

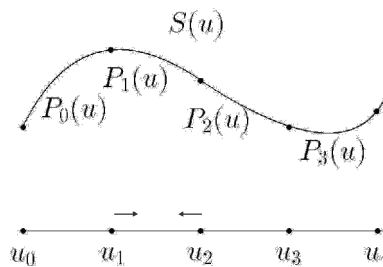
B-Spline Curves

Multiple knots

Modeling for computer graphics
Faramarz Samavati

Multiple knots

- ❖ Vary the spacing between knots can control the shape
- ❖ multiple knot in the limit

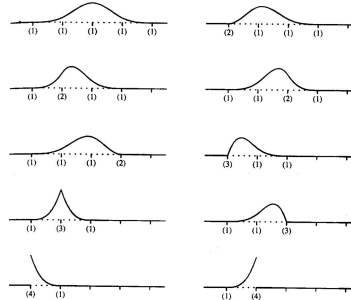


Modeling for Computer
Graphics
Faramarz Samavati

What happens for B-Splines?

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

- ❖ The fraction will be set zero if its denominator is zero .
- ❖ Cubic B-spline
- ❖ Multiplicities are indicated

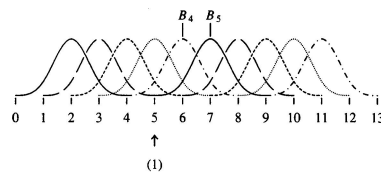


The effect of the multiple knot: Multiplicity 1

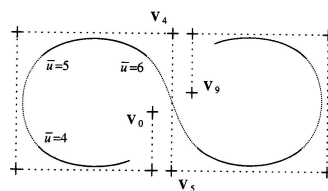
- ❖ Consider the uniform cubic ($k=4$) B-spline curve

$$u = \{0, 1, \dots, 13\}$$

$$m=9, k=4 \implies [u_3, u_{10}) \implies 7 \text{ segments}$$

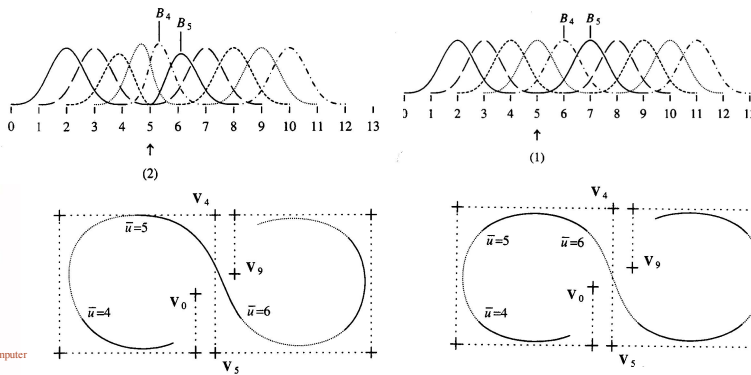


➤ continuity = 2



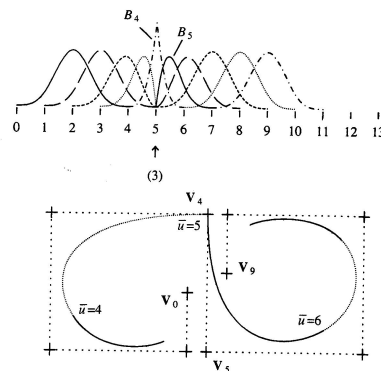
The effect of the multiple knot: Multiplicity 2

- ❖ Double knot at 5 or $\mu_i = 2$
- ❖ $\{0, 1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12\}$
- ❖ 6 segments
- ❖ continuity = 1



Multiplicity 3

- ❖ Triple knot at 5 or $\mu_i = 3$
- ❖ $\{0, 1, 2, 3, 4, 5, 5, 5, 6, 7, 8, 9, 10, 11\}$
- ❖ 5 segments
- ❖ Interpolating the corresponding control points

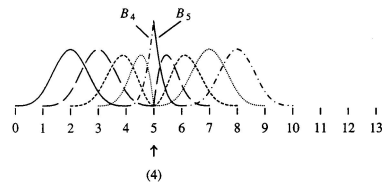


❖ continuity = 0

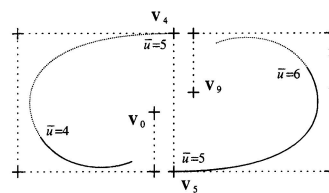
❖ the height of N_4 is 1!

