전산 선박 설계

Part 2. 곡선, 곡면

2006.9.

서울대학교 조선해양공학과

이규열

Advanced
Ship
Design
Automation
Laboratory

목 차

Ch1. 학습목표

1.1 주어진 점을 지나는 곡선 만들기

1.2 주어진 점을 지나는 곡면 만들기

1.1 주어진 점을 지나는 곡선 만들기

1.1.1 함수/곡선의 다항식 표현

1.1.2 3차 Bezier 곡선 보간

1.1.3 3차 B-spline 곡선 보간

1.1.4 선형 표현을 위한 주요 곡선 – Section line

1.1.5 선형 표현을 위한 주요 곡선 – Water line 생성

Advanced
Ship
Design
Automation
Laboratory

1.1.1 함수/곡선의 다항식 표현

- Given : $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$
- Find : cubic polynomial function $\mathbf{r}(t)$

The monomial form :

$$\mathbf{r}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2 + \mathbf{a}_3 t^3 = \begin{bmatrix} \mathbf{a_0} & \mathbf{a_1} & \mathbf{a_2} & \mathbf{a_3} \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}$$

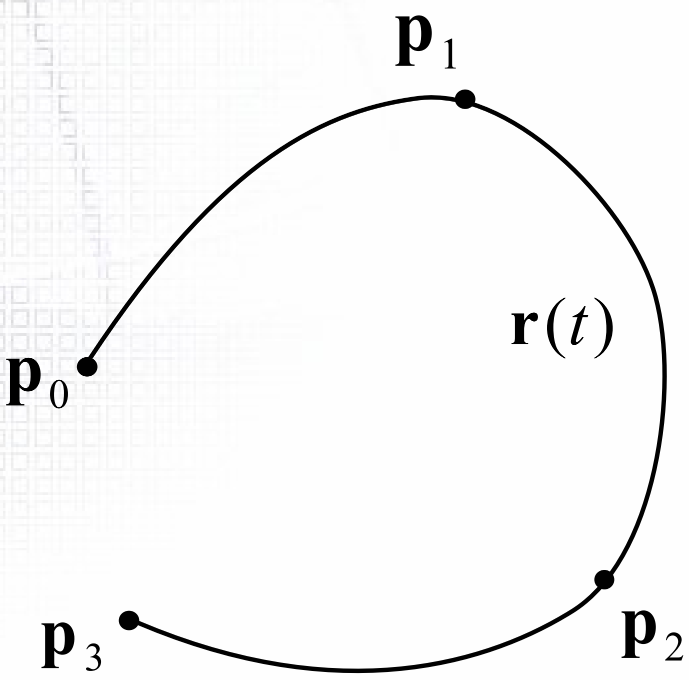
The Bezier form :

$$\begin{aligned} \mathbf{r}(t) &= (1-t)^3 \mathbf{b}_0 + 3(1-t)^2 t \mathbf{b}_1 + 3(1-t)t^2 \mathbf{b}_2 + t^3 \mathbf{b}_3 \\ &= B_0^3(t)\,\mathbf{b}_0 + B_1^3(t)\,\mathbf{b}_1 + B_2^3(t)\,\mathbf{b}_2 + B_3^3(t)\,\mathbf{b}_3 = \begin{bmatrix} \mathbf{b_0} & \mathbf{b_1} & \mathbf{b_2} & \mathbf{b_3} \end{bmatrix} \begin{bmatrix} B_0^3\mathbf{(t)} \\ B_1^3\mathbf{(t)} \\ B_2^3\mathbf{(t)} \\ B_3^3\mathbf{(t)} \end{bmatrix} \end{aligned}$$

$$\begin{bmatrix} \mathbf{b_0} & \mathbf{b_1} & \mathbf{b_2} & \mathbf{b_3} \end{bmatrix} = \begin{bmatrix} \mathbf{a_0} & \mathbf{a_1} & \mathbf{a_2} & \mathbf{a_3} \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{\mathbf{-1}}$$

1.1.2 3차 Bezier 곡선 보간방법 (1)

- If we are given fitting points $\mathbf{P}_i$ and we wish to pass a curve through them. There, the points are 2D, but the curve might as well be 3D. This is called “curve interpolation”.



- We may choose among many kinds of curves; for right now, we will use a cubic Bezier curve.

→ “cubic Bezier curve interpolation”

$$\mathbf{r}(t) = B_0^3(t)\,\mathbf{b}_0 + B_1^3(t)\,\mathbf{b}_1 + B_2^3(t)\,\mathbf{b}_2 + B_3^3(t)\,\mathbf{b}_3$$

- Given parameter t_i corresponding to $\mathbf{P}_i$, we want a cubic Bezier curve such that:

$$\mathbf{r}(t_i) = \mathbf{p}_i; \quad i = 0,1,2,3.$$

1.1.2 3차 Bezier 곡선 보간방법 (2)

- The cubic Bezier curve of the form:

$$\mathbf{r}(t) = B_0^3(t)\mathbf{b}_0 + B_1^3(t)\mathbf{b}_1 + B_2^3(t)\mathbf{b}_2 + B_3^3(t)\mathbf{b}_3.$$

- All interpolation conditions are:

$$\begin{aligned}
\mathbf{p}_0 &= B_0^3(t_0)\enclose{circle}{\mathbf{b}_0} + B_1^3(t_0)\enclose{circle}{\mathbf{b}_1} + B_2^3(t_0)\enclose{circle}{\mathbf{b}_2} + B_3^3(t_0)\enclose{circle}{\mathbf{b}_3},\\
\mathbf{p}_1 &= B_0^3(t_1)\enclose{circle}{\mathbf{b}_0} + B_1^3(t_1)\enclose{circle}{\mathbf{b}_1} + B_2^3(t_1)\enclose{circle}{\mathbf{b}_2} + B_3^3(t_1)\enclose{circle}{\mathbf{b}_3},\\
\mathbf{p}_2 &= B_0^3(t_2)\enclose{circle}{\mathbf{b}_0} + B_1^3(t_2)\enclose{circle}{\mathbf{b}_1} + B_2^3(t_2)\enclose{circle}{\mathbf{b}_2} + B_3^3(t_2)\enclose{circle}{\mathbf{b}_3},\\
\mathbf{p}_3 &= B_0^3(t_3)\enclose{circle}{\mathbf{b}_0} + B_1^3(t_3)\enclose{circle}{\mathbf{b}_1} + B_2^3(t_3)\enclose{circle}{\mathbf{b}_2} + B_3^3(t_3)\enclose{circle}{\mathbf{b}_3}.
\end{aligned}$$

4 Unknown Vectors, 4 Vector Equations

1.1.2 3차 Bezier 곡선 보간방법 (3)

- To find the solution of these four equations for four unknowns, we can write in matrix form:

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = \begin{bmatrix} B_0^3(t_0) & B_1^3(t_0) & B_2^3(t_0) & B_3^3(t_0) \\ B_0^3(t_1) & B_1^3(t_1) & B_2^3(t_1) & B_3^3(t_1) \\ B_0^3(t_2) & B_1^3(t_2) & B_2^3(t_2) & B_3^3(t_2) \\ B_0^3(t_3) & B_1^3(t_3) & B_2^3(t_3) & B_3^3(t_3) \end{bmatrix} \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}.$$

- To abbreviate the above form as: $\mathbf{P} = \mathbf{MB}.$
- The solution is: $\mathbf{B} = \mathbf{M}^{-1}\mathbf{P}.$
- Although it is looks like the solution to one linear system but it is the two or three systems depending on the dimensionality of the $\mathbf{p}_i$.

$$ex)\ \mathbf{p}_0 = [x_0 \quad y_0]^T \quad or \quad [x_0 \quad y_0 \quad z_0]^T$$

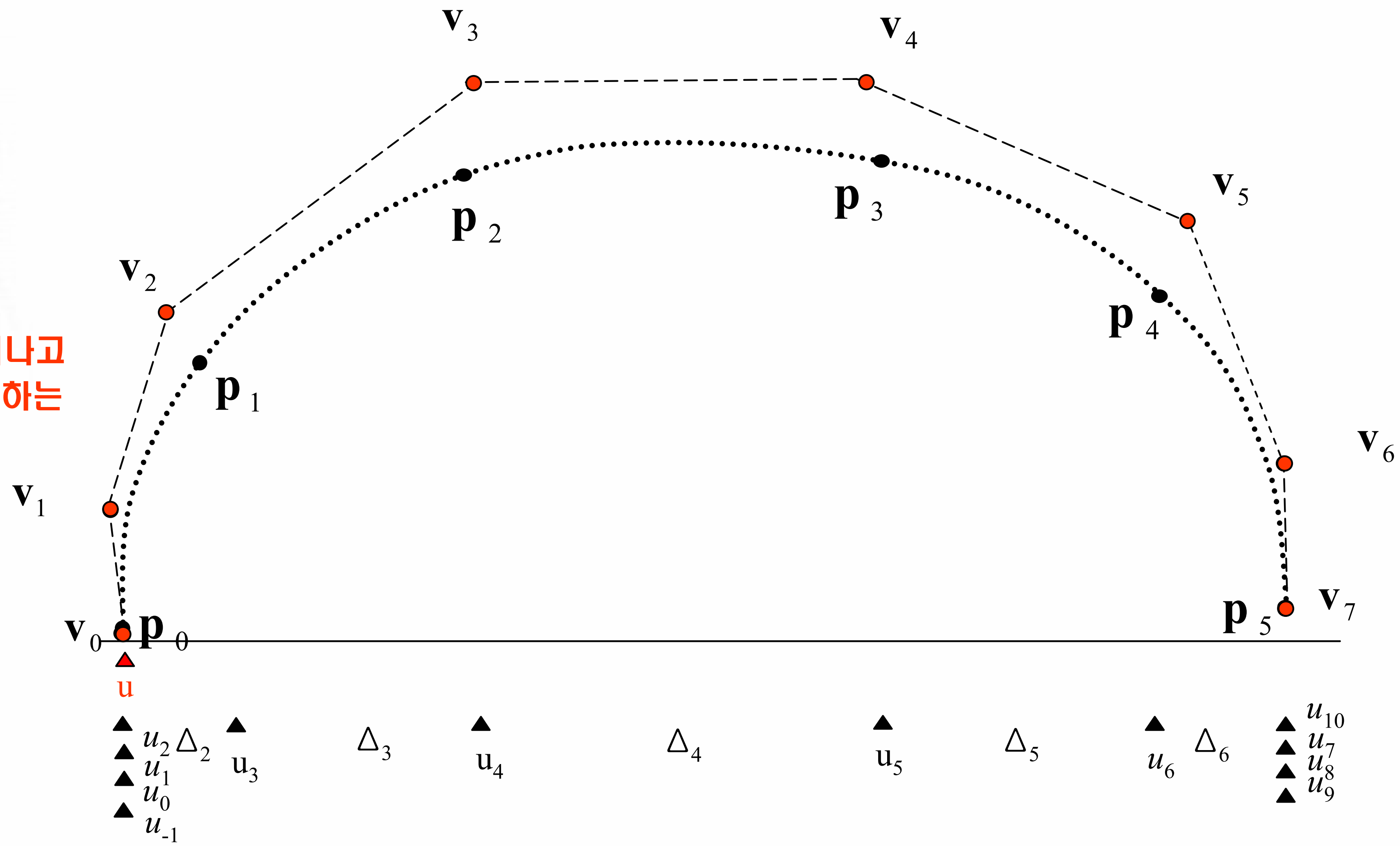
1.1.3 3차 B-spliine 곡선 보간방법

Given:

곡선 상의 점 p_i
곡선의 놋트 u_i
양끝단의 접선 벡터

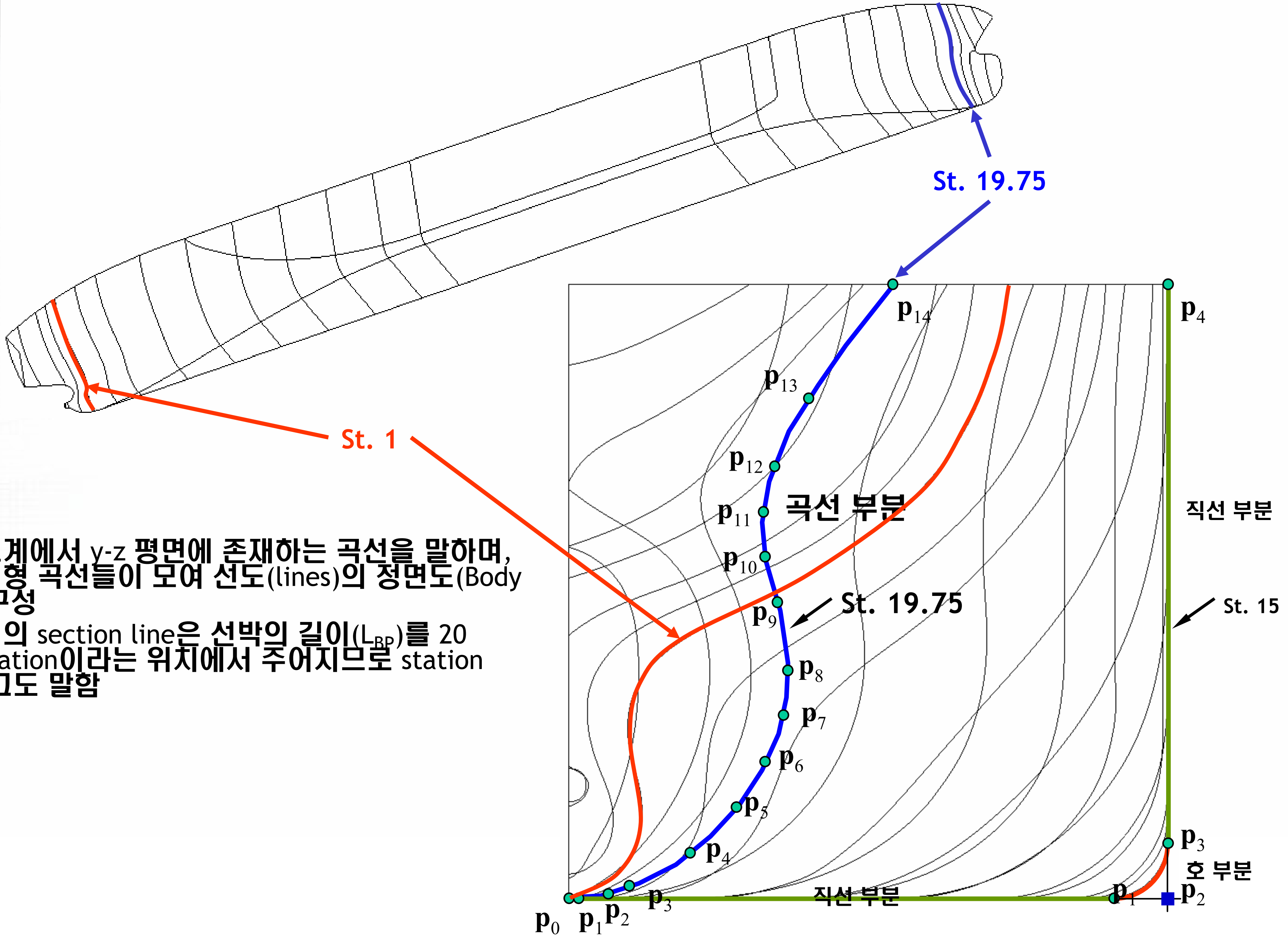
Find:

곡선 상의 점 p_i을 지나고
C^2 연속 조건을 만족하는
3차 B-Spline 곡선
(B-Spline 조정점)



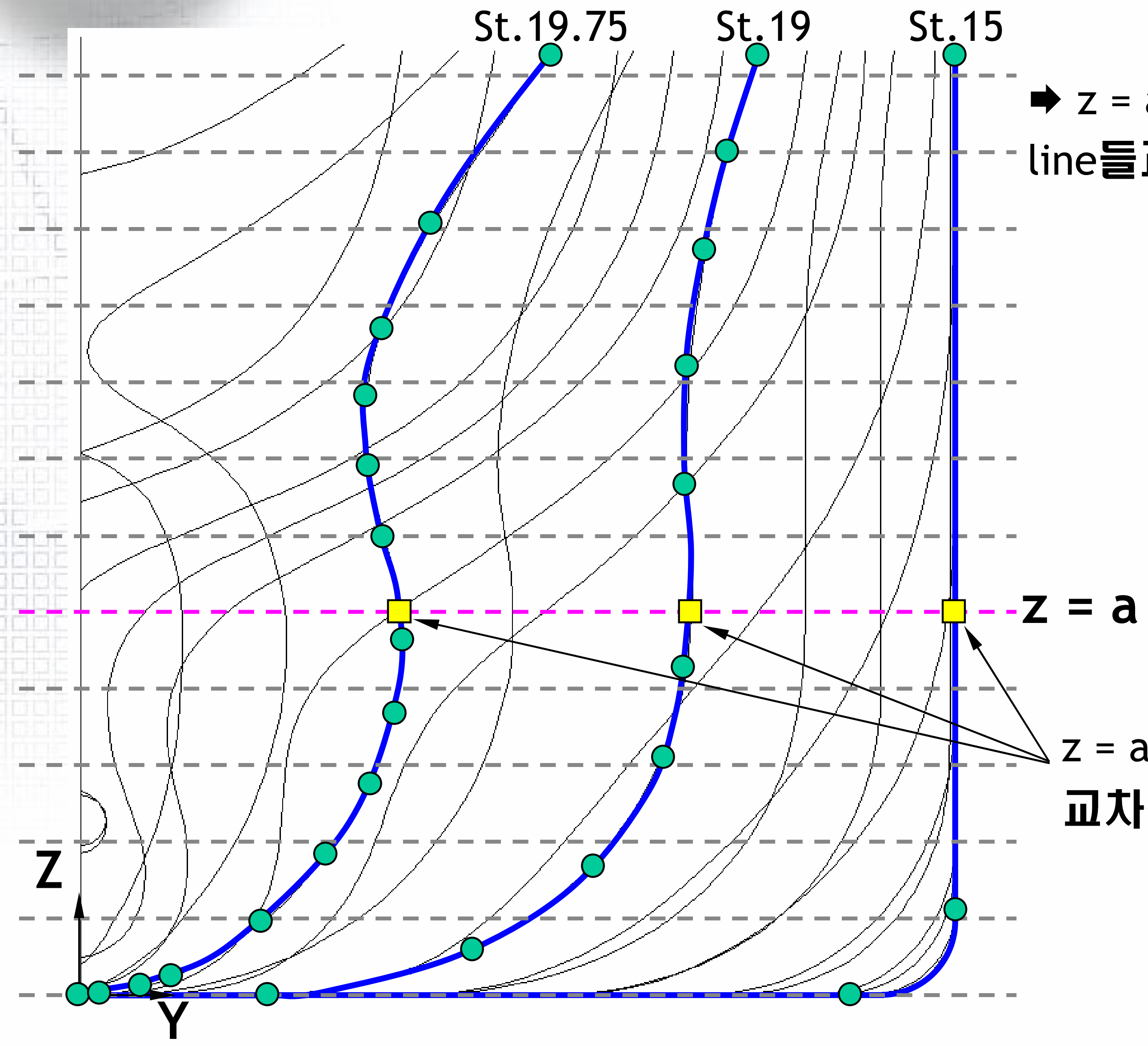
가정 : 각 곡선 세그먼트는 3차 Bezier Curve 이다.
연결점에서는 C1, C2 연속조건을 만족한다.

1.1.4 선형 표현을 위한 주요 곡선들 - Section Line



- 선형 좌표계에서 y-z 평면에 존재하는 곡선을 말하며, 이러한 선형 곡선들이 모여 선도(lines)의 정면도(Body Plan)를 구성
- 보통 선형의 section line은 선박의 길이(L_{BP})를 20 등분한 station이라는 위치에서 주어지므로 station line이라고도 말함

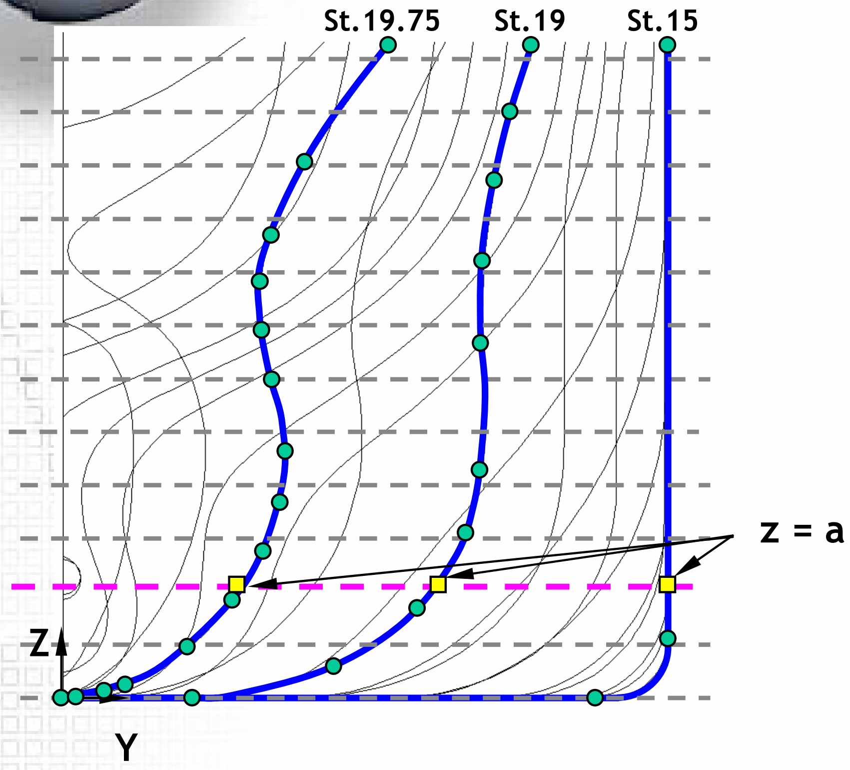
1.1.5 선형 표현을 위한 주요 곡선들 - Water Line 생성 (1)

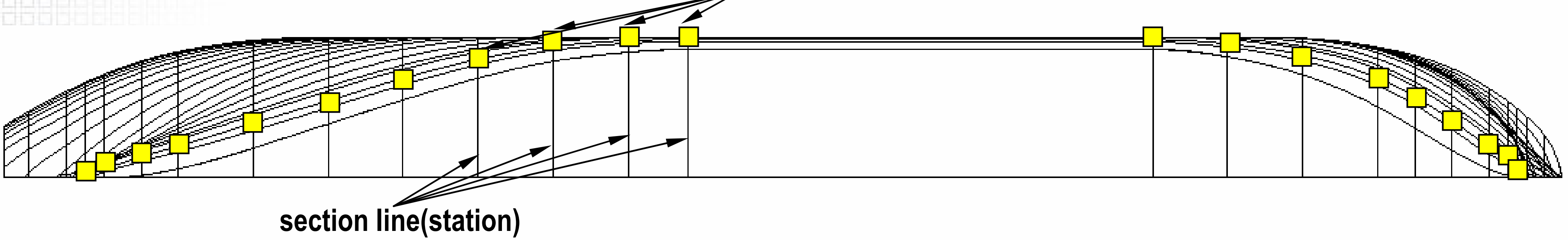


➡ z = a 평면과 주요 곡선 및 모든 section line들과의 교차 계산을 통해 waterline 생성

z = a에서의 waterline 생성을 위한 교차점

1.1.5 선형 표현을 위한 주요 곡선들 - Water Line 생성 (2)



