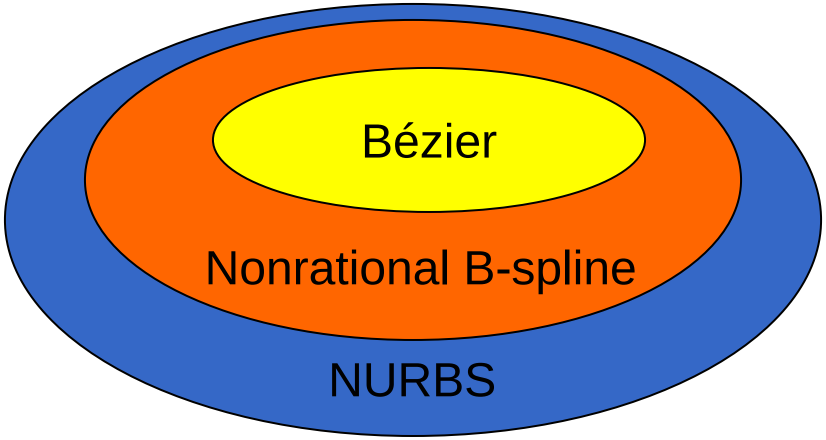
State Key Lab of CAD&CG
Zhejiang University

Contents

Rational B-Spline Curves

Rational B-spline curves – Overview

- Bézier and nonrational B-splines are a subset (special case) of rational B-splines (NURBS)
  - Bézier is a subset of nonrational B-splines
  - Non-Uniform Rational B-Spline



Rational B-spline curves – Overview

- Rational B-splines provide a single precise mathematical form for:
  - lines
  - planes
  - conic sections (circles, ellipses . . .)
  - free form curves
  - quadric surfaces
  - sculptured surfaces

Rational B-spline curves – Overview

Ken Versprille

First to discuss rational B-splines PhD dissertation at Syracuse University

Rational B-spline curves – Definition

- Defined in 4-D homogeneous coordinate space
- Projected back into 3-D physical space

In 4-D homogeneous coordinate space

$$P(t) = \sum_{i=1}^{n+1} B_i^h N_{i,k}(t)$$

where

- B_i^h are the 4-D homogeneous control vertices
- $N_{i,k}(t)$s are the nonrational B-spline basis functions
- k is the order of the basis functions

Rational B-spline curves – Definition

- Projected back into 3-D physical space

Divide through by homogeneous coordinate

$$P(t) = \frac{\sum_{i=1}^{n+1} B_i h_i N_{i,k}(t)}{\sum_{i=1}^{n+1} h_i N_{i,k}(t)} = \sum_{i=1}^{n+1} B_i R_{i,k}(t)$$

B_is are the 3-D control vertices

$$R_{i,k}(t) = \frac{h_i N_{i,k}(t)}{\sum_{i=1}^{n+1} h_i N_{i,k}(t)} \qquad h_i \geq 0$$

$R_{i,k}(t)$s are the rational B-spline basis functions

Rational B-spline curves – Properties

- $\sum_{i=1}^{n+1} R_{i,k}(t) \equiv 1$ for all t
- $R_{i,k}(t) \geqslant 0$ for all t
- $R_{i,k}(t)$, $k > 1$ has precisely one maximum
- Maximum degree $= n$, $k_{max} = n+1$
- Exhibits variation diminishing property

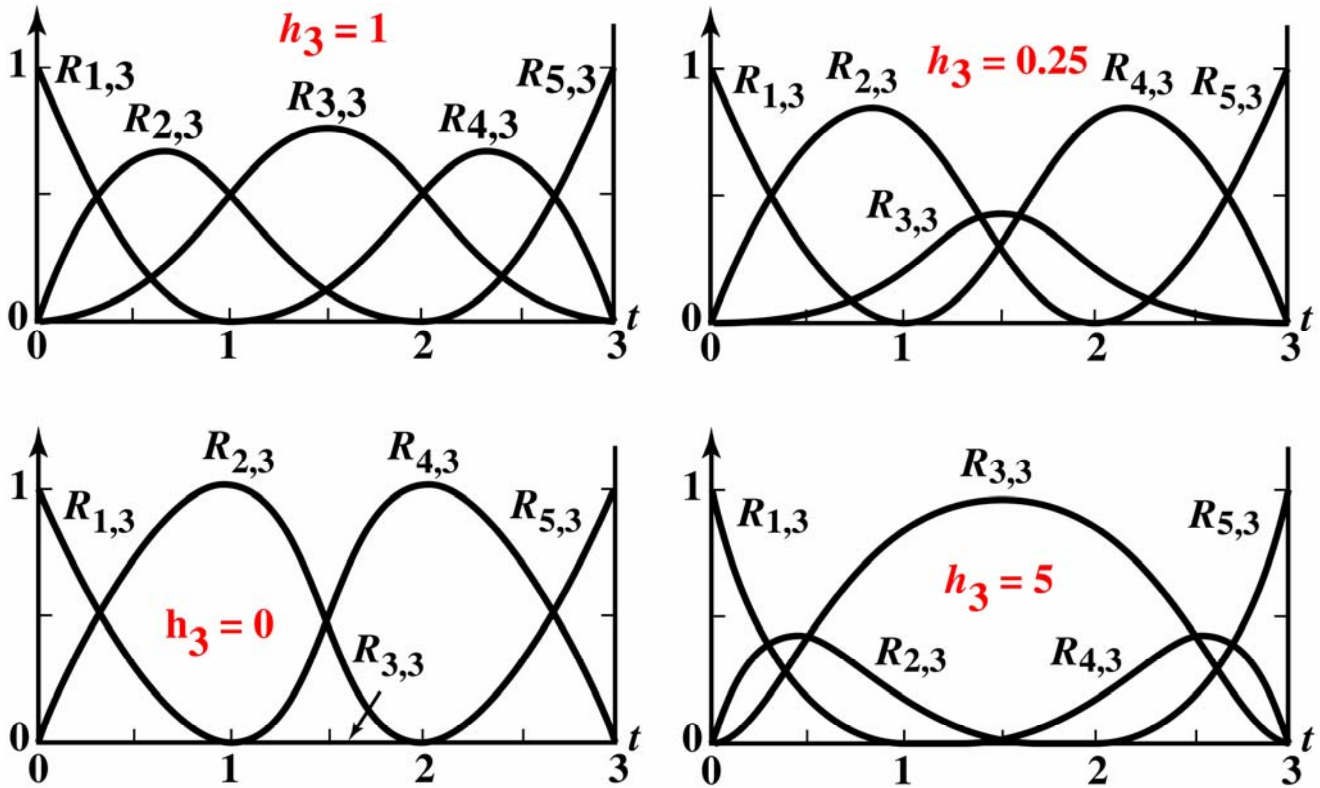
Rational B-spline curves – Properties

- Follows shape of the control polygon
- Transforms curve ←→ transforms control polygon
- Lies within union of convex hulls of k successive control vertices if $h_i>0$
- Everywhere C^{k-2} continuous

Rational B-spline basis functions

Comparisons: $n+1=5$, $k=3$

$[X]=[0\ \ 0\ \ 0\ \ 1\ \ 2\ \ 3\ \ 3\ \ 3]$, $[H] = [1\ \ 1\ \ h_3\ \ 1\ \ 1]$



Rational B-spline curves – Control

Same as nonrational B-splines

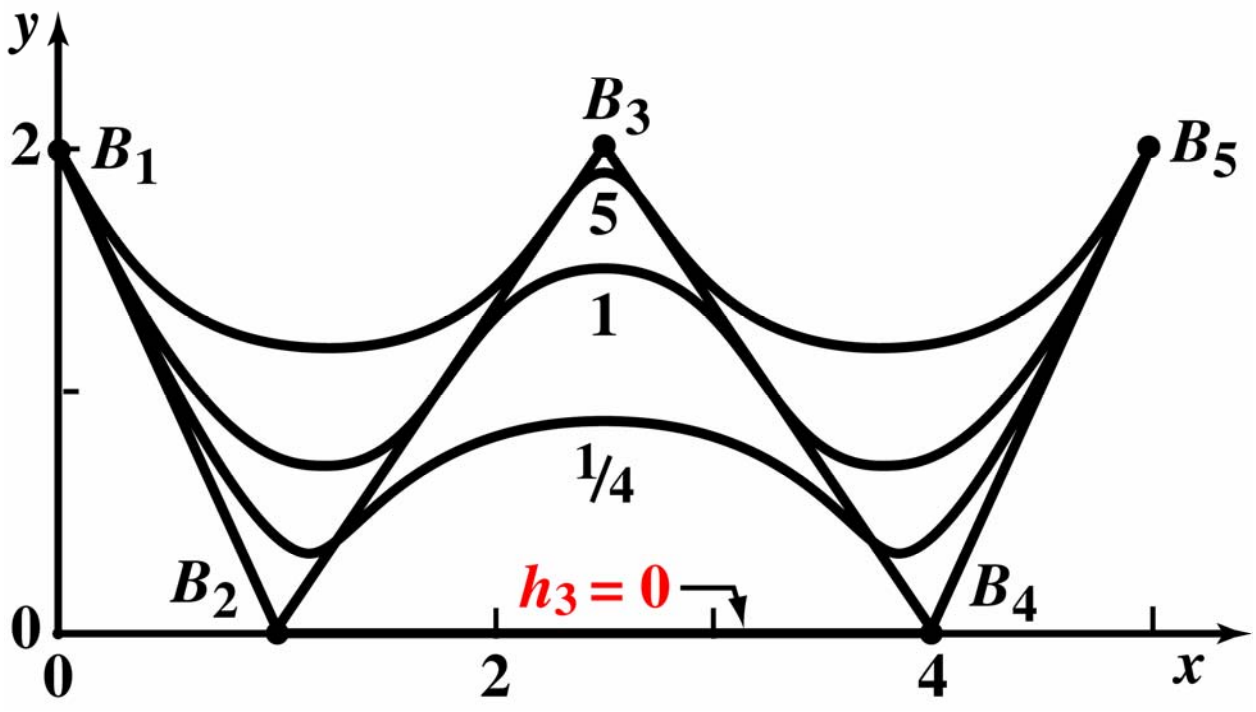
plus

Manipulation of the homogeneous weighting factor

Rational B-spline curves – Control

Homogeneous weighting factor : $n+1=5, k=3$

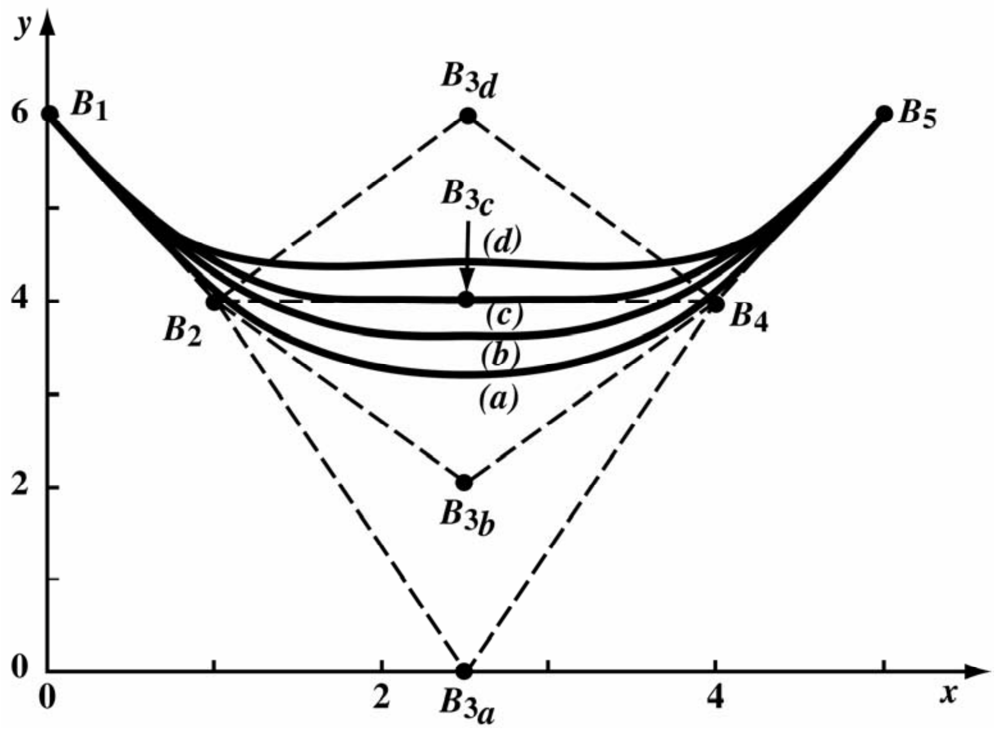
$[X]=[0\ \ 0\ \ 0\ \ 1\ \ 2\ \ 3\ \ 3\ \ 3]$ $\qquad [H]=[1\ \ 1\ \ h_3\ \ 1\ \ 1]$



