Suranaree University
of Technology

COMPUTER GRAPHICS
School of Computer Engineering

Lecture 10 3-Dimensional Object Representations (Part III)

Paramate Horkaew

School of Computer Engineering, Institute of Engineering
Suranaree University of Technology

Lecture Outline

- 3-D Object Representations
  - Polygonal Surfaces
  - Planes in 3D Space
  - Quadric Surfaces and Blobs
- Spline Representations
  - Interpolation and Approximation Spline
  - Continuity Conditions
  - Cubic Spline Interpolation
  - Bezier Curves and Surfaces
  - B-Spline Curves and Surfaces
  - Other Splines and Their Conversions
- Introduction to OpenGL
  - OpenGL Programming using C/C++

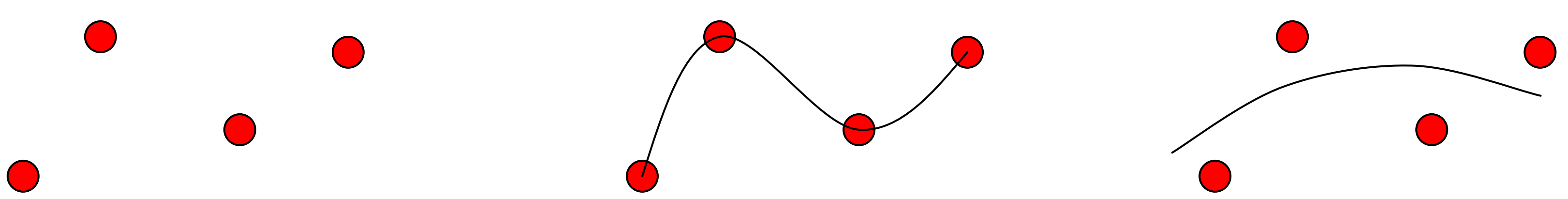
Interpolation and Approximation

การนิยามเส้นโค้งแบบ spline ทำได้โดยระบุ กลุ่มของพิกัด เรียกว่า control points ซึ่งกำหนดรูปทรงของ เส้นโค้ง โดยที่การลากเส้นโค้งตาม control points สามารถทำได้ 2 วิธี

- ลากโดยให้เส้นโค้งผ่าน ตำแหน่ง ของจุดที่กำหนดทุกๆ จุด (Interpolation)
  เหมาะสำหรับ การลากเส้นตามแบบ และ การกำหนดเส้นทางของการเคลื่อนไหวของวัตถุ (animation)

- ลากโดยให้เส้นโค้งผ่าน เส้นทาง ของจุดที่กำหนด โดยไม่จำเป็นต้องผ่านจุดนั้นๆ ก็ได้ (Approximation)
  เหมาะสำหรับ การออกแบบโครงสร้างของ พื้นผิวของวัตถุ

Interpolation **Approximation**



Spline Specifications

เราสามารถ สร้างเส้นโค้ง spline ได้ 3 วิธี

1. กำหนด เงื่อนไข ที่จุดปลายของเส้นโค้ง (Boundary Conditions)
2. กำหนด matrix ที่บ่งชี้คุณสมบัติของเส้นโค้ง
3. กำหนด ชุดของฟังก์ชันผสม (Blending Functions หรือ Basis Functions) ซึ่งระบุชุดเงื่อนไขทาง Geometry เพื่อใช้ในการคำนวณจุดพิกัด (x, y, z) สำหรับ (u) ใดๆ บนเส้นโค้ง

วิธีที่ 1 สำหรับเส้นโค้งพหุนาม $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad 0 \le u \le 1$

ถ้ากำหนด x (0), x (1), x′ (0) และ x′ (1) เราสามารถหาค่าคงที่ a, b, c, d ได้

โดยการแก้ระบบสมการ 4 ตัวแปร 4 สมการ (สำหรับ y, z คิดเหมือนกัน)

$$x(0) = d_x$$

$$x(1) = a_x + b_x + c_x + d_x$$

$$x'(0) = c_x$$

$$x'(1) = 3a_x + 2b_x + c_x$$

Suranaree University
of Technology

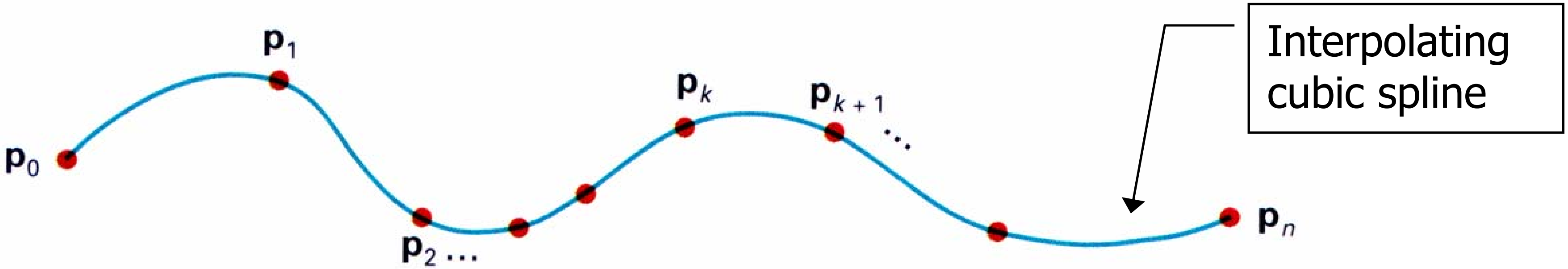
Cubic Spline Interpolation

Cubic Spline เหมาะสำหรับใช้ในการกำหนดเส้นทางการเคลื่อนไหวของวัตถุ หรือ กำหนดโครงร่างของวัตถุที่ได้จากการเก็บภาพภาพด้วยคอมพิวเตอร์ (เช่น laser digitization) เนื่องจาก เส้นโค้งจะผ่าน control points ทุกๆจุด

ฟังก์ชันพหุนามกำลัง 3 ใช้งานประเภทเหล่านี้อย่างแพร่หลายเนื่องจาก มีความยืดหยุ่น เทียบกับ ความเร็วในการคำนวณ ที่เหมาะสม ซึ่งมีขั้นตอนดังต่อไปนี้

ขั้นที่ 1 กำหนดพิกัดจุด control points จำนวน (n + 1) จุด

$$p_k = (x_k, y_k, z_k), \quad k = 0,1,2,\ldots,n$$



Solving Spline Coefficient

ขั้นที่ 2 สำหรับเส้นโค้งแต่ละช่วง (p_k, p_{k+1}) เราจะแก้สมการเพื่อหาสัมประสิทธิ์ $(a, b, c, d)_k$ ที่อธิบายเส้นโค้งในช่วงนั้นๆ

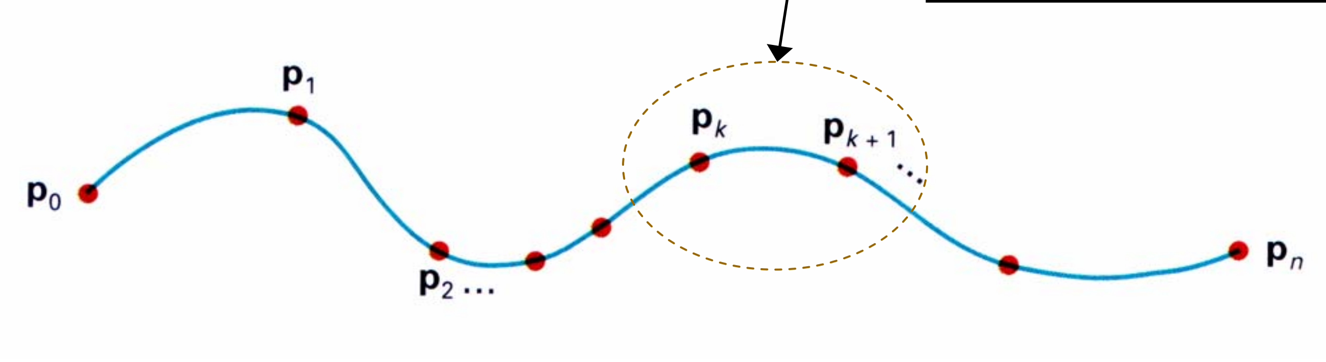
$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y \qquad 0 \le u \le 1$$