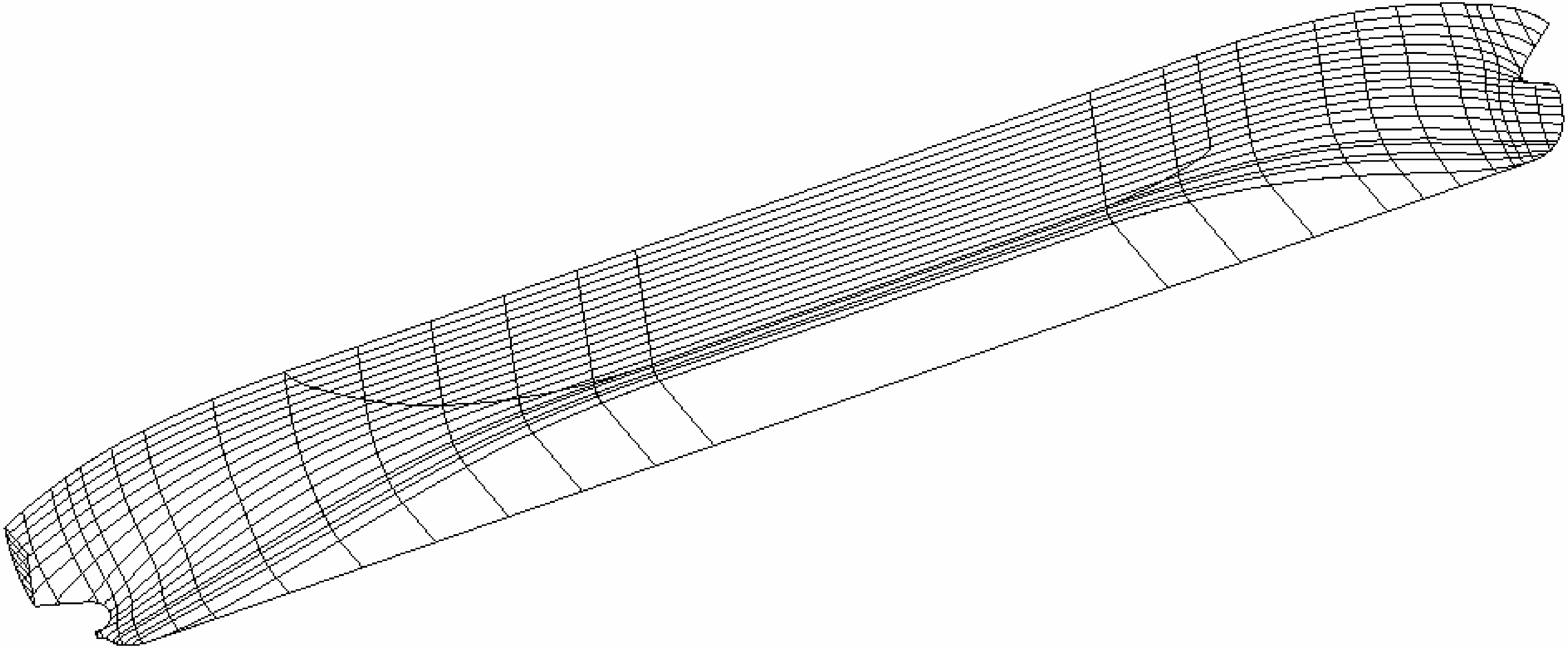


z = a **위치에서의 모든 교차점들을** NURB **곡선을 이용하여 보간**(fitting)
➡ z = a**에서의** waterline **생성**

➡ Waterline **생성**

원하는 z 위치에 대해 위 과정을 반복 수행

1.1.5 선형 표현을 위한 주요 곡선들 - Water Line 생성 (3)



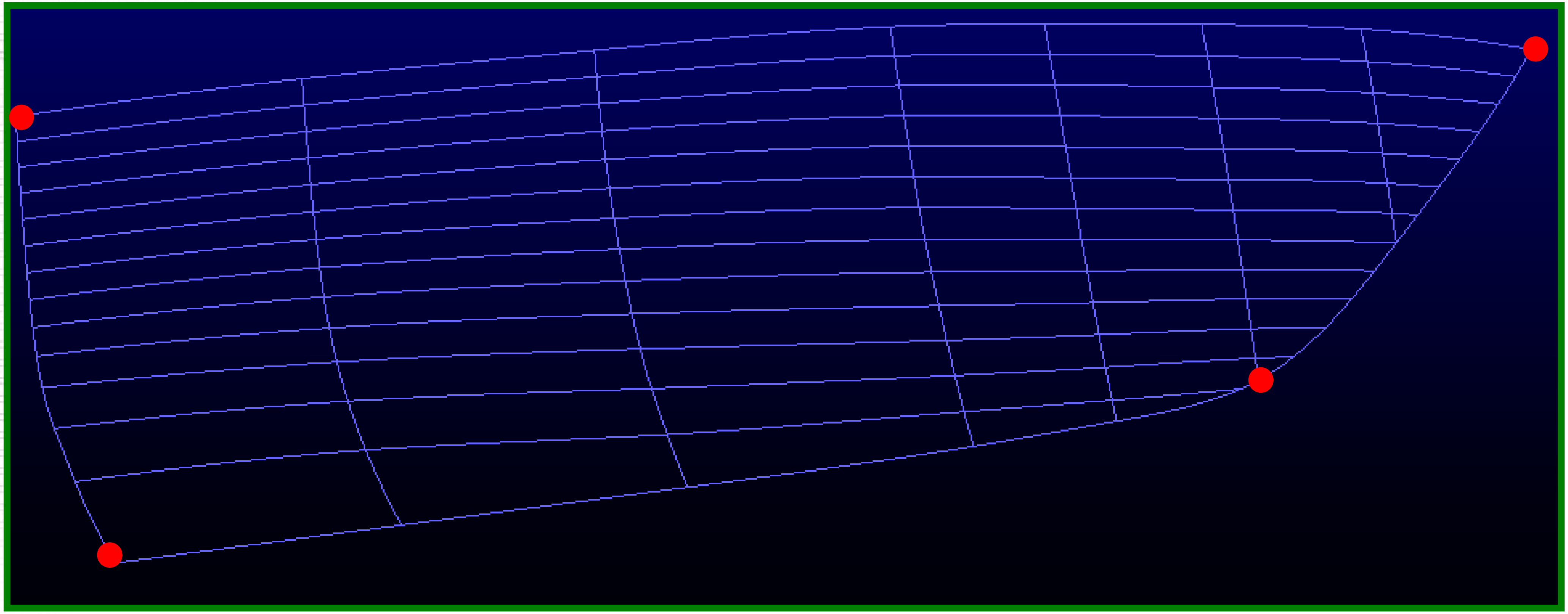
1.2 주어진 점을 지나는 곡면 만들기

1.2.1 요트 형상의 선박형상곡면 생성

1.2.2 구상선수를 갖는 선박형상곡면 생성

1.2.1 요트 형상의 선박형상 곡면 생성 (1)

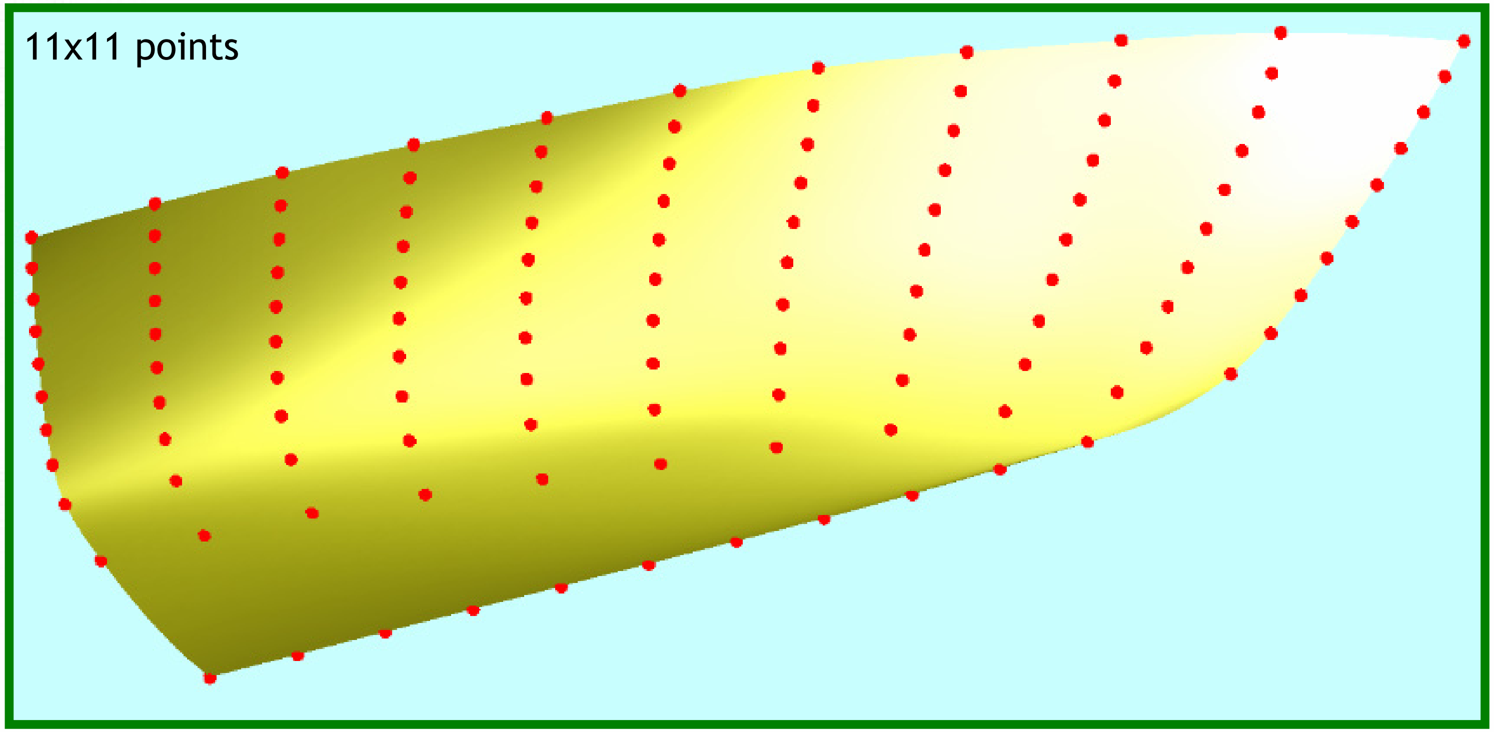
* 출처 : 서울대 조선해양공학과 2005년 2학년 교과목 『 조선해양공학계획』 강좌 중에 학생들이 설계한 선형

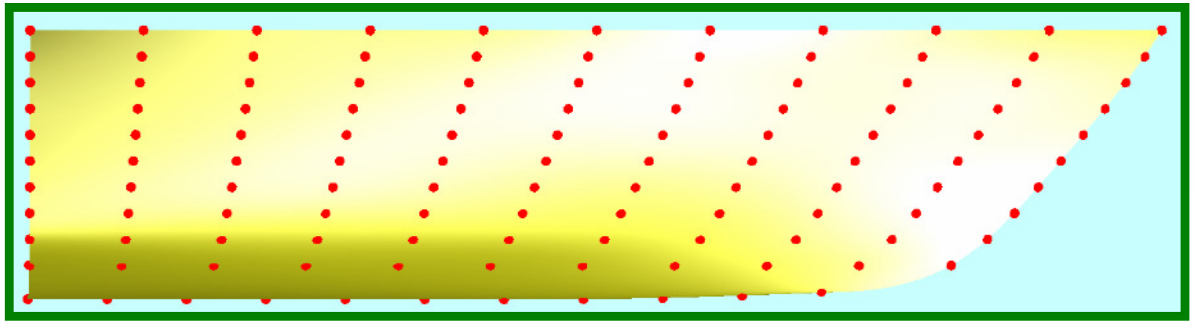


사각형 패치의 꼭지점 결정

1.2.1 요트 형상의 선박형상 곡면 생성 (2)

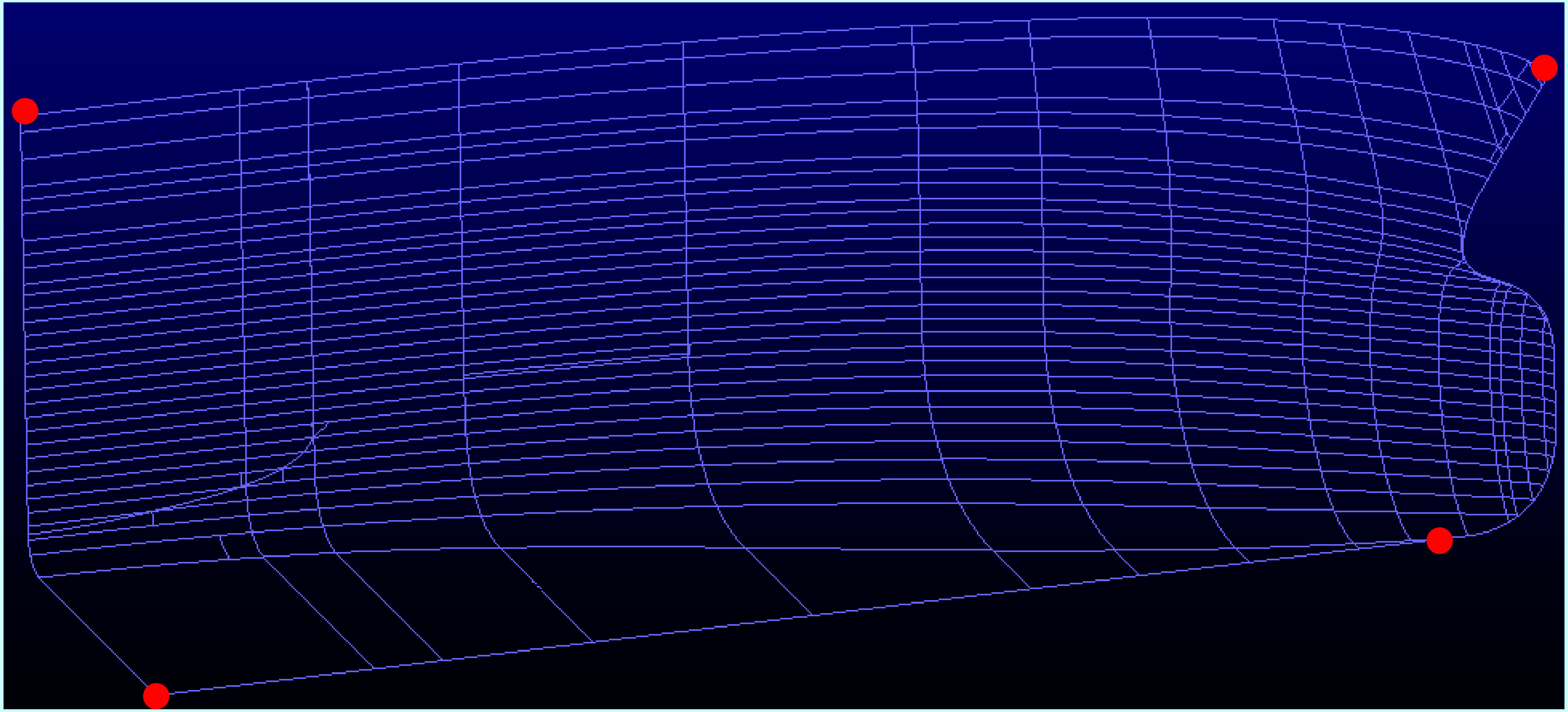
□ 점들의 x좌표 사이의 거리가 거의 일정하도록 점을 추출한 후
→ 이 점 data로부터 선형곡면을 생성한 결과





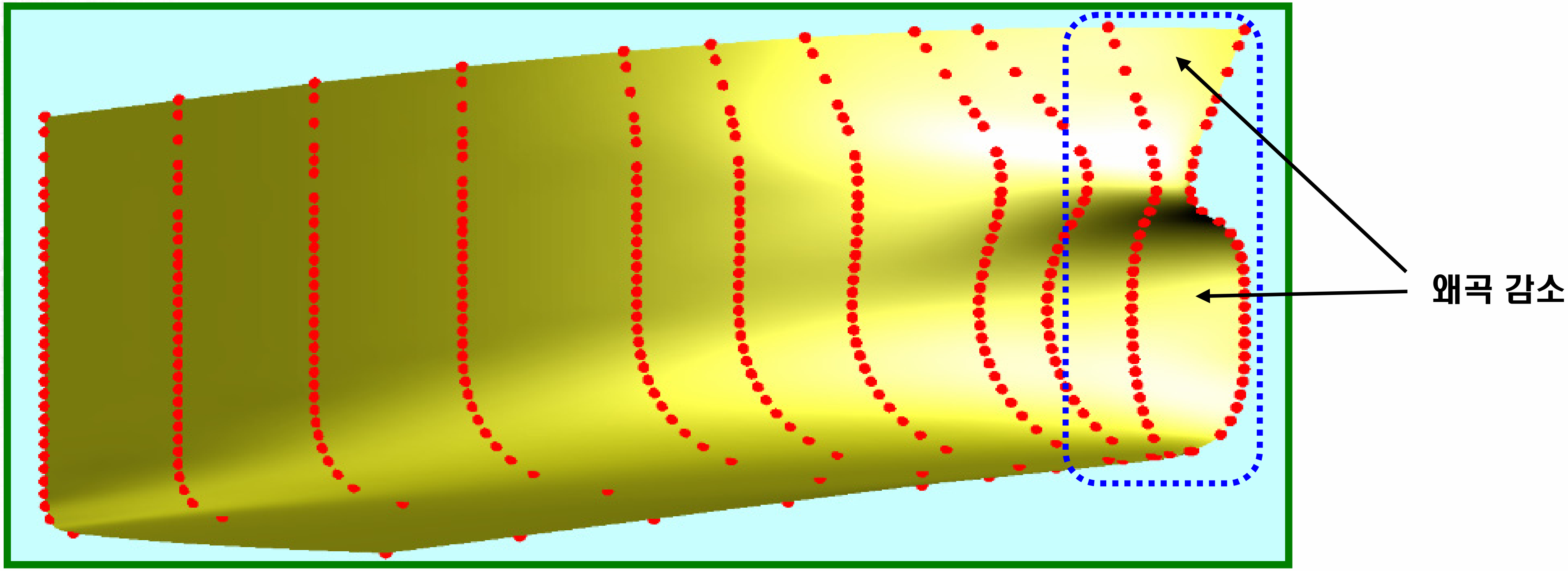
1.2.2 구상선수를 갖는 선박형상곡면 생성 (1)

선수부



1.2.2 구상선수를 갖는 선박형상곡면 생성 (2)

□ 주어진 곡선그물망 이외의 보조선을 생성하여 곡선 보간에 적합한 점 data를 생성한 후, 점 data로부터 선수부 선형곡면을 생성한 결과



Ch 2. 곡선(Curves)

2.1 Parametric function/curves

2.2 Bezier Curves

2.3 B-spline Curves

Advanced
Ship
Design
Automation
Laboratory

2.1 Parametric function/ curves

2.1.1 양함수/음함수/매개변수 함수

2.1.2 매개변수 함수의 특징

2.1.3 일반함수의 매개변수 함수 표현

2.1.4 매개변수 함수의 직관적 표현방법

2.1.1 Explicit / Implicit / Parametric function

- Explicit function(양함수식)
  - 함수가 y=f(x) 의 형태로 나타내어지면 양함수라고 함
  - x가 주어지면 y를 쉽게 구할 수 있음

ex) $y = \sqrt{r^2 - x^2}$

- Implicit function(음함수식)
  - 변수가 x, y 두 개일 때, 함수 f(x, y) = 0 와 같은 형태로 표현되는 식을 음함수라고 함
  - 주어진 점이 곡선의 내부 또는 외부, 왼쪽 또는 오른쪽)에 있는지 판단하기 쉬움
  - f(x, y) = 0인 함수를 y에 대해 풀어서 y = f(x)의 형태가 얻어지면 양함수로 변환이 가능함. 모든 음함수가 양함수로 변환될 수 있는 것은 아님

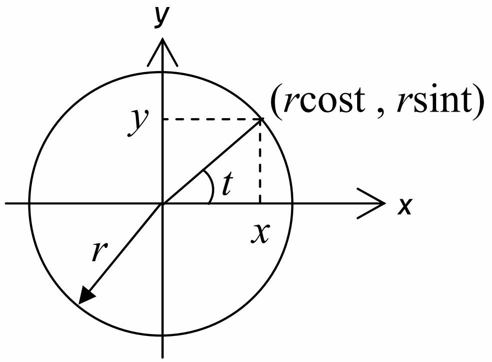
ex) $x^2 + y^2 - r^2 = 0$

ex) $(0)^2 + (0)^2 - r^2 < 0$

$(r)^2 + (r)^2 - r^2 > 0$

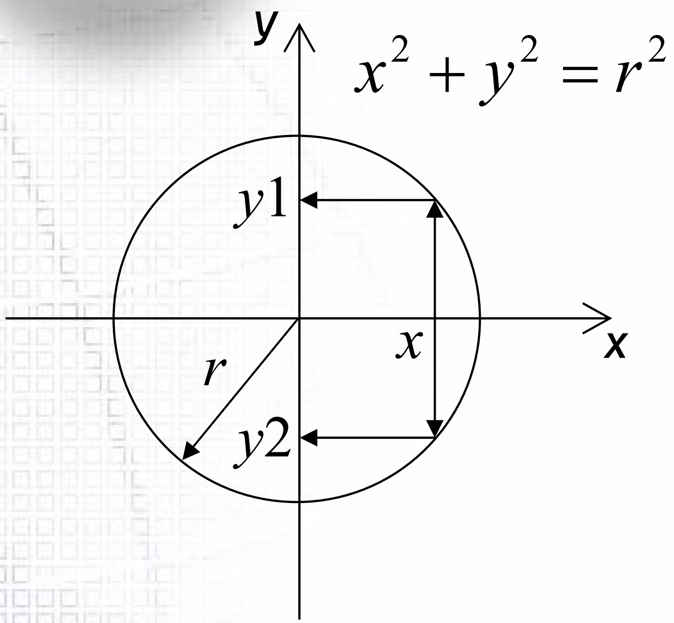
ex) $y = \pm\sqrt{r^2 - x^2}$

- Parametric function(매개 변수 함수식)
  - 2변수 함수가 매개 변수 t를 이용하여, x = f(t), y = g(t)로 나타내어지면 이를 매개 변수 함수라고 함
  - 즉, 매개 변수 t를 매개로 하여 x와 y를 표현하는 형식
  - 모든 양함수식은 매개 변수 함수식으로 변환이 가능함



ex) $x(t) = r\cos t, \ \ y(t) = r\sin t$

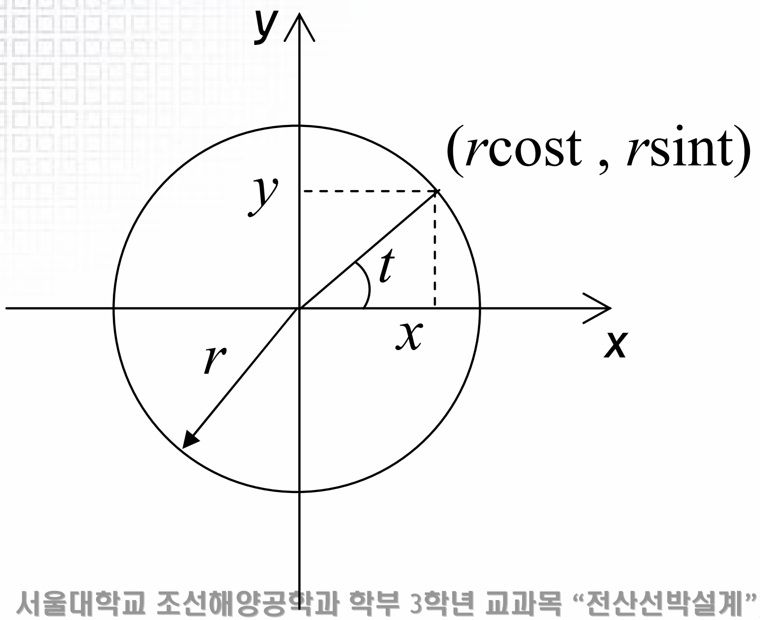
2.1.2 매개변수 함수(Parametric function)의 특징 (1)



- **일반적인 함수**
  - **하나의** x **값에 대해** y **값이 두 개이상 나올 수 있음** (multi-value function)

    $$x^2 + y^2 = r^2 \qquad y = \pm\sqrt{r^2 - x^2}$$

  - **미분 계수를 표현하기 어려운 경우가 있음** $\left.\dfrac{dy}{dx}\right|_{,x=r} = \infty$



- **매개 변수 함수**
  - **하나의 매개 변수 값에 대해 하나의 값만이 나옴**

    $$x(t) = r\cos t, y(t) = r\sin t$$

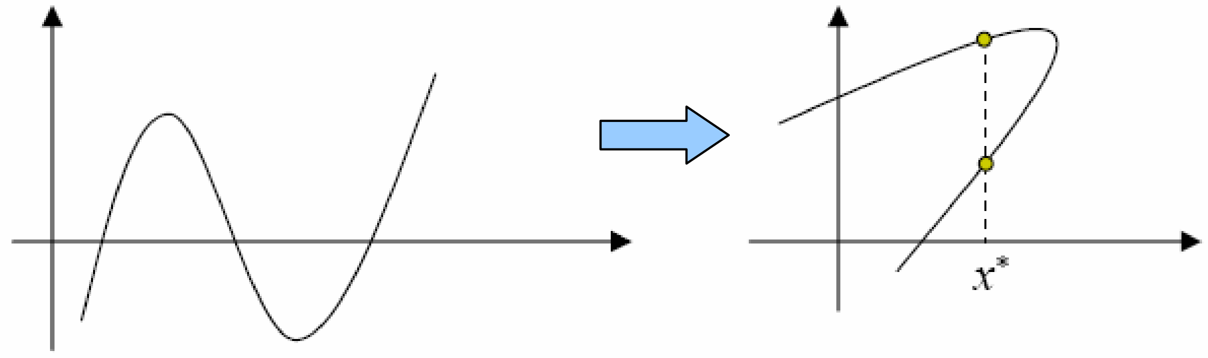
  - **미분 계수를 표현하기 쉬움**
    →dy/dx**를 각 요소별로 나누어서 계산함**: dy/dt, dx/dt

    $$\frac{dx}{dt} = -r\sin t, \quad \frac{dy}{dt} = r\cos t$$

2.1.2 매개변수 함수(Parametric function)의 특징 (2)

- y = f(x)의 양함수식
  - 원래의 양함수 곡선을 회전,이동 등을 통하여 변형한 후, 이를 다시 양함수 곡선으로 표현* 하기 어려움

  *차원확장



$y = a_0 + a_1 x^1 + a_2 x^2 + a_3 x^3$

$y = ??$

- f(x, y) = 0의 음함수식
  - 곡선 상의 점을 순차적으로 계산할 수 없다
  - 차원 확장이 어렵다.

