2. B-Spline Curves
Overview

• B-Spline Basis Functions
• B-Spline Curves
• deBoor Algorithm
• End Conditions
• Interpolation
B-Spline Curves

• Disadvantages of Bézier curves:
  – Global support of the basis functions
  – Insertion of new control points comes along with degree elevation
  – $C^r$-continuity between individual segments of a Bézier curve

$\Rightarrow$ B-Spline bases help to overcome these problems (Local support, continuity control, arbitrary knot vector)
• A B-Spline curve:
  – Endpoint interpolation?
  – Continuity control
  – Uniform
B-Spline Bases of Different Degree

konstant
\[ N_0^0(u) \]

linear
\[ N_2^1(u) \]

quadratisch
\[ N_5^2(u) \]
B-Spline Functions

• Definition:
  – A B-Spline curve $s(u)$ built from piecewise polynomial bases
    $$s(u) = \sum_{i=0}^{k} d_i N_{i}^{n}(u)$$
  – Coefficients $d_i$ of the B-Spline basis function are called *de Boor* points
  – Bases are piecewise, recursively defined polynomials over a sequence of knots $u_0 < u_1 < u_2 < \ldots$
  – Defined by a knot vector $T = u = [u_0, \ldots, u_{k+n+1}]$
• B-Spline bases:
  – Different degrees
  – Piecewise polynomial
  – Local support
  – uniform / non-uniform
  – B-Splines-Bernstein polynomials
B-Spline Functions

• Properties:
  – Partition of Unity: \( \sum_i N_i^n(u) \equiv 1 \)
  – Positivity: \( N_i^n(u) \geq 0 \)
  – Compact support: \( N_i^n(u) = 0, \ \forall u \notin [u_i, u_{i+n+1}] \)
  – Continuity: \( N_i^n \) is \((n-1)\) times continuously differentiable
B-Spline Functions

- From the recurrence formula we obtain:

\[
N_i^1(u) = \begin{cases} 
\frac{u-u_i}{u_{i+1}-u_i}, & u \in [u_i, u_{i+1}] \\
\frac{u_{i+2}-u}{u_{i+2}-u_{i+1}}, & u \in [u_{i+1}, u_{i+2}] 
\end{cases}
\]

\[
N_i^2(u) = \frac{u-u_i}{u_{i+2}-u_i} N_i^1(u) + \frac{u_{i+3}-u}{u_{i+3}-u_{i+1}} N_{i+1}^1(u)
\]

\[
= \begin{cases} 
\frac{u-u_i}{u_{i+2}-u_i} \cdot \frac{u-u_i}{u_{i+1}-u_i}, & i \in [u_i, u_{i+1}] \\
\frac{u-u_i}{u_{i+2}-u_i} \cdot \frac{u_{i+2}-u_i}{u_{i+2}-u_{i+1}} + \frac{u_{i+3}-u}{u_{i+3}-u_{i+1}} \cdot \frac{u-u_i}{u_{i+2}-u_{i+1}}, & i \in [u_{i+1}, u_{i+2}] \\
\frac{u_{i+3}-u_{i+1}}{u_{i+3}-u} \cdot \frac{u_{i+3}-u}{u_{i+3}-u_{i+2}}, & i \in [u_{i+2}, u_{i+3}] 
\end{cases}
\]
B-Spline Functions

- Recurrence relation:

\[
N_i^n(u) = (u - u_i) \frac{N_i^{n-1}(u)}{u_{i+n} - u_i} + (u_{i+n+1} - u) \frac{N_{i+1}^{n-1}(u)}{u_{i+n+1} - u_{i+1}}
\]

where:

\[
N_i^0(u) = \begin{cases} 
1, & u \in [u_i, u_{i+1}] \\
0, & \text{else}
\end{cases}
\]

The student might verify that B-Spline bases of degree n have support over \( n+1 \) intervals of the knot vector.
So-called B-Spline filters are widely used in signal processing. Cardinal B-Splines over uniform knot sequences can be computed using the convolution operator as:

\[ N_i^n = N^{n-1} \ast N^0 = \int_0^x N^{n-1}(t)N^0(x-t)dt \]

\[ N^0 : box - function \]
B-Spline Functions

- **uniform B-Splines vs. non-uniform B-Splines**

  *Continuity: Curve is globally $C^{n-1}$ continuous.*

- Exception:
  multiple knots of order $p$ with $u_j = \ldots = u_{j+p-1}$
  lead to $C^{n-p}$ continuous curves ($p < n+1$)

- Properties:
  
  - $\Rightarrow$ *variation diminishing property*: More restrictive,
    for $n+1$ adjacent deBoor points
  - $\Rightarrow$ *convex hull property*: More restrictive,
    for $n+1$ adjacent deBoor points
deBoor Algorithm

- Generalization of deCasteljau’s method.
- Evaluation of a point on the curve at $u = t$.
- For a given $t \in [u_l, u_{l+1}]$ all $N^k_i(u)$ are vanishing in spite of $i \in \{l-n, \ldots, l\}$.