Rational B-spline Curves – Control

Move single vertex, $n + 1 = 5, k = 4$

$[X] = [0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 2\ \ 2\ \ 2\ \ 2]$, $[H] = [1\ \ 1\ \ 1/4\ \ 1\ \ 1]$



Rational B-spline Curves – Control

Multiple vertices

$[X]=[0\ 0\ 0\ 0\ 1\ 2\ 2\ 2\ 2]$ $n+1=5,\ k=4$

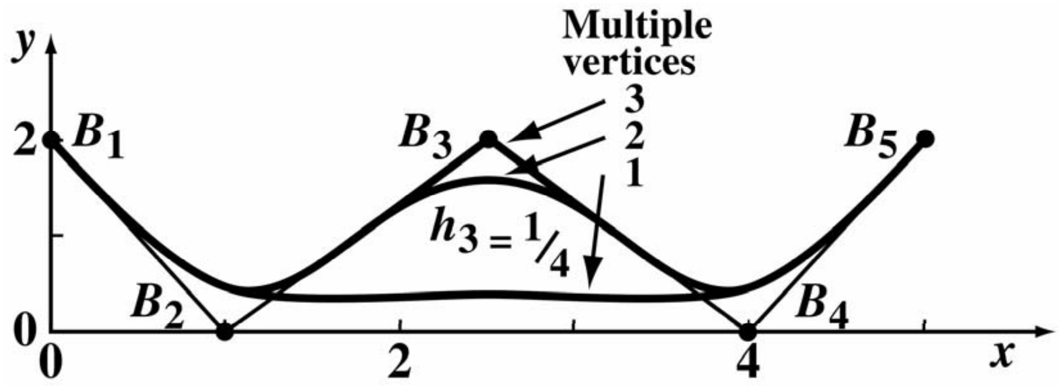
$[H]=[1\ 1\ 1/4\ 1\ 1]$ single vertex

$[X]=[0\ 0\ 0\ 0\ 1\ 2\ 3\ 3\ 3\ 3]$ $n+1=6,\ k=4$

$[H]=[1\ 1\ 1/4\ 1/4\ 1\ 1]$ double vertex

$[X]=[0\ 0\ 0\ 0\ 1\ 2\ 3\ 4\ 4\ 4\ 4]$ $n+1=7,\ k=4$

$[H]=[1\ 1\ 1/4\ 1/4\ 1/4\ 1\ 1]$ triple vertex



Rational B-spline Curves – Conic Sections

- Conic sections described by quadratic curves
- Consider quadratic rational B-spline

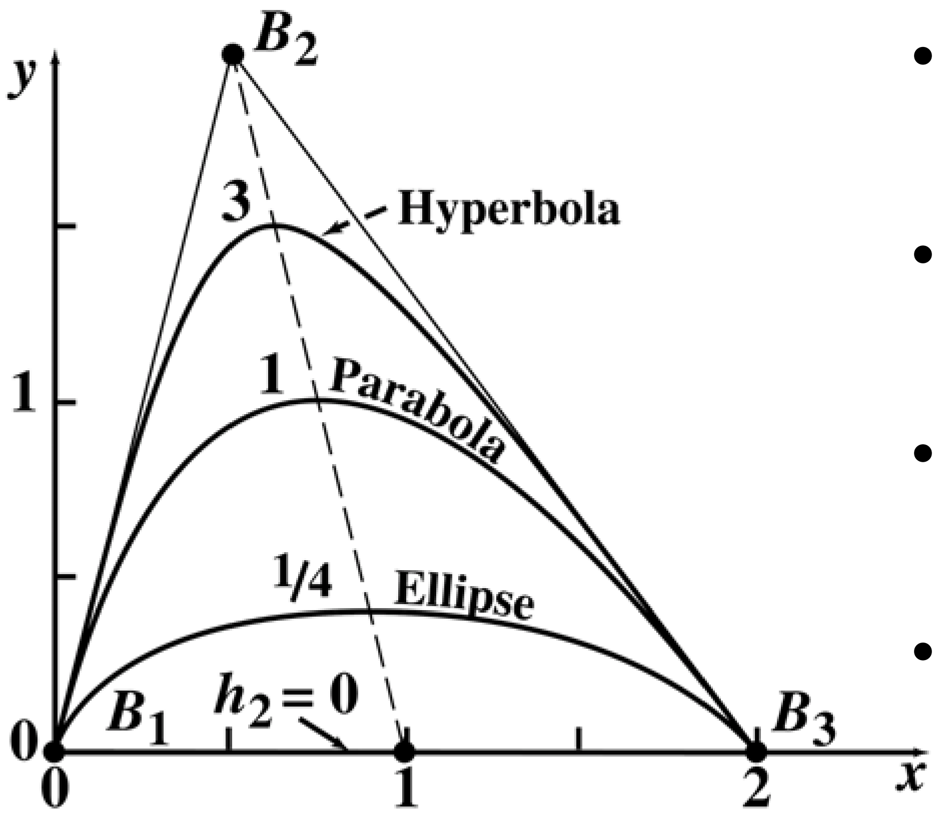
  $[X]=[0\ 0\ 0\ 1\ 1\ 1]$; $n+1=3$, $k=3$

$$P(t) = \frac{h_1 N_{1,3}(t) B_1 + h_2 N_{2,3}(t) B_2 + h_3 N_{3,3}(t) B_3}{h_1 N_{1,3}(t) + h_2 N_{2,3}(t) + h_3 N_{3,3}(t)}$$

- A third-order rational Bézier curve
- Convenient to assume $h_1 = h_3 = 1$

$$P(t) = \frac{N_{1,3}(t) B_1 + h_2 N_{2,3}(t) B_2 + N_{3,3}(t) B_3}{N_{1,3}(t) + h_2 N_{2,3}(t) + N_{3,3}(t)}$$

Rational B-spline Curves – Conic Sections



- $h_2 = 0$
  a straight line
- $0 < h_2 < 1$
  an elliptic curve segment
- $h_2 = 1$
  a parabolic curve segment
- $h_2 > 1$
  a hyperbolic curve segment

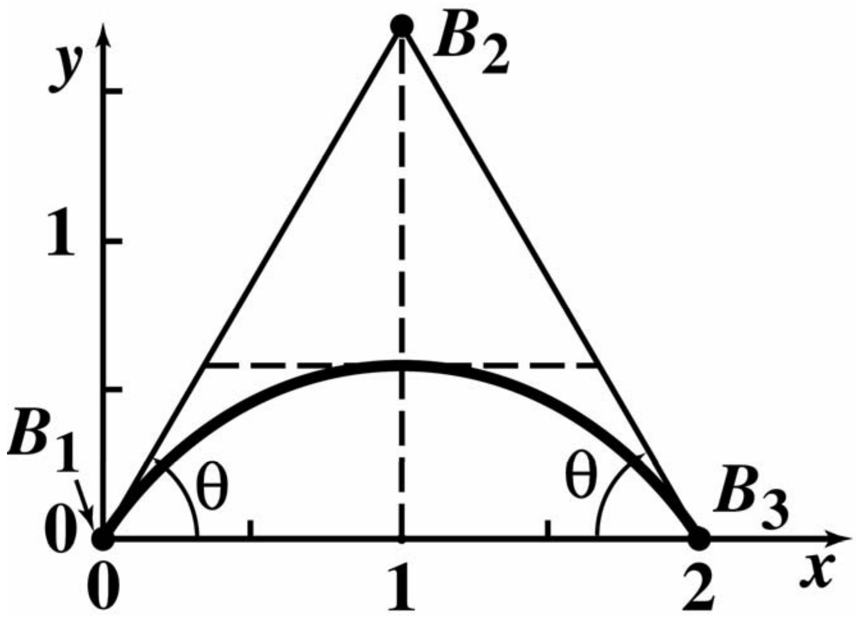
Rational B-spline Curves – Circles

Control vertices form isosceles triangle

Multiple internal knot values

Specific value of the homogeneous weight, $h_2 = ½$

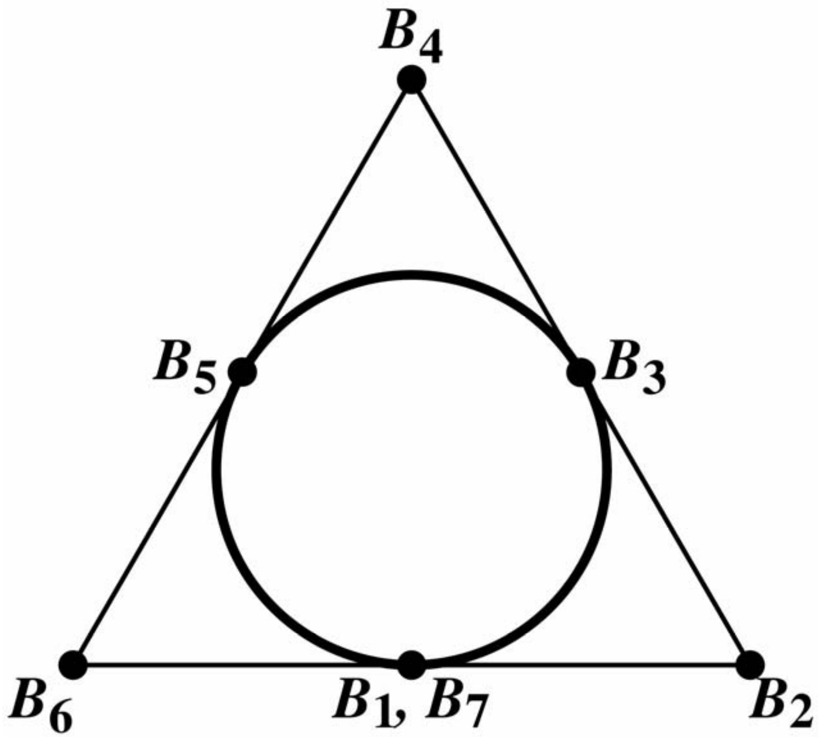
$n + 1 = 3$, $k = 3$, $[X]=[0\ 0\ 0\ 1\ 1\ 1]$, $[H] = [1\ 1/2\ 1\ ]$



Rational B-spline Curves – Circles

Three 120° arcs

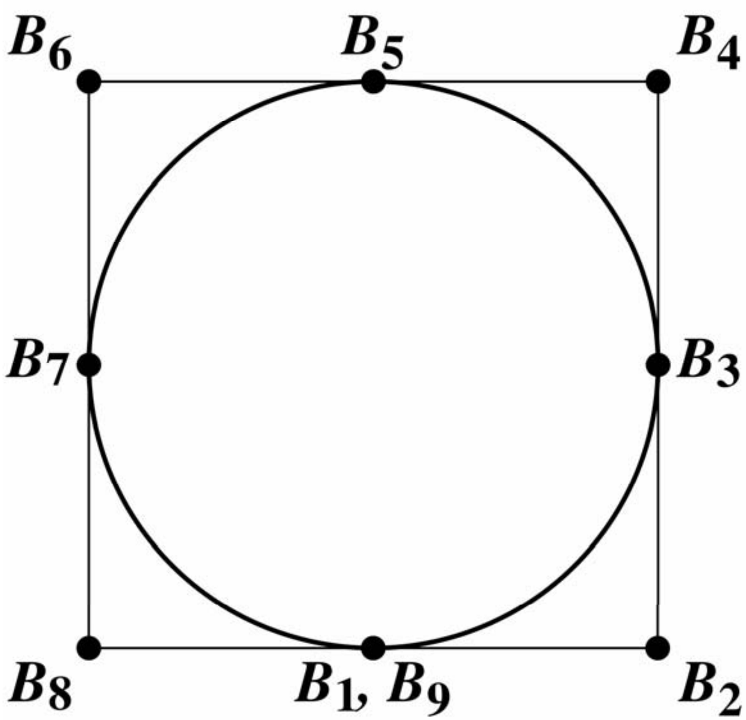
$[X] = [0\ 0\ 0\ 1\ 1\ 2\ 2\ 3\ 3\ 3]$; $k = 3$; $[H] = [1\ 1/2\ 1\ 1/2\ 1\ 1/2\ 1\ ]$



Rational B-spline Curves – Circles

Four 90° arcs $[X]=[0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 2\ \ 2\ \ 3\ \ 3\ \ 4\ \ 4\ \ 4]$;

$k = 3$; $[H] = [1\ \ \sqrt{2}/2\ \ 1\ \ \sqrt{2}/2\ \ 1\ \ \sqrt{2}/2\ \ 1\ \ \sqrt{2}/2\ \ 1\ ]$