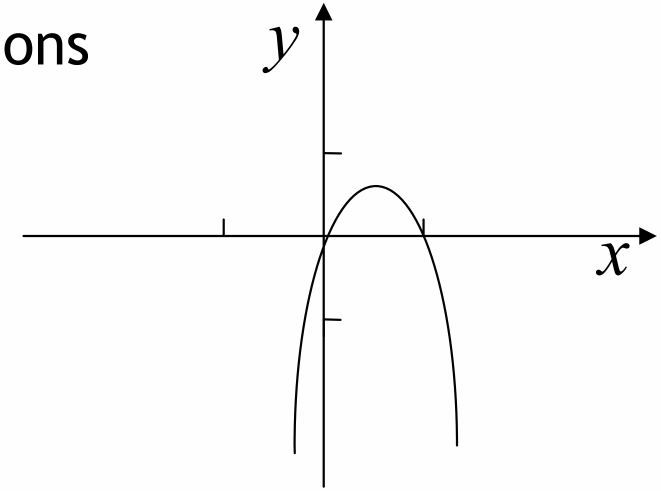
- x = f(t), y = g(t)의 매개 변수식
  - 매개 변수를 통해 곡선 상의 점을 순차적으로 계산할 수 있다.
  - 차원을 쉽게 확장할 수 있다.

    ➡ CAD 시스템에서 매개 변수식을 많이 사용하는 이유

2.1.2 매개변수 함수(Parametric function)의 특징 (3)

- The curve is defined by parametric functions

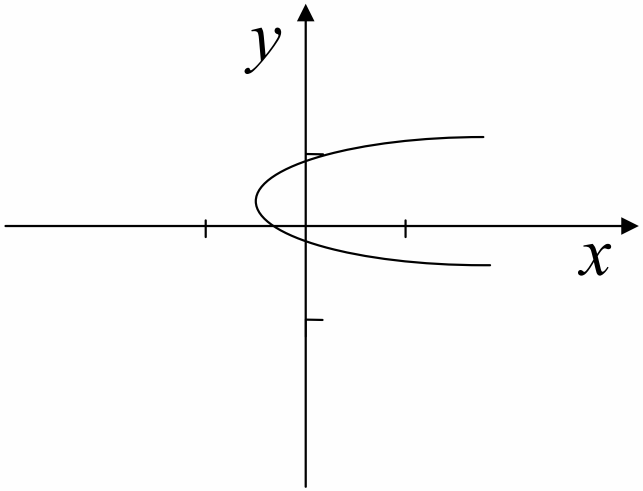
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t \\ 2t - 2t^2 \end{bmatrix}$$



- If the curve is rotated by 90°,

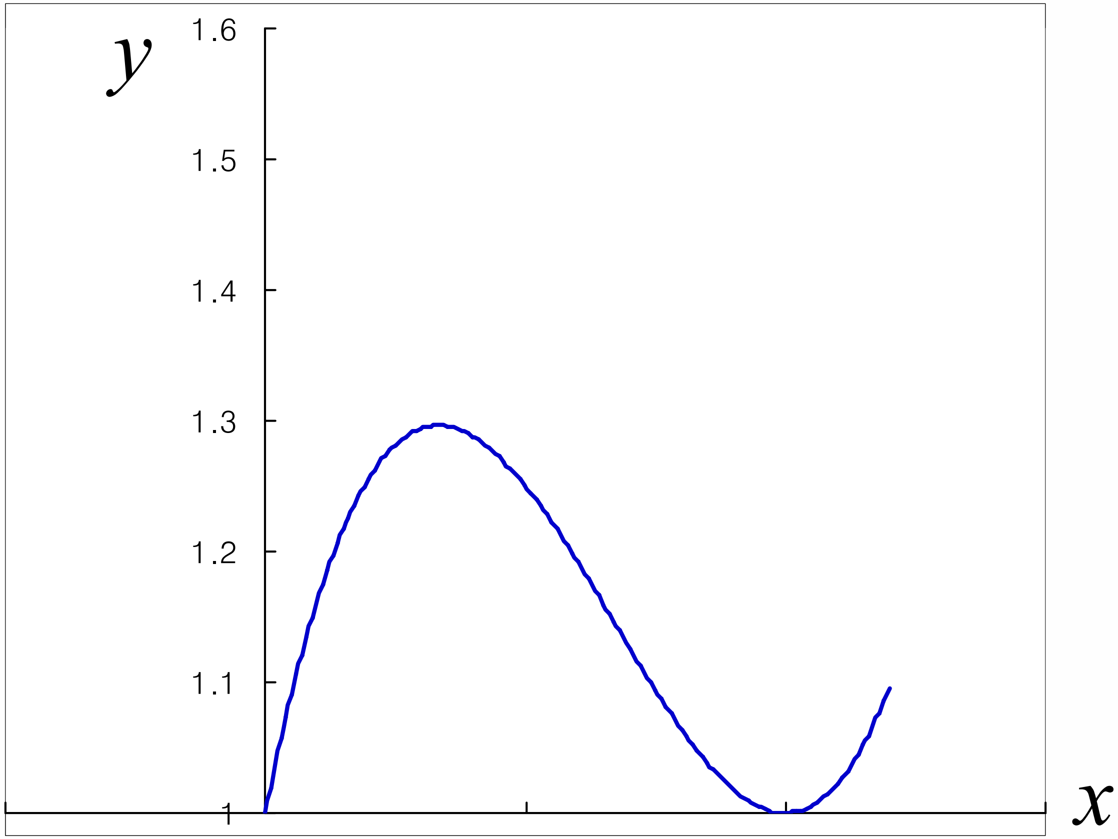
형상은 변하지 않고, 위치만 변함

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -2t + 2t^2 \\ t \end{bmatrix}$$



2.1.3 일반 함수의 매개변수 함수 표현 (1)

Given: $y = 2x^3 - 4x^2 + 2x + 1$



2.1.3 일반 함수의 매개변수 함수 표현 (2)

$$y = 2x^3 - 4x^2 + 2x + 1$$

$$\mathbf{r}(t) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} t \\ 2t^3 - 4t^2 + 2t + 1 \end{bmatrix}$$

- 이러한 함수식과 계수 2, -4, 2, 1 으로는 그래프의 모양을 "직관적"으로 예상하기 어려움

$$\Downarrow$$

- 함수식을 아래와 같은 형태로 표현할 수 있다면

$$\mathbf{r}(t) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} (1-t)^3 x_0 + 3t(1-t)^2 x_1 + 3t^2(1-t)x_2 + t^3 x_3 \\ (1-t)^3 y_0 + 3t(1-t)^2 y_1 + 3t^2(1-t)y_2 + t^3 y_3 \end{bmatrix}$$

$$\begin{bmatrix} t \\ 2t^3 - 4t^2 + 2t + 1 \end{bmatrix} = \begin{bmatrix} (1-t)^3 \enclose{circle}{x_0} + 3t(1-t)^2 \enclose{circle}{x_1} + 3t^2(1-t)\enclose{circle}{x_2} + t^3 \enclose{circle}{x_3} \\ (1-t)^3 \enclose{circle}{y_0} + 3t(1-t)^2 \enclose{circle}{y_1} + 3t^2(1-t)\enclose{circle}{y_2} + t^3 \enclose{circle}{y_3} \end{bmatrix}$$

▶

▶

2.1.3 일반 함수의 매개변수 함수 표현 (3)

$$(1-t)^3 x_0 + 3t(1-t)^2 x_1 + 3t^2(1-t)x_2 + t^3 x_3 = t$$

상수의 계수: $x_0 = 0$

u의 계수: $-3x_0 + 3x_1 = 1$

u^2의 계수: $3x_0 - 6x_1 + 3x_2 = 0$

u^3의 계수: $-x_0 + 3x_1 - 3x_2 + x_3 = 0$

➡

$x_0 = 0$

$x_1 = 1/3$

$x_2 = 2/3$

$x_3 = 1$

$$b_{x_i}^0 = x_i = \frac{i}{n}$$

Linear
Precision

2.1.3 일반 함수의 매개변수 함수 표현 (4)

$$(1-t)^3 y_0 + 3t(1-t)^2 y_1 + 3t^2(1-t) y_2 + t^3 y_3 = 2t^3 - 4t^2 + 2t + 1$$

상수의 계수: $y_0 = 1$

t의 계수: $-3y_0 + 3y_1 = 2$

t2의 계수: $3y_0 - 6y_1 + 3y_2 = -4$

t3의 계수: $-y_0 + 3y_1 - 3y_2 + y_3 = 2$

➡

$y_0 = 1$

$y_1 = 5/3$

$y_2 = 1$

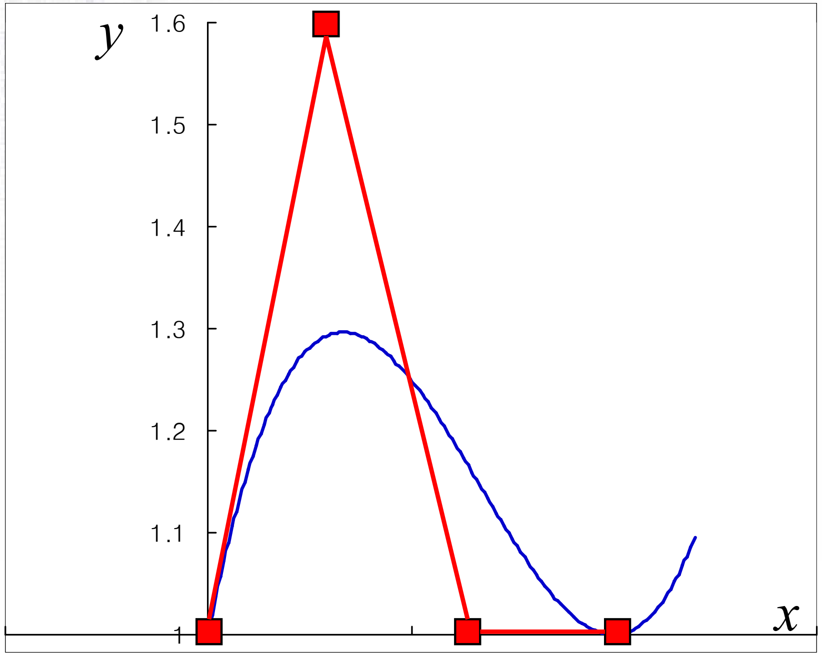
$y_3 = 1$

2.1.3 일반 함수의 매개변수 함수 표현 (5)

$$\mathbf{r}(t)=\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}x(t)\\y(t)\end{bmatrix}=\begin{bmatrix}t\\2t^3-4t^2+2t+1\end{bmatrix}=\begin{bmatrix}(1-t)^3\cdot 0+3t(1-t)^2\cdot\dfrac{1}{3}+3t^2(1-t)\cdot\dfrac{2}{3}+t^3\cdot 1\\(1-t)^3\cdot 1+3t(1-t)^2\cdot\dfrac{5}{3}+3t^2(1-t)\cdot 1+t^3\cdot 1\end{bmatrix}$$

$$=(1-t)^3\begin{bmatrix}0\\1\end{bmatrix}+3t(1-t)^2\begin{bmatrix}1/3\\5/3\end{bmatrix}+3t^2(1-t)\begin{bmatrix}2/3\\1\end{bmatrix}+t^3\begin{bmatrix}1\\1\end{bmatrix}$$

$$=B_0^3(t)\begin{bmatrix}0\\1\end{bmatrix}+B_1^3(t)\begin{bmatrix}1/3\\5/3\end{bmatrix}+B_2^3(t)\begin{bmatrix}2/3\\1\end{bmatrix}+B_3^3(t)\begin{bmatrix}1\\1\end{bmatrix}$$



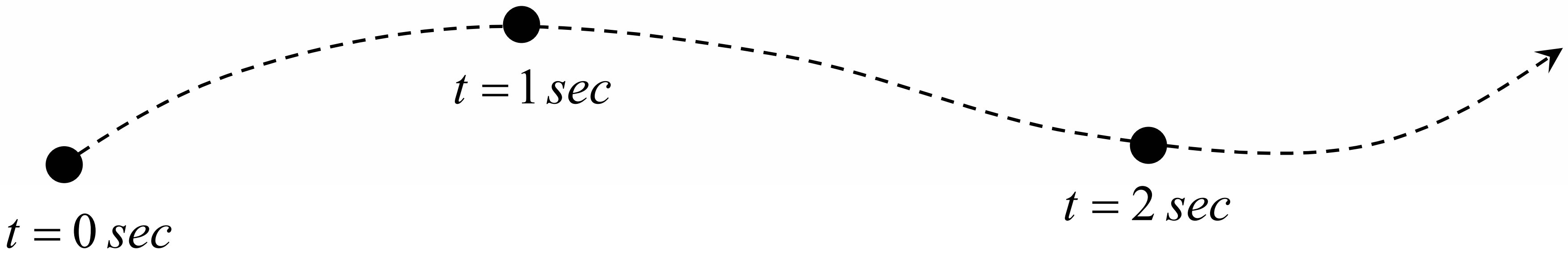
- 새로운 함수들에 곱해지는 계수들을 점으로 가시화 하면, 처음 점과 마지막 점은 곡선을 지나고,
- 점들을 직선으로 연결하면, 곡선의 모양과 비슷한 형태임을 알 수 있다.

2.1.3 일반 함수의 매개변수 함수 표현 (6)

$$\mathbf{r}(t)=\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}x(t)\\ y(t)\end{bmatrix}=\begin{bmatrix}t\\ 2t^3-4t^2+2t+1\end{bmatrix}=\begin{bmatrix}(1-t)^3\cdot 0+3t(1-t)^2\cdot\frac{1}{3}+3t^2(1-t)\cdot\frac{2}{3}+t^3\cdot 1\\ (1-t)^3\cdot 1+3t(1-t)^2\cdot\frac{5}{3}+3t^2(1-t)\cdot 1+t^3\cdot 1\end{bmatrix}$$

- **매개변수 t를 시각이라고 생각하면, r(t)는 어떠한 물체(rigid body)가 이동하는 궤적(trajectory)을 표현한 것으로 생각할 수 있다.**
- **양함수, 음함수 함수식에서는 이동하는 물체의 궤적의 형상만 표현할 수 있지만, 매개변수 함수식으로는 이동하는 물체의 궤적의 형상뿐만 아니라 특정시각 t 에서의 위치 r(t)의 관계를 함께 살펴볼 수 있다.**

$\mathbf{r}(t)$: 물체의 궤적, $\dot{\mathbf{r}}(t)$: 물체의 속도, $\ddot{\mathbf{r}}(t)$: 물체의 가속도



2.1.4 매개 변수 함수의 '직관적' 표현 방법

- For any polynomial degree, on the cubic case $n = 3$

$$\mathbf{r}(t) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -(1-t)^3 + t^3 \\ 3(1-t)^2 t - 3(1-t)t^2 \end{bmatrix}$$

위와 같이 어떠한 함수도 매개 변수 형태로 표현할 수 있음. 그러나 r(t)가 다항식으로 표현되는 경우는 직관적으로 곡선의 형상을 예상하기 어려움

<문제 제기> 어떻게 하면 함수를 직관적으로 알아 볼 수 있게 표현할 수 있을까?

- The polynomial in terms of a combination of points;

$$\mathbf{r}(t) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -(1-t)^3 + t^3 \\ 3(1-t)^2 t - 3(1-t)t^2 \end{bmatrix}$$

$$= (1-t)^3 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + 3(1-t)^2 t \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 3(1-t)t^2 \begin{bmatrix} 0 \\ -1 \end{bmatrix} + t^3 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$= B_0^3 \begin{bmatrix} -1 \\ 0 \end{bmatrix} + B_1^3 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + B_2^3 \begin{bmatrix} 0 \\ -1 \end{bmatrix} + B_3^3 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

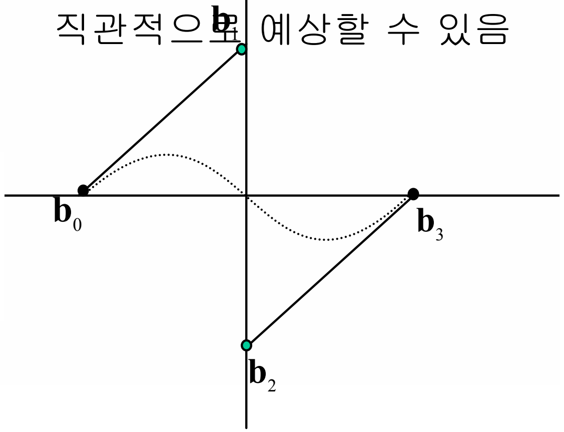
$$= B_0^3 \mathbf{b}_0 + B_1^3 \mathbf{b}_1 + B_2^3 \mathbf{b}_2 + B_3^3 \mathbf{b}_3$$

→ 프랑스 르노자동차의 엔지니어인 P. Bezier가 1971년 발견

- 시작 조정점과 끝 조정점은 곡선상에 위치하고,

조정점들을 연결한 직선의 곡선의 모양과 비슷한 형상이다

→ 조정점으로 곡선의 형태를 직관적으로 예상할 수 있음



공간 상의 점(point)들을 3차 함수들과 "blending"하여 곡선을 표현할 수 있음

또한, 이러한 점을 움직이면 곡선의 형상이 변하므로, 곡선의 조정점(control points)이라고 함

2.2 Bezier curves

2.2.1 Definition & Characteristics of Bezier curves

2.2.1.1 Definition of cubic “Bezier” curves

- The cubic Bezier curve is defined by

$$\mathbf{r}(t)=\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{y}(t)\end{bmatrix}\ or\ \begin{bmatrix}\mathbf{x}(t)\\ \mathbf{y}(t)\\ \mathbf{z}(t)\end{bmatrix}=(1-t)^3\,\mathbf{b}_0+3(1-t)^2t\,\mathbf{b}_1+3(1-t)t^2\,\mathbf{b}_2+t^3\,\mathbf{b}_3$$

$$=\underline{B_0^3(t)}\,\mathbf{b}_0+\underline{B_1^3(t)}\,\mathbf{b}_1+\underline{B_2^3(t)}\,\mathbf{b}_2+\underline{B_3^3(t)}\,\mathbf{b}_3$$

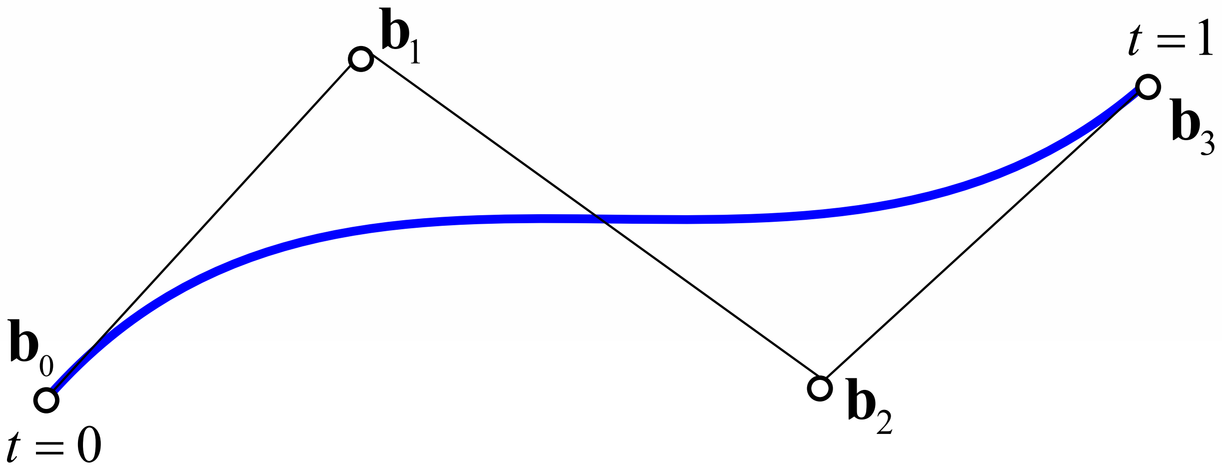
linearly independent
(어떤 하나라도 다른 것으로 표현할 수 없음)

where, $\mathbf{b}_i$: Bezier control points (b_{ix}, b_{iy}) or (b_{ix}, b_{iy} , b_{iz})

$B_i^3(t)$: cubic Bernstein polynomial
or Bernstein basis function

$$\sum_{i=0}^{3}B_i^3(t)=1,\quad B_i^3(t)\geq 0$$

$0\leq t\leq 1$: Bezier curve parameter



2.2.1.2 1st Derivatives of Cubic Bezier Curves (1)

$$\mathbf{r}(t) = (1-t)^3\mathbf{b}_0 + 3(1-t)^2 t\mathbf{b}_1 + 3(1-t)t^2\mathbf{b}_2 + t^3\mathbf{b}_3$$

- First derivatives: Tangent vector of the curve

  : "**시간 t에서의 물체의 속도**"

$$\frac{d\mathbf{r}(t)}{dt} = -3(1-t)^2\mathbf{b}_0 + [3(1-t)^2 - 6(1-t)t]\mathbf{b}_1 + [6(1-t)t - 3t^2]\mathbf{b}_2 + 3t^2\mathbf{b}_3$$

$$= 3[\mathbf{b}_1 - \mathbf{b}_0](1-t)^2 + 6[\mathbf{b}_2 - \mathbf{b}_1](1-t)t + 3[\mathbf{b}_3 - \mathbf{b}_2]t^2$$

$$= 3\Delta\mathbf{b}_0(1-t)^2 + 6\Delta\mathbf{b}_1(1-t)t + 3\Delta\mathbf{b}_2 t^2$$

$$= 3(\Delta\mathbf{b}_0 B_0^2 + \Delta\mathbf{b}_1 B_1^2 + \Delta\mathbf{b}_2 B_2^2)$$

where, $\Delta\mathbf{b}_i = \mathbf{b}_{i+1} - \mathbf{b}_i$: forward differences

2.2.1.2 1st Derivatives of Cubic Bezier Curves(2)

- The derivative of the cubic curve is quadratic curve.