$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

สำหรับแต่ละสมการเราต้องการเงื่อนไขที่รอยต่อ 4 เงื่อนไข ซึ่งจะได้อธิบายต่อไป

Solving for each curve segment (k, k+1)



Natural Cubic Spline

พิจารณาเฉพาะพิกัด x ในระบบ Cartesian ซึ่งเป็นฟังก์ชันของ u

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad 0 \le u \le 1$$

เส้นโค้งที่มี n + 1 control points ประกอบด้วยจุดปลาย 2 จุด และ จุดภายใน n – 1 จุด โดยที่จุดภายในแต่ละจุด จะระบุเงื่อนไข parametric ด้วยอนุพันธ์อันดับที่ 0, 1 และ 2 ได้ 4 สมการ และ เงื่อนไขที่จุดปลายได้ 4 สมการ

อนุพันธ์อันดับ 1

$$x'(0) = c_x$$

$$x'(1) = 3a_x + 2b_x + c_x$$

อนุพันธ์อันดับ 2

$$x''(0) = 2b_x$$

$$x''(1) = 6a_x + 2b_x$$

อนุพันธ์อันดับ 0

$$x(0) = d_x$$

$$x(1) = a_x + b_x + c_x + d_x$$

Solving for the k – 1, k<u>th</u> Segments

สำหรับการหาสัมประสิทธิ์ (a, b, c, d)$^{k-1, k}$ (ตัวแปร 8 ตัว) ของเส้นโค้ง 2 เส้น ที่เชื่อมต่อที่ control point ที่ p_k สามารถสรุปได้ดังนี้

อนุพันธ์อันดับที่ 1 และ 2 ของเส้นโค้งที่ k – 1 และ k มีค่าเท่ากันที่จุด p_k

$$x'_{k-1}(1) = x'_k(0) \qquad x''_{k-1}(1) = x''_k(0)$$

อนุพันธ์อันดับที่ 0 ของเส้นโค้งที่ k – 1 และ k มีค่าเท่ากันที่จุด p_k เท่ากับ p_k

$$x_{k-1}(1) = p_k \qquad x_k(0) = p_k$$

ถ้าเรามีเส้นโค้ง n เส้น จุด p_k ที่เป็นจุดเชื่อมต่อของเส้นโค้งสองเส้น มีเพียง n – 1 จุด (จุดภายใน) ดังนั้นเราจะมีสมการเพียง 4 (n – 1) = 4n – 4 สมการ

แต่เส้นโค้ง n เส้นต้องการ 4n สมการ ดังนั้นเราต้องหาสมการเพิ่มอีก 4 สมการ

Specifying Boundary Conditions

สมการอีก 4 สมการที่เหลือ สามารถสร้างได้จาก จุดปลายของเส้นโค้ง คือจุด p_0 และ จุด p_n

อนุพันธ์อันดับที่ 2 ที่จุดปลายของ เส้นโค้งที่ 0 และเส้นโค้งที่ n – 1 มีค่าเท่ากับ 0

$$x_0''(0) = 0 \qquad x_{n-1}''(1) = 0$$

อนุพันธ์อันดับที่ 0 ที่จุดปลายของ เส้นโค้งที่ 0 และ n – 1 มีค่าเท่ากับพิกัดจุดนั้น

$$x_0(0) = p_0 \qquad x_{n-1}(1) = p_n$$

ดังนั้นเราจะมีสมการที่ได้จาก จุดภายใน p_k จำนวน 4n – 4 สมการ และ สมการที่ได้จากเงื่อนไขขอบเขต ที่จุดปลายอีก 4 สมการ รวมทั้งสิ้น 4n สมการ ซึ่งเพียงพอสำหรับคำนวณหาค่าสัมประสิทธิ์ของเส้นโค้ง n เส้น

$(a, b, c, d)^k$ โดยที่ k = 0 ถึง n – 1

An Example

ถ้ากำหนด control points 3 จุด จะมีทั้งหมด (3 – 1) x 4 = 8 สมการ

จุดภายใน

เงื่อนไขขอบเขต

$$
\begin{bmatrix}
x_0'(1) = x_1'(0) \\
x_0''(1) = x_1''(0) \\
x_0(1) = p_1 \\
x_1(0) = p_1 \\
x_0(0) = p_0 \\
x_1(1) = p_2 \\
x_0''(0) = 0 \\
x_1''(1) = 0
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
3 & 2 & 1 & 0 & 0 & 0 & -1 & 0 \\
6 & 2 & 0 & 0 & 0 & -2 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 6 & 2 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
a_x^0 \\ b_x^0 \\ c_x^0 \\ d_x^0 \\ a_x^1 \\ b_x^1 \\ c_x^1 \\ d_x^1
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 0 \\ p_1 \\ p_1 \\ p_0 \\ p_2 \\ 0 \\ 0
\end{bmatrix}
$$

ดังนั้นในการหา สัมประสิทธิ์ a, b, c, d ของตัวแปร x สำหรับ spline ที่มี control points 3 จุดสามารถทำได้โดยแก้สมการ 8 สมการ สำหรับ 8 ตัวแปร

Hermite Interpolation

ข้อเสียของ Natural Cubic Spline คือถ้าเราเปลี่ยนตำแหน่งของ control point เพียงตำแหน่งเดียว จะทำให้รูปร่างของเส้นโค้งเปลี่ยนไปโดยสิ้นเชิง (ไม่สามารถควบคุมเส้นโค้งเฉพาะบริเวณที่กำหนดได้)

Hermite Spline แก้ไขข้อจำกัดนี้ โดยที่สำหรับเส้นโค้งแต่ละช่วงจะขึ้นอยู่กับจุดปลาย 2 จุดของช่วงนั้น ถ้า **P** (u) แทนฟังก์ชันพหุนามกำลัง 3 สำหรับเส้นโค้งช่วงที่อยู่ระหว่าง control point p_k กับ p_{k+1} (*วิธีที่ 1 กำหนดเงื่อนไขอนุพันธ์*)

$$\mathbf{P}(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d} \quad 0 \le u \le 1$$

แล้วนิยามเงื่อนไขขอบเขต ดังนี้

$$\mathbf{P}(0) = \mathbf{p}_k \qquad \mathbf{P}'(0) = \mathbf{D}\mathbf{p}_k$$

$$\mathbf{P}(1) = \mathbf{p}_{k+1} \qquad \mathbf{P}'(1) = \mathbf{D}\mathbf{p}_{k+1}$$

โดยที่ **D** แทน ตัวดำเนินการอนุพันธ์ อันดับที่ 1

Hermite Spline Matrix

จากเงื่อนไขของ Hermite เราสามารถเขียนในรูปของ Matrix ได้ดังนี้

0th order

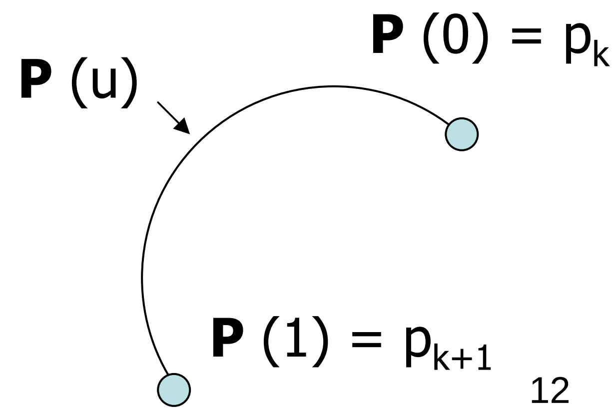
$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

1st order

$$\mathbf{P}'(u) = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

แทนค่าตัวแปร u = 0 และ 1 แล้วหาสมการตามเงื่อนไขของ Hermite Spline ได้

$$\begin{bmatrix} \mathbf{P}(0) \\ \mathbf{P}(1) \\ \mathbf{P}'(0) \\ \mathbf{P}'(1) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$



Solving for Coefficients

จากสมการ Matrix ของ Hermite Spline เราสามารถแก้หาเวคเตอร์ สัมประสิทธิ์ **a**, **b**, **c**, **d** ได้โดยการหา inverse ของ Matrix (*วิธีที่ 2 เขียนในรูป Matrix*)