Non-Rational B-Spline Surfaces

- Definition
- Properties
- Control
- Additional Topics

Non-Rational B-Spline Surfaces: Definition

$$Q(u,w) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j} N_{i,k}(u) M_{j,\ell}(w)$$

where

$$N_{i,1}(u) = \begin{cases} 1 & \textbf{if } x_i \le u < x_{i+1} \\ 0 & \textbf{otherwise} \end{cases}$$

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

and

$$M_{j,1}(w) = \begin{cases} 1 & \textbf{if } y_j \le w < y_{j+1} \\ 0 & \textbf{otherwise} \end{cases}$$

$$M_{j,\ell}(w) = \frac{(w - y_j)M_{j,\ell-1}(w)}{y_{j+l-1} - y_j} + \frac{(y_{j+\ell} - w)M_{j+1,\ell-1}(w)}{y_{j+\ell} - y_{j+1}}$$

Non-Rational B-Spline Surfaces: Properties

- Maximum order, k, l is the number of control vertices in each parametric direction
- Continuity C^{k-2}, C^{l-2} in each parametric direction
- Variation diminishing property is not known
- Transform surface – transform control net

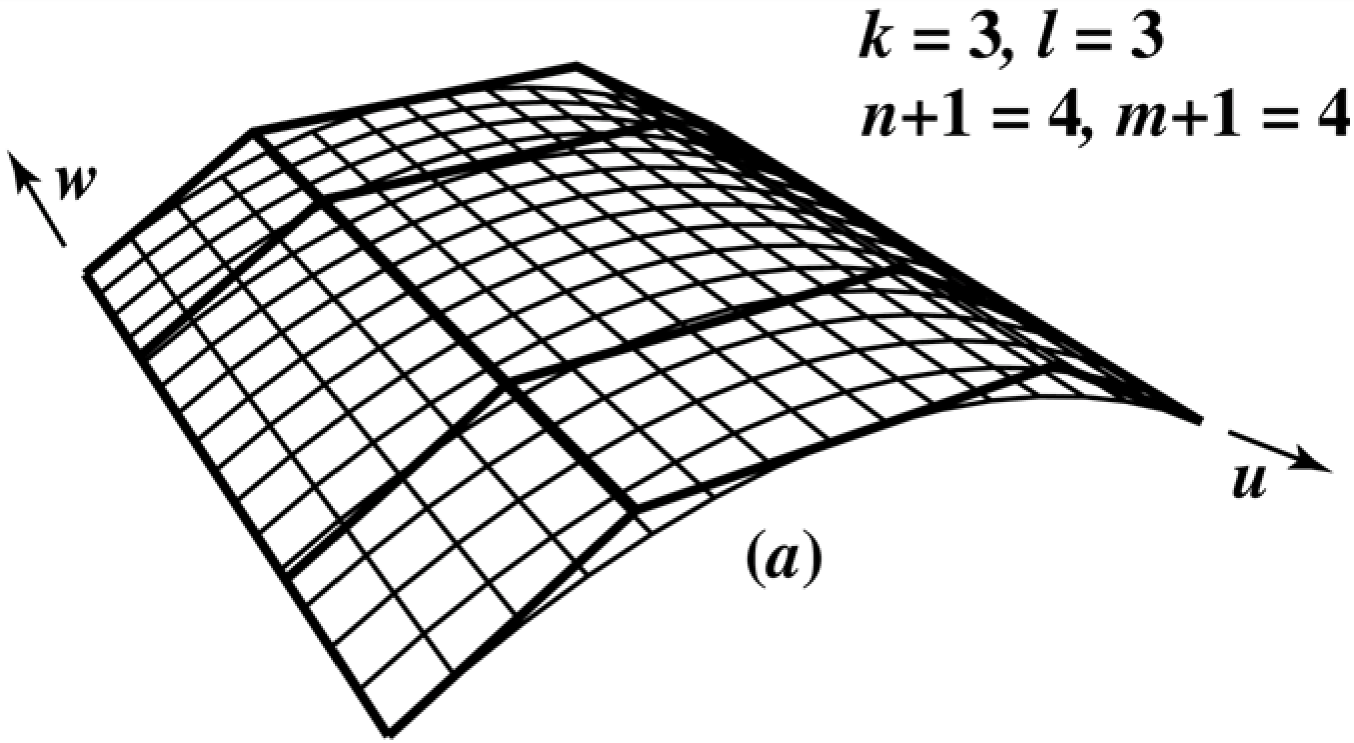
Non-Rational B-Spline Surfaces: Properties

- Influence of single control vertex is $\pm k/2$, $\pm l/2$
- If $n+1=k$, $m+1=l$ a Bézier surface results
- Triangulated, the control net forms a planar approximation to the surface
- Lies within the union of convex hulls of k, l neighboring control vertices

Non-Rational B-Spline Surfaces: Control

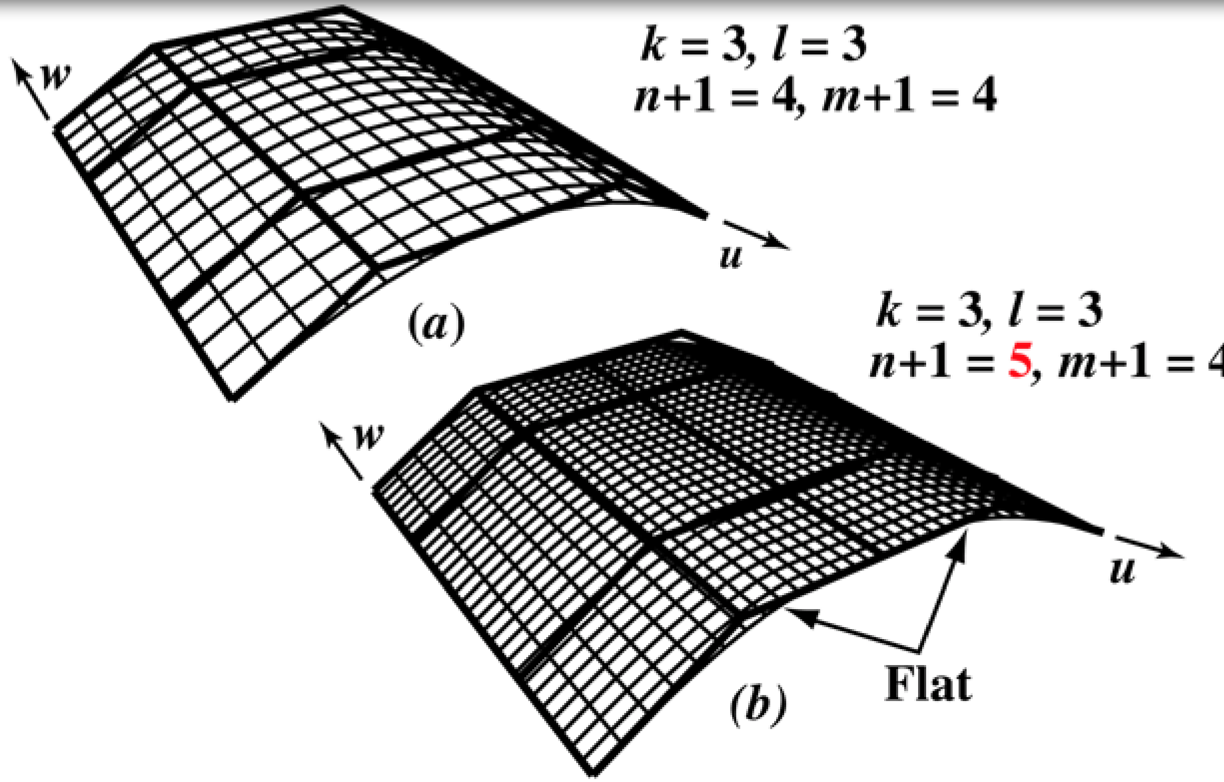
- Order/degree
- Knot vectors (single/multiple)
- Number of Control points
- Control points (single/multiple)

Non-Rational B-Spline Surfaces: Colinear net lines



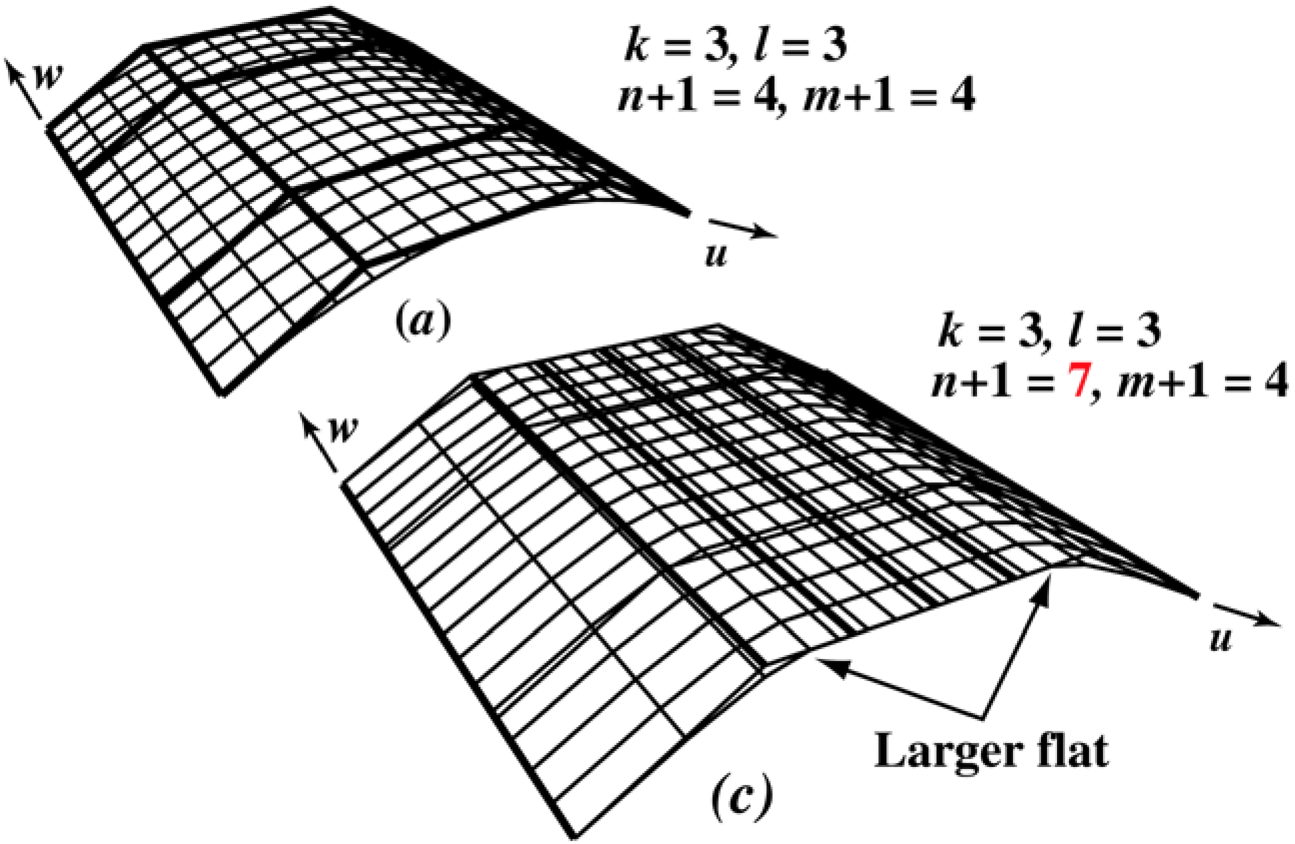
- Ruled in the w direction
- Smoothly curved in the u direction