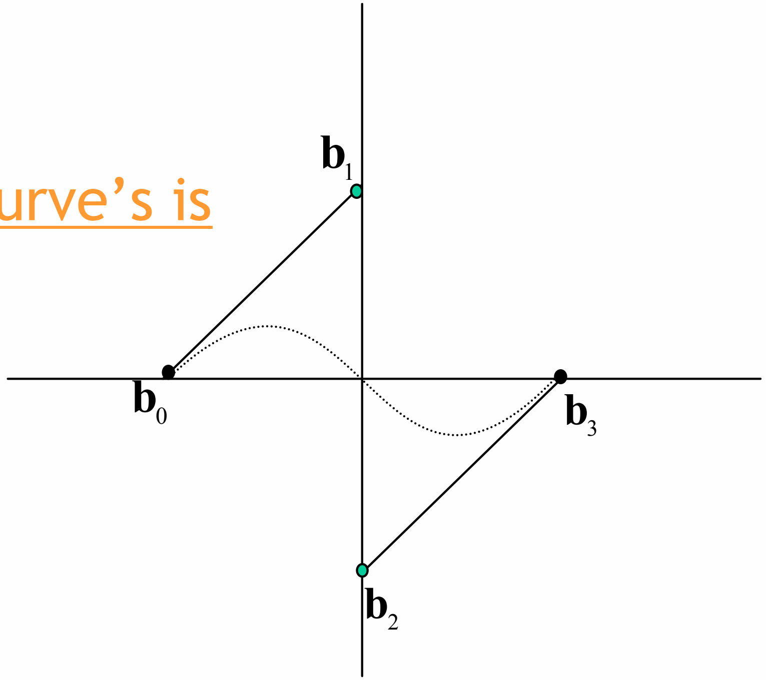
$$\dot{\mathbf{r}}(t) = \frac{d\mathbf{r}(t)}{dt} = 3(\Delta\mathbf{b}_0 B_0^2 + \Delta\mathbf{b}_1 B_1^2 + \Delta\mathbf{b}_2 B_2^2).$$

  - where, B_i^2 : quadratic Bernstein basis function.

- Most important tangent vectors at the curve's is Endpoints tangent vectors:

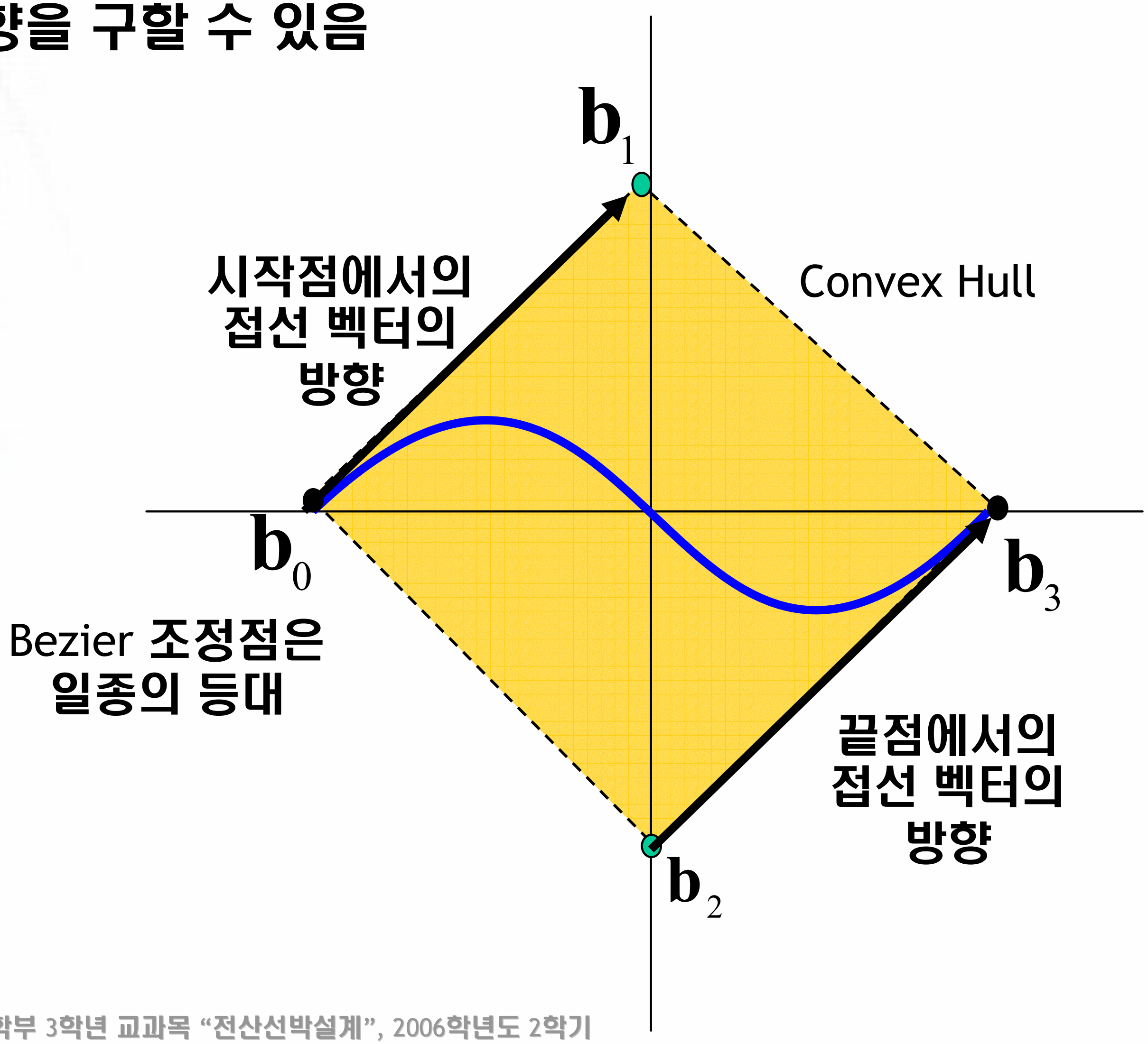
$$\dot{\mathbf{r}}(0) = 3\Delta\mathbf{b}_0 = 3(\mathbf{b}_1 - \mathbf{b}_o),$$

$$\dot{\mathbf{r}}(1) = 3\Delta\mathbf{b}_2 = 3(\mathbf{b}_3 - \mathbf{b}_2)$$

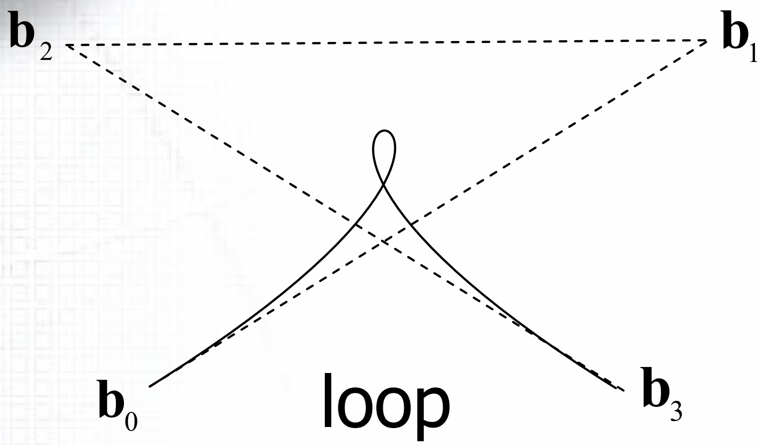


2.2.1.3 Characteristics of Bezier Curves (1)

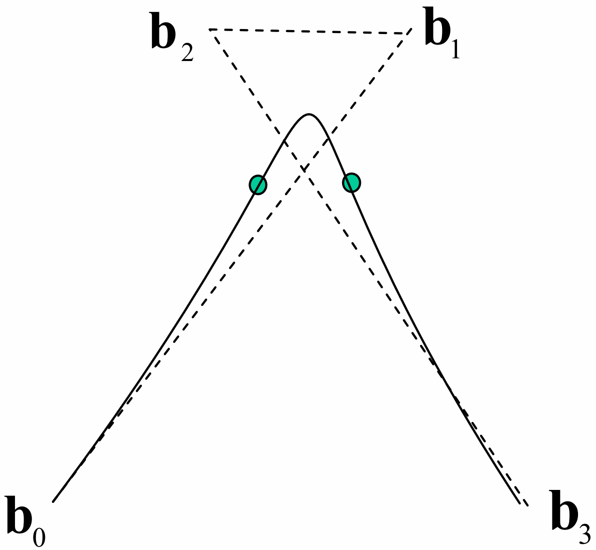
- Bezier 곡선은 최외각 조정점들로 구성되는 Convex Hull 내에 존재함 $\left(\because \sum_{i=0}^{3} B_i^3(t) = 1\right)$
- 처음 두 개의 조정점과 마지막 두 개의 조정점으로부터 시작점 및 끝점에서의 접선 벡터의 방향을 구할 수 있음



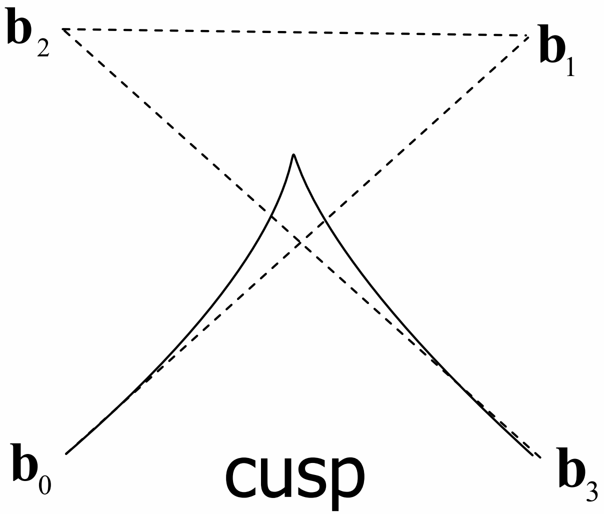
2.2.1.3 Characteristics of Bezier Curves (2)



loop



Two inflection points



cusp

2.2.1.4 Higher order Bezier Curves (1)

- A Bezier Curve of degree n can be defined by;

$$\mathbf{r}(t) = \mathbf{b}_0 B_0^n(t) + \mathbf{b}_1 B_1^n(t) + ... + \mathbf{b}_n B_n^n(t).$$

- where, $B_i^n(t)$: Bernstein Polynomial Function.

$$B_i^n(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i},$$

$$\begin{pmatrix} n \\ i \end{pmatrix} = {}_nC_i = \begin{cases} \dfrac{n!}{i!(n-i)!} & \textbf{if } 0 \le i \le n \\ \mathbf{0} & \textbf{else} \end{cases}$$

$$B_i^n(t) = tB_{i-1}^{n-1}(t) + (1-t)B_i^{n-1}(t) \quad \text{with } B_0^0(t) \equiv 1$$

- For cubic case, the Bezier curve as:

$$\mathbf{r}(t) = \mathbf{b}_0 B_0^3(t) + \mathbf{b}_1 B_1^3(t) + \mathbf{b}_2 B_2^3(t) + \mathbf{b}_3 B_3^3(t).$$

2.2.1.4 Higher order Bezier Curves (2)

- Bernstein Polynomial Function:

$$\begin{aligned}[(1-t)+t]^2 &= (1-t)^2 + 2(1-t)t + t^2 \\ &= B_0^2(t) + B_1^2(t) + B_2^2(t),\end{aligned}$$

$$\begin{aligned}[(1-t)+t]^3 &= [(1-t)+t]^2[(1-t)+t] \\ &= (1-t)^3 + 3(1-t)^2t + 3(1-t)t^2 + t^3 \\ &= B_0^3(t) + B_1^3(t) + B_2^3(t) + B_3^3(t),\end{aligned}$$

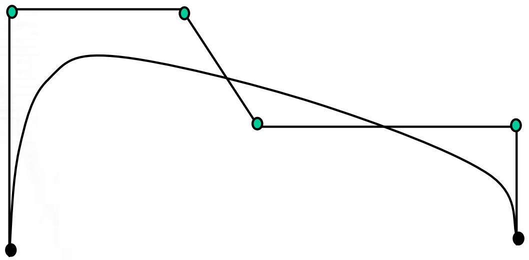
$$\begin{aligned}[(1-t)+t]^4 &= [(1-t)+t]^3[(1-t)+t] \\ &= (1-t)^4 + 4(1-t)^3t + 6(1-t)^2t^2 + 4(1-t)t^3 + t^4 \\ &= B_0^4(t) + B_1^4(t) + B_2^4(t) + B_3^4(t) + B_4^4(t)\end{aligned}$$

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

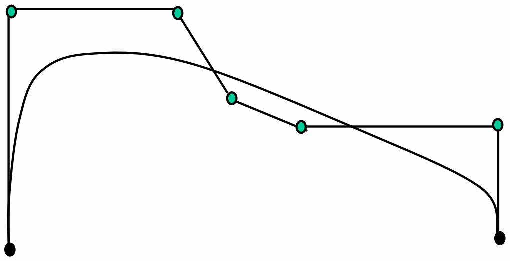
파스칼의 삼각형

2.2.1.4 Higher order Bezier Curves (3)

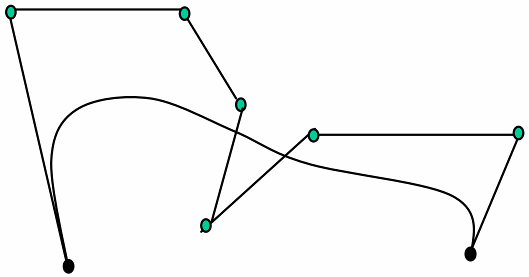
5th-degree Bezier curve



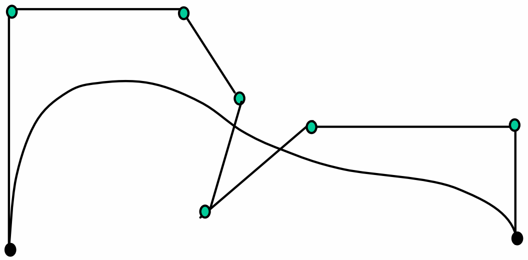
6th-degree Bezier curve



7th-degree Bezier curve



7th-degree Bezier curve



2.2.1.5 Derivatives of Higher Order Bezier Curves (1)

- For Cubic Case(n=3),

$$\dot{\mathbf{r}}(t) = 3[\Delta\mathbf{b}_0 B_0^2 + \Delta\mathbf{b}_1 B_1^2 + \Delta\mathbf{b}_2 B_2^2].$$

- For degree=n,

$$\dot{\mathbf{r}}(t) = n[\Delta\mathbf{b}_0 B_0^{n-1} + \Delta\mathbf{b}_1 B_1^{n-1} + \dots\dots\dots + \Delta\mathbf{b}_{n-1} B_{n-1}^{n-1}].$$

where $\Delta\mathbf{b}_i = \mathbf{b}_{i+1} - \mathbf{b}_i$: forward difference.

- Bezier Curve→ differentiated by more than one by parameter $'t'$.

- For the k^{th} times derivative:

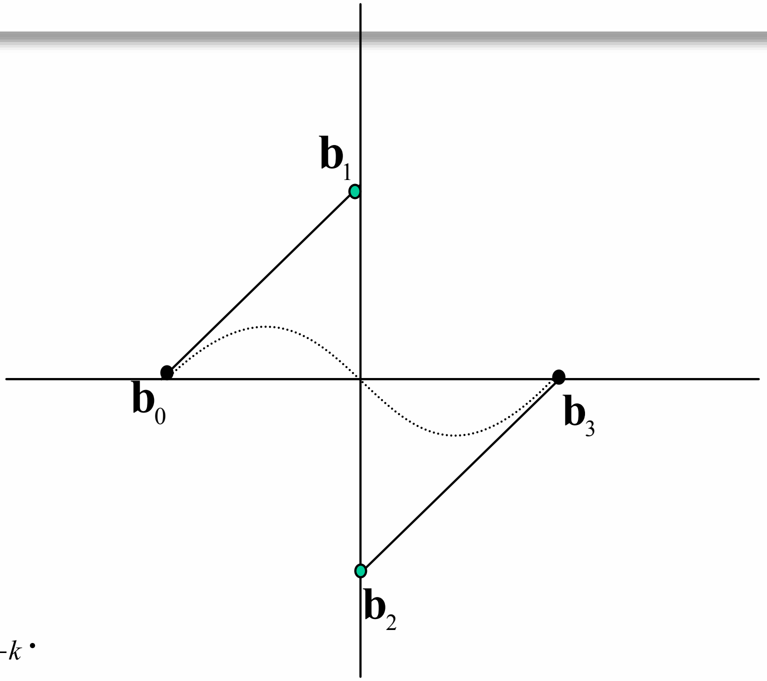
$$\frac{d^k\mathbf{r}(t)}{dt^k} = \frac{n!}{(n-k)!}[\Delta^k\mathbf{b}_0 B_0^{n-k}(t) + \Delta^k\mathbf{b}_1 B_1^{n-k}(t)\dots\dots + \Delta^k\mathbf{b}_{n-k} B_{n-k}^{n-k}(t)].$$

2.2.1.5 Derivatives of Higher Order Bezier Curves (2)

- where, $\boldsymbol{\Delta}^k$:forward operator.
- we can get $\Delta^k \mathbf{b}_i = \Delta^{k-1}\mathbf{b}_{i+1} - \Delta^{k-1}\mathbf{b}_i$.

  where, $\Delta^0 \mathbf{b}_i = \mathbf{b}_i$.
- for k=2 : $\mathbf{b}_{i+2} - 2\mathbf{b}_{i+1} + \mathbf{b}_i$.
- for k=3 : $\mathbf{b}_{i+3} - 3\mathbf{b}_{i+2} + 3\mathbf{b}_{i+1} - \mathbf{b}_i$.
- for k=4 : $\mathbf{b}_{i+4} - 4\mathbf{b}_{i+3} + 6\mathbf{b}_{i+2} - 4\mathbf{b}_{i+1} + \mathbf{b}_i$.
- the k^{th} derivative of $\mathbf{r}(0)$ and $\mathbf{r}(1)$;

$$\mathbf{r}^k(0) = \frac{n!}{(n-k)!}\Delta^k \mathbf{b}_0 \text{ and } \mathbf{r}^k(1) = \frac{n!}{(n-k)!}\Delta^k \mathbf{b}_{n-k}.$$



- For n=3, k=2;

$$\begin{aligned}\mathbf{r}^2(0) &= \frac{3!}{(3-2)!}\Delta^2 \mathbf{b}_0 \\ &= 6(\Delta^1 \mathbf{b}_1 - \Delta^1 \mathbf{b}_0) \\ &= 6((\Delta^0 \mathbf{b}_2 - \Delta^0 \mathbf{b}_1) - (\Delta^0 \mathbf{b}_1 - \Delta^0 \mathbf{b}_0)) \\ &= 6(\Delta^0 \mathbf{b}_2 - 2\Delta^0 \mathbf{b}_1 + \Delta^0 \mathbf{b}_0) \\ &= 6(\mathbf{b}_2 - 2\mathbf{b}_1 + \mathbf{b}_0)\end{aligned}$$

$$\begin{aligned}\mathbf{r}^2(1) &= \frac{3!}{(3-2)!}\Delta^2 \mathbf{b}_1 \\ &= 6(\Delta^1 \mathbf{b}_2 - \Delta^1 \mathbf{b}_1) \\ &= 6((\Delta^0 \mathbf{b}_3 - \Delta^0 \mathbf{b}_2) - (\Delta^0 \mathbf{b}_2 - \Delta^0 \mathbf{b}_1)) \\ &= 6(\Delta^0 \mathbf{b}_3 - 2\Delta^0 \mathbf{b}_2 + \Delta^0 \mathbf{b}_1) \\ &= 6(\mathbf{b}_3 - 2\mathbf{b}_2 + \mathbf{b}_1)\end{aligned}$$

2.2.1.6 Matrix form of Bezier curves(1)

- Cubic Bezier Curve

$$\mathbf{r}(t) = (1-t)^3\mathbf{b}_0 + 3(1-t)^2 t\mathbf{b}_1 + 3(1-t)t^2\mathbf{b}_2 + t^3\mathbf{b}_3$$

- applying the dot product to above equation;

$$\mathbf{r}(t) = \begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} \begin{bmatrix} (1-t)^3 \\ 3(1-t)^2 t \\ 3(1-t)\, t^2 \\ t^3 \end{bmatrix}$$

2.2.1.6 Matrix form of Bezier curves(2)

- The Matrix form of Bezier Curve is

$$\mathbf{r}(t) = \begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} \begin{bmatrix} (1-t)^3 \\ 3(1-t)^2 t \\ 3(1-t)\, t^2 \\ t^3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} \begin{bmatrix} B_0^3(\mathbf{t}) \\ B_1^3(\mathbf{t}) \\ B_2^3(\mathbf{t}) \\ B_3^3(\mathbf{t}) \end{bmatrix}$$

Conversion to the monomial form: $\mathbf{r}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2 + \mathbf{a}_3 t^3$

$$\mathbf{r}(t) = \begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} \begin{bmatrix} (1-t)^3 \\ 3(1-t)^2 t \\ 3(1-t)\, t^2 \\ t^3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}$$

2.2.1.6 Matrix form of Bezier curves(3)

- The Matrix form of Monomial Curve is

$$\mathbf{r}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2 + \mathbf{a}_3 t^3 = \begin{bmatrix} \mathbf{a_0} & \mathbf{a_1} & \mathbf{a_2} & \mathbf{a_3} \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}$$

Conversion to the Bezier form:

$$\begin{aligned} \mathbf{r}(t) &= (1-t)^3 \mathbf{b}_0 + 3(1-t)^2 t \mathbf{b}_1 + 3(1-t) t^2 \mathbf{b}_2 + t^3 \mathbf{b}_3 \\ &= B_0^3(t)\,\mathbf{b}_0 + B_1^3(t)\,\mathbf{b}_1 + B_2^3(t)\,\mathbf{b}_2 + B_3^3(t)\,\mathbf{b}_3 \\ &= \begin{bmatrix} \mathbf{b_0} & \mathbf{b_1} & \mathbf{b_2} & \mathbf{b_3} \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \end{aligned}$$

$$\therefore \begin{bmatrix} \mathbf{b_0} & \mathbf{b_1} & \mathbf{b_2} & \mathbf{b_3} \end{bmatrix} = \begin{bmatrix} \mathbf{a_0} & \mathbf{a_1} & \mathbf{a_2} & \mathbf{a_3} \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{\mathbf{-1}}$$

2.2.1.7 Sample code of Bezier Curve class(1)

```
#ifndef __BezierCurve_h__
#define __BezierCurve_h__

#include “vector.h”

class BezierCurve {
public:
    int m_nDegree;
    Vector* m_ControlPoint;   int m_nControlPoint;
    BezierCurve();
    ~BezierCurve();

    void SetDegree(int nDegree);
    void SetControlPoint(Vector* pControlPoint, int nControlPoint);
    Vector CalcPoint(double t);
    double B (int i, double t);                          // Bernstein Polynomial
};
#endif
```

2.2.1.7 Sample code of Bezier Curve class(2)

```
BezierCurve::BezierCurve () {
   m_ControlPoint = 0;  m_nDegree = 0;
   m_nControlPoint = 0;
}
BezierCurve::~BezierCurve () {
   if(m_ControlPoint) delete[] m_ControlPoint;
}
void BezierCurve::SetControlPoint(Vector* pControlPoint, int nControlPoint) {
   SetDegree( nControlPoint-1 );
   if(m_ControlPoint) delete[] m_ControlPoint;
   m_ControlPoint = new Vector[nControlPoint];
   for(int i=0; i < nControlPoint; i++) {
      m_ControlPoint[i] = pControlPoint[i];
   }
}
void BezierCurve::SetDegree(int nDegree){
   m_nDegree = nDegree;
}
```

2.2.1.7 Sample code of Bezier Curve class(3)

```
Vector BezierCurve:: CalcPoint(double t) {
    Vector PointOnCurve(0,0,0);
    if ( t < 0.0 || t > 1.0 ) {
        return PointOnCurve;
    }
    for(int i = 0; i < m_nControlPoint; i++){
        PointOnCurve = PointOnCurve + m_ControlPoint[i] * B(i,t);
    }
    return PointOnCurve;
}

double BezierCurve:: B (int i, double t) {
   double result = 0;
   // Calculate ith Berstein Polynomial at parameter t
    …….
   return result;
}
```

2.2.2 Degree Elevation / Reduction of Bezier curves

- 2.2.2.1 Degree Elevation
- 2.2.2.2 Degree Reduction
- 2.2.2.3 Repeated Degree Elevation

Advanced
Ship
Design
Automation
Laboratory

2.2.2.1 Degree Elevation (1)

- **목적**
  - **서로 다른 차수의 곡선을 같은 차수로 연결할 때 사용**
    (3**차** Bezier **곡선** + 4**차** Bezier **곡선** → 4**차** Bezier **곡선** + 4**차** Bezier **곡선**)
  - **보다 많은 조정점으로 곡선을 보다 자유롭게 설계**
    (Bezier **조정점의 개수** = **차수**+1)