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} = \mathbf{M}_H \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix}$$

โดยที่ M_H คือ Hermite Matrix ซึ่งมีค่าเฉพาะสำหรับ Hermite Spline

Hermite Closed Form

เมื่อแก้สมการหา ค่าเวกเตอร์ สัมประสิทธิ์ได้แล้ว เราสามารถเขียน Hermite Spline สำหรับจุด **P** (u) ใดๆ ระหว่าง control point ทั้งสอง ได้

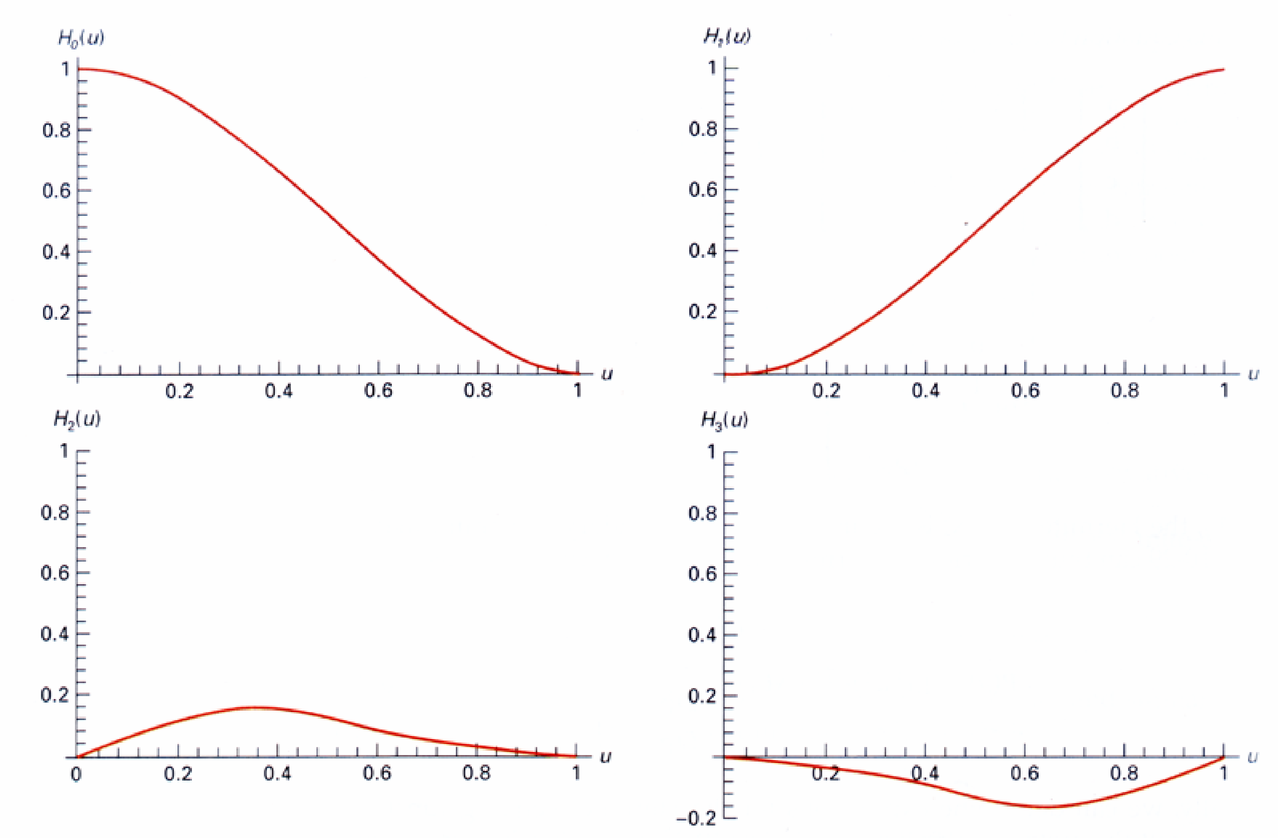
$$\mathbf{P}(u)=\begin{bmatrix}u^3 & u^2 & u^1 & 1\end{bmatrix}\cdot\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\\ \mathbf{c}\\ \mathbf{d}\end{bmatrix}=\begin{bmatrix}u^3 & u^2 & u^1 & 1\end{bmatrix}\cdot\mathbf{M}_H\cdot\begin{bmatrix}\mathbf{p}_k\\ \mathbf{p}_{k+1}\\ \mathbf{Dp}_k\\ \mathbf{Dp}_{k+1}\end{bmatrix}$$

$$=\begin{bmatrix}u^3 & u^2 & u^1 & 1\end{bmatrix}\cdot\begin{bmatrix}2 & -2 & 1 & 1\\ -3 & 3 & -2 & -1\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\cdot\begin{bmatrix}\mathbf{p}_k\\ \mathbf{p}_{k+1}\\ \mathbf{Dp}_k\\ \mathbf{Dp}_{k+1}\end{bmatrix}$$

จากสมการพบว่าผู้ใช้ต้องกำหนด ค่าความชัน **Dp** ที่ control points ด้วย

Hermite Spline as a Blending Function

$$\mathbf{P}(u) = \mathbf{p}_k(2u^3 - 3u^2 + 1) + \mathbf{p}_{k+1}(-2u^3 + 3u^2) + \mathbf{Dp}_k(u^3 - 2u^2 + u) + \mathbf{Dp}_{k+1}(u^3 - u^2)$$
$$= \mathbf{p}_k H_0(u) + \mathbf{p}_{k+1} H_1(u) + \mathbf{Dp}_k H_2(u) + \mathbf{Dp}_{k+1} H_3(u)$$



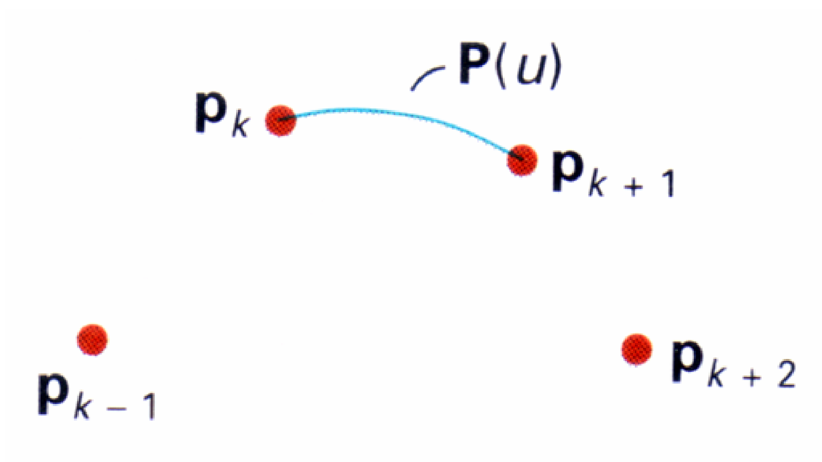
จากขั้นตอนที่แล้ว เราสามารถหาพิกัดของจุดใดๆ ภายในช่วงได้ โดยการหาผลบวกถ่วงน้ำหนักของ vector โดยที่ค่าผลคูณขึ้นอยู่กับระยะ u ระหว่าง $\mathbf{p}_k$ และ $\mathbf{p}_{k+1}$

วิธีที่ 3 ใช้ Basis Function

Cardinal Spline

ข้อเสียของ Hermite Spline คือผู้ออกแบบต้องกำหนด ทั้งพิกัดของ control points (**p**) และ ความชันของเส้นโค้ง ณ จุด control points (**Dp**) ซึ่งไม่สะดวก สำหรับการใช้งานบางประเภท

<u>Cardinal Spline</u> แก้ปัญหานี้โดยทำการคำนวณค่าความชันโดยอัตโนมัติ ด้วยวิธีการ Finite Difference (FD) ทั้งนี้ สำหรับเส้นโค้ง 1 ช่วงต้องพิจารณา control points จำนวน 4 จุด พร้อมๆ กัน ดังรูป และ สมการต่อไปนี้