Non-Rational B-Spline Surfaces: Colinear net lines



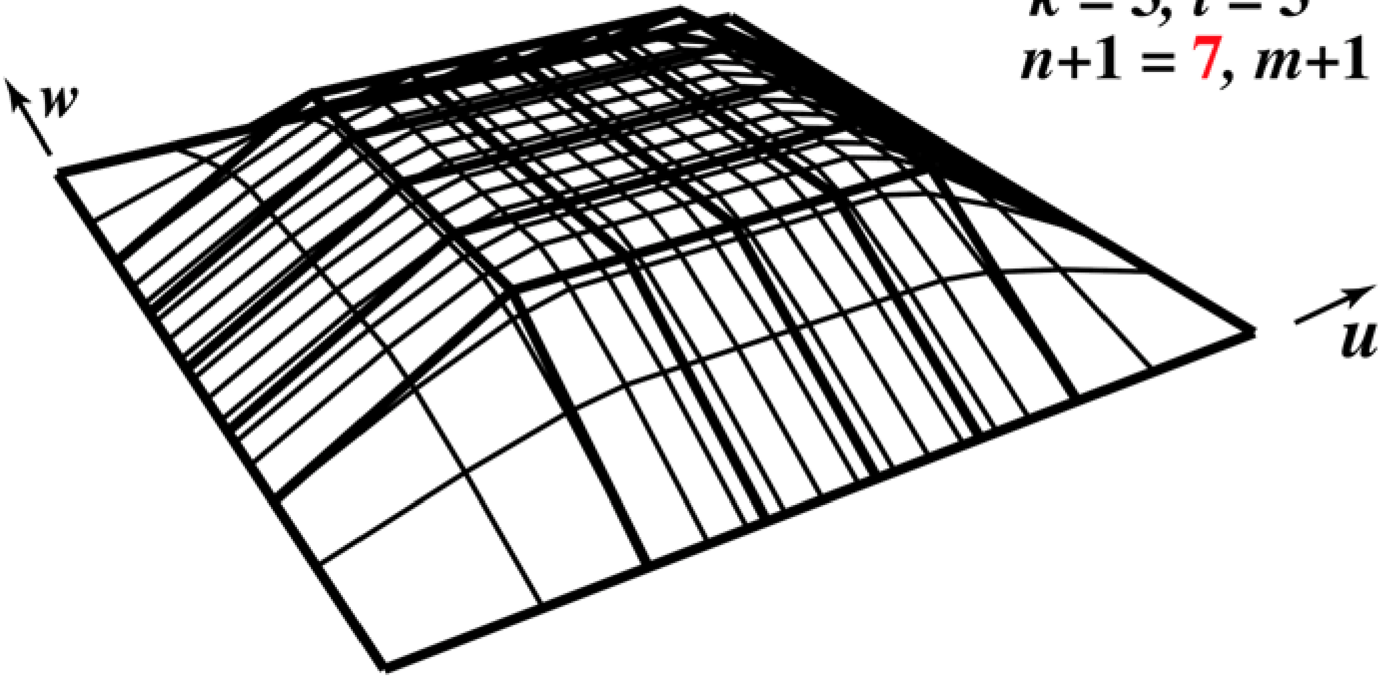
- Ruled in the w direction
- Embedded flat area in the u direction

Non-Rational B-Spline Surfaces: Colinear net lines



Larger embedded flat area in the u direction

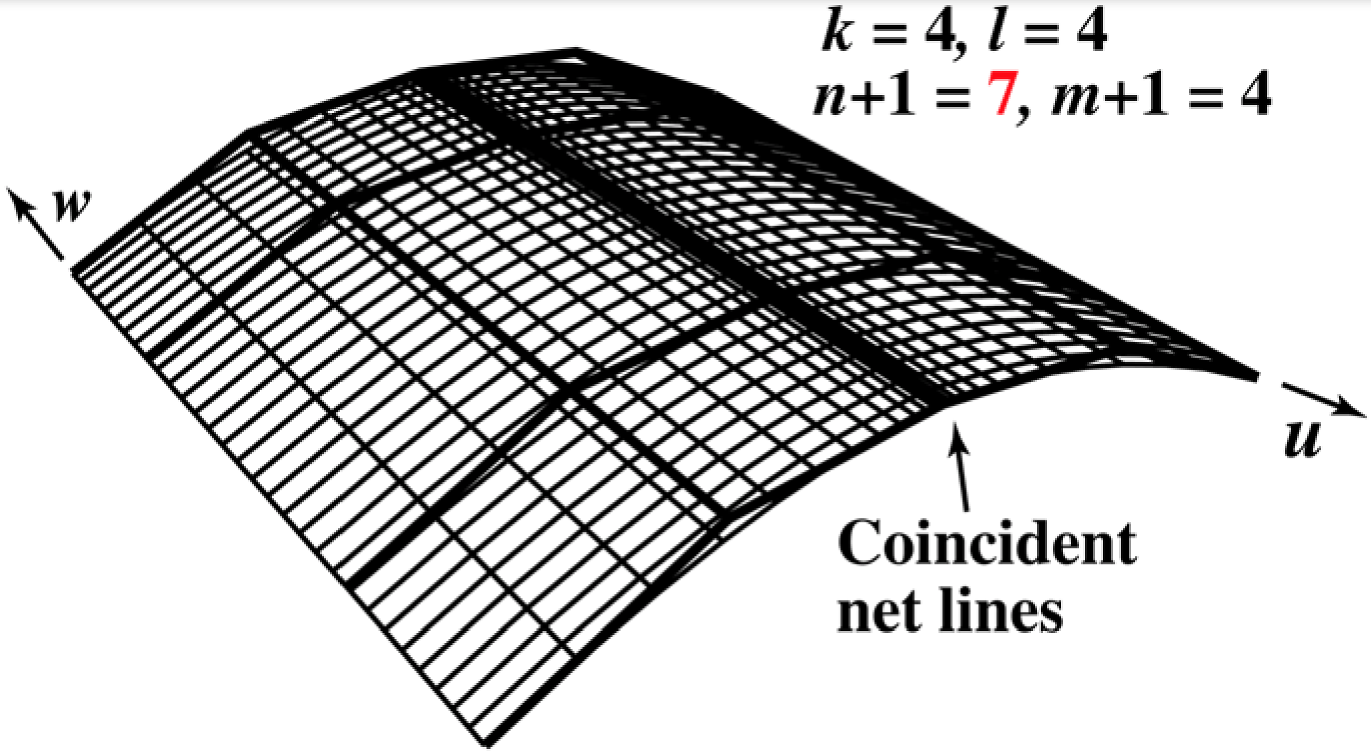
Non-Rational B-Spline Surfaces: Colinear net lines



$k = 3, l = 3$
$n+1 = 7, m+1 = 7$

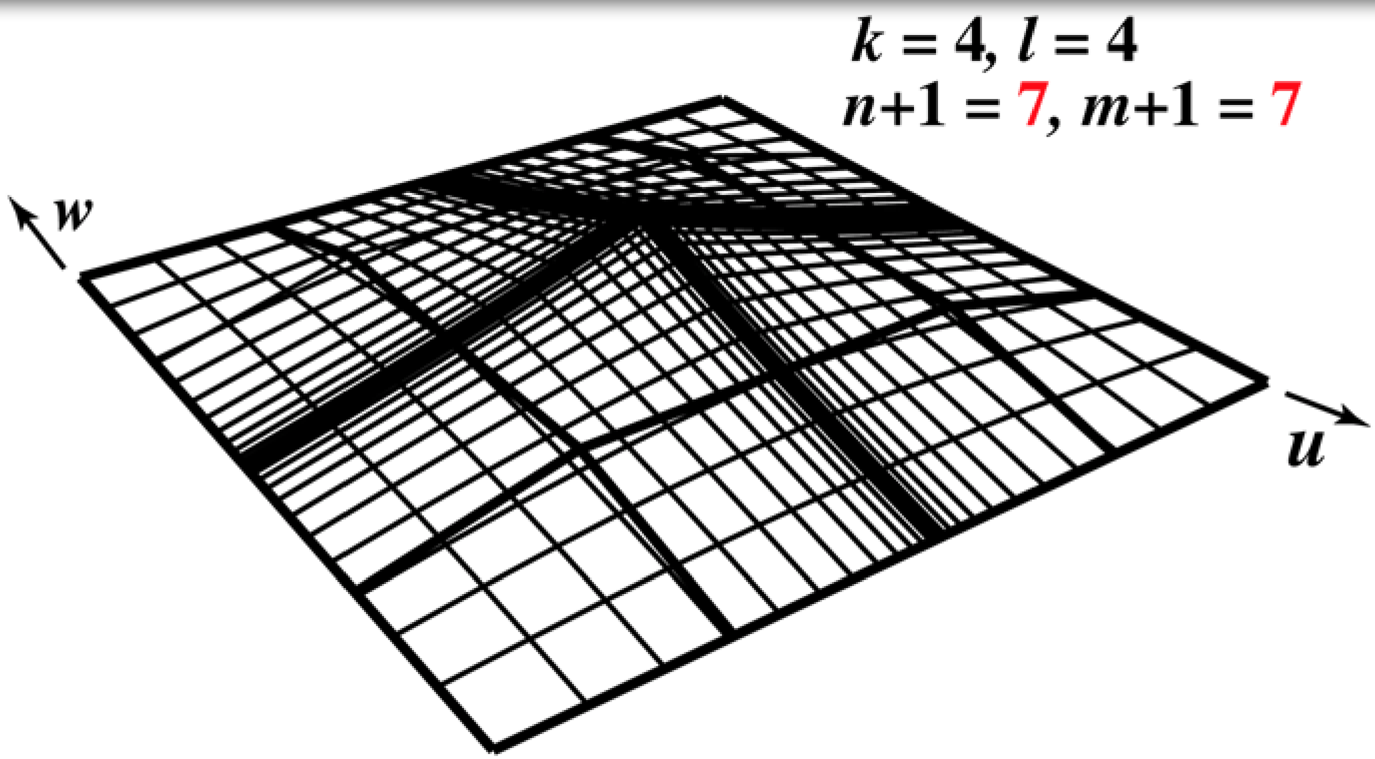
- Embedded flat area in the center
- Embedded flat area on each side
- Curved corners

Non-Rational B-Spline Surfaces: Colinear net lines



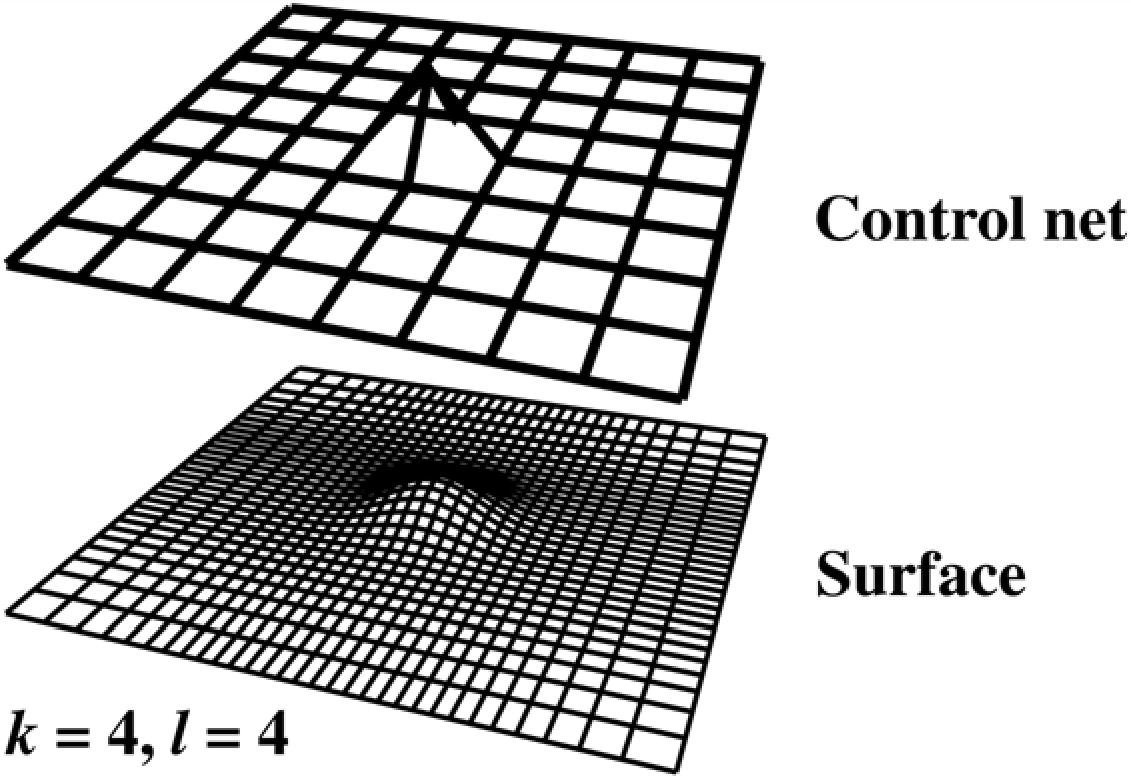
- Three coincident net lines in the w direction generate hard line in the surface
- Still C^{k-2}, C^{l-2} continuous in both parametric directions

Non-Rational B-Spline Surfaces: Colinear net lines



- Three coincident net lines in the u and w directions generate two hard lines and a point in the surface
- Still C^{k-2}, C^{l-2} continuous in both parametric directions

Non-Rational B-Spline Surfaces: Local control



$k = 4, l = 4$

$n + 1 = 9, m + 1 = 9$

Local influence is $\pm k/2,\ \pm l/2$

Non-Rational B-Spline Surfaces: Additional Topics

- Degree elevation and reduction
- Derivatives
- Knot insertion
- Subdivision
- Reparameterization

Rational B-Spline Surfaces: NURBS

- Definition
- Properties
- Weight Effects
- Algorithms
- Additional Topics

NURBS: Definition

In four-dimensional homongeneous coordinate space

$$Q(u,w) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j}^h N_{i,k}(u) M_{j,\ell}(w)$$