Multiplicity 4

- ❖ Quadruple knot at 5 $\mu_i = 4$
- ❖ 4 segments



- ❖ continuity = -1



An interesting case: the standard Knot Sequence

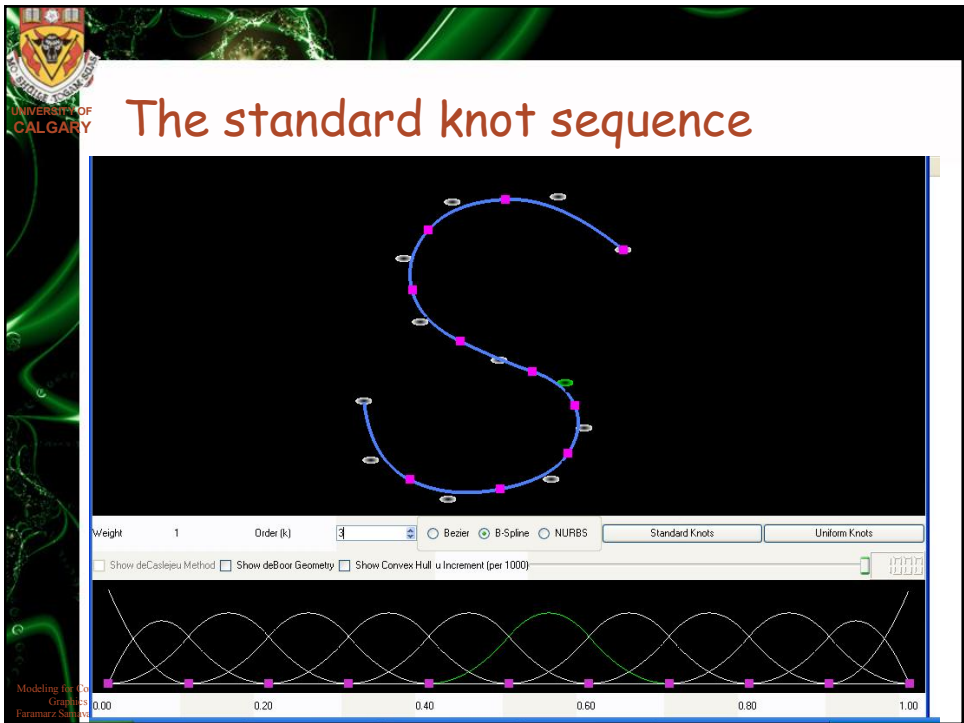
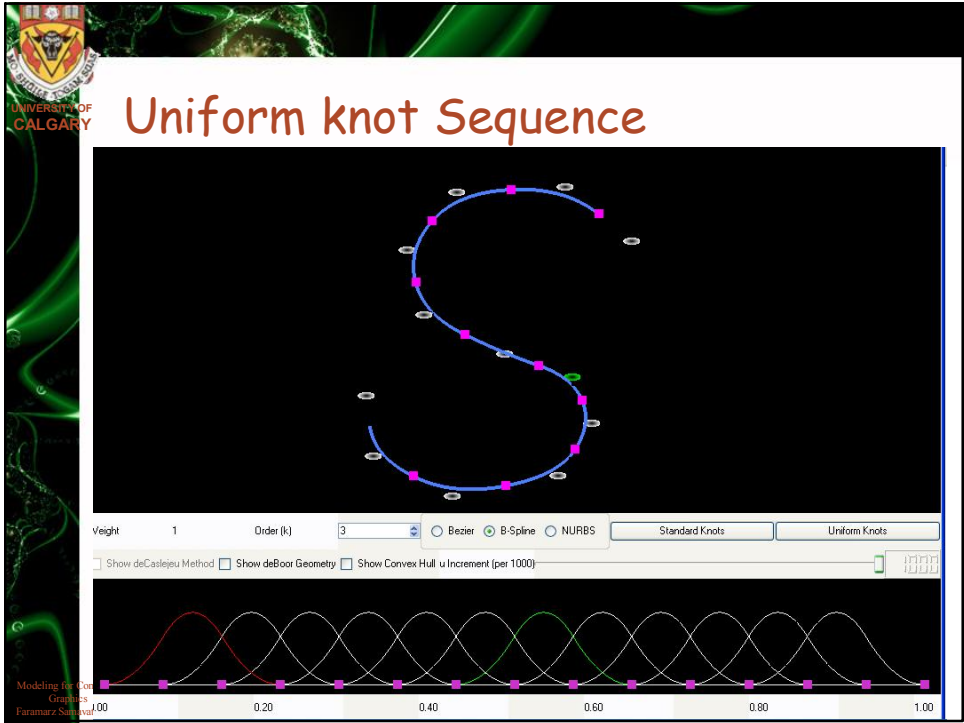
- ❖ For interpolating the first and last control points (clamped curve)
- ❖ Multiple knots in the beginning and the end of knots
- ❖ Using uniform knot spacing between (0,1),