อนุพันธ์ลำดับที่ 0 ที่จุดปลายมีค่าเท่ากับพิกัด

$$\mathbf{P}(0) = \mathbf{p}_k \qquad \mathbf{P}(1) = \mathbf{p}_{k+1}$$

อนุพันธ์ลำดับที่ 1 ที่จุดปลายหาได้จาก FD

$$\mathbf{P}'(0) = 0.5(1-t)(\mathbf{p}_{k+1} - \mathbf{p}_{k-1})$$

$$\mathbf{P}'(1) = 0.5(1-t)(\mathbf{p}_{k+2} - \mathbf{p}_k)$$

ค่าตัวแปร t เป็นตัวควบคุม ความตึง หรือ tension

Cardinal Spline Formula

หมายเหตุ สำหรับจุดปลายสุด p_0 จะไม่สามารถหา p_{0-1} ได้ และ สำหรับจุด p_{n-1} จะไม่สามารถหา $p_{(n-1)+2}$ ได้ ซึ่งจะทำให้สมการหายไป 2 สมการ คือ $P_0'(0)$ และ $P_{n-1}'(1)$ วิธีการแก้ไขคือ อาจแทนสมการที่หายไป ด้วยเงื่อนไขอนุพันธ์อันดับสอง เหมือนกรณี natural cubic spline ได้ โดย กำหนดให้ $P_0''(0) = 0$ และ $P_{n-1}''(1) = 0$ สำหรับจุดภายในที่ k สัมประสิทธิ์ ของเส้นโค้งที่ k หาได้จากชุดสมการ

$$\mathbf{P}_k(u) = \mathbf{a}_k u^3 + \mathbf{b}_k u^2 + \mathbf{c}_k u + \mathbf{d}_k$$

$$\mathbf{P}_k(0) = \mathbf{p}_k \longrightarrow \mathbf{P}_k(0) = \mathbf{d}_k = \mathbf{p}_k$$

$$\mathbf{P}_k(1) = \mathbf{p}_{k+1} \longrightarrow \mathbf{P}_k(1) = \mathbf{a}_k + \mathbf{b}_k + \mathbf{c}_k + \mathbf{d}_k = \mathbf{p}_{k+1}$$

$$\mathbf{P}_k'(u) = 3\mathbf{a}_k u^2 + 2\mathbf{b}_k u + \mathbf{c}_k \quad , \quad s = 0.5(1-t)$$

$$\mathbf{P}'(0) = 0.5(1-t)(\mathbf{p}_{k+1} - \mathbf{p}_{k-1}) \longrightarrow \mathbf{c}_k = s(\mathbf{p}_{k+1} - \mathbf{p}_{k-1})$$

$$\mathbf{P}'(1) = 0.5(1-t)(\mathbf{p}_{k+2} - \mathbf{p}_k) \longrightarrow 3\mathbf{a}_k + 2\mathbf{b}_k + \mathbf{c}_k = s(\mathbf{p}_{k+2} - \mathbf{p}_k)$$

Cardinal Spline Matrix

นำระบบสมการ ไปจัดรูป (จาก slide projector) แล้วเขียนในรูป Matrix ได้ดังนี้

$$\begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dfrac{s-1}{s} & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ \dfrac{3}{s} & \dfrac{2}{s} & \dfrac{1}{s} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_k \\ \mathbf{b}_k \\ \mathbf{c}_k \\ \mathbf{d}_k \end{bmatrix}$$

หา Inverse ของ Matrix เพื่อแก้สมการหาสัมประสิทธิ์ a, b, c, d

$$\begin{bmatrix} \mathbf{a}_k \\ \mathbf{b}_k \\ \mathbf{c}_k \\ \mathbf{d}_k \end{bmatrix} = \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix}$$

ให้ นักศึกษา ไปทดลองพิสูจน์ที่มาของสมการนี้ด้วยตนเอง

Cardinal Closed Form

เราสามารถเขียน Cardinal Spline สำหรับเส้นโค้งที่ k ในรูป matrix ได้ดังนี้

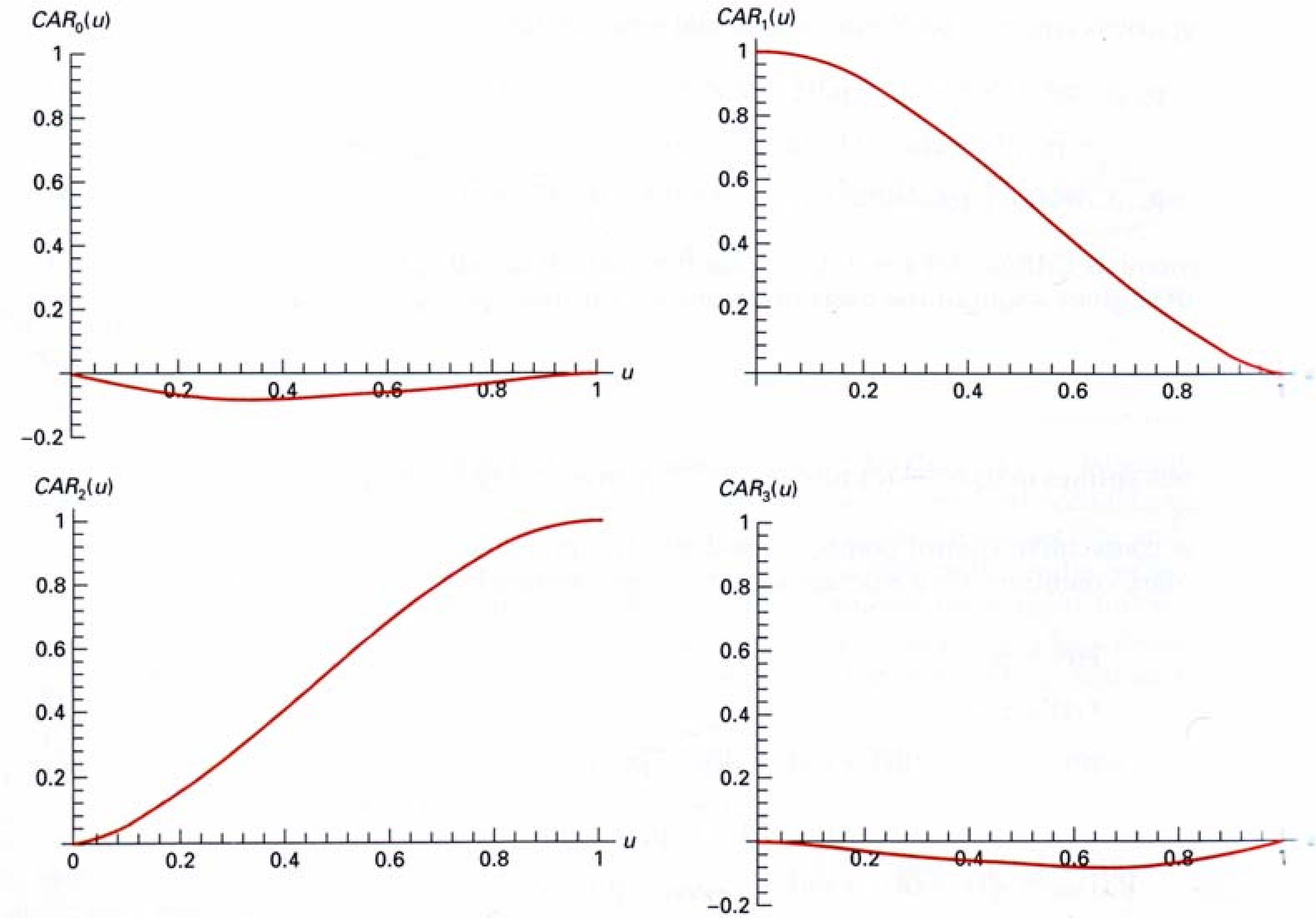
$$\mathbf{P}_k(u) = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a}_k \\ \mathbf{b}_k \\ \mathbf{c}_k \\ \mathbf{d}_k \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \mathbf{M}_{CAR} \cdot \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix}$$

$$= \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix}$$

เมื่อกระจายแล้วจะอยู่ในรูป

$$\mathbf{P}_k(u) = \mathbf{p}_{k-1} CAR_0(u) + \mathbf{p}_k CAR_1(u) + \mathbf{p}_{k+1} CAR_2(u) + \mathbf{p}_{k+2} CAR_3(u)$$