And projecting back into three space

$$Q(u,w) = \frac{\sum_{i=1}^{n+1} \sum_{j=1}^{m+1} h_{i,j} B_{i,j} N_{i,k}(u) M_{j,\ell}(w)}{\sum_{i=1}^{n+1} \sum_{j=1}^{m+1} h_{i,j} N_{i,k}(u) M_{j,\ell}(w)} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j} S_{i,j}(u,w)$$

where $B_{i,j}$s are the 3-D control net vertices
$S_{i,j}$s are the bivariate rational B-spline surface basis functions

NURBS: Definition

Basis functions

$$Q(u,w) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j} S_{i,j}(u,w)$$

where

$$S_{i,j}(u,w) = \frac{h_{i,j} N_{i,k}(u) M_{j,\ell}(w)}{\sum_{i1=1}^{n+1} \sum_{j1=1}^{m+1} h_{i1,j1} N_{i1,k}(u) M_{j1,\ell}(w)} = \frac{h_{i,j} N_{i,k}(u) M_{j,\ell}(w)}{\mathbf{Sum}(u,w)}$$

and

$$\mathbf{Sum}(u,w) = \sum_{i1=1}^{n+1} \sum_{j1=1}^{m+1} h_{i1,j1} N_{i1,k}(u) M_{j1,\ell}(w)$$

Convenient, but not necessary, to assume $h_{i,j} \geqslant 0$ for all i, j

NURBS: Definition

Basis functions

$$S_{i,j}(u,w) = \frac{h_{i,j} N_{i,k}(u) M_{j,\ell}(w)}{\mathbf{Sum}(u,w)}$$

$$\mathbf{Sum}(u,w) = \sum_{i1=1}^{n+1} \sum_{j1=1}^{m+1} h_{i1,j1} N_{i1,k}(u) M_{j1,\ell}(w)$$

$S_{i,j}(u,w)$s are not the product of $R_{i,k}(u)$ and $R_{j,l}(w)$

Similar shapes and characteristics to $N_{i,k}(u)M_{j,l}(w)$

NURBS: Properties

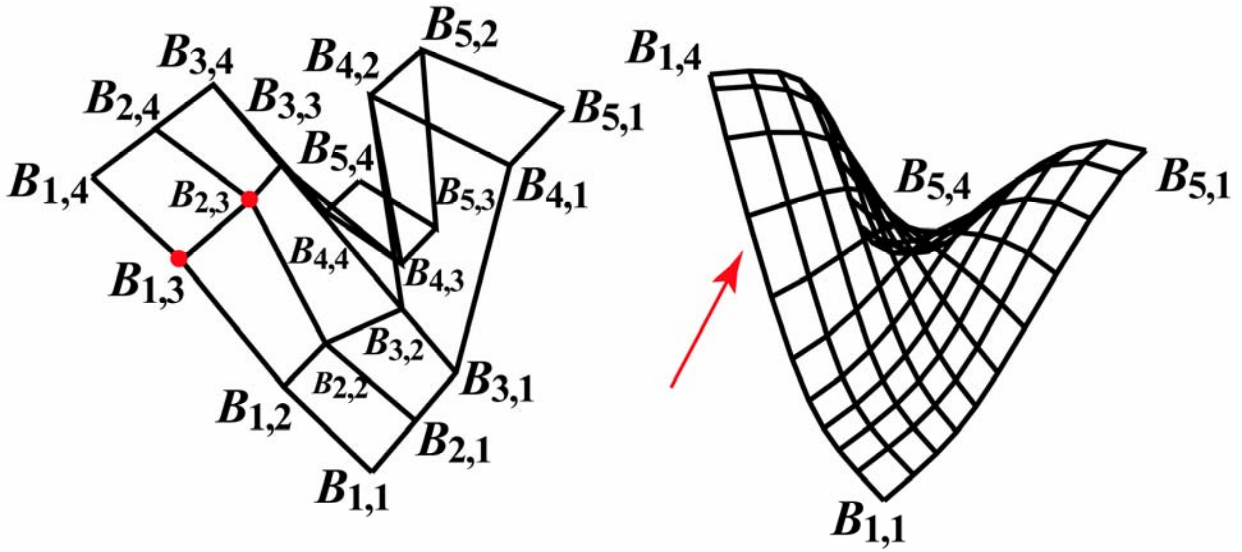
- $\sum_{i=1}^{n+1} \sum_{j=1}^{m+1} S_{i,j}(u, w) \equiv 1$
- $S_{i,j}(u,w) \geqslant 0$
- Maximum order is the number of control vertices in each parametric direction
- Continuity C^{k-2}, C^{l-2} in each parametric direction
- Transform surface – transform control net
- The variation-diminishing property not known

NURBS: Properties

- Influence of single control vertex is $\pm k/2, \pm l/2$
- If $n+1=k, m+1=l$, a rational Bézier surface results
- If $n+1=k, m+1=l$ and $h_{ij}=1$, a nonrational Bézier surface results
- Triangulated, the control net forms a planar approximation to the surface
- If $h_{i,j} \geqslant 0$, surface lies within union of convex hulls of k,l neighboring control vertices

NURBS: Weight Effects

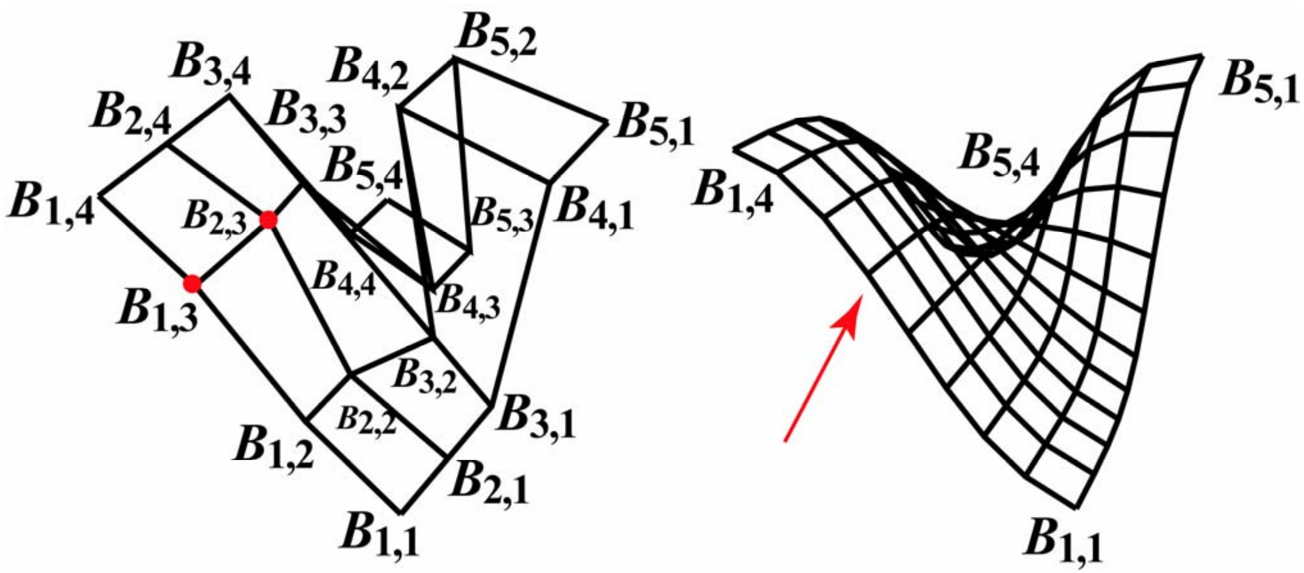
$h_{i,j} \geqslant 0$ effect of zero weights



$n + 1 = 5$, $m + 1 = 4$, $k = l = 4$, $h_{1,3} = h_{2,3} = 0$,
Notice the straight edge and flat surface indicated by the red arrow