This is a direct consequence of the local support of the bases.

- Point $s(t)$ computed by successive linear interpolation
- Control point in $k$-th step

$$d^k_i = (1 - a^k_i) d^k_{i-1} + a^k_i d^k_i$$

$$a^k_i = \frac{t - u_i}{u_{i+n+1-k} - u_i}$$

where $d^0_i = d_i$, $d^n_i = s(t)$
deBoor Algorithm (non-uniform)
• deBoor algorithm:
  – Successive linear interpolation
  – Local support (Principles of locality)
  – Bernstein polynomials
  – Different end conditions
deBoor Algorithm

• Special case: First and last knot have multiplicity of $n+1$:

$$0 = u_0 = u_1 = \ldots = u_n < u_{n+1} = u_{n+2} = \ldots = u_{2n+1}$$

• with $u_{n+k} = 1$ for $k \in [1,\ldots,n+1]$ we obtain:

$$d_i^k(u) = ud_i^{k-1}(u) + (1-u)d_{i+1}^{k-1}(u)$$

(de Casteljau-Algorithm)
End Conditions

• Open curves:
  – Design of endpoint interpolating B-Spline curves of degree $n$ by knot vectors of type:
    
    $$ u = T = (u_0 = u_1 = \ldots = u_{n-1} = u_n, u_k = u_{k+1} = \ldots = u_{k+n}) $$
  
  – Sequencing of knots influences the sweep of the curve
  – Example: Cubic bases with $T_1 = (0,0,0,1,2,3,4,5,5,5,5)$
    and $T_2 = (0,0,0,0,1,2.75,3.25,4,5,5,5,5)$:
    In both cases we get different bases at the boundaries
    $$ N_0^3(0) = 1 = N_0^3(5) $$
End Conditions

• Closed curves:
  - Periodic repetition of the deBoor points and knots by $d_0 = d_{k+1}$
    
    $u_{k+1} = u_0$
    
    $u_{k+2} = u_{k+1} + (u_1 - u_0)$
    
    $u_{k+3} = u_{k+2} + (u_2 - u_1)$
    
    ... 
    
  - The knot vector:

    $T = (u_0, u_1, ..., u_k, u_{k+1} = u_0, u_{k+2} = u_2, ..., u_{k+n} = u_{n-1})$
End Conditions

- **Parametric B-Spline curve:**
  \[ s(u) = \sum_{i=0}^{k} d_i N_i^n(u), \quad u \in [u_0, u_{n-1}] \]

- **Support of the bases:**
  \[ N_0^n \Rightarrow [u_0, \ldots, u_{n+1}] \]
  \[ N_1^n \Rightarrow [u_1, \ldots, u_{n+2}] \]
  \[ N_2^n \Rightarrow [u_2, \ldots, u_{n+3}] \]
  
  ... 

  \[ N_{k-2}^n \Rightarrow [u_{k-2}, u_{k-1}, u_k, u_0, \ldots, u_{n-2}] \]
  \[ N_{k-1}^n \Rightarrow [u_{k-1}, u_k, u_0, \ldots, u_{n-1}] \]
  \[ N_k^n \Rightarrow [u_k, u_0, \ldots, u_n] \]
B-Spline Interpolation

- Interpolate a given set of $k+1$ points $p_j$
- Let $u_j \in [u_0, \ldots, u_{k+n+1}]$ a straightforward insertion yields
  \[ s(u_j) = \sum_{i=0}^{k} d_i N_i^0(u_j) = p_j \]
- However, the curve needs $n+1$ active bases in the interval of definition
- System is under-determined
- We need more control points $d_0, \ldots, d_{k+n-1}$
  \[ s(u_j) = \sum_{i=0}^{k+n-1} d_i N_i^n(u_j) = p_j \]
Interpolating B-Spline

Endpoints $p_3 = d_0$ and $p_8 = d_7$ as well as tangents ($q_a = d_1$ and $r_b = d_6$) have to be preset.
B-Spline Interpolation

- For endpoint interpolating splines we need \( n+k \) equations, whereof \( k-1 \) define the interior intervals and \( n+1 \) the boundaries.

- Interpolation costs two equations:
  \[
  d_0 = p_0, \quad d_{k+n-1} = p_k
  \]

- Others can be used to specify tangency, curvature etc.
  \[
  t_0 = d_1 - d_0, \quad t_k = d_{k+n-2} - d_{k+n-1}
  \]
Again: JAVA-Applet

- Illustration of the interpolation problem
B-Spline Interpolation

- For a cubic:

\[
\begin{bmatrix}
1 & 0 \\
-1 & 1 \\
\vdots & \vdots \\
1 & -1 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
d_0 \\
d_1 \\
\vdots \\
d_{n+k-1}
\end{bmatrix}
= \begin{bmatrix}
p_0 \\
t_0 \\
p_1 \\
p_{k-1} \\
t_k \\
p_k
\end{bmatrix}
\]