Cardinal Basis Function



Bezier Curves and Surfaces

Bezier Spline เป็น Approximating Spline (เส้นโค้งไม่จำเป็นต้องผ่าน control point ทุกจุด) ซึ่งมีคุณสมบัติคือ ผู้ใช้งานสามารถ ออกแบบเส้นโค้งได้สะดวก ทำให้มีใช้แพร่หลายในซอฟท์แวร์ Graphics เกือบทุกชนิด

โดยทฤษฎีแล้วเราสามารถสร้าง Bezier Curve ได้ 3 วิธีเหมือนกับ Interpolating Spline แต่วิธีการกำหนดพหุนามของ Basis/Blending Function จะสะดวกที่สุด

กำหนด control point จำนวน (n+1) จุด $\mathbf{p}_k = (x_k, y_k, z_k)$ โดยที่ k = 0 … n แล้ว Blending Function ของเส้นโค้ง <u>ทั้งเส้น</u> **P** (u) นิยามโดย

$$\mathbf{P}(u) = \sum_{k=0}^{n} \mathbf{p}_k BEZ_{k,n}(u)$$

โดยที่ $BEZ_{k,n}$ คือ Bernstein Polynomials อันดับที่ n ของจุดที่ k

$$BEZ_{k,n}(u) = C(n,k)u^k(1-u)^{n-k} \quad \text{โดยที่} \quad C(n,k) = \frac{n!}{k!(n-k)!}$$

Bezier Basis Function

เราสามารถเขียน Bezier Basis Function ในรูปความสัมพันธ์เวียนบังเกิดได้ดังนี้

$$BEZ_{k,n}(u) = (1-u)BEZ_{k,n-1}(u) + uBEZ_{k-1,n-1}(u) \quad n > k \geq 1$$

โดยที่ $BEZ_{k,k}(u) = u^k \quad BEZ_{0,k}(u) = (1-u)^k$

หรืออาจใช้การกระจาย Binomial มาช่วยได้ $C(n,k) = \dfrac{n-k+1}{k} C(n,k-1)$

$$BEZ_{0,n}(u) = C(n,0) \cdot u^0 \cdot (1-u)^{n-0}$$

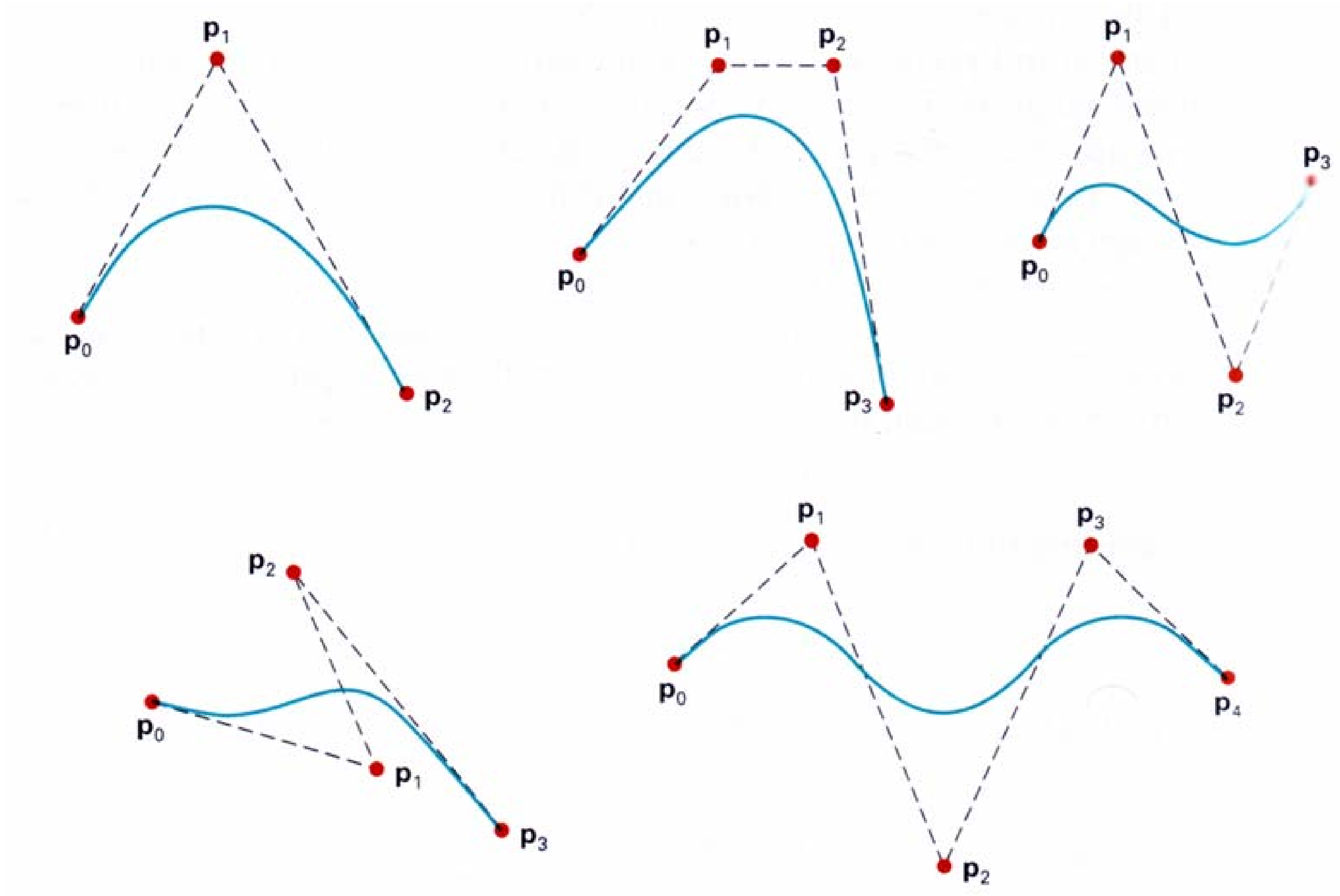
$$= \frac{n!}{0!(n-0)!} \cdot (1) \cdot (1-u)^n = (1-u)^n$$

$$BEZ_{1,n}(u) = C(n,1) \cdot u^1 \cdot (1-u)^{n-1}$$

$$= \frac{n-1+1}{1} \cdot C(n,0) \cdot u^1 (1-u)^{n-1} = n \cdot u^1 (1-u)^{n-1}$$

Bezier Curve Appearance

อันดับ (degree) ของเส้นโค้ง Bezier จะมีค่าน้อยกว่าจำนวน control point อยู่ 1



Properties of Bezier Curves

- เส้นโค้งจะผ่าน control point จุดปลายสองจุดเสมอ

$$\mathbf{P}(0)=\mathbf{p}_0 \quad \mathbf{P}(1)=\mathbf{p}_n$$

- ค่าอนุพันธ์อันดับที่ 1 ของจุดปลายเส้นโค้ง = ความชัน ของเส้นตรง ส่วนปลาย

$$\mathbf{P}'(0)=-n\mathbf{p}_0+n\mathbf{p}_1 \quad \mathbf{P}(1)=-n\mathbf{p}_{n-1}+n\mathbf{p}_n$$

- ค่าอนุพันธ์อันดับที่ 2 ของจุดปลายเส้นโค้ง = ความโค้ง ของเส้นโค้ง ส่วนปลาย

$$\mathbf{P}''(0)=n(n-1)[(\mathbf{p}_2-\mathbf{p}_1)-(\mathbf{p}_1-\mathbf{p}_0)]$$
$$\mathbf{P}''(1)=n(n-1)[(\mathbf{p}_{n-2}-\mathbf{p}_{n-1})-(\mathbf{p}_{n-1}-\mathbf{p}_n)]$$