for example step = 0.1

- ❖ k=2 , [0, 0, 0.1, 0.2, ..., 0.9, 1, 1]
- ❖ k=3 , [0, 0, 0, 0.1, 0.2, ..., 0.9, 1, 1, 1]
- ❖ k=4 , [0, 0, 0, 0, 0.1, 0.2, ..., 0.9, 1, 1, 1, 1]

- ❖ Sample program

- ❖ What is the parameter range in each case of above?





Bezier curves are B-spline curves

- ❖ Converting the designs with the Bezier curves to the B-splines
- ❖ Changing PS or Metafont formats to TTF
- ❖ Changing 3D formats
- ❖ Rendering Bezier models by B-spline software



Bezier Curve From B-splines

- ❖ Connection between the Bernstein polynomials and the B-splines
- ❖ B-spline of order $d+1$ over a special knot sequence \implies Bernstein degree d
 - ❖ knot sequence $= \underbrace{\{0, 0, \dots, 0\}}_{d+1}, \underbrace{\{1, 1, \dots, 1\}}_{d+1}$
 - ❖ Simple case , $\{0,0,0,0,1,1,1,1\}$ where $d=3$
 - ❖ What is m ? What is the parameter range?
 - ❖ $m=3$, $[0,1]$ \implies 1 segment





Sum of B-Splines

- ❖ Curve and surface use sum of B-splines
- ❖ Curve in space

$$Q(u) = \begin{bmatrix} X(u) \\ Y(u) \\ Z(u) \end{bmatrix} = \begin{bmatrix} \sum x_i N_{i,k}(u) \\ \sum y_i N_{i,k}(u) \\ \sum z_i N_{i,k}(u) \end{bmatrix}, p_i = (x_i, y_i, z_i)$$

$$\sum E_i N_{i,k}(u)$$

General notation for all three components
 E_i 's are given real numbers



B-Spline Algorithm



Computing sums of B-splines

$$S(u) = \sum_i E_i N_{i,k}(u)$$

- ❖ It appears in curve, surface, NURBS and their derivatives
- ❖ u is a given and fixed parameter value
- ❖ E_i given real coefficient (such as x_i)



Efficient Algorithm

- ❖ To avoid re-calculating of Basis functions
- ❖ To deal with just non-zero basis (the local support property)



Non Zero B-splines at Given u

- ❖ We know $u_\delta \leq u < u_{\delta+1}$
- ❖ Many B-splines are zero

$$\begin{array}{cccccc}
 0 & 0 & 0 & \dots & N_{\delta-k+1,k} \\
 0 & 0 & 0 & \dots & \vdots \\
 0 & 0 & N_{\delta-2,3} & \dots & \vdots \\
 0 & N_{\delta-1,2} & N_{\delta-1,3} & \dots & N_{\delta-1,k} \\
 N_{\delta,1} & N_{\delta,2} & N_{\delta,3} & \dots & N_{\delta,k} \\
 0 & 0 & 0 & \dots & 0
 \end{array}$$

Modeling for Computer
Graphics
Farmanz Samanati



δ Index

- ❖ Given knot sequence : $\{u_0, u_1, \dots, u_{m+k}\}$
- ❖ Example : $\{0, 1, 2, 3, 3, 4, 5, 5, 5, 6\}$, $k=3$, $m=6$
- ❖ u is given parameter (for example $u=2.8$)
- ❖ We define δ as the unique index such that

$$u_\delta \leq u < u_{\delta+1}$$

Example : $\delta(2.8)=2$
 $\delta(3)=4$
 $\delta(3.1)=4$
 $\delta(5)=8$

```

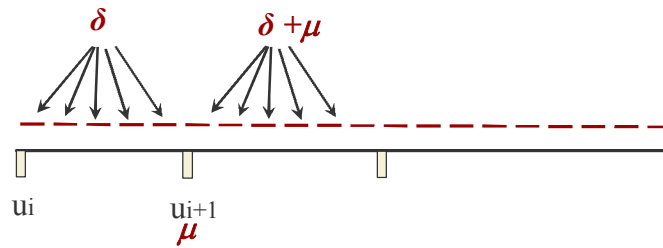
function delta(u)
// u[i], m and k are global variables
for (i=0 to m+k-1
  if (u>=u[i] and u<u[i+1]) then
    return i;
end;
return(-1);
//it is not in this knot sequence