NURBS: Weight Effects

$h_{i,j} \geqslant 0$ **effect of homogeneous weights**

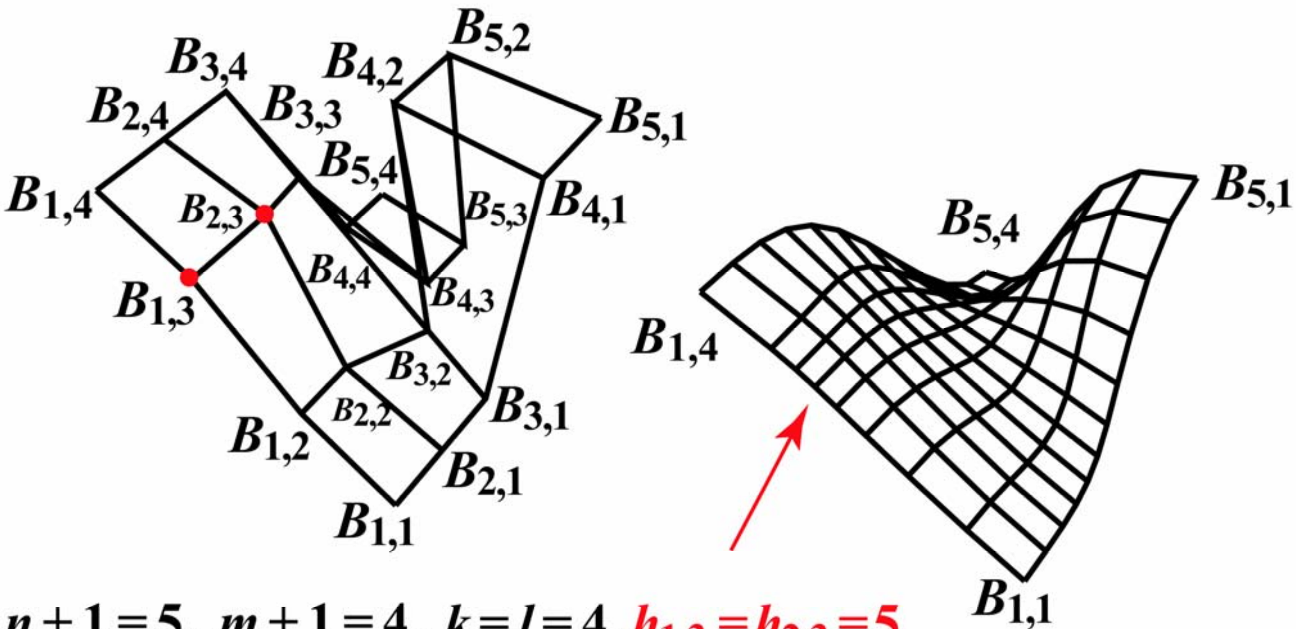


$n+1=5$, $m+1=4$, $k=l=4$, $h_{1,3}=h_{2,3}=1$

Notice the curved edge and surface indicated by the red arrow

NURBS: Weight Effects

$h_{i,j} \geqslant 0$ **effect of homogeneous weights**

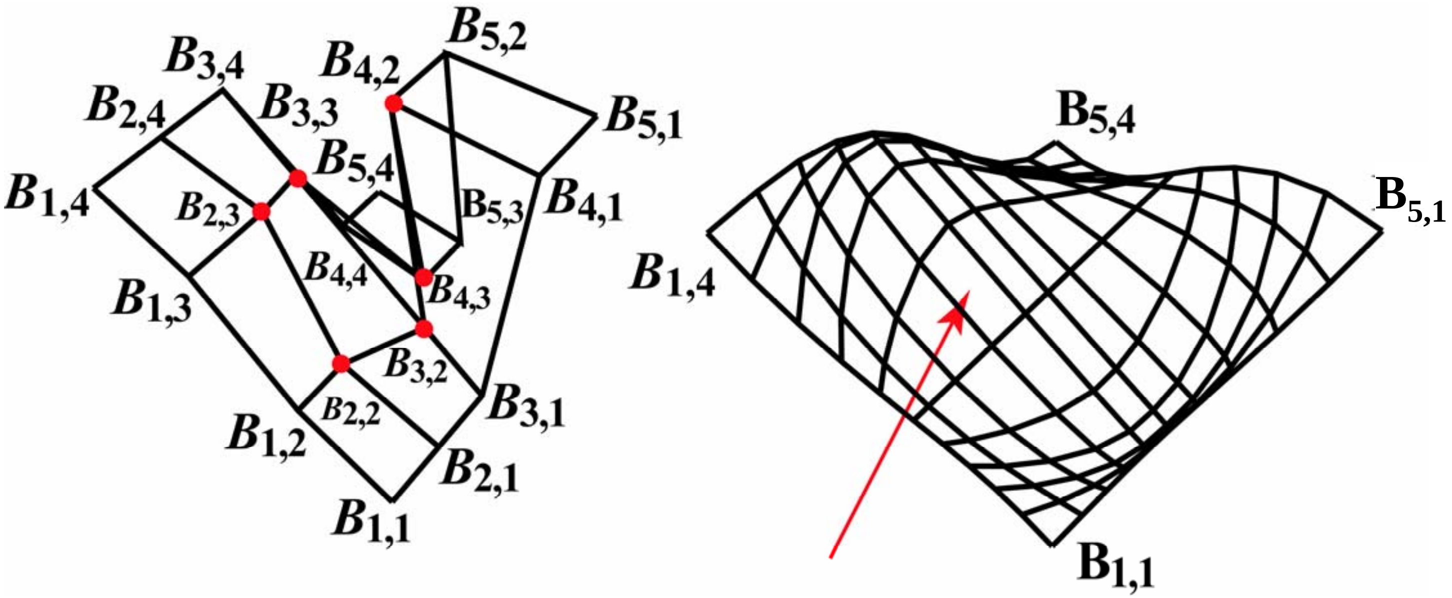


$$n+1=5,\ m+1=4,\ k=l=4,\ h_{1,3}=h_{2,3}=5$$

Notice the curved edge and surface indicated by the red arrow

NURBS: Weight Effects

$h_{i,j} \geqslant 0$ **effect of homogeneous weights**

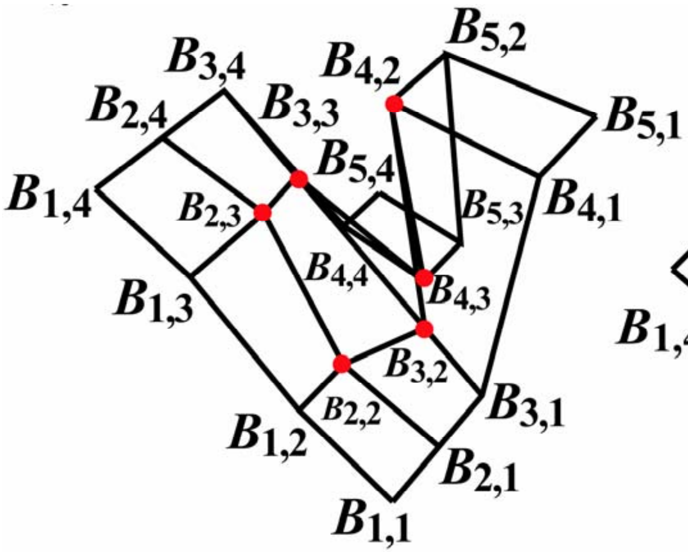


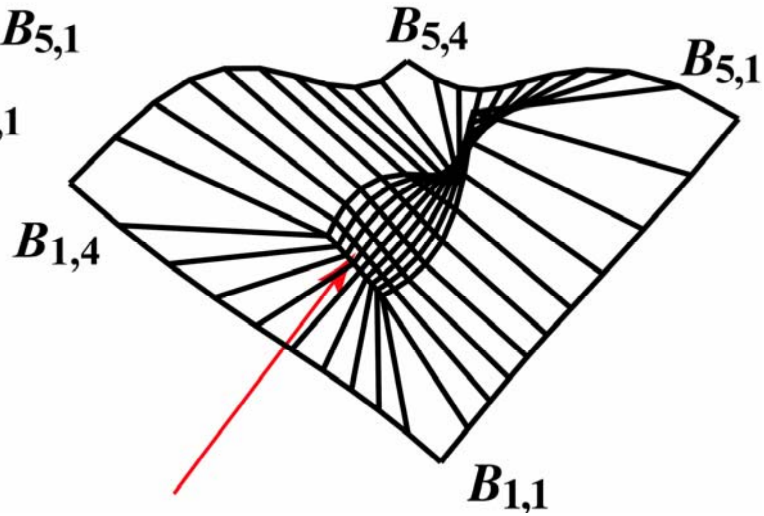
$n+1=5$, $m+1=4$, $k=l=4$, All interior $h_{i,j}=0$