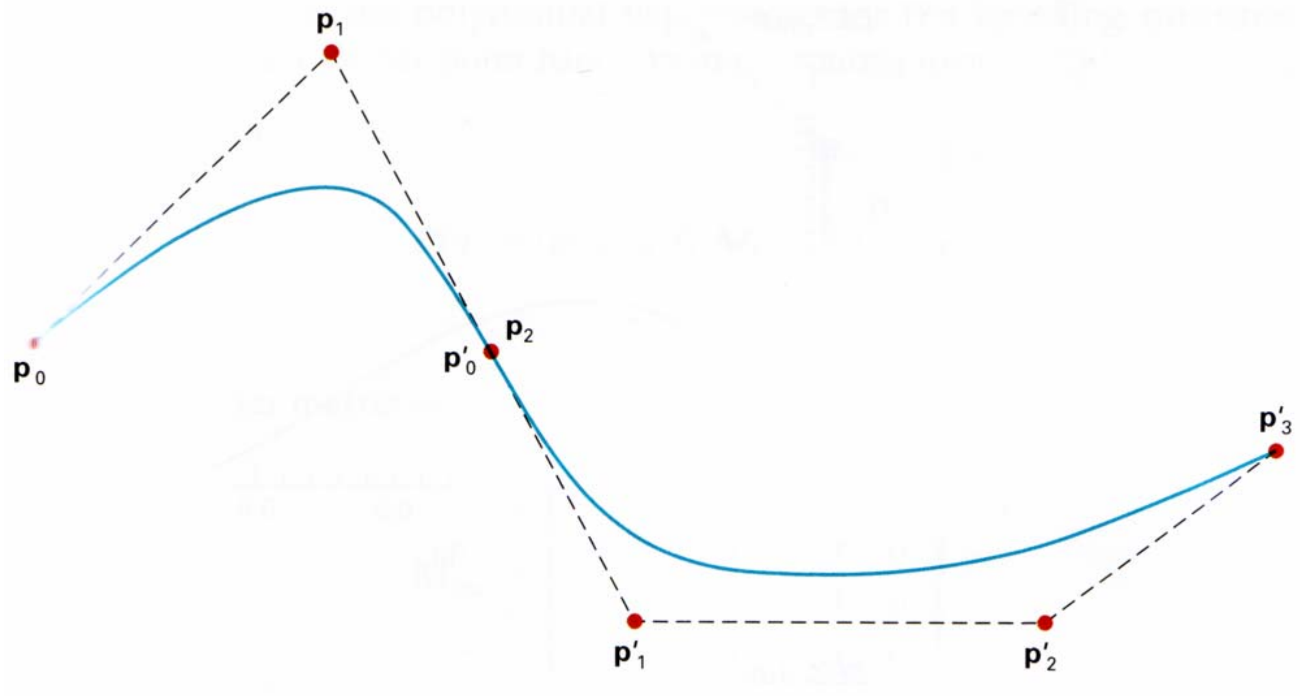
โดยที่ขนาดของอนุพันธ์ในข้อ 1 และ 2 แปรผันตรงกับ n และ n^2 ตามลำดับ

- เส้นโค้ง Bezier จะอยู่ใน Convex Hull เสมอ

$$\sum_{k=0}^{n} BEZ_{k,n}(u)=1$$

Design Techniques

จุดใดๆ บนเส้นโค้ง Bezier คำนวณมาจากฟังก์ชันพหุนาม อันดับที่ n ซึ่งถึงแม้ว่าจะเพิ่มความเร็วโดยใช้ recursion แล้วก็ยังไม่เหมาะสมสำหรับ การวาดเส้นโค้งที่ซับซ้อน นักออกแบบจึงนิยม นำเส้นโค้ง Bezier อันดับต่ำมาต่อกันดังรูป



- จุดแรกของเส้นโค้งที่ 2 (p'_0) อยู่ที่พิกัดเดียวกับจุดสุดท้ายของเส้นโค้งแรก (p_2)
- เส้นตรง $p'_0\, p'_1$ อยู่ในแนวเส้นตรงเดียวกันกับ $p_1\, p_2$ และมีความยาวเท่ากัน

Cubic Bezier Curves

ด้วยหลักการเดียวกันนี้เราสามารถสร้างเส้นโค้งใดๆ จากเส้นโค้ง Bezier อันดับที่ 3 (Cubic Bezier) ได้ ซึ่งนิยามในรูปของ Blending Function ดังนี้

$$BEZ_{0,3} = (1-u)^3$$

$$BEZ_{1,3} = 3u(1-u)^2$$

$$BEZ_{2,3} = 3u^2(1-u)$$

$$BEZ_{3,3} = u^3$$

โดยมีเงื่อนไขอนุพันธ์อันดับที่ 1 ดังนี้ (แก้ใน Text ด้วย)

$$\mathbf{P}'(0) = 3(\mathbf{p}_1 - \mathbf{p}_0)$$

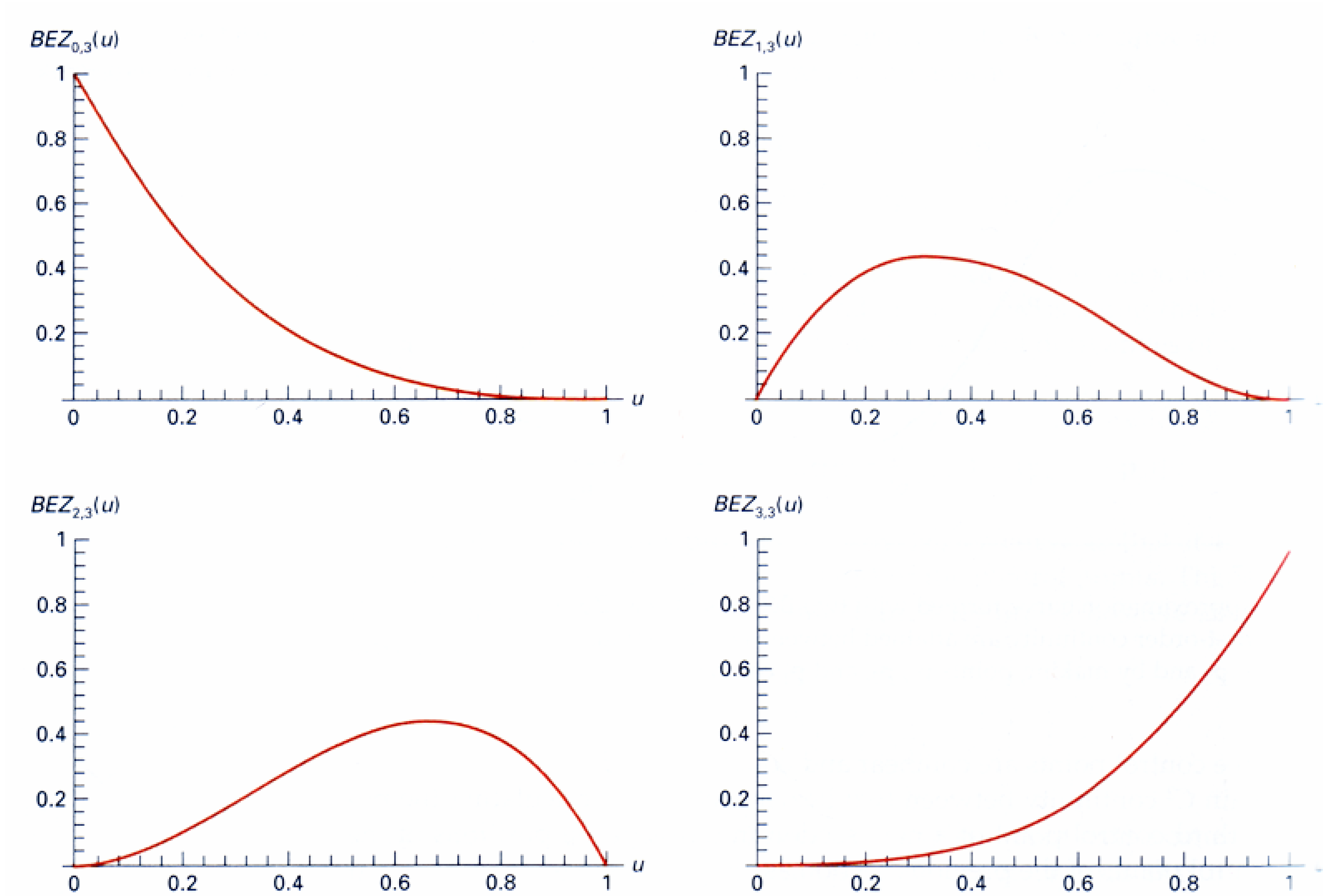
$$\mathbf{P}'(1) = 3(\mathbf{p}_3 - \mathbf{p}_2)$$