- **참고:**

n **차** B-Spline **곡선은 주어진 놋트 상에서 부드럽게 연결된**
n 차 Bezier 곡선의 집합 **이라고 볼 수 있다.**
따라서, B-Spline **곡선의 차수 증가 방법은**
Bezier **곡선의 차수 증가 방법으로 설명될 수 있다.**

2.2.2.1 Degree Elevation (2)

- 2차 Bézier curve -> 3차 Bézier curve

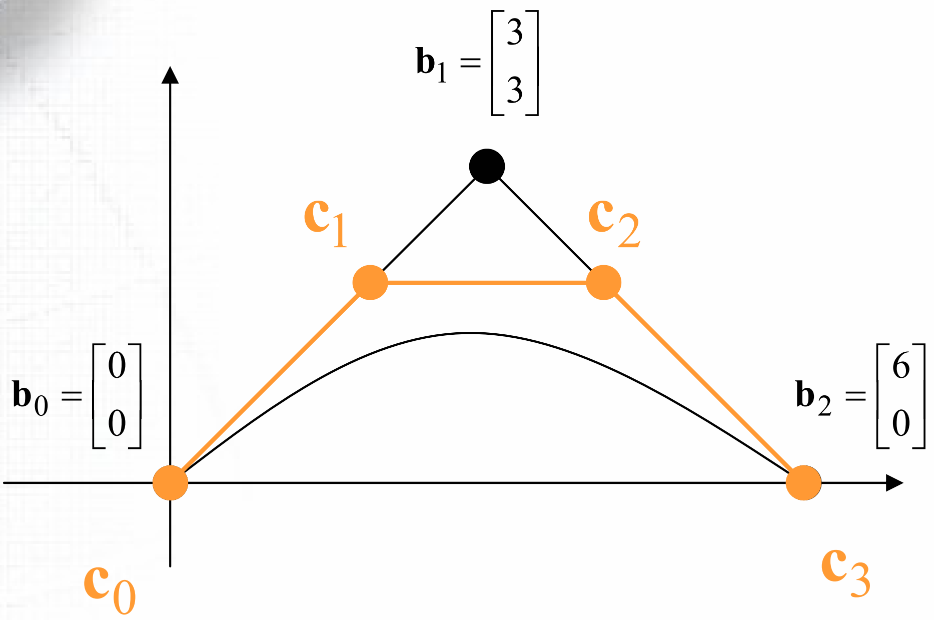
$$\mathbf{r}(t) = (1-t)^2\mathbf{b}_0 + 2(1-t)t\mathbf{b}_1 + t^2\mathbf{b}_2 \quad \times [t + (1-t)]$$

$$\mathbf{r}(t) = [t(1-t)^2 + (1-t)^3]\mathbf{b}_0 + 2[t^2(1-t) + (1-t)^2 t]\mathbf{b}_1 + [t^3 + t^2(1-t)]\mathbf{b}_2$$

$$\boxed{\mathbf{r}(t) = (1-t)^3\boxed{\mathbf{b}_0} + 3(1-t)^2 t\boxed{[\frac{1}{3}\mathbf{b}_0 + \frac{2}{3}\mathbf{b}_1]} + 3(1-t)t^2\boxed{[\frac{2}{3}\mathbf{b}_1 + \frac{1}{3}\mathbf{b}_2]} + t^3\boxed{\mathbf{b}_2}}$$

⇨ 즉, □를 새로운 control point로 갖는 3차 Bézier curve

2.2.2.1 Degree Elevation (3)



$$\mathbf{c}_0 = \mathbf{b}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\mathbf{c}_1 = [\frac{1}{3}\mathbf{b}_0 + \frac{2}{3}\mathbf{b}_1] = \begin{bmatrix} 2 \\ 2 \end{bmatrix},$$

$$\mathbf{c}_2 = [\frac{2}{3}\mathbf{b}_1 + \frac{1}{3}\mathbf{b}_2] = \begin{bmatrix} 4 \\ 2 \end{bmatrix},$$

$$\mathbf{c}_3 = \mathbf{b}_2 = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$$

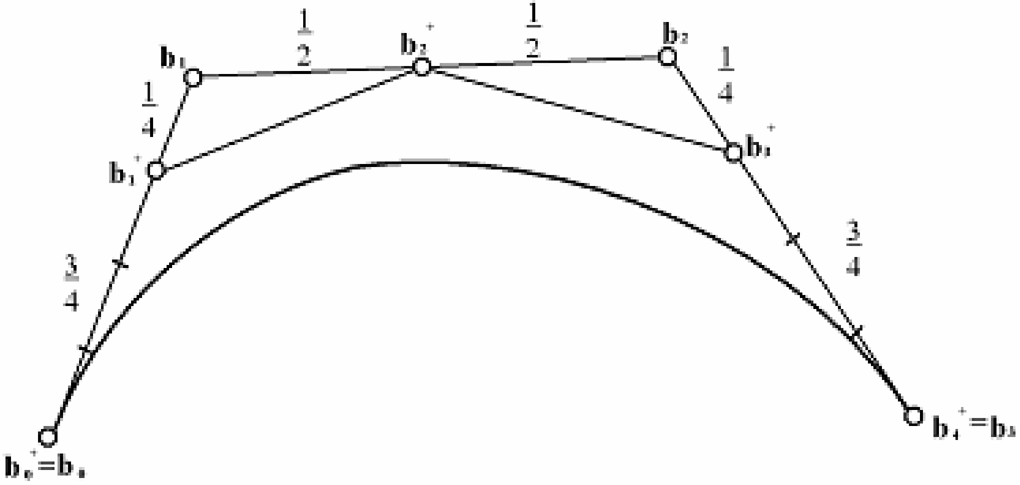
$$\mathbf{r}(t) = (1-t)^2\mathbf{b}_0 + 2(1-t)t\mathbf{b}_1 + t^2\mathbf{b}_2$$

$$\mathbf{r}(t) = (1-t)^3\mathbf{b}_0 + 3(1-t)^2 t[\frac{1}{3}\mathbf{b}_0 + \frac{2}{3}\mathbf{b}_1] + 3(1-t)t^2[\frac{2}{3}\mathbf{b}_1 + \frac{1}{3}\mathbf{b}_2] + t^3\mathbf{b}_2$$

2.2.2.1 Degree Elevation (4)

- $\mathbf{b}_0,\ldots,\mathbf{b}_n$ 를 control point로 가지는 n차 Bézier curve를 n+1차로 degree elevation하면

$$\mathbf{c}_0 = \mathbf{b}_0,$$
$$\vdots$$
$$\mathbf{c}_i = \frac{i}{n+1}\mathbf{b}_{i-1} + \left(1 - \frac{i}{n+1}\right)\mathbf{b}_i,$$
$$\vdots$$
$$\mathbf{c}_{n+1} = \mathbf{b}_n$$



3차 → 4차 degree elevation

2.2.2.1 Degree Elevation (5)

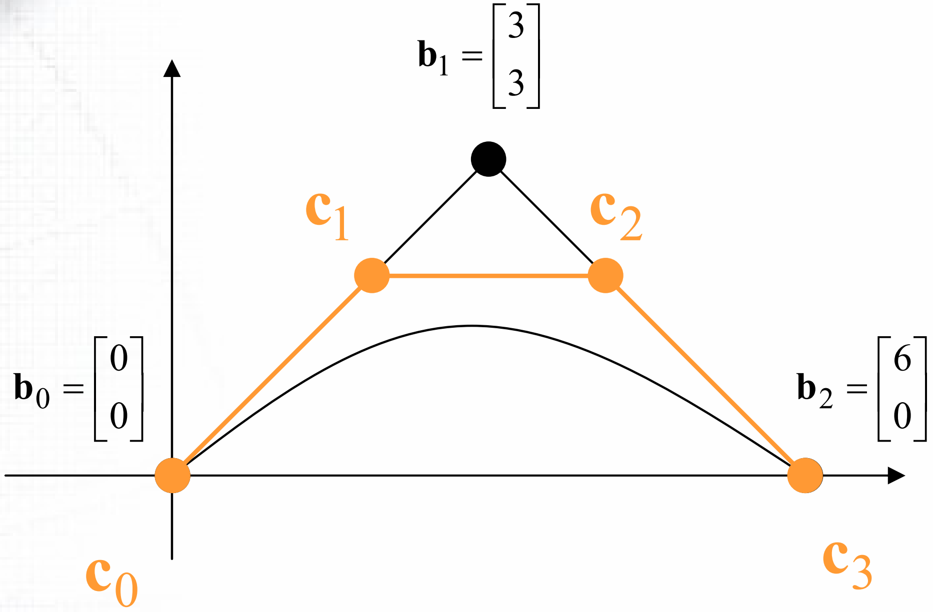
- $\mathbf{b}_0,\ldots,\mathbf{b}_n$ 를 control point로 가지는 n차 Bézier curve를 n+1차로 degree elevation하면

$$
\begin{aligned}
&\mathbf{c}_0 = \mathbf{b}_0,\\
&\quad\vdots\\
&\mathbf{c}_i = \frac{i}{n+1}\mathbf{b}_{i-1} + (1-\frac{i}{n+1})\mathbf{b}_i,\\
&\quad\vdots\\
&\mathbf{c}_{n+1} = \mathbf{b}_n
\end{aligned}
$$

$$
n+2\ \ rows\left\{\overbrace{\begin{bmatrix}
1 & & & & & \\
* & * & & & & \\
 & * & * & & & \\
 & & & \vdots & \vdots & \\
 & & & & * & * \\
 & & & & & 1
\end{bmatrix}}^{n+1\ \ colums}\right.
\begin{bmatrix}\mathbf{b}_0\\ \vdots\\ \mathbf{b}_n\end{bmatrix}
=
\begin{bmatrix}\mathbf{c}_0\\ \vdots\\ \mathbf{c}_{n+1}\end{bmatrix}
$$

$$\mathbf{DB} = \mathbf{C}$$

2.2.2.1 Degree Elevation (6)



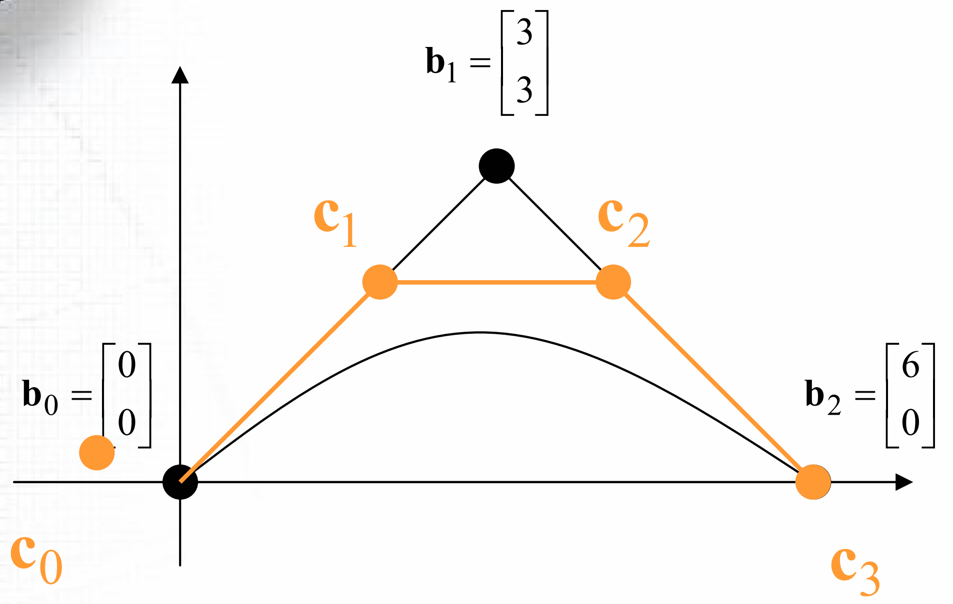
$$\mathbf{c}_0 = \mathbf{b}_0,$$
$$\mathbf{c}_1 = [\frac{1}{3}\mathbf{b}_0 + \frac{2}{3}\mathbf{b}_1],$$
$$\mathbf{c}_2 = [\frac{2}{3}\mathbf{b}_1 + \frac{1}{3}\mathbf{b}_2],$$
$$\mathbf{c}_3 = \mathbf{b}_2$$

$$\mathbf{DB} = \mathbf{C}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1/3 & 2/3 & 0 \\ 0 & 2/3 & 1/3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 3 & 3 \\ 6 & 0 \end{bmatrix} = \mathbf{C}$$

$$\therefore \mathbf{C} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 4 & 2 \\ 6 & 0 \end{bmatrix}$$

2.2.2.2 Degree Reduction



$$\mathbf{c}_0 = \mathbf{b}_0,$$
$$\mathbf{c}_1 = [\frac{1}{3}\mathbf{b}_0 + \frac{2}{3}\mathbf{b}_1],$$
$$\mathbf{c}_2 = [\frac{2}{3}\mathbf{b}_1 + \frac{1}{3}\mathbf{b}_2],$$
$$\mathbf{c}_3 = \mathbf{b}_2$$

$$\mathbf{DB} = \mathbf{C}$$

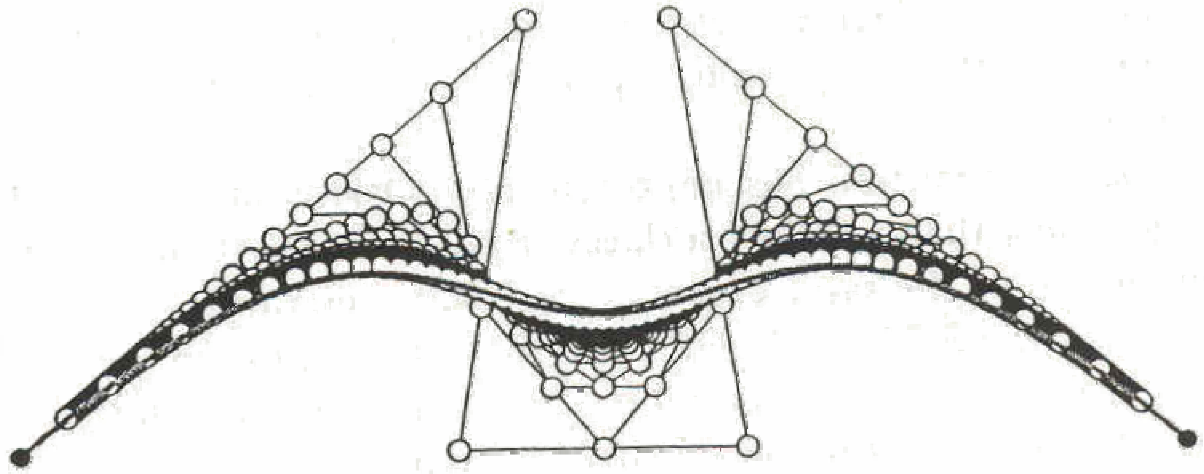
$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 1/3 & 2/3 & 0 \\ 0 & 2/3 & 1/3 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \\ 4 & 2 \\ 6 & 0 \end{bmatrix}$$

$$\mathbf{D}^T\mathbf{DB} = \mathbf{D}^T\mathbf{C}$$

$$\therefore \mathbf{B} = \left(\mathbf{D}^T\mathbf{D}\right)^{-1}\mathbf{D}^T\mathbf{C}$$

$$\mathbf{D}^T\mathbf{D} = \frac{1}{9}\begin{bmatrix} 10 & 2 & 0 \\ 2 & 8 & 2 \\ 0 & 2 & 10 \end{bmatrix}, \quad \mathbf{D}^T\mathbf{C} = \frac{1}{3}\begin{bmatrix} 2 & 2 \\ 12 & 8 \\ 22 & 2 \end{bmatrix}, \quad \therefore \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 3 & 3 \\ 6 & 0 \end{bmatrix}$$

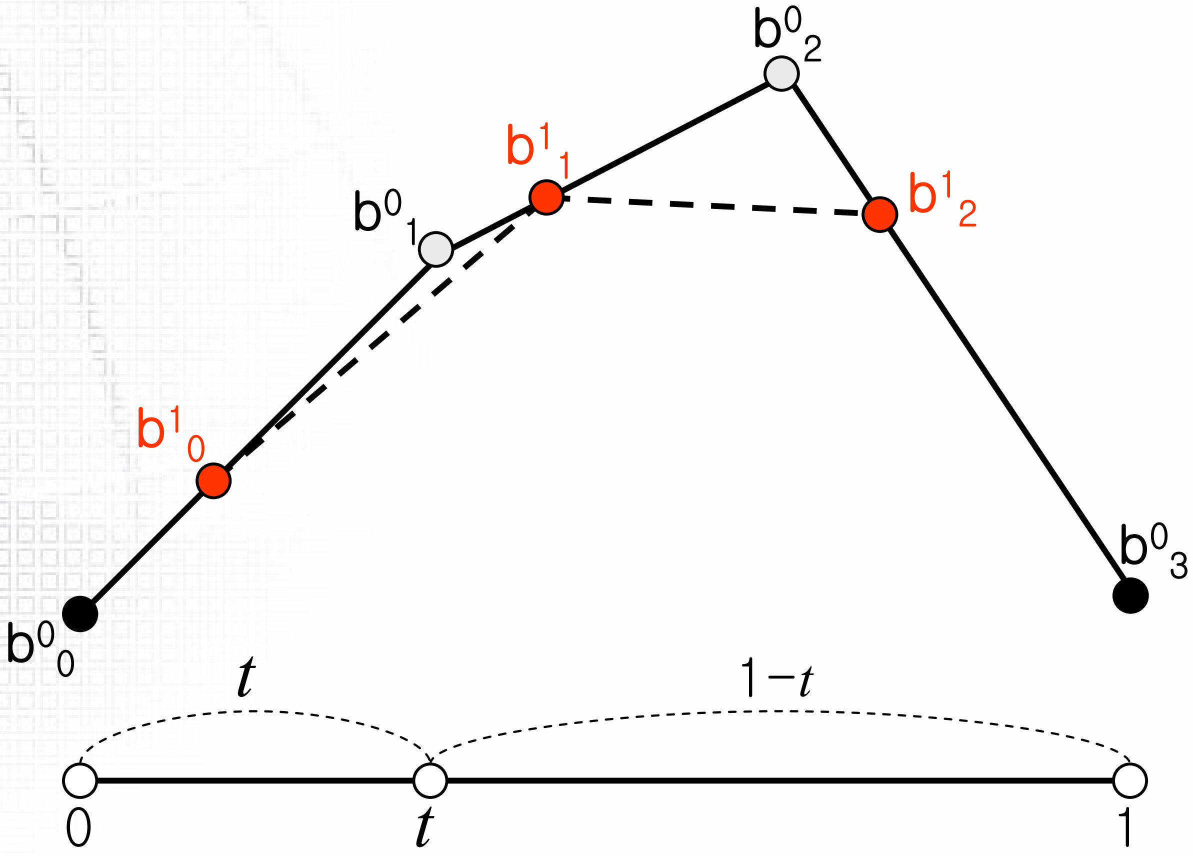
2.2.2.3 Repeated Degree Elevation



무한히 반복하면 polygon이 curve에 근접해간다.

2.2.3 de Casteljau algorithm

2.2.3.1 de Casteljau algorithm & Bezier curves (1)



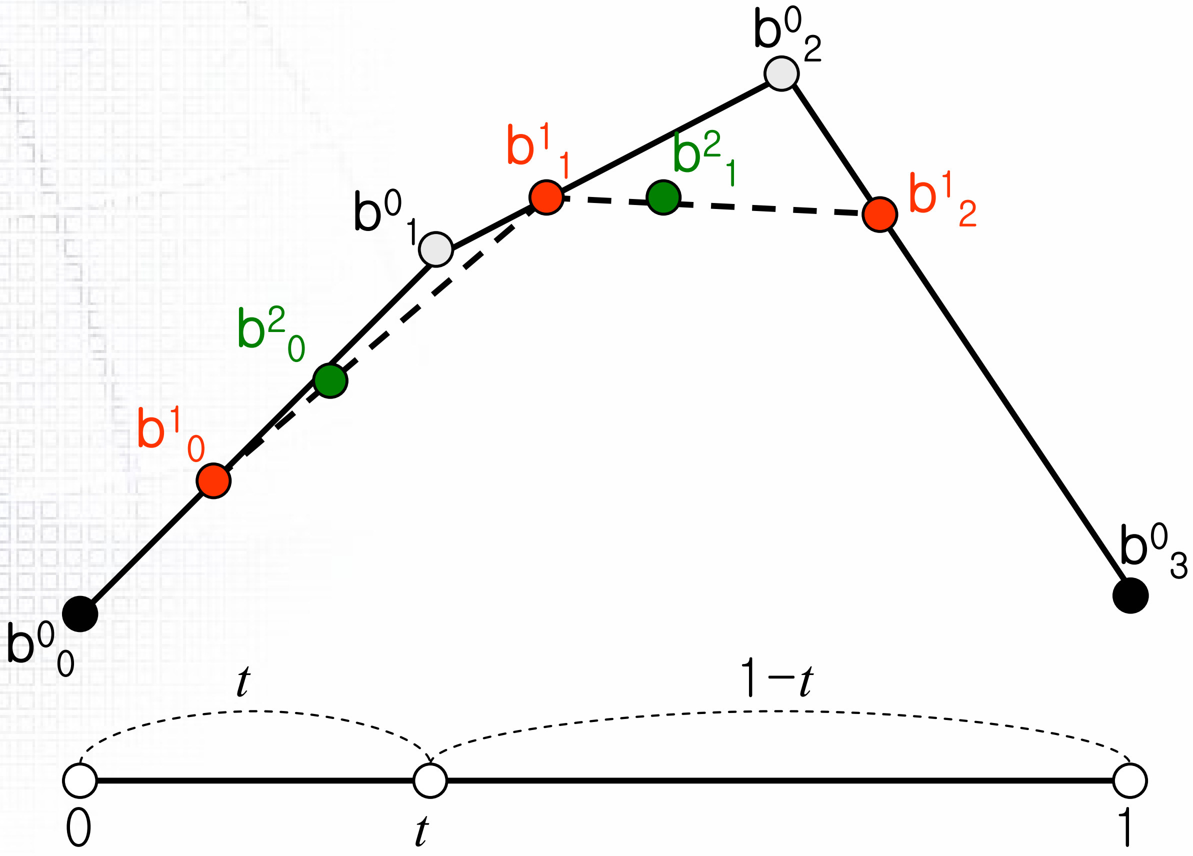
Linear interpolation

$$\mathbf{b}_0^1(t) = (1-t)\mathbf{b}_0^0 + t\mathbf{b}_1^0$$

$$\mathbf{b}_1^1(t) = (1-t)\mathbf{b}_1^0 + t\mathbf{b}_2^0$$

$$\mathbf{b}_2^1(t) = (1-t)\mathbf{b}_2^0 + t\mathbf{b}_3^0$$

2.2.3.1 de Casteljau algorithm & Bezier curves (2)