Notice the curved edge and surface indicated by the red arrow

NURBS: Weight Effects

$h_{i,j} \geqslant 0$ **effect of homogeneous weights**



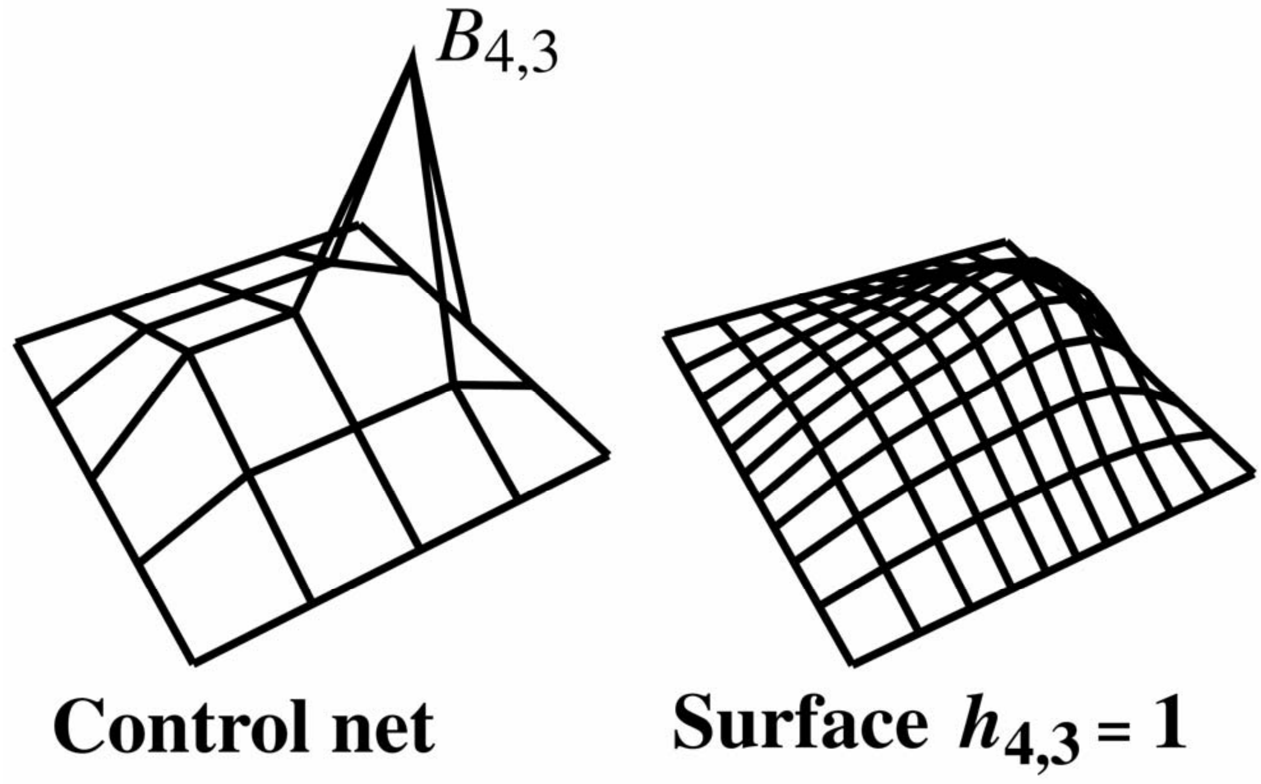


$n+1=5$, $m+1=4$, $k=l=4$, All interior $h_{i,j}=500$

Notice the curved edge and surface indicated by the red arrow

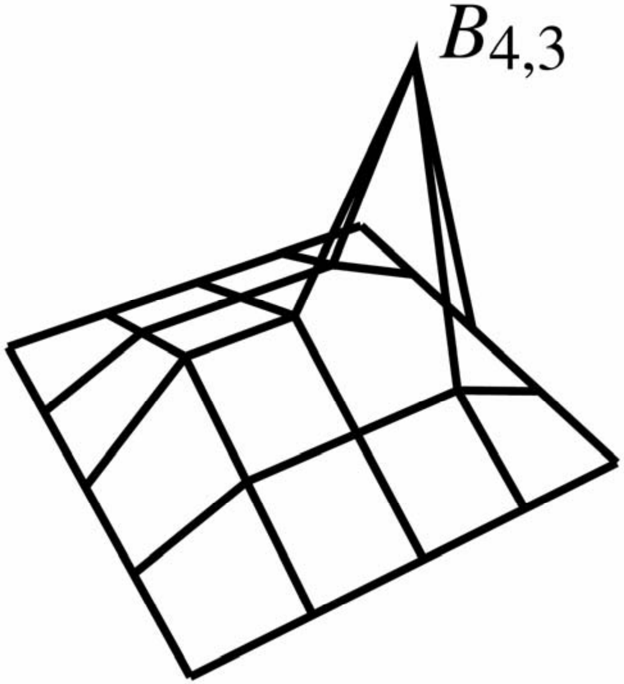
NURBS: Weight Effects

$h_{i,j} \geqslant 0$, effect of homogeneous weights

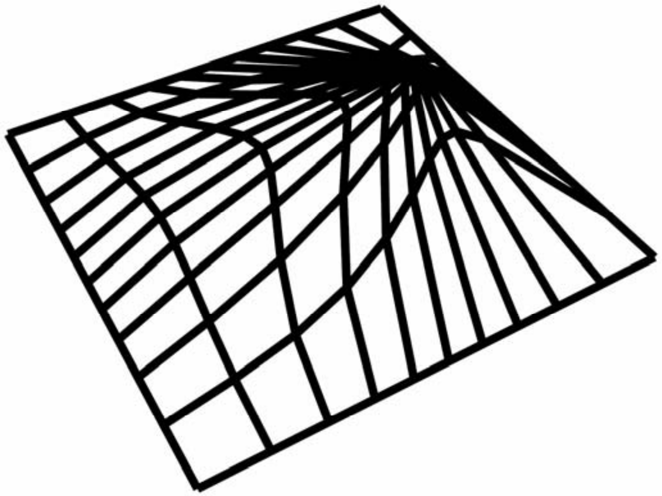


NURBS: Weight Effects

$h_{i,j} \geqslant 0$, effect of homogeneous weights



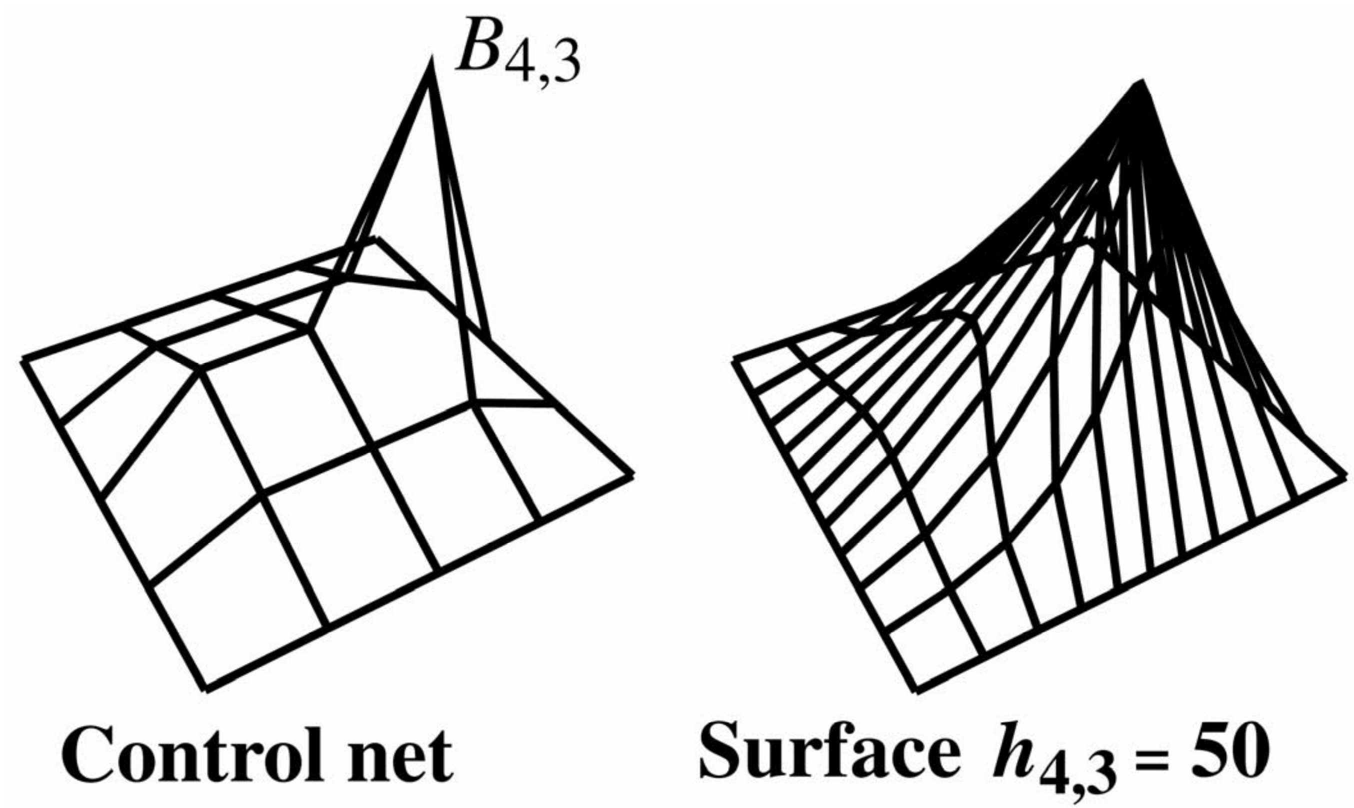
Control net



Surface $h_{4,3} = 5$

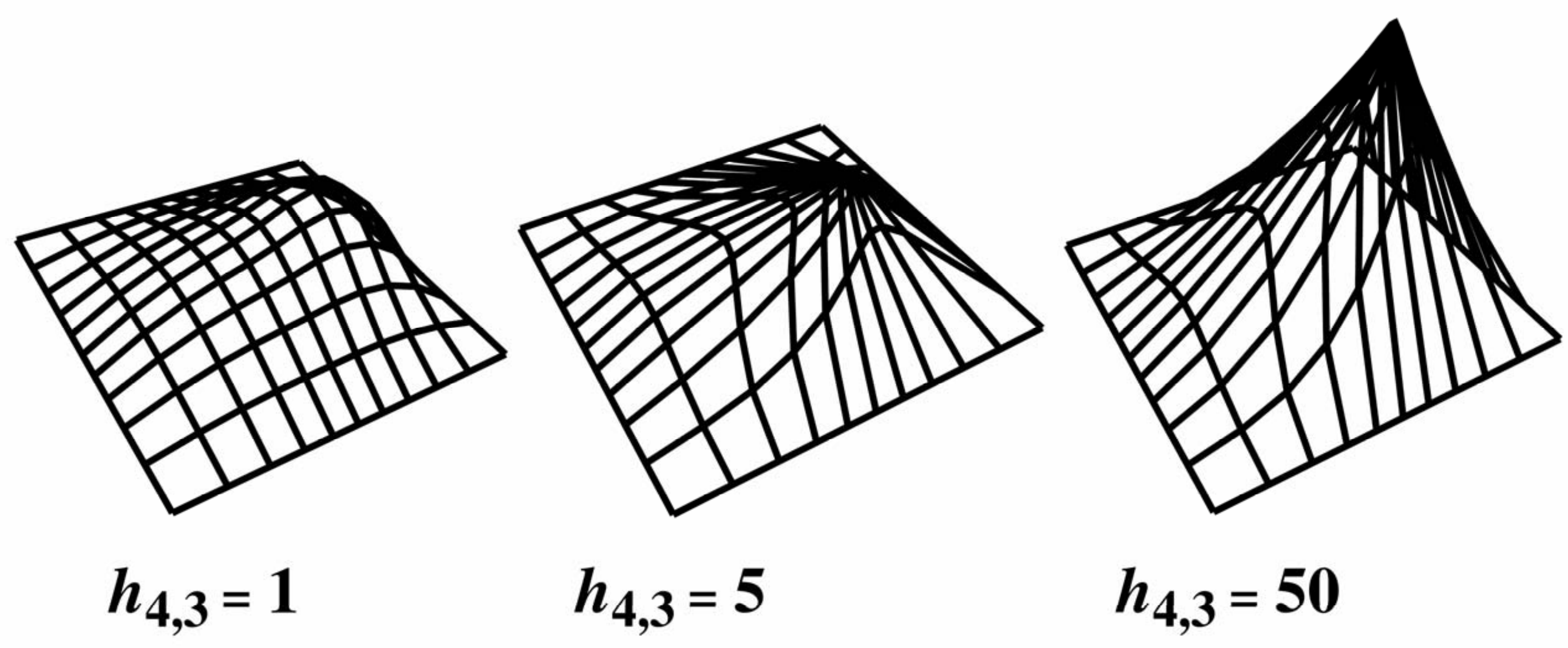
NURBS: Weight Effects

$h_{i,j} \geqslant 0$, effect of homogeneous weights



NURBS: Weight Effects

$h_{i,j} \geqslant 0$, effect of homogeneous weights - comparison



NURBS Surfaces: Algorithms

Nonrational B-spline surface – $h_{i,j}=1$ for all i,j

Hence

$$\mathbf{Sum}(u,w) = \sum_{i1=1}^{n+1}\sum_{j1=1}^{m+1} h_{i1,j1} N_{i1,k}(u) M_{j1,\ell}(w) = 1 \quad \mathbf{for\ \ all} \quad u,w$$

and $S_{i,j}(u,w)$ reduces to

$$S_{i,j}(u,w) = \frac{h_{i,j} N_{i,k}(u) M_{j,\ell}(w)}{\mathbf{Sum}(u,w)} = N_{i,k}(u) M_{j,\ell}(w)$$

which yields

$$Q(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j} S_{i,j}(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j} N_{i,k}(u) M_{j,\ell}(w)$$

which suggests that the core algorithm is two nested loops

NURBS Surfaces: Algorithms -- Example

Writing out for $n+1=4$, $m+1=4$, $k=l=4$ yields

$$Q(u,w) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j} N_{i,k}(u) M_{j,\ell}(w) = \sum_{i=1}^{4} \sum_{j=1}^{4} B_{i,j} N_{i,4}(u) M_{j,4}(w)$$

or

$$\begin{aligned} Q(u,w) = {} & N_{1,4}(B_{1,1}M_{1,4} + B_{1,2}M_{2,4} + B_{1,3}M_{3,4} + B_{1,4}M_{4,4}) \\ & + N_{2,4}(B_{2,1}M_{1,4} + B_{2,2}M_{2,4} + B_{2,3}M_{3,4} + B_{2,4}M_{4,4}) \\ & + N_{3,4}(B_{3,1}M_{1,4} + B_{3,2}M_{2,4} + B_{3,3}M_{3,4} + B_{3,4}M_{4,4}) \\ & + N_{4,4}(B_{4,1}M_{1,4} + B_{4,2}M_{2,4} + B_{4,3}M_{3,4} + B_{4,4}M_{4,4}) \end{aligned}$$

The inner loop is within the ( )

The outer loop is the multiplier $N_{i,j}(\ )$

The knot vectors and basis functions are also needed

NURBS Surfaces: Algorithms

Naive nonrational B-spline surface algorithm

Specify number of control vertices in the u, w directions
Specify order in each of the u, w directions
Specify number of isoparametric lines in each of the u, w direction
Specify the control net, store in an array

Calculate the knot vector in the u direction, store in an array
Calculate the knot vector in the w direction, store in an array
For each parametric value, u
Calculate the basis functions, $N_{i,k}(u)$, store in an array

For each parametric value, w
    Calculate the basis functions, $M_{j,l}(w)$, store in an array
        For each control vertex in the u direction
            For each control vertex in the w direction
                Calculate the surface point, $Q(u,w)$, store in an array
            end loop
        end loop
    end loop
end loop

NURBS Surfaces: Algorithms

Rational B-spline (NURBS) surface

$$Q(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j}\frac{h_{i,j}N_{i,k}(u)M_{j,\ell}(w)}{\mathbf{Sum}(u,w)}$$

and

$$\mathbf{Sum}(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j}N_{i,k}(u)M_{j,\ell}(w)$$

Two differences from the nonrational B-spline surface:

- Calculate and divide by the $\mathbf{Sum}(u,w)$ function
- Multiply by h_{ij}

Let's look at calculating the $\mathbf{Sum}(u,w)$ function

NURBS Surfaces: Algorithms

Calculating the **Sum**(u,w) function

$$\mathbf{Sum}(u,w) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} h_{i,j} N_{i,k}(u) M_{j,\ell}(w)$$

Writing this out for $n+1=m+1=4$, $k=l=4$ yields

$$\begin{aligned}
\mathbf{Sum}(u,w) &= \sum_{i=1}^{4} \sum_{j=1}^{4} h_{i,j} N_{i,4}(u) M_{j,4}(w) \\
&= N_{1,4}(h_{1,1}M_{1,4} + h_{1,2}M_{2,4} + h_{1,3}M_{3,4} + h_{1,4}M_{4,4}) \\
&\quad + N_{2,4}(h_{2,1}M_{1,4} + h_{2,2}M_{2,4} + h_{2,3}M_{3,4} + h_{2,4}M_{4,4}) \\
&\quad + N_{3,4}(h_{3,1}M_{1,4} + h_{3,2}M_{2,4} + h_{3,3}M_{3,4} + h_{3,4}M_{4,4}) \\
&\quad + N_{4,4}(h_{4,1}M_{1,4} + h_{4,2}M_{2,4} + h_{4,3}M_{3,4} + h_{4,4}M_{4,4})
\end{aligned}$$

Same form as the nonrational B-spline surface except h_{ij} instead of B_{ij} – use the same algorithm

NURBS Surfaces: Algorithms

Algorithm for the Sum(u,w) function

Assume the $N_{i,k}$ and $M_{j,l}$ basis functions are available

Assume the homogeneous weights, $h_{i,j}$, are available

For each control vertex in the u direction

  For each control vertex in the w direction

    Calculate and store the **Sum**(u,w) function

  end loop

end loop

NURBS Surfaces: Algorithms

Naive rational B-spline (NURBS) surface algorithm

The inner loop now becomes

For each parametric value, u
    Calculate the basis functions, $N_{i,k}(u)$, store in an array
    For each parametric value, w
        Calculate the basis functions, $M_{j,l}(w)$, store in an array
    ➔ Calculate the **Sum**(u,w) function
            For each control vertex in the u direction
                For each control vertex in the w direction
                    Calculate and store the surface point, $Q(u,w)$
                end loop
            end loop
    end loop
end loop

NURBS Surfaces: Algorithms

Nonrational B-spline surface

$$Q(u,w) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j} N_{i,k}(u) M_{j,\ell}(w)$$

Rational B-spline (NURBS) surface

$$Q(u,w) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} h_{i,j} \frac{B_{i,j} N_{i,k}(u) M_{j,\ell}(w)}{\mathbf{Sum}(u,w)}$$

Comparing shows the NURBS algorithm requires

- an additional multiply
- a division
- calculation of the **Sum**(u,w) function

Results in approximately 1/3 more computational effort

NURBS Surfaces: Algorithms

These naive algorithms are very memory efficient

However, they are computationally inefficient

Computational efficiency improved by

- avoiding the division by the **Sum**(u,w) function by converting it to a multiply using the reciprocal
- avoiding entire computations

NURBS Surfaces: Algorithms

More efficient NURBS algorithm

Recall for $n + 1 = m + 1 = 3, k = l = 3$ the NURBS surface is

$$
\begin{aligned}
Q(u,w) = {} & \frac{N_{1,3}}{\text{Sum}}(h_{1,1}B_{1,1}M_{1,3} + h_{1,2}B_{1,2}M_{2,3} + h_{1,3}B_{1,3}M_{3,3}) \\
& + \frac{N_{2,3}}{\text{Sum}}(h_{2,1}B_{2,1}M_{1,3} + h_{2,2}B_{2,2}M_{2,3} + h_{2,3}B_{2,3}M_{3,3}) \\
& + \frac{N_{3,3}}{\text{Sum}}(h_{3,1}B_{3,1}M_{1,3} + h_{3,2}B_{3,2}M_{2,3} + h_{3,3}B_{3,3}M_{3,3})
\end{aligned}
$$

Recall that in many cases the basis functions are zero

- If $N_{i,j}(u,w) = 0$, then we can avoid the entire calculation in ( ) and the division (multiply) by $\mathbf{Sum}(u,w)$ (the reciprocal)
- If $M_{i,j}(u,w) = 0$, then we can avoid three multiplies in ( )

Storing the reciprocal of $\mathbf{Sum}(u,w)$ saves a divide at the expense of a multiply

NURBS Surfaces: Algorithms

More efficient rational B-spline (NURBS) surface algorithm

The inner loop now becomes

For each parametric value, u

Calculate the basis functions, $N_{i,k}(u)$, store in an array

For each parametric value, w

Calculate the basis functions, $M_{j,l}(w)$, store in an array

➔ Calculate and save the reciprocal of $\mathbf{Sum}(u,w)$

For $i = 1$ to $n + 1$ //*For each control vertex in the* u *direction*

➔ If $N_{i,k}(u) \neq 0$ then

For $j = 1$ to $m + 1$ //*For each control vertex in the* w *direction*

➔ If $M_{j,l}(w) \neq 0$ then

Calculate $Q(u,w) = Q(u,w) + h_{i,j}N_{i,k}(u)M_{j,l}(w)*\mathbf{Sum}(u,w)$

end if

end loop

end if

end loop

Store $Q(u,w)$; Reinitalize $Q(u,w) = 0$

end loop

end loop

NURBS Surfaces: Algorithms

- The improved naive algorithms are still very memory efficient
- The simple changes, based on the underlying mathematics, increase the computational efficiency by 25% or more
- In the late 1970s this algorithm provided the basis for a real time interactive nonrational B-spline surface design system based on directly manipulating the control net – SIGGRAPH ’80 paper
- The machine was a 16 bit minicomputer with 64 Kbytes of memory driving an Evans & Sutherland Picture System I
- Can we do better – Yes!