หรือในรูปของ Matrix

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \mathbf{M}_{BEZ} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{M}_{BEZ} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Cubic Bezier Basis Functions

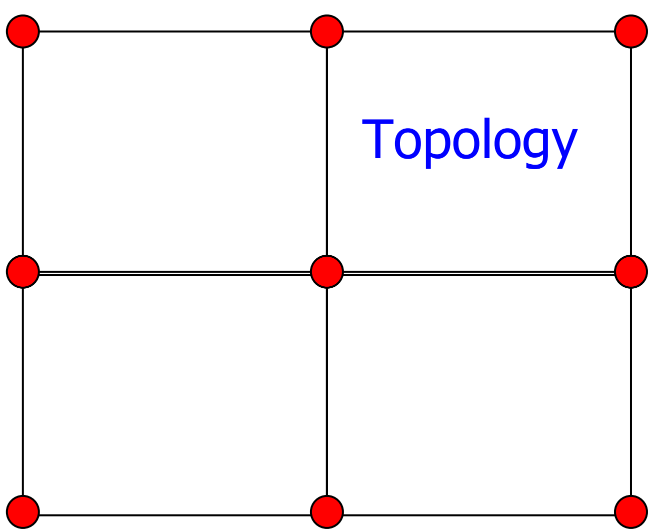


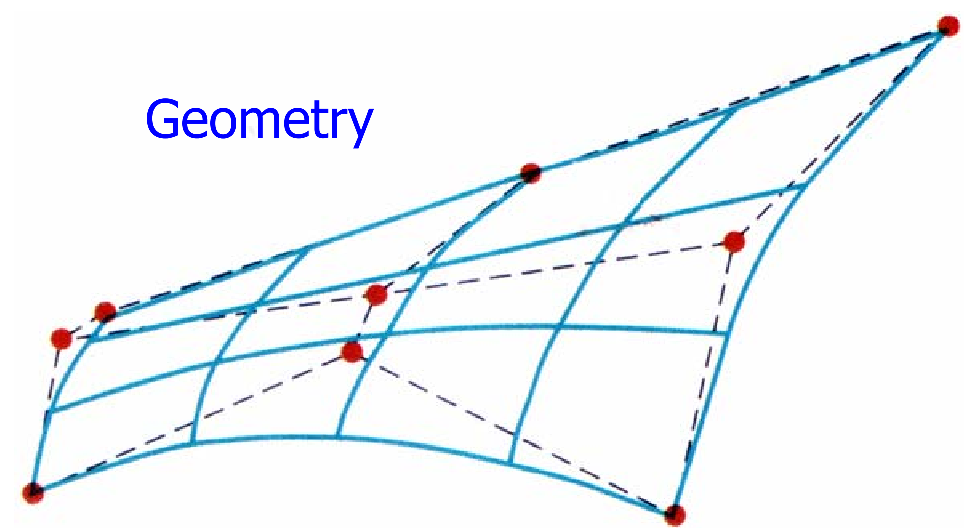
Bezier Surfaces

พื้นผิว Bezier (ใช้ในการออกแบบครั้งแรกสำหรับ ตัวถังรถ Renault) สร้างได้จาก ผลคูณ Tensor (Tensor Product) ของเส้นโค้ง Bezier สองชุด ดังนี้

$$\mathbf{P}(u,v)=\sum_{j=0}^{m}\sum_{k=0}^{m}\mathbf{p}_{j,k}BEZ_{j,m}(u)BEZ_{k,n}(u)$$

โดยที่ $p_{j,k}$ คือ control point หนึ่งจาก $(m + 1)\times(n + 1)$ control points





m = 2, n = 2

B-Spline Curves and Surfaces

B-Spline Curves เป็นรูปแบบเส้นโค้งที่ใช้มากที่สุด ในศาสตร์หลายแขนง ตั้งแต่ Computer Graphics จนถึง อุตุนิยมวิทยา และ อนุกรมวิธาน (ชีววิทยา) เนื่องจากมีข้อดี เมื่อเทียบกับ Bezier ตรงที่

- ผู้ใช้สามารถกำหนด อันดับของพหุนามได้อิสระ (กำหนดความโค้ง) โดยไม่ขึ้นกับจำนวน Control Points (แต่มีข้อจำกัดบางประการ)

- B-Spline มีความยืดหยุ่นสูง สามารถปรับแต่งเส้นโค้ง เฉพาะบริเวณได้ (คล้ายกับ Hermite) แต่มีความซับซ้อนมากกว่า Bezier Curves

กำหนด control point จำนวน (n+1) จุด $\mathbf{p}_k = (x_k, y_k, z_k)$ โดยที่ k = 0 … n แล้ว Blending Function ของเส้นโค้ง <u>ทั้งเส้น</u> **P** (u) นิยามโดย

$$\mathbf{P}(u) = \sum_{k=0}^{n} \mathbf{p}_k B_{k,d}(u), \quad u_{\min} \le u \le u_{\max}, \quad 2 \le d \le n+1$$

โดยที่ $B_{k,d}$ คือพหุนามอันดับ d – 1 โดย d เป็นจำนวนเต็มระหว่าง 2 ถึง n + 1

B-Spline Basis Function

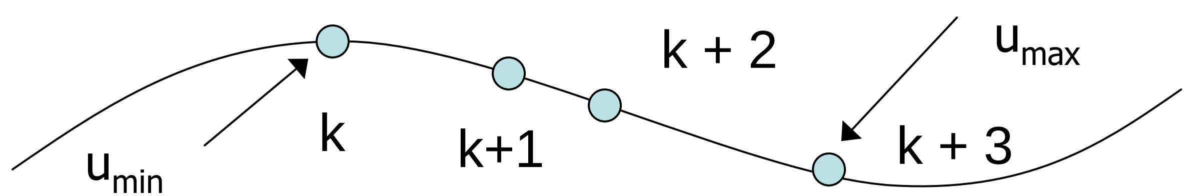
เราสามารถเขียน B-Spline Basis Function ในรูปความสัมพันธ์เวียนบังเกิด (สูตรของ Cox-deBoor) ได้ดังนี้

$$B_{k,1}(u)=\begin{cases}1, & u_k \le u \le u_{k+1}\\ 0, & \text{else}\end{cases}$$

$$B_{k,d}(u)=\frac{u-u_k}{u_{k+d-1}-u_k}B_{k,d-1}(u)+\frac{u_{k+d}-u}{u_{k+d}-u_{k+1}}B_{k+1,d-1}(u)$$

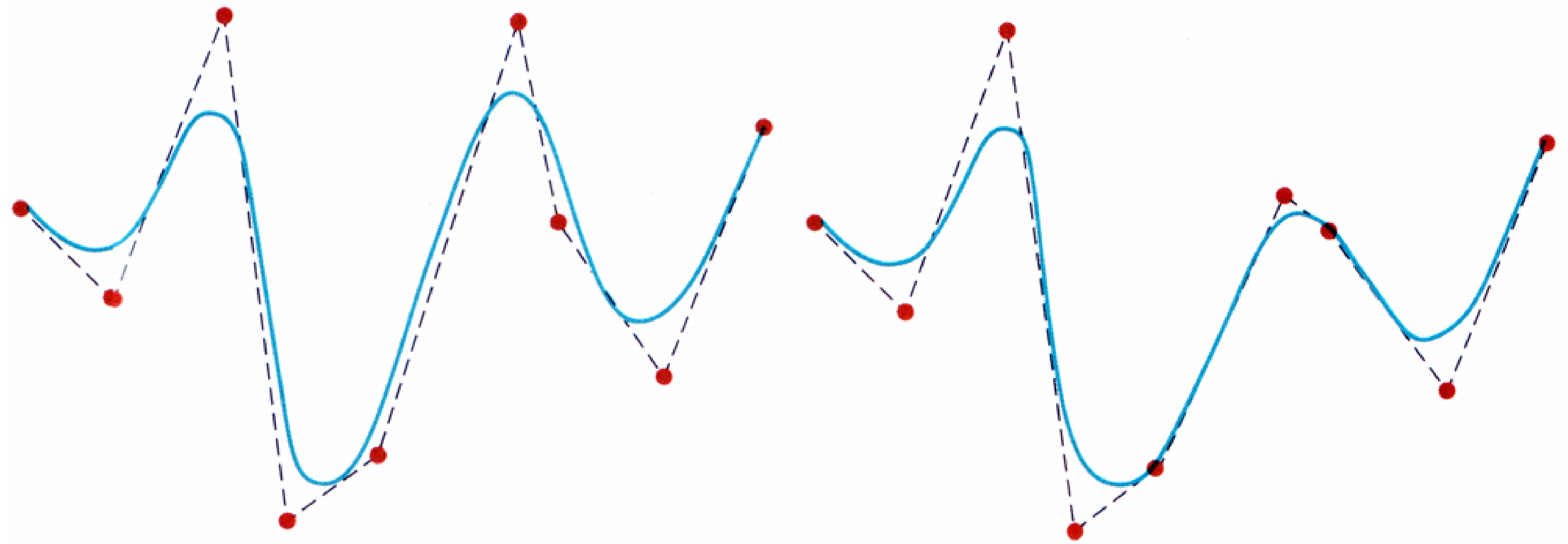
- Basis Function นี้นิยามบน ช่วงย่อยๆ d ช่วง จากค่า u ที่เป็นไปได้ทั้งหมด
- เรียกจุดปลายของแต่ละช่วงว่า knot vector โดยมีเงื่อนไขว่า $u_j \le u_{j+1}$
- เศษส่วนตัวคูณที่เป็น 0/0 กำหนดให้มีค่าเท่ากับ 0