Linear interpolation

$$\mathbf{b}^1_0(t) = (1-t)\mathbf{b}^0_0 + t\mathbf{b}^0_1$$

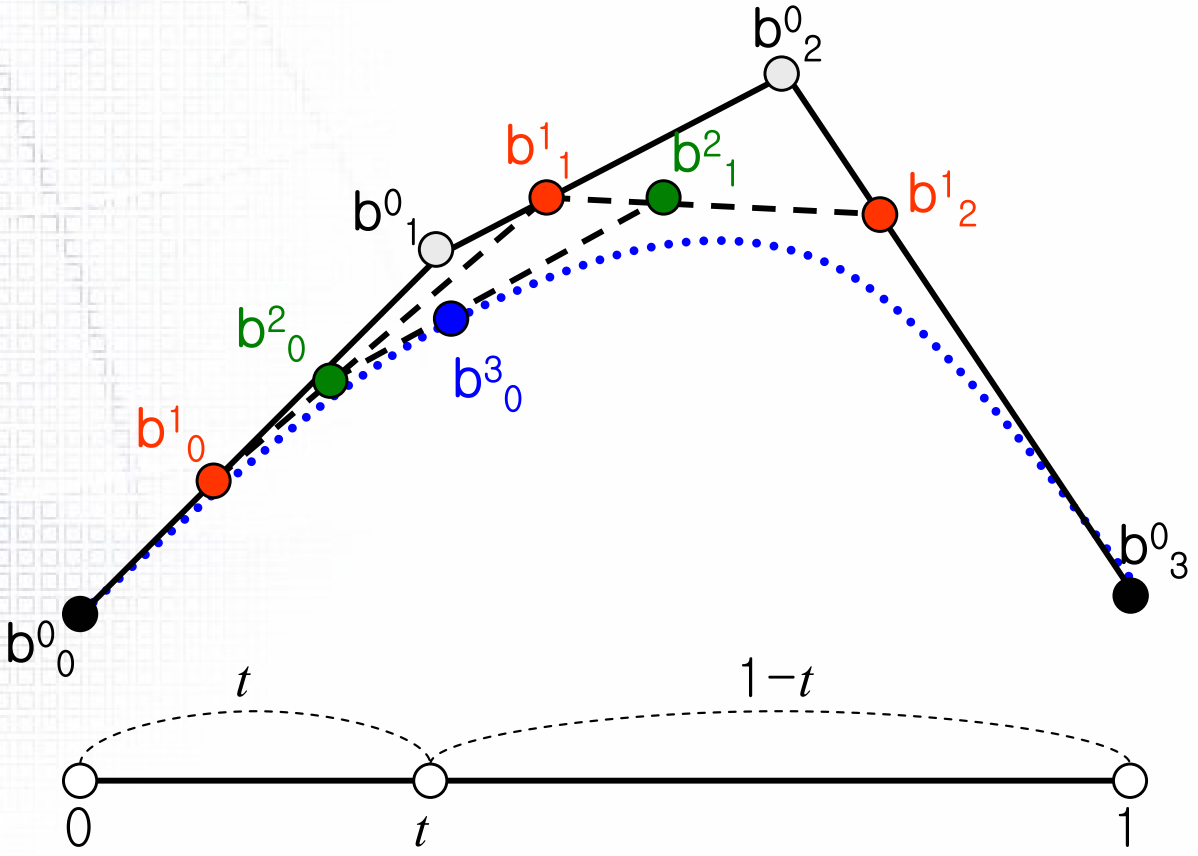
$$\mathbf{b}^1_1(t) = (1-t)\mathbf{b}^0_1 + t\mathbf{b}^0_2$$

$$\mathbf{b}^1_2(t) = (1-t)\mathbf{b}^0_2 + t\mathbf{b}^0_3$$

$$\mathbf{b}^2_0(t) = (1-t)\mathbf{b}^1_0 + t\mathbf{b}^1_1$$

$$\mathbf{b}^2_1(t) = (1-t)\mathbf{b}^1_1 + t\mathbf{b}^1_2$$

2.2.3.1 de Casteljau algorithm & Bezier curves (3)



Linear interpolation

$$\mathbf{b}_0^1(t) = (1-t)\mathbf{b}_0^0 + t\mathbf{b}_1^0$$

$$\mathbf{b}_1^1(t) = (1-t)\mathbf{b}_1^0 + t\mathbf{b}_2^0$$

$$\mathbf{b}_2^1(t) = (1-t)\mathbf{b}_2^0 + t\mathbf{b}_3^0$$

$$\mathbf{b}_0^2(t) = (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1$$

$$\mathbf{b}_1^2(t) = (1-t)\mathbf{b}_1^1 + t\mathbf{b}_2^1$$

$$\mathbf{b}_0^3(t) = (1-t)\mathbf{b}_0^2 + t\mathbf{b}_1^2$$

3차 Bezier curves 와 동일한 함수식 !!!

$$\mathbf{b}_0^3(t) = (1-t)^3\mathbf{b}_0^0 + 3t(1-t)^2\mathbf{b}_1^0 + 3t^2(1-t)\mathbf{b}_2^0 + t^3\mathbf{b}_3^0$$

2.2.3.1 de Casteljau algorithm & Bezier curves (4)

- de Casteljau **알고리즘**: "Constructive Approach"

  Input: $\mathbf{b}_i$ (Bezier control points)

  Processor: n**번 순차적** 'linear interpolation'

  Output : n**차 곡선상의 점**

  → Bernstein basis function(polynomial) **형태로 표현 됨**

- Bezier **곡선식**: "Bernstein Function evaluation Approach"

  Input: $\mathbf{b}_i$ (Bezier control points)

  Processor: **공간 상의 점** $\mathbf{b}_i$ **와** Bernstein Basis function**을** "blending"**하여 함수를 값을 계산하면 곡선상의 점을 구할 수 있음**

  Output: Bernstein basis function(polynomial) **과** $\mathbf{b}_i$ **의 혼합 함수 형태로 표현**

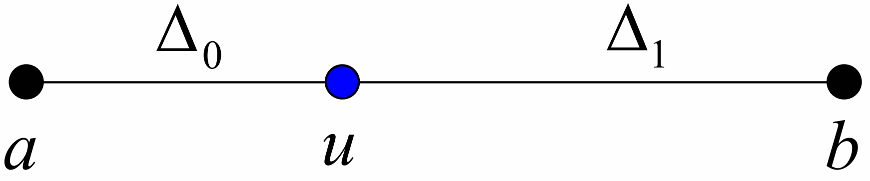
2.2.3.2 Parameter Transformation

- Parameter Transformation
- the affine map for the interval of $t \in [0,1] \to u \in [a,b],$
- We get

$$t = \frac{u-a}{b-a}. \quad \text{(or)} \quad 1-t = \frac{b-u}{b-a}.$$

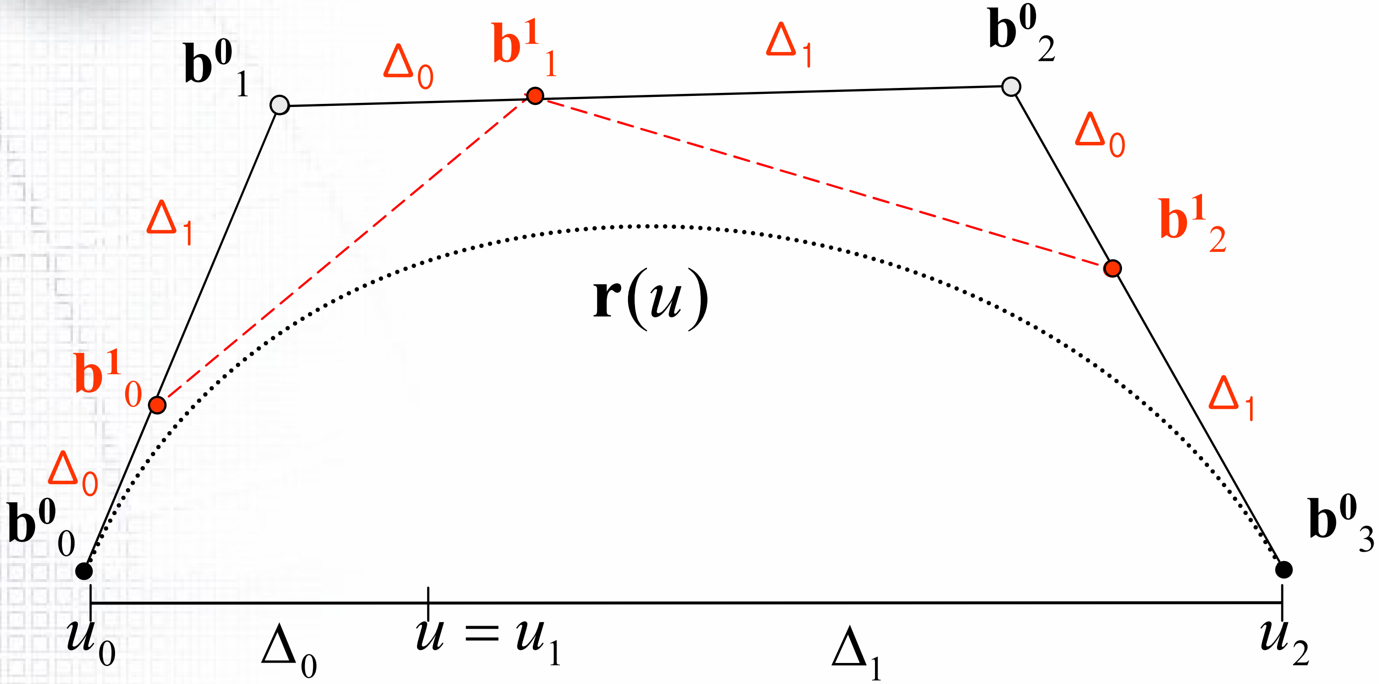
- u → global parameter, t → local parameter
- the process of changing interval is called parameter transformation.

2.2.3.3 Linear Interpolation on [a, b]



$$u-a:b-u=\Delta_0:\Delta_1$$
$$\Delta_0(b-u)=\Delta_1(u-a)$$
$$(\Delta_0+\Delta_1)u=\Delta_1 a+\Delta_0 b$$
$$u=\frac{\Delta_1 a+\Delta_0 b}{\Delta_0+\Delta_1}$$
$$\therefore u=\frac{\Delta_1}{\Delta_0+\Delta_1}a+\frac{\Delta_0}{\Delta_0+\Delta_1}b$$
$$ratio(a,u,b)=\frac{\Delta_0}{\Delta_1}$$

2.2.3.4 매개변수 구간이 $[u_0, u_2]$인 경우, $u = u_1$ 에서의 곡선상의 점 구하기: de Casteljau Algorithm



$$\mathbf{b}_0^1(u) = \frac{u_2 - u}{u_2 - u_0}\mathbf{b}_0^0 + \frac{u - u_0}{u_2 - u_0}\mathbf{b}_1^0$$

$$= \frac{\Delta_1}{\Delta}\mathbf{b}_0^0 + \frac{\Delta_0}{\Delta}\mathbf{b}_1^0$$

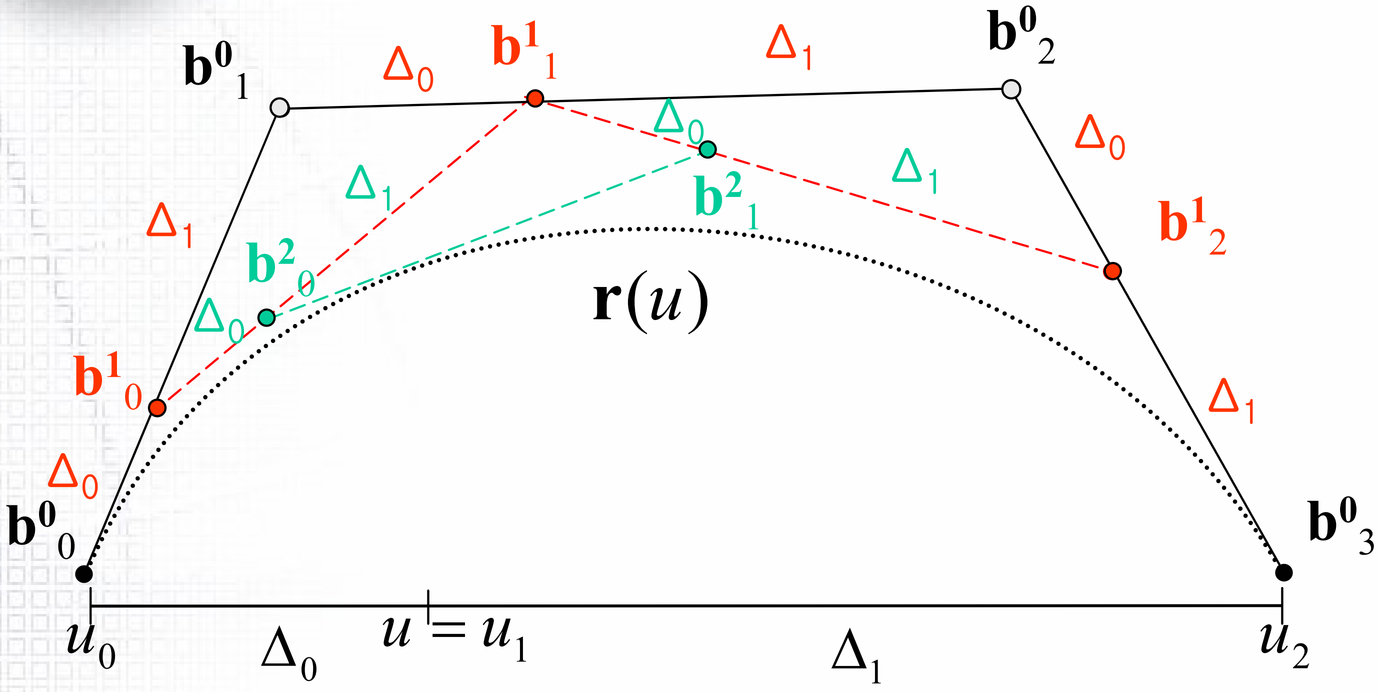
$$\mathbf{b}_1^1(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_1^0 + \frac{\Delta_0}{\Delta}\mathbf{b}_2^0$$

$$\mathbf{b}_2^1(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_2^0 + \frac{\Delta_0}{\Delta}\mathbf{b}_3^0$$

$$\Delta = u_2 - u_0, \quad \Delta_1 = u_2 - u_1, \quad \Delta_0 = u_1 - u_0, \quad \Delta = \Delta_0 + \Delta_1$$

$$\text{ratio}(b_0^2, b_0^3, b_1^2) = \frac{u - u_0}{u_2 - u} = \frac{\Delta_0}{\Delta_1}$$

2.2.3.4 매개변수 구간이 $[u_0, u_2]$인 경우, $u = u_1$ 에서의 곡선상의 점 구하기: de Casteljau Algorithm



$$\mathbf{b}_0^1(u) = \frac{u_2 - u}{u_2 - u_0}\mathbf{b}_0^0 + \frac{u - u_0}{u_2 - u_0}\mathbf{b}_1^0$$

$$= \frac{\Delta_1}{\Delta}\mathbf{b}_0^0 + \frac{\Delta_0}{\Delta}\mathbf{b}_1^0$$

$$\mathbf{b}_1^1(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_1^0 + \frac{\Delta_0}{\Delta}\mathbf{b}_2^0$$

$$\mathbf{b}_2^1(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_2^0 + \frac{\Delta_0}{\Delta}\mathbf{b}_3^0$$

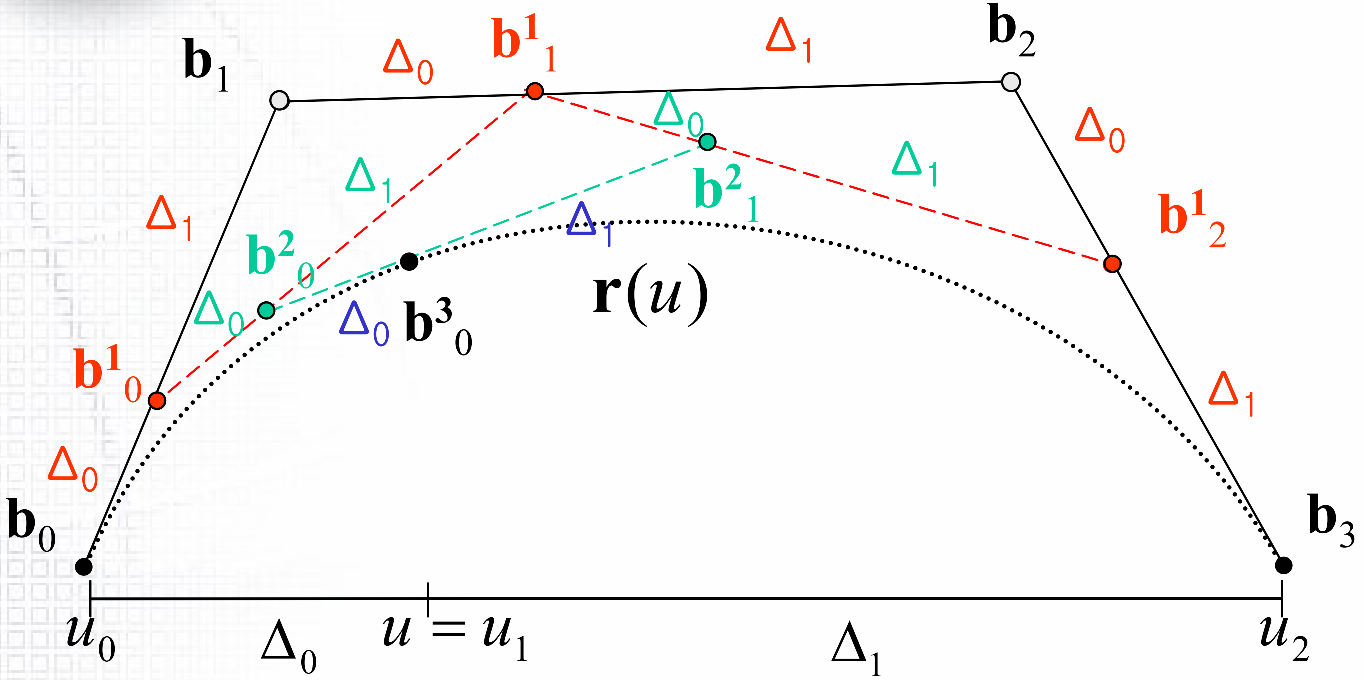
$$\mathbf{b}_0^2(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_0^1 + \frac{\Delta_0}{\Delta}\mathbf{b}_1^1$$

$$\mathbf{b}_1^2(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_1^1 + \frac{\Delta_0}{\Delta}\mathbf{b}_2^1$$

$$\Delta = u_2 - u_0, \quad \Delta_1 = u_2 - u_1, \quad \Delta_0 = u_1 - u_0, \quad \Delta = \Delta_0 + \Delta_1$$

$$\text{ratio}(b_0^2, b_0^3, b_1^2) = \frac{u - u_0}{u_2 - u} = \frac{\Delta_0}{\Delta_1}$$

2.2.3.4 매개변수 구간이 $[u_0, u_2]$인 경우, $u = u_1$ 에서의 곡선상의 점 구하기: de Casteljau Algorithm



$$\Delta = u_2 - u_0, \quad \Delta_1 = u_2 - u_1, \quad \Delta_0 = u_1 - u_0, \quad \Delta = \Delta_0 + \Delta_1$$

$$\text{ratio}(b_0^2, b_0^3, b_1^2) = \frac{u - u_0}{u_2 - u} = \frac{\Delta_0}{\Delta_1}$$

$$\mathbf{b}_0^1(u) = \frac{u_2 - u}{u_2 - u_0}\mathbf{b}_0^0 + \frac{u - u_0}{u_2 - u_0}\mathbf{b}_1^0$$

$$= \frac{\Delta_1}{\Delta}\mathbf{b}_0^0 + \frac{\Delta_0}{\Delta}\mathbf{b}_1^0$$

$$\mathbf{b}_1^1(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_1^0 + \frac{\Delta_0}{\Delta}\mathbf{b}_2^0$$

$$\mathbf{b}_2^1(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_2^0 + \frac{\Delta_0}{\Delta}\mathbf{b}_3^0$$

$$\mathbf{b}_0^2(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_0^1 + \frac{\Delta_0}{\Delta}\mathbf{b}_1^1$$

$$\mathbf{b}_1^2(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_1^1 + \frac{\Delta_0}{\Delta}\mathbf{b}_2^1$$

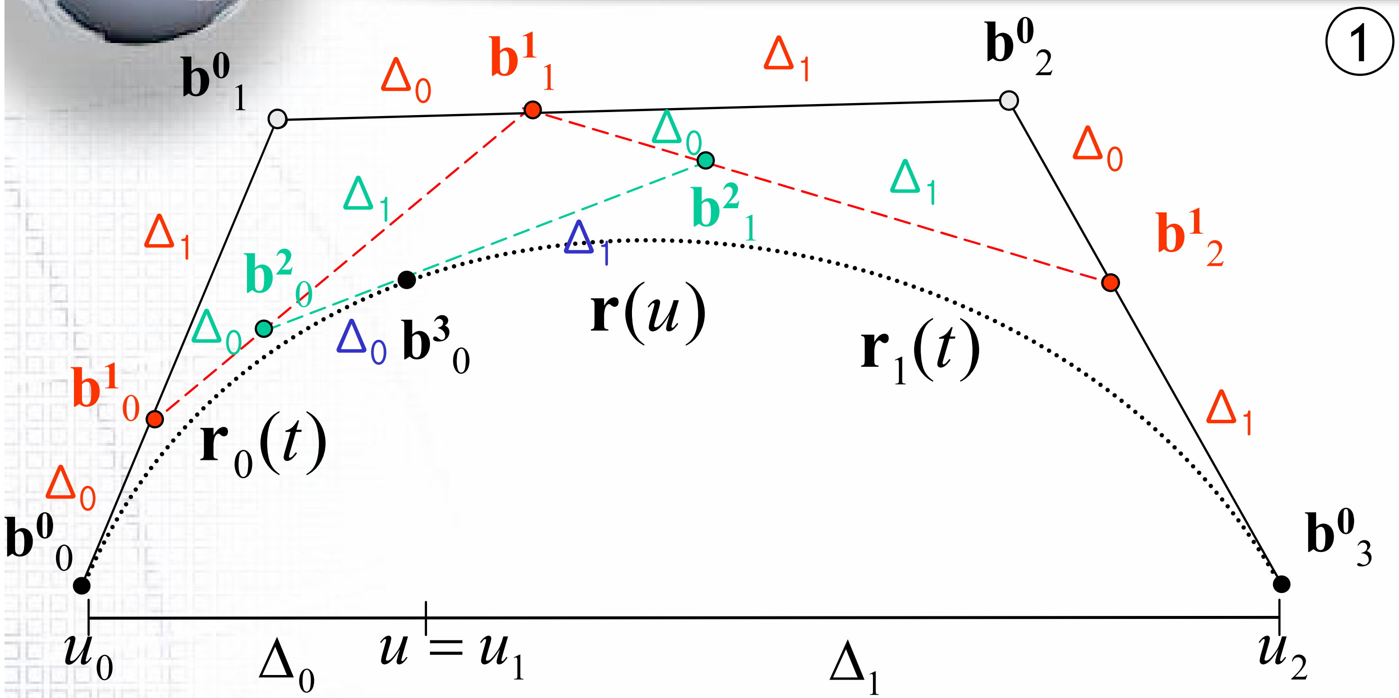
$$\mathbf{b}_0^3(u) = \frac{\Delta_1}{\Delta}\mathbf{b}_0^2 + \frac{\Delta_0}{\Delta}\mathbf{b}_1^2$$

Let $t = \dfrac{u - u_0}{u_2 - u_0}$,

3차 Bezier curves 와 동일한 함수식 !!!

$$\mathbf{b}_0^3(u) = \mathbf{r}(t) = (1-t)^3\mathbf{b}_0 + 3(1-t)^2 t\mathbf{b}_1 + 3(1-t)t^2\mathbf{b}_2 + t^3\mathbf{b}_3$$

2.2.3.5 de Casteljau Algorithm의 특성 (1)



$\Delta = u_2 - u_0$, $\Delta_1 = u_2 - u_1$, $\Delta_0 = u_1 - u_0$, $\Delta = \Delta_0 + \Delta_1$

$$\text{ratio}(b_0^2, b_0^3, b_1^2) = \frac{u - u_0}{u_2 - u} = \frac{\Delta_0}{\Delta_1}$$

1. Bezier 곡선의 $u = u_1$에서의 점 계산

2. 곡선 분할:

   $u = u_1$에서 2개의 Bezier 곡선으로 분할됨.