NURBS Surfaces: Algorithms

- When modifying a B-spline surface, a designer typically works with a control net:
  - of constant control net size, $n + 1$, $m + 1$, in each direction
  - of constant order, k, l, in each parametric direction
  - with a constant number, p_1, p_2, of isoparametric lines in each parametric direction

  Hence, $n + 1$, $m + 1$, k, l, p_1 and p_2 do not change
- If these values do not change, neither do the basis functions, $N_{i,k}(u)$ and $M_{j,l}(w)$, nor the $\mathbf{Sum}(u,w)$ function
- Thus, precalculating and storing the product $N_{i,k}(u)M_{j,l}(w)/\mathbf{Sum}(u,w)$ further increases the efficiency
- However, we leave this specific efficiency increase as an exercise

NURBS Surfaces: Algorithms

When modifying a NURBS surface control net, a designer typically manipulates:

a single control net vertex, B_{ij}

or

the value of a single homongeneous weight, h_{ij}

Also, assume $n+1$, $m+1$, k, l, p_1 and p_2 do not change

Writing the NURBS surface equation for both the new and old surfaces and subtracting yields

$$\mathbf{Sum}_{\mathrm{new}}(u,w)Q_{\mathrm{new}}(u,w) = \mathbf{Sum}_{\mathrm{old}}(u,w)\, Q_{\mathrm{old}}(u,w) + (h_{i,j\mathrm{new}}B_{i,j\mathrm{new}} - h_{i,j\mathrm{old}}B_{i,j\mathrm{old}})\, N_{i,k}(u)\, M_{j,l}(w)$$

which represents an incremental calculation for the new surface

NURBS Surfaces: Algorithms

Only a single control vertex changes

If $h_{i,j}$ does not change, then $\mathbf{Sum}(u,w)$ does not change and

$$\mathbf{Sum}_{\text{new}}(u,w)Q_{\text{new}}(u,w) = \mathbf{Sum}_{\text{old}}(u,w)\, Q_{\text{old}}(u,w) + (h_{i,j\text{new}}B_{i,j\text{new}} - h_{i,j\text{old}}B_{i,j\text{old}})\, N_{i,k}(u)\, M_{j,l}(w)$$

becomes

$$Q_{\text{new}}(u,w) = Q_{\text{old}}(u,w) + (B_{i,j_{\text{new}}} - B_{i,j_{\text{old}}})\frac{h_{i,j}(u)N_{i,k}(u)M_{j,\ell}(w)}{\mathbf{Sum}(u,w)}$$

Thus, incremental calculation of the new surface requires four multiplies, one subtract, one add for each u,w

NURBS Surfaces: Algorithms

Only a single homogeneous weight changes

If $h_{i,j}$ changes, then $\mathbf{Sum}(u,w)$ does not change and

$$\mathbf{Sum}_{\text{new}}(u,w)Q_{\text{new}}(u,w) = \mathbf{Sum}_{\text{old}}(u,w)\, Q_{\text{old}}(u,w)$$

$$+ (h_{i,j\text{new}}B_{i,j\text{new}} - h_{i,j\text{old}}B_{i,j\text{old}})\, N_{i,k}(u)\, M_{j,l}(w)$$

becomes

$$Q_{\text{new}}(u,w) = \frac{\mathbf{Sum}_{\text{old}}(u,w)}{\mathbf{Sum}_{\text{new}}(u,w)}\, Q_{\text{old}}(u,w) + \left(h_{i,j_{\text{new}}} - h_{i,j_{\text{old}}}\right) \frac{B_{i,j} N_{i,k}(u) M_{j,\ell}(w)}{\mathbf{Sum}_{\text{new}}(u,w)}$$

Thus, incremental calculation of the new surface requires six multiplies, one subtract, one add , calculation of the new $\mathbf{Sum}\,(u,w)$ function for each u,w

NURBS Surfaces: Algorithms

Incremental $\mathbf{Sum}(u,w)$ calculation

Writing the $\mathbf{Sum}(u,w)$ expression for both the new and old surfaces and subtracting yields

$$\mathbf{Sum}_{\mathrm{new}}(u,w) = \mathbf{Sum}_{\mathrm{old}}(u,w)+(h_{i,j\mathrm{new}}-h_{i,j\mathrm{old}})N_{i,k}(u)M_{j,l}(w)$$

which represents an incremental calculation for the new $\mathbf{Sum}(u,w)$ function

Thus, calculating the new $\mathbf{Sum}(u,w)$ requires two multiplies, a subtract and an add

If either $N_{i,k}(u)$ or $M_{j,l}(w)$ are zero, the $\mathbf{Sum}(u,w)$ function does not change

NURBS Surfaces: Algorithms

Nonrational B-spline surface incremental calculation

Recall

$$\mathbf{Sum}_{\mathrm{new}}(u,w)Q_{\mathrm{new}}(u,w) = \mathbf{Sum}_{\mathrm{old}}(u,w)Q_{\mathrm{old}}(u,w)$$
$$+ (h_{i,j\mathrm{new}}B_{i,j\mathrm{new}} - h_{i,j\mathrm{old}}B_{i,j\mathrm{old}})N_{i,k}(u)M_{j,l}(w)$$

If $\mathbf{Sum}(u,w)=1$ and all $h_{i,j}=1$, a nonrational B-spline surface is generated. The result is

$$Q_{\mathrm{new}}(u,w) = Q_{\mathrm{old}}(u,w)+(B_{i,j\mathrm{new}}-B_{i,j\mathrm{old}})N_{i,k}(u)M_{j,l}(w)$$

Thus, calculating the new surface requires two multiplies, a subtract and an add for each u,w

If either $N_{i,k}(u)$ or $M_{j,l}(w)$ are zero, the surface point at u,w does not change

NURBS Surfaces: Algorithms

Implemented in 1981 and published in 1982

The algorithms provide

- dynamic real time interactive manipulation of
  - spatial position control net vertex
  - homogeneous weight
- on modest computer systems

Fast NURBS Surface Algorithm

Use $itest = (n+1) + (m+1)k + l + p_1 + p_2$ to determine if a complete new surface is required

if ($itest \neq (n+1) + (m+1)k + l + p_1 + p_2$) then
    calculate complete new surface (see previous)
else
    calculate incremental change to the surface
end if

Fast NURBS Surface Algorithm

```
if (itest == (n + 1) + (m + 1)k + l + p_1 + p_2) then
    calculate incremental change, if any,
    in the spatial coordinate or homogeneous
    weight of the vertex being manipulated
    if (any coordinate or weight changed) then
        if (homogeneous weight changed) then
            save the old Sum(u,w) function
            calculate the new Sum(u,w) function
            if (no change in homogeneous weight) then
                control net vertex changed
                calculate change in surface for each u,w
            else
                homogeneous weight changed
                calculate change in surface for each u,w
            end if
        end if
        save current vertex coordinates as old
        save current homogeneous weight as old
    end if
end if
```

Fast NURBS Surface Algorithm

Efficiency improvement

only spatial coordinate changes – factor of 38

only homogeneous weight changes – factor of 15

over the naive algorithms

Additional Topics

- Effiect of multiple coincident knot values
- Effiect of internal nonuniform knot values
- Effiect of negative weights
- Reparameterization
- Derivatives – Curvature
- Bilinear surfaces
- Ruled/Developable surfaces
- Sweep surfaces
- Surfaces of revolution
- Conic volumes
- Subdivision
- Trim surfaces
- Surface fitting
- Constrained surface fitting

Catmull-Rom Spline

- The Catmull-Rom Spline is a local interpolating spline developed for computer graphics and CAGD
  - Data points
  - Tangents at data points
- Development of the matrix form of Catmull-Rom Spline

Ferguson's Parametric Cubic Curves

Given

the two control points $\mathbf{P}_0$ and $\mathbf{P}_1$,

the slopes of the tangents $\mathbf{P}_0'$ and $\mathbf{P}_1'$ at each point,

Define a parametric cubic curve that

passes through $\mathbf{P}_0$ and $\mathbf{P}_1$,

with the respective slopes $\mathbf{P}_0'$ and $\mathbf{P}_1'$ at $\mathbf{P}_0$ and $\mathbf{P}_1$

By equating the coefficients of the following polynomial function

$$\mathbf{P}(t)=a_0+a_1t+a_2t^2+a_3t^3$$

with the values above, namely

$$\mathbf{P}(0)=a_0 \qquad \mathbf{P}(1)=a_1$$

$$\mathbf{P}'(0)=a_1 \qquad \mathbf{P}'(1)=a_1+2a_2+2a_3$$

Ferguson's Parametric Cubic Curves

Solving these equations simultaneously for a_0, a_1, a_2 and a_3, we obtain

$a_0 = \mathbf{P}(0)$ $\qquad$ $a_1 = \mathbf{P}'(0)$

$a_2 = 3[\mathbf{P}(1) - \mathbf{P}(0)] - 2\mathbf{P}'(0) - \mathbf{P}'(1)$ $\qquad$ $a_3 = 3[\mathbf{P}(1) - \mathbf{P}(0)] - 2\mathbf{P}'(0) - \mathbf{P}'(1)$

Substituting these into the original polynomial equation and simplifying to isolate the terms with $\mathbf{P}(0)$ and $\mathbf{P}(1)$, $\mathbf{P}'(0)$ and $\mathbf{P}'(1)$ we have

$$
\begin{aligned}
\mathbf{P}(t) = {} & (1-3t^2+2t^3)\mathbf{P}(0) \\
& +(3t^2-2t^3)\mathbf{P}(1) \\
& +(t-2t^2+t^3)\mathbf{P}'(0) \\
& +(-t^2+t^3)\mathbf{P}'(1)
\end{aligned}
$$

Ferguson's Parametric Cubic Curves

It is clearly in a cubic polynomial form. Alternatively, this can be written in the following matrix form

$$\mathbf{P}(u) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}(0) \\ \mathbf{P}(1) \\ \mathbf{P}'(0) \\ \mathbf{P}'(1) \end{bmatrix}$$

This method can be used to obtain a curve through a more general set of control points $\{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n\}$ by considering pairs of control points and using the Ferguson method for two points as developed above. It is necessary, however, to have the slopes of the tangents at each control point.

Catmull-Rom Spline

Given $n+1$ control points $\{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n\}$, find a curve that

- **interpolates** these control points (i.e. passes through them all)
- is **local** in nature (i.e. if one of the control points is moved, it only affects the curve locally)

For the curve on the segment $\mathbf{P}_i\mathbf{P}_{i+1}$, using $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ as two control points, specifying the tangents to the curve at the ends to be

$$\frac{\mathbf{P}_{i+1} - \mathbf{P}_{i-1}}{2} \quad \text{and} \quad \frac{\mathbf{P}_{i+2} - \mathbf{P}_{i}}{2}$$

Substituting these tangents into Ferguson's method, we obtain the matrix equation

Catmull-Rom Spline

$$\mathbf{P}(u)=\begin{bmatrix}1 & t & t^2 & t^3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ -3 & 3 & -2 & -1\\ 2 & -2 & 1 & 1\end{bmatrix}\begin{bmatrix}\mathbf{P}_i\\ \mathbf{P}_{i+1}\\ \dfrac{\mathbf{P}_{i+1}-\mathbf{P}_{i-1}}{2}\\ \dfrac{\mathbf{P}_{i+2}-\mathbf{P}_{i}}{2}\end{bmatrix}$$

$$\mathbf{P}(u)=\begin{bmatrix}1 & t & t^2 & t^3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ -3 & 3 & -2 & -1\\ 2 & -2 & 1 & 1\end{bmatrix}\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ -1/2 & 0 & 1/2 & 0\\ 0 & -1/2 & 0 & 1/2\end{bmatrix}\begin{bmatrix}\mathbf{P}_{i-1}\\ \mathbf{P}_i\\ \mathbf{P}_{i+1}\\ \mathbf{P}_{i+2}\end{bmatrix}$$

Catmull-Rom Spline

Multiplying the two inner matrices, we obtain

$$\mathbf{P}(u) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} M \begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix}$$

where

$$M = \frac{1}{2}\begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

For the first and last segments in which $\mathbf{P}_0'$ and $\mathbf{P}_n'$ must be defined by a different method.

Catmull-Rom Spline: Example