ตัวอย่าง d = 3



B-Spline Local Control

สำหรับ B-Spline ผู้ใช้สามารถเปลี่ยนตำแหน่งของ knot vector เพื่อปรับเส้นโค้งได้เฉพาะบริเวณ โดยที่กระทบบริเวณอื่นน้อยที่สุด



นอกจากนี้ผู้ใช้ยังสามารถ
เพิ่ม knot vector ไปในส่วนใดของเส้นโค้ง หรือ ลบ knot vector หนึ่งๆ ออกจากเส้นโค้ง ภายหลังได้ โดยที่ค่า d เหมือนเดิม

B-Spline Properties

เส้นโค้งพหุนาม ที่มีอันดับเท่ากับ d – 1 จะมีความต่อเนื่อง C^{d-2} (สามารถหาอนุพันธ์ได้ถึงอันดับที่ d – 2)

เส้นโค้งที่มี control point เท่ากับ n + 1 จุดจะนิยามด้วย Blending Function จำนวน n + 1 ฟังก์ชัน

สำหรับ Blending Function $B_{k,d}$ ใดๆ จะนิยามอยู่บนช่วยย่อย d ช่วงจากค่า u ที่เป็นไปได้ทั้งหมด โดยเริ่มจาก control point ที่ k (u_k)

Parameter u ทั้งหมดจะแบ่งออกเป็นช่วงย่อย n + d ช่วง ด้วย knot vector จำนวน n + d + 1 จุด ถ้ามี knot vector $\{u_0, u_1, \ldots u_{n+d}\}$ จะวาดเส้นโค้ง B-Spline ได้เฉพาะช่วงระหว่างจุด u_{d-1} ถึงจุด u_{n+1} เท่านั้น

เส้นโค้งแต่ละช่วงจะเปลี่ยนก็ต่อเมื่อ control point ใดในจำนวน d จุดรอบๆ เปลี่ยน และ เมื่อปรับ control point 1 จุดจะทำให้เส้นโค้งอย่างมาก d ช่วงเปลี่ยน

ผลรวมของ Basis Function มีค่าเท่ากับ 1 (อยู่ภายใน Convex Hull)

Uniform, Periodic B-Spline

คือเส้นโค้ง B-Spline ที่มีระห่างระหว่าง knot vector คงที่ (uniform) ซึ่งสามารถกำหนดได้หลายแบบ ซึ่งแต่ละแบบเหมาะกับการคำนวณที่แตกต่างกันไป

แบบทั่วไป คือค่า u_{min} และ u_{max} เป็นเท่าใดก็ได้ เช่น {-1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, 2.0}

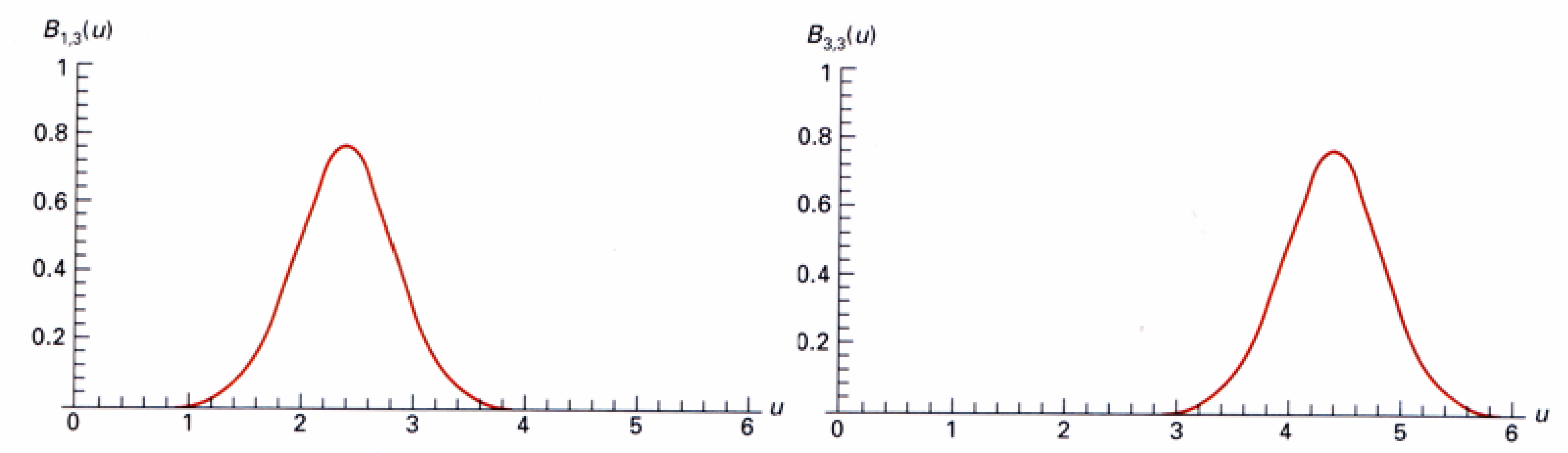
แบบ normalized คือปรับให้ค่า u_{min} และ u_{max} อยู่ระหว่าง 0 และ 1 ซึ่งจากตัวอย่างด้านบนแปลงได้ดังนี้ {0.0, 0.2, 0.4, 0.6, 0.8, 1.0}

แบบจำนวนเต็ม คือเริ่มจาก u_{min} เท่ากับ 0 และระยะห่างระหว่าง knot vector มีค่าเป็นจำนวนเต็มเท่ากับ 1 เช่น {0, 1, 2, 3, 4, 5}

Periodic หมายความว่า **รูปร่าง** ของ Blending Function เหมือนกันทุกๆ จุดสำหรับค่า n และ d เดียวกัน Blending Function ที่ติดกันเพียงแต่เลื่อนตำแหน่งไปเท่านั้น

Periodic Blending Functions

รูปด้านล่างแสดง Periodic Blending Function สำหรับกรณีที่ n เท่ากับ 3 และ d = 3 โดยเปรียบเทียบเมื่อ k เท่ากับ 1 (ซ้ายมือ) และ k = 3 (ขวามือ)



$$B_{k,d}(u) = B_{k+1,d}(u + \Delta u) = B_{k+2,d}(u + 2\Delta u)$$

โดยที่ Δu คือระยะห่างระหว่าง ค่า knot ที่ติดกัน (เนื่องจากเป็น uniform ดังนั้นระยะนี้จะเท่ากัน) ดูตัวอย่างจาก slide projector

Conclusions

- 3-D Object Representations
  - Polygonal Surfaces
  - Planes in 3D Space
  - Quadric Surfaces and Blobs
- Spline Representations
  - Interpolation and Approximation Spline
  - Continuity Conditions
  - Cubic Spline Interpolation
  - Bezier Curves and Surfaces
  - B-Spline Curves and Surfaces
  - Other Splines and Their Conversions
- Introduction to OpenGL
  - OpenGL Programming using C/C++