   즉, 왼쪽 곡선 $\mathbf{r}_0(t)$은 Bezier points $\mathbf{b}^0{}_0$, $\mathbf{b}^1{}_0$, $\mathbf{b}^2{}_0$, $\mathbf{b}^3{}_0$으로 이루어지고,
   오른쪽 곡선 $\mathbf{r}_1(t)$은 Bezier points $\mathbf{b}^3{}_0$, $\mathbf{b}^2{}_1$, $\mathbf{b}^1{}_2$, $\mathbf{b}^0{}_3$으로 이루어짐

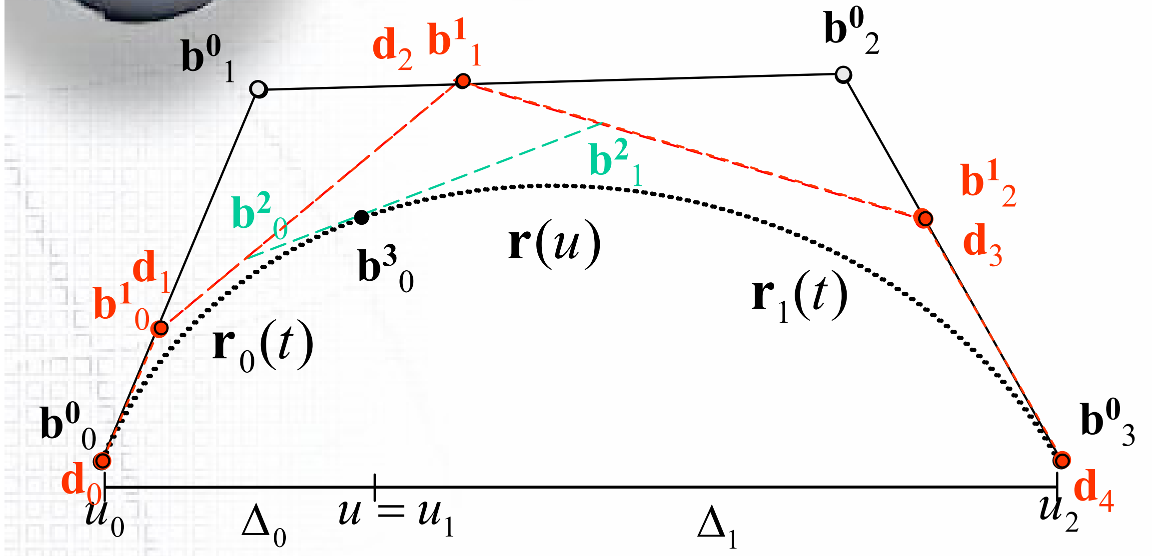
②

$$\mathbf{r}(t) = (1-t)^3\mathbf{b}_0^0 + 3(1-t)^2t\mathbf{b}_1^0 + 3(1-t)t^2\mathbf{b}_2^0 + t^3\mathbf{b}_3^0,\ t = \frac{u-u_0}{u_2-u_0}$$

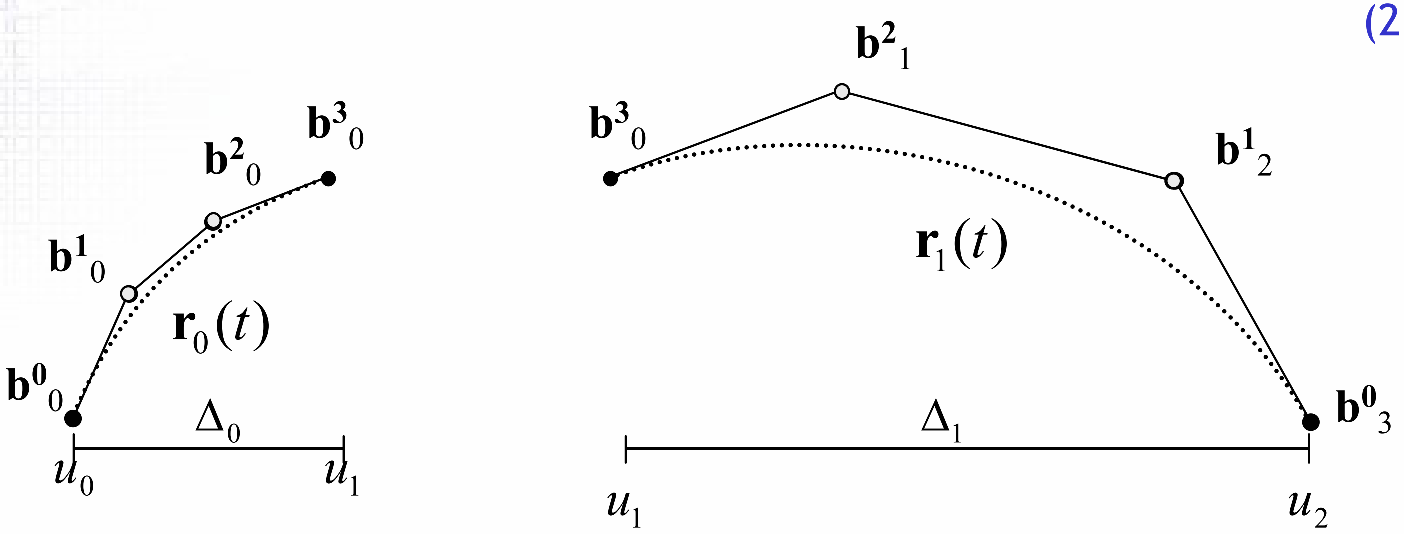
$$\mathbf{r}_0(t) = (1-t)^3\mathbf{b}_0^0 + 3(1-t)^2t\mathbf{b}_0^1 + 3(1-t)t^2\mathbf{b}_0^2 + t^3\mathbf{b}_0^3,\ t = \frac{u-u_0}{u_1-u_0}$$

$$\mathbf{r}_1(t) = (1-t)^3\mathbf{b}_0^3 + 3(1-t)^2t\mathbf{b}_1^2 + 3(1-t)t^2\mathbf{b}_2^1 + t^3\mathbf{b}_3^0,\ t = \frac{u-u_1}{u_2-u_1}$$

2.2.3.5 de Casteljau Algorithm의 특성 (2)



(1) 곡선 $\mathbf{r}(u)$은 떨어져 있는 두 개의
Bezier 곡선 $\mathbf{r}_0(t)$, $\mathbf{r}_1(t)$를 $u = u_1$에서
de Casteljau algorithm을 만족하도록
연결한 것으로 생각할 수 있음. 이 때
u_1은 곡선을 하나로 묶는 매듭이란
의미로 knot 라고 부름



(2) 다른 의미로 보면,
곡선 $\mathbf{r}(u)$은 떨어져 있는
두 개의 Bezier 곡선
$\mathbf{r}_0(t)$, $\mathbf{r}_1(t)$를 $u = u_1$에서
C^0, C^1, C^2, C^3 연속조건을
만족하도록 연결한
것으로 생각할 수 있다

$$\mathbf{r}_0(t) = (1-t)^3\mathbf{b}_0^0 + 3(1-t)^2 t\mathbf{b}_0^1 + 3(1-t)t^2\mathbf{b}_0^2 + t^3\mathbf{b}_0^3,$$

$$t = \frac{u-u_0}{u_1-u_0}$$

$$\mathbf{r}_1(t) = (1-t)^3\mathbf{b}_0^3 + 3(1-t)^2 t\mathbf{b}_1^2 + 3(1-t)t^2\mathbf{b}_2^1 + t^3\mathbf{b}_3^0,$$

$$t = \frac{u-u_1}{u_2-u_1}$$

2.2.3.6 Sample code of de Casteljau algorithm (1)

```
#ifndef __BezierCurve_h__
#define __BezierCurve_h__

#include “vector.h”

class BezierCurve {
public:
    int m_nDegree;
    Vector* m_ControlPoint;   int m_nControlPoint;
    BezierCurve();
    ~BezierCurve();

    void SetDegree(int nDegree);
    void SetControlPoint(Vector* pControlPoint, int nControlPoint);
    Vector CalcPoint(double t);
    Vector deCasteljau(double t);                    // CalcPoint by de Casteljau algorithm
    double B (int i, double t);
};
#endif
```

2.2.3.6 Sample code of de Casteljau algorithm (2)

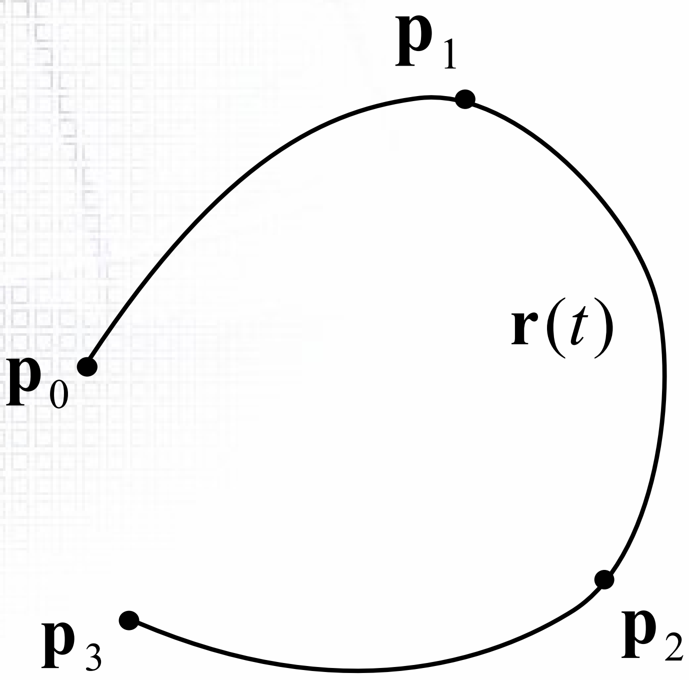
```
Vector BezierCurve:: deCasteljau (double t) {
    Vector* TmpControlPoint = new Vector [m_nControlPoint];
    for(int i = 0; i < m_nControlPoints; i++) TmpControlPoint[i] = m_ControlPoint[i];


    for(i = 1; i < m_nControlPoint; i++){
        for(int j = 0; j < m_nDegree – i; j++){
            TmpControlPoint[j] = (1-t)*TmpControlPoint[j]  + t*TmpControlPoint[j+1];
            //        b_j^i                          b_j^{i-1}                      b_{j+1}^{i-1}
        }
    }
    Vector result = TmpControlPoint[0];  // b_0^3
    delete[] TmpControlPoint;
   return result;
}
```

$$\begin{array}{llll}
\mathbf{b}_0^0 & \mathbf{b}_0^1 & \mathbf{b}_0^2 & \mathbf{b}_0^3 \\
\mathbf{b}_1^0 & \mathbf{b}_1^1 & \mathbf{b}_1^2 & \\
\mathbf{b}_2^0 & \mathbf{b}_2^1 & & \\
\mathbf{b}_3^0 & & &
\end{array}$$

2.2.4 Bezier Curve Interpolation / Approximation

2.2.4.1 Introduction to Curve Interpolation (1)

- If we are given fitting points $\mathbf{P}_i$ and we wish to pass a curve through them. There, the points are 2D, but the curve might as well be 3D. This is called "curve interpolation".



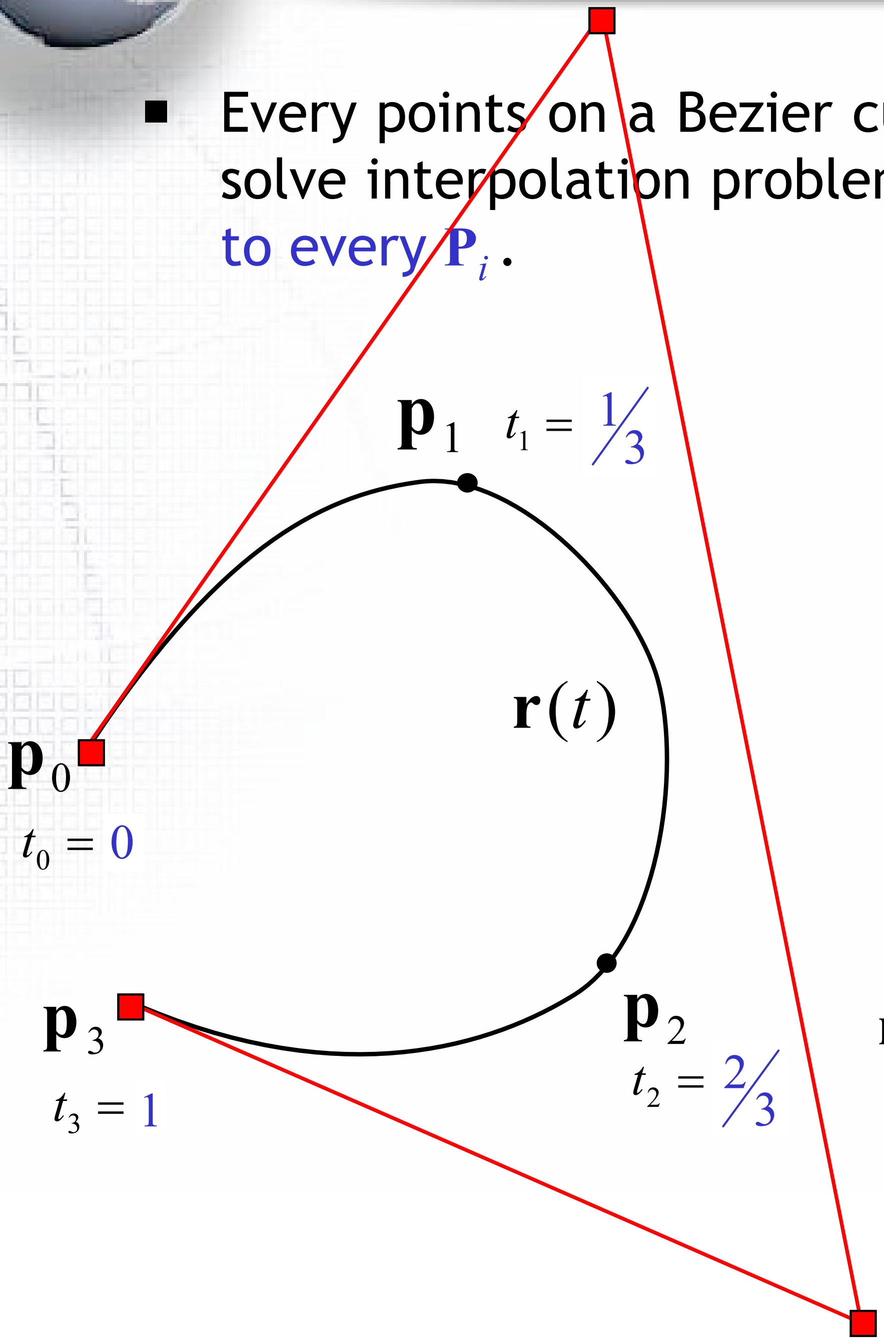
- We may choose among many kinds of curves; for right now, we will use a cubic Bezier curve.

→ "cubic Bezier curve interpolation"

$$\mathbf{r}(t) = (1-t)^3\mathbf{b}_0 + 3(1-t)^2 t\mathbf{b}_1 + 3(1-t)t^2\mathbf{b}_2 + t^3\mathbf{b}_3$$

2.2.4.1 Introduction to Curve Interpolation (2)

- Every points on a Bezier curve has a parameter value t; in order to solve interpolation problem, we have to assign a parameter value $\boldsymbol{t}_i$ to every $\mathbf{P}_i$.



$$0 = t_0 < t_1 < t_2 < t_3 = 1$$

- A natural choice is to associate each $\mathbf{P}_i$ with a uniform parameter $\boldsymbol{t}_i = i/3$.
- Then, we want a cubic Bezier curve such that:

$$\mathbf{r}(t_i) = \mathbf{p}_i; \quad i = 0,1,2,3.$$

$$\mathbf{r}(t) = (1-t)^3\mathbf{b}_0 + 3(1-t)^2 t\mathbf{b}_1 + 3(1-t)t^2\mathbf{b}_2 + t^3\mathbf{b}_3$$

2.2.4.2 Cubic Bezier curve interpolation (1)

- The cubic Bezier curve of the form:

$$\mathbf{r}(t) = B_0^3(t)\mathbf{b}_0 + B_1^3(t)\mathbf{b}_1 + B_2^3(t)\mathbf{b}_2 + B_3^3(t)\mathbf{b}_3.$$

- All interpolation conditions are:

$$\begin{aligned}
\mathbf{p}_0 &= B_0^3(t_0)\boxed{\mathbf{b}_0} + B_1^3(t_0)\boxed{\mathbf{b}_1} + B_2^3(t_0)\boxed{\mathbf{b}_2} + B_3^3(t_0)\boxed{\mathbf{b}_3}, \\
\mathbf{p}_1 &= B_0^3(t_1)\boxed{\mathbf{b}_0} + B_1^3(t_1)\boxed{\mathbf{b}_1} + B_2^3(t_1)\boxed{\mathbf{b}_2} + B_3^3(t_1)\boxed{\mathbf{b}_3}, \\
\mathbf{p}_2 &= B_0^3(t_2)\boxed{\mathbf{b}_0} + B_1^3(t_2)\boxed{\mathbf{b}_1} + B_2^3(t_2)\boxed{\mathbf{b}_2} + B_3^3(t_2)\boxed{\mathbf{b}_3}, \\
\mathbf{p}_3 &= B_0^3(t_3)\boxed{\mathbf{b}_0} + B_1^3(t_3)\boxed{\mathbf{b}_1} + B_2^3(t_3)\boxed{\mathbf{b}_2} + B_3^3(t_3)\boxed{\mathbf{b}_3}.
\end{aligned}$$

4 Unknown Vectors, 4 Vector Equations

2.2.4.2 Cubic Bezier curve interpolation (2)

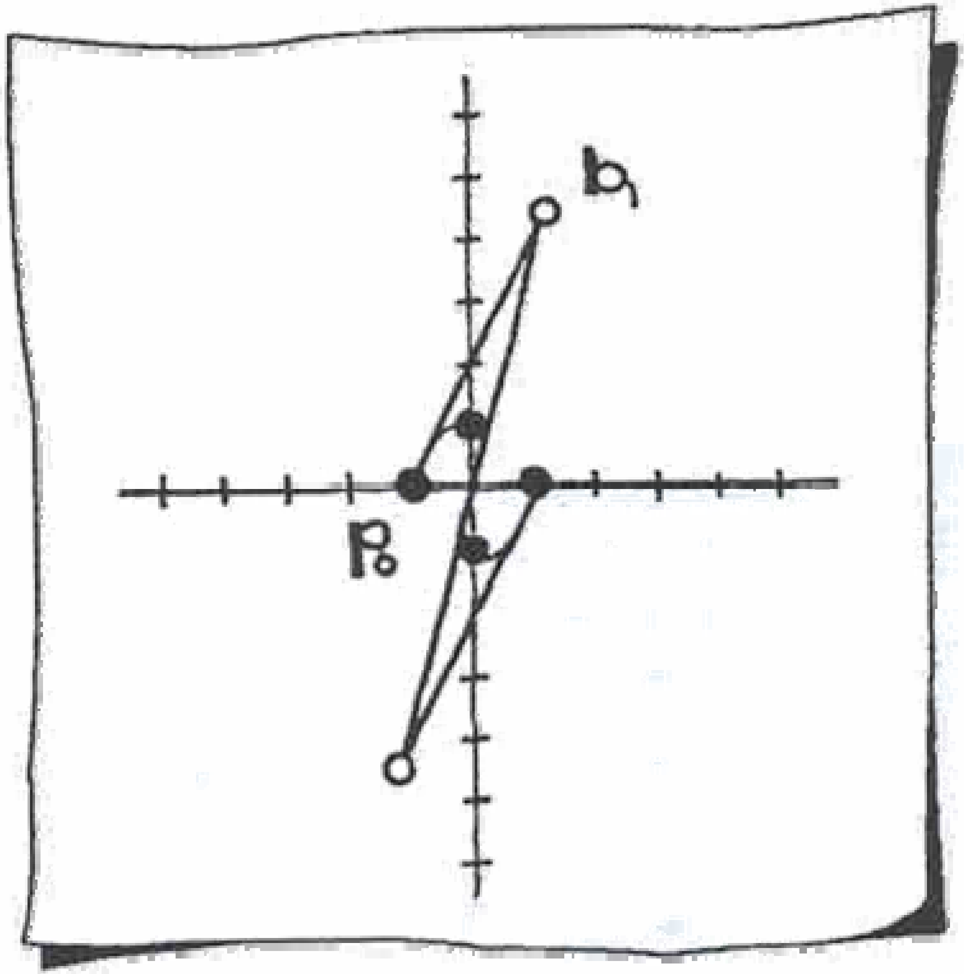
- To find the solution of these four equations for four unknowns, we can write in matrix form:

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = \begin{bmatrix} B_0^3(t_0) & B_1^3(t_0) & B_2^3(t_0) & B_3^3(t_0) \\ B_0^3(t_1) & B_1^3(t_1) & B_2^3(t_1) & B_3^3(t_1) \\ B_0^3(t_2) & B_1^3(t_2) & B_2^3(t_2) & B_3^3(t_2) \\ B_0^3(t_3) & B_1^3(t_3) & B_2^3(t_3) & B_3^3(t_3) \end{bmatrix} \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}.$$

- To abbreviate the above form as: $\mathbf{P} = \mathbf{MB}$.
- The solution is: $\mathbf{B} = \mathbf{M}^{-1}\mathbf{P}$.
- Although it is looks like the solution to one linear system but it is the two or three systems depending on the dimensionality of the $\mathbf{p}_i$.

$$ex)\ \mathbf{p}_0 = [x_0 \quad y_0]^T \ \ or \ \ [x_0 \quad y_0 \quad z_0]^T$$

2.2.4.2 Cubic Bezier curve interpolation (3)



Cubic Bezier interpolation.

2.2.4.3 Bezier curve interpolation beyond Cubics (1)

- Polynomial interpolation can also works for more than four data points.
- Given: points $\mathbf{p}_0, \cdots, \mathbf{p}_m$ and
  corresponding parameter values $0 = t_0 < t_1 < \ldots < t_{m-1} < t_m = 1.$
- If we choose a Bezier curve of degree n for interpolation,
  we have "$m+1$ vector equations" for "$n+1$ unknown vectors".
- $n > m$: underdetermined system,
  We need *additional conditions* to solve the interpolation problem
- $n = m$: determinate linear system → "Interpolation problem"
- $n < m$: overdetermined system → "Approximation problem"

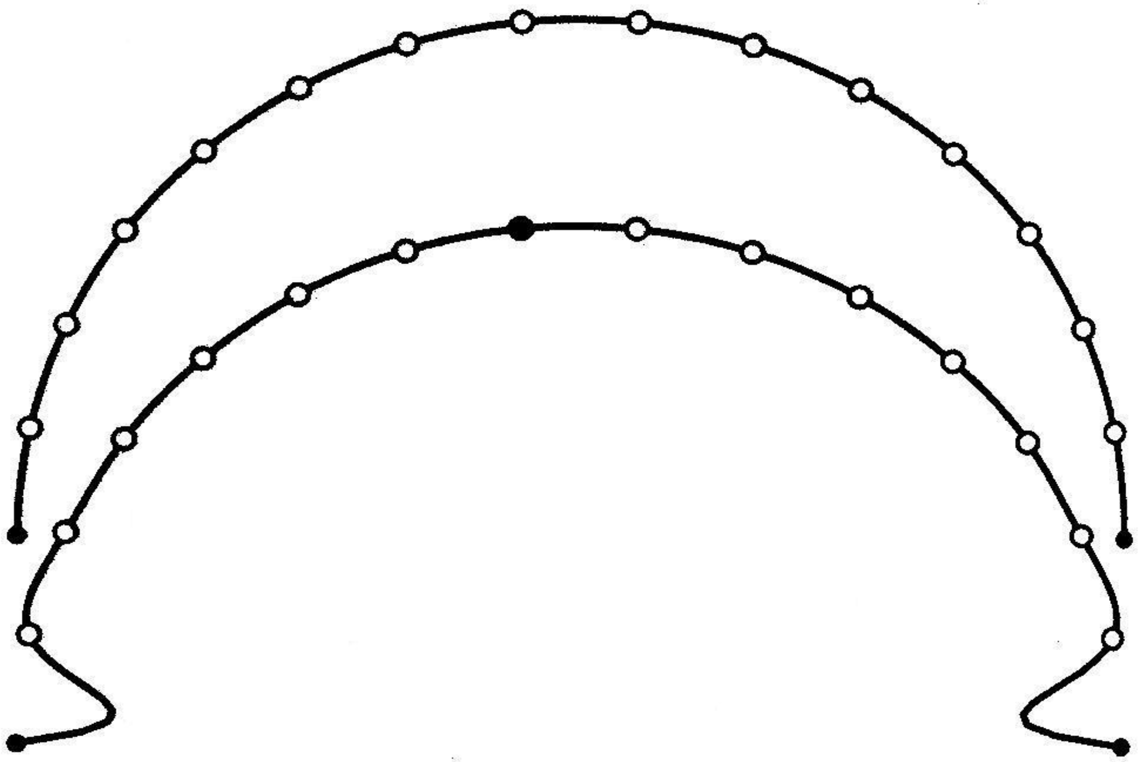
2.2.4.3 Bezier curve interpolation beyond Cubics (2)

- Given: points $\mathbf{p}_0,\ldots\mathbf{p}_m$ and
  corresponding parameter values $0=t_0<t_1<\ldots<t_{m-1}<t_m=1.$
- If we use a Bezier curve of degree n $(=m)$,
  we have a linear system: $\mathbf{P}=\mathbf{MB}.$
- $\mathbf{M}$ is an $(m+1)\times(m+1)$ matrix with elements;

$$e_{ij}=B_j^m(t_i)$$

- It can be solved with any linear solver.
- Polynomial interpolation does not provide satisfied result for higher degrees. Figure in the next slide should be convincing enough.