```

Modeling for Computer
Graphics
Farmanz Samanati

Sequential values for u

- ❖ in this case delta(u) can upgrade to a more efficient algorithm



Order Reduction

$$\begin{aligned}
 \sum_{i=0}^m E_i N_{i,k}(u) &= \sum_{i=0}^m E_i \left[\frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u) \right] \\
 &= \sum_{i=0}^m E_i \frac{u - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(u) + \sum_{i=0}^m E_i \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(u) \\
 &= \sum_{i=0}^m \left[\frac{u - u_i}{u_{i+k-1} - u_i} E_i + \frac{u_{i+k-1} - u}{u_{i+k-1} - u_i} E_{i-1} \right] N_{i,k-1}(u) \\
 &= \sum_{i=0}^m E_i^{[k-1]} N_{i,k-1}(u)
 \end{aligned}$$

- ❖ Where E-1 is taken zero

Repetition of Degree Reduction

- ❖ We know

$$S(u) = \sum_{i=0}^m E_i N_{i,k}(u) = \sum_{i=0}^m E_i^{[k-1]} N_{i,k-1}(u)$$

- ❖ We can use non zero terms so that:

$$\sum_{i=\delta-k+1}^{\delta} E_i N_{i,k}(u) = \sum_{i=\delta-k+2}^{\delta} E_i^{[k-1]} N_{i,k-1}(u)$$

- ❖ We can also use the same result for right summation again

$$S(u) = \sum_{i=0}^{m+k-1} E_i^{[1]} N_{i,1}(u) = \sum_{i=\delta}^{\delta} E_i^{[1]} N_{i,1}(u) = E_{\delta}^{[1]}$$

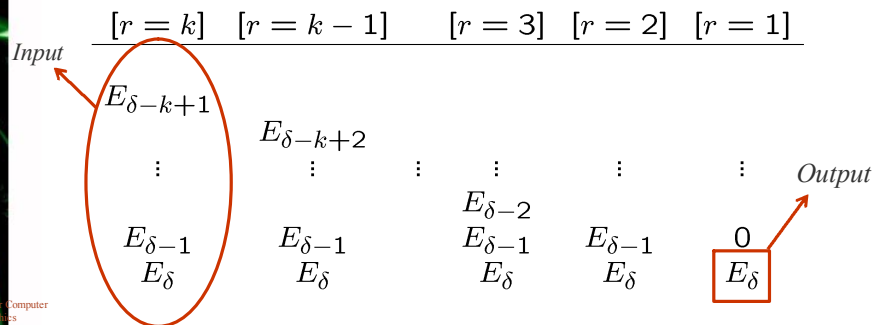
Diagram

- ❖ Where

$$E_i^{[k]} = E_i, \quad i = \delta - k + 1, \delta - k + 2, \dots, \delta$$

$$E_i^{[r-1]} = \frac{u-u_i}{u_i+r-1-u_i} E_i^{[r]} + \frac{u_i+r-1-u}{u_i+r-1-u_i} E_{i-1}^{[r]}, \quad i = \delta - r + 2, \delta - r + 3, \dots, \delta$$

$$r = k, k-1, \dots, 2$$



Updating

- ❖ One step in the recurrence

$$\frac{\begin{matrix} [r] & [r-1] \\ E_{i-1} \end{matrix}}{E_i} \xrightarrow[\omega]{1-\omega} E_i \quad \omega = \frac{u-u_i}{u_{i+r-1}-u_i}$$

- ❖ We move from left to right and from the bottom to up.

Pseudo Code for Sums of B-spline

```

Input k,m, E[],u[],u
//k: order of B-spline
//m: number of control points
//E[]: coefficient vector( can be x[], y[], z[] of the control points
//u[]: knot sequence
//u: fixed parameter value
δ =delta(u);
for i=0 to k-1
    c[i]=E[δ-i]; //nonzero coefficients
endfor;
for r=k to 2 step -1
    i= δ;
    for s=0 to r-2
        omega=(u-u[i])/(u[i+r-1]-u[i]);
        c[s]=omega*c[s]+(1-omega)*c[s+1];
        i=i-1;
    endfor;
endfor;
output c[0];
    
```