2.2.4.3 Bezier curve interpolation beyond Cubics (3)



Top: Data from a circle; Bottom: one point slightly modified.

- The processes of a small change in data can lead large change in the interpolating curve is called ill-conditioned.
- Different polynomial forms will give the identical result.

2.2.4.4 Bezier curve approximation (1)

- One is given more data points than should be interpolated by a polynomial curve (i.e. number of data points more than degree of curve)

  → We can solve the problem
  by interpolating with a higher degree Bezier curve,
  but higher degree interpolation becomes ill-conditioned.

- In such cases, an approximating curve will be needed, which does not pass through the data points exactly; rather it passes near them.
  - the best technique to find such curves
  - →'least squares approximation'.

2.2.4.4 Bezier curve approximation (2)

- Given: points $\mathbf{p}_0,\ldots\mathbf{p}_m$ and
  corresponding parameter values $0 = t_0 < t_1 < \ldots < t_{m-1} < t_m = 1.$

- We wish to find a polynomial curve $r(t)$ of a given degree $n\ (< m)$ such that

$$\left\| \mathbf{p}_i - \mathbf{r}(t_i) \right\| \rightarrow minimize \quad (or) \quad \mathbf{p}_i = \mathbf{r}(t_i); \quad i = 0,1,\ldots\ldots,m$$

- Polynomial curve is of the Bezier form:

$$\mathbf{r}(t) = \mathbf{b}_0 B_0^n(t) + \mathbf{b}_1 B_1^n(t) + \ldots\ldots\ldots + \mathbf{b}_n B_n^n(t).$$

2.2.4.4 Bezier curve approximation (3)

- We would like the following to hold:

$$\begin{aligned}
\mathbf{p}_0 &= \mathbf{b}_0 B_0^n(t_0) + \dots\dots\dots + \mathbf{b}_n B_n^n(t_0) \\
\mathbf{p}_1 &= \mathbf{b}_0 B_0^n(t_1) + \dots\dots\dots + \mathbf{b}_n B_n^n(t_1) \\
\vdots \quad & \quad \vdots \qquad\qquad\qquad\quad \vdots \\
\mathbf{p}_m &= \mathbf{b}_0 B_0^n(t_m) + \dots\dots\dots + \mathbf{b}_n B_n^n(t_m)
\end{aligned}
\quad \Longrightarrow \quad
\begin{bmatrix} B_0^n(t_0) & \cdots & B_n^n(t_0) \\ \vdots & & \vdots \\ B_0^n(t_m) & & B_n^n(t_m) \end{bmatrix}
\begin{bmatrix} \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_n \end{bmatrix}
=
\begin{bmatrix} \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_m \end{bmatrix}$$

$$\mathbf{MB} = \mathbf{P}$$

<u>$(m+1)*(2 \text{ or } 3)$ Unknowns $<$ $(n+1)*(2 \text{ or } 3)$ Equations</u>

2.2.4.4 Bezier curve approximation (4)

- Multiply both sides by :$\mathbf{M}^T$

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

← Normal equation

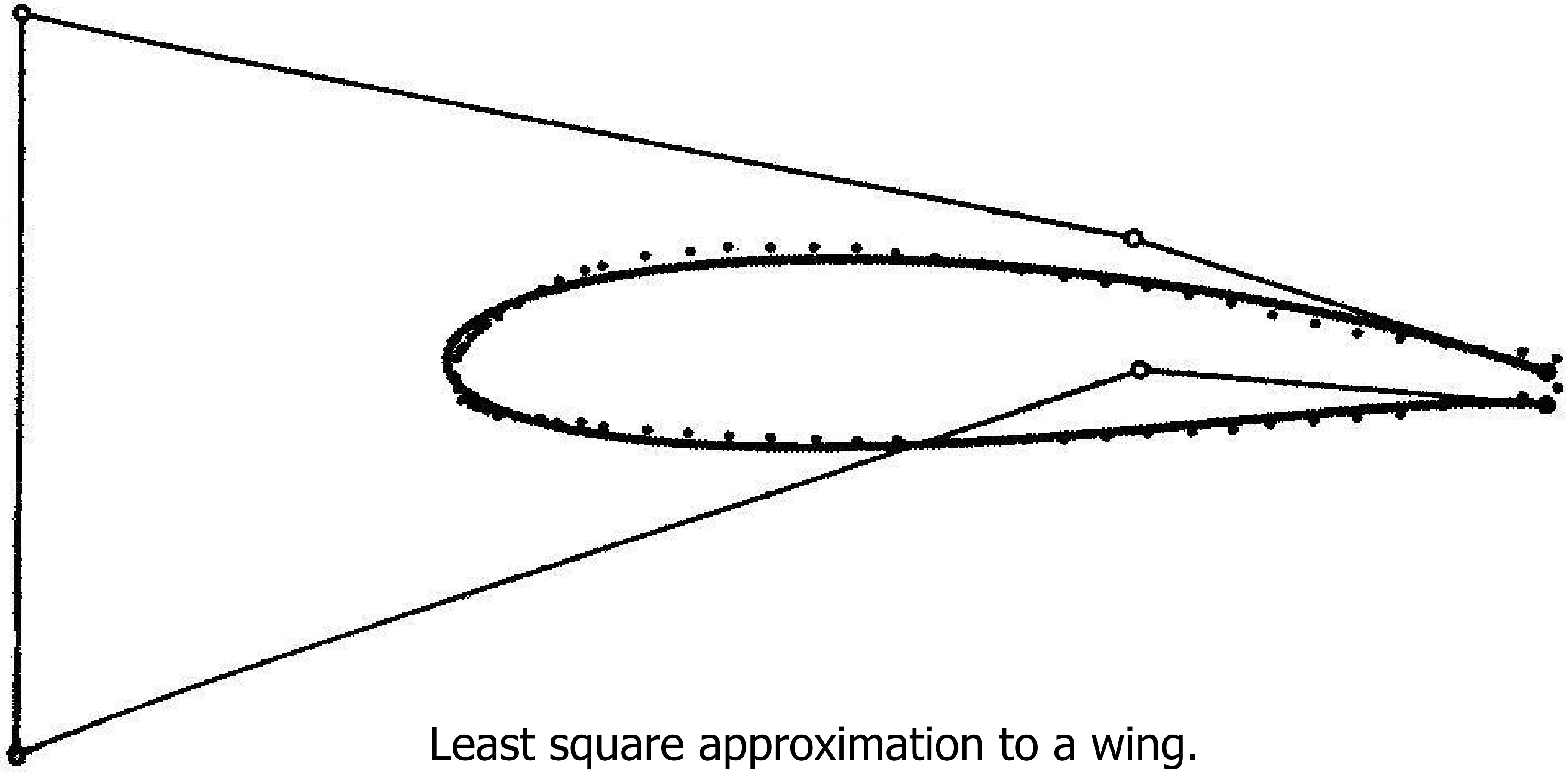
where $\mathbf{M}^T\mathbf{M}$ is a square and symmetric matrix, which is always invertible.

- The curve $\mathbf{B}$ minimizes the sum of the $\left\|\mathbf{p}_i - \mathbf{r}(t_i)\right\|,\quad i = 0,1,\ldots..,m$

$$\therefore\ \mathbf{B} = \left(\mathbf{M}^T\mathbf{M}\right)^{-1}\mathbf{M}^T\mathbf{P}.$$

note that any modification of the t_i would result in an entirely different solution.

2.2.4.4 Bezier curve approximation (5)



Least square approximation to a wing.
A quintic Bezier curve with chord length parameters assigned to the data.

2.2.4.5 Finding the right parameters (1)

- In both interpolation & approximation curve,
  in practice, the parameter value t_i are not normally given,
  and have to be made up.

- There are two types to be made up:

(1) Uniform sets of parameters;

  - If there are $(m+1)$ points $\mathbf{p}_i$,
  - then set $t_i = i/l$.

(2) chord length parameters;

  - if the distance between two points is relatively large, then their parameter values should also be fairly different.

$$t_0 = 0$$

$$t_1 = t_0 + \|\mathbf{p}_1 - \mathbf{p}_0\|$$

$$\vdots$$

$$t_l = t_{l-1} + \|\mathbf{p}_l - \mathbf{p}_{l-1}\|$$

2.2.4.5 Finding the right parameters (2)

- If desired (it makes no difference to the interpolation or approximation result), the parameters may be normalized by scaling the parameters to live between zero and one:

$$t_i = \frac{t_i - t_0}{t_m - t_0}.$$

- In general, chord length parameterization method is superior to the uniform method, because it takes into account the geometry of the data.

2.2.4.6 Sample code of Interpolation/Approximation (1)

```
#include “vector.h”

class BezierCurve {
public:
    int m_nDegree;
    Vector* m_ControlPoint;   int m_nControlPoint;
    ……

    void SetDegree(int nDegree);
    void SetControlPoint(Vector* pControlPoint, int nControlPoint);
    Vector CalcPoint(double t);
    double B (int i, double t);
    int Approximation(int nDegree, int nType, Vector* FittingPoint, int nPoint);
    int Interpolation(int nType, Vector* FittingPoint, int nPoint);
    void Parameterization(int nType, Vector* FittingPoint, int nPoint, double* t);
};
```

2.2.4.6 Sample code of Interpolation/Approximation (2)

```
void BezierCurve:: Parameterization (int nType, Vector* FittingPoint, int nPoint, double* t){
    // assume t is allocated out of function
    if( nType == 1) {   // Uniform Set
       for (int i = 0; i < nPoint; i++)
                         t[i] = 1./(nPoint-1);
    } else if ( nType == 2) { // Chord length
       t[0] = 0.;
       for (int i=0; i < nPoint-1; i++)
           t[i+1] = t[i] + (FittingPoint[i+1] – FittingPoint[i]).Magnitude();
       double t0 = t[0], tm = t[nPoint-1];
       for (int i=0; i < nPoint; i++)
           t[i] = (t[i] -  t0)/(tm – t0);      // Normalize
    }
}
```

2.2.4.6 Sample code of Interpolation/Approximation (3)

```
int BezierCurve:: Approximation(int nDegree, int nType, Vector* FittingPoint, int nPoint){
    m_nDegree = nDegree;
    m_nControlPoint = m_nDegree+1;
    if(m_ControlPoint) = delete[] m_ControlPoint;
    m_ControlPoint = new Vector[m_nControlPoint];

    double* t = new double[nPoint];
    Parameterization(nType, FittingPoint, nPoint, t);

    // Solve normal equation
    ....
   delete[] t;
}
```

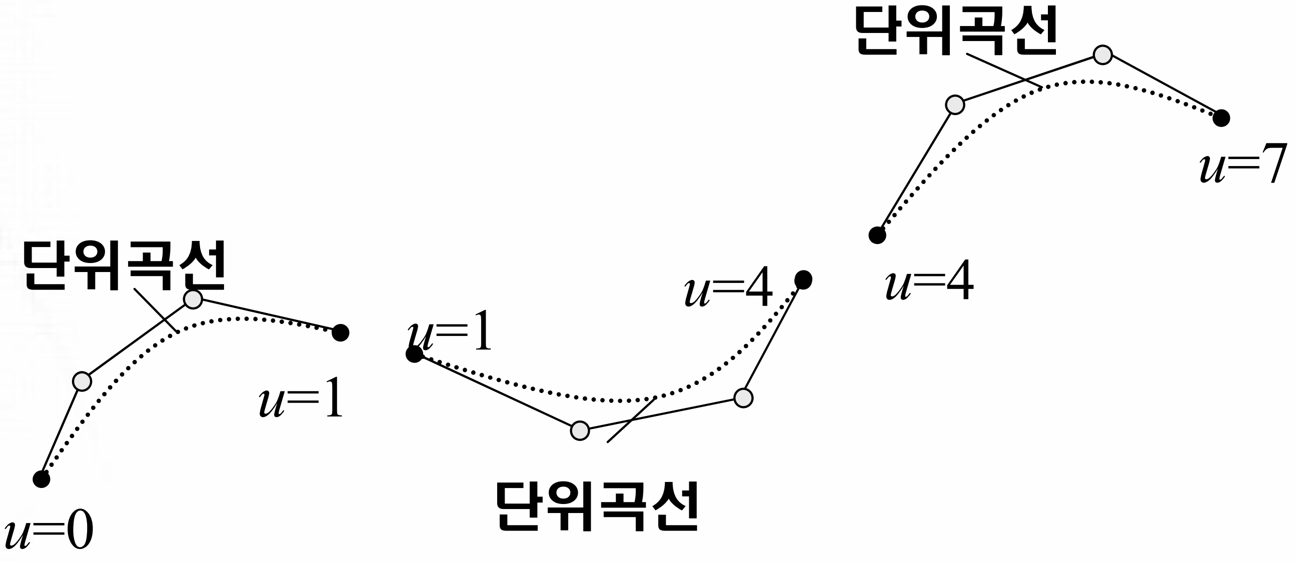
2.2.4.6 Sample code of Interpolation/Approximation (4)

```
int BezierCurve:: Interpolation(int nType, Vector* FittingPoint, int nPoint){
    m_nControlPoint = nPoint;
    m_nDegree = m_nControlPoint-1;
    if(m_ControlPoint) = delete[] m_ControlPoint;
    m_ControlPoint = new Vector[m_nControlPoint];

    double* t = new double[nPoint];
    Parameterization(nType, FittingPoint, nPoint, t);

    // Solve MB = P
    ....
    delete[] t;
}
```

2.3 B[asis]-spline curves

2.3.1 Definition of B-spline curves

2.3.1.1 Knot & Spline curves



놋트 = {…, 0, 1, 4, 7, …}

- **단위 곡선들을 "부드럽게" 연결한 곡선:** Spline curve
- **단위 곡선을 묶는 매듭 :** **놋트**(knot)

2.3.1.2 Definition of B-spline curves

- Ex): Cubic B-spline curves

Given: $\mathbf{d}_i$, u_j

$$\mathbf{r}(u)=\mathbf{d}_0N_0^3(u)+\mathbf{d}_1N_1^3(u)+\mathbf{d}_2N_2^3(u)+\cdots+\mathbf{d}_{D-1}N_{D-1}^3(u)$$

$\mathbf{d}_i$: de Boor points (control points), $i=0,1,\ldots,D-1$

$N_i^n(u)$: B-splines basis function of degree $n(=3)$

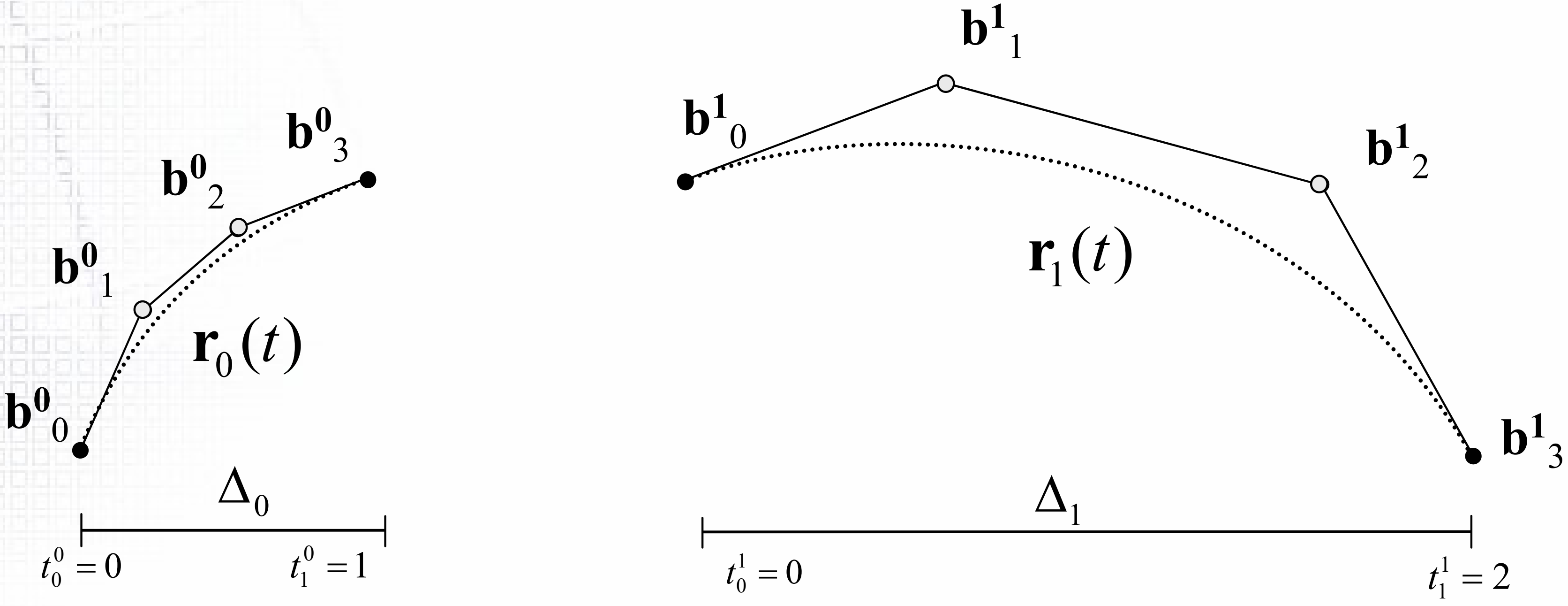
u_j : knots, $j=0,1,\ldots,K-1$, where $K=D+n+1$

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

$$N_i^0(u)=\begin{cases}1 & \textbf{if} \;\; u_{i-1}\le u<u_i \\ 0 & \textbf{else}\end{cases} \quad , \sum_{i=0}^{D-1}N_i^n(u)=1$$

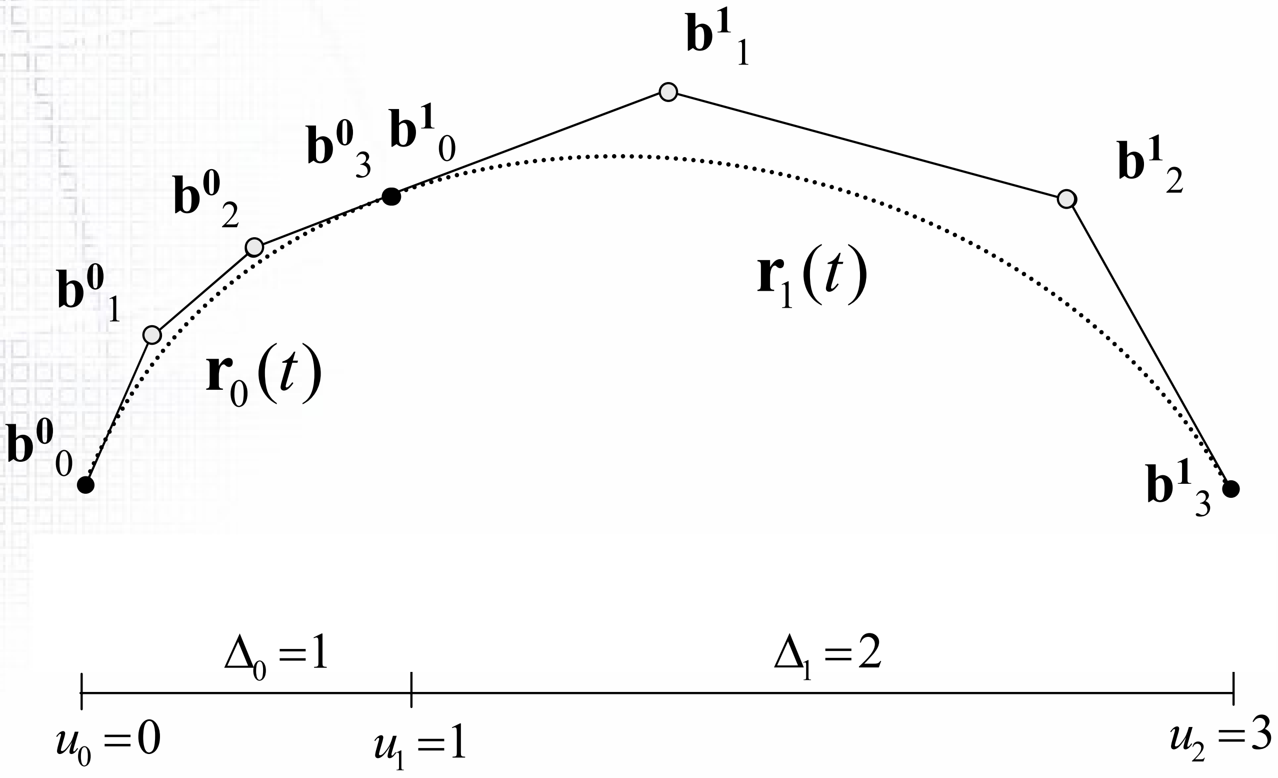
2.3.1.3 Geometric meanings of cubic B-spline curve (1)

- 'Cubic' B-spline curve consist of 'cubic' Bezier curves, which are connected with the C^2 continuity condition ▶



2.3.1.3 Geometric meanings of cubic B-spline curve (1)

- 'Cubic' B-spline curve consist of 'cubic' Bezier curves, which are connected with the C^2 continuity condition



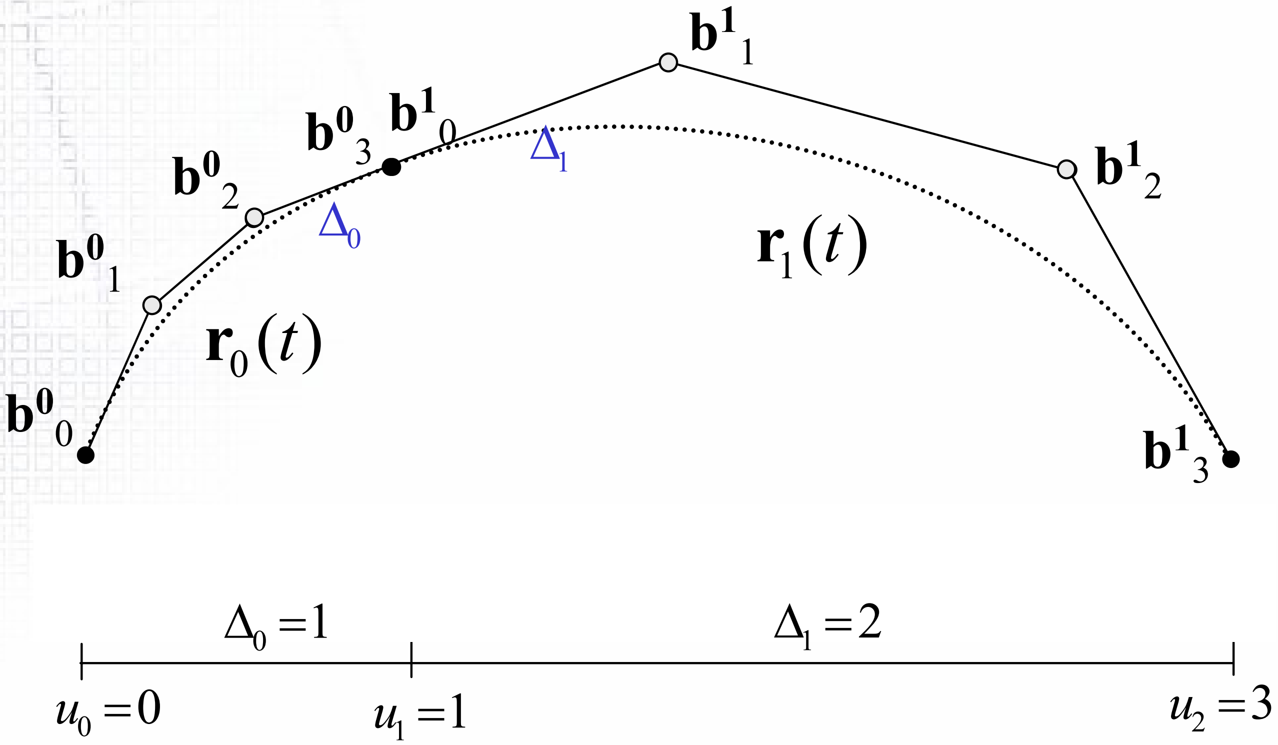
Assign new global parameter u to jointed curve

C^0 continuity condition

$$\mathbf{b}_3^0 = \mathbf{b}_0^1$$

2.3.1.3 Geometric meanings of cubic B-spline curve (1)

- ‘Cubic’ B-spline curve consist of ‘cubic’ Bezier curves, which are connected with the C^2 continuity condition



Assign new global parameter u to jointed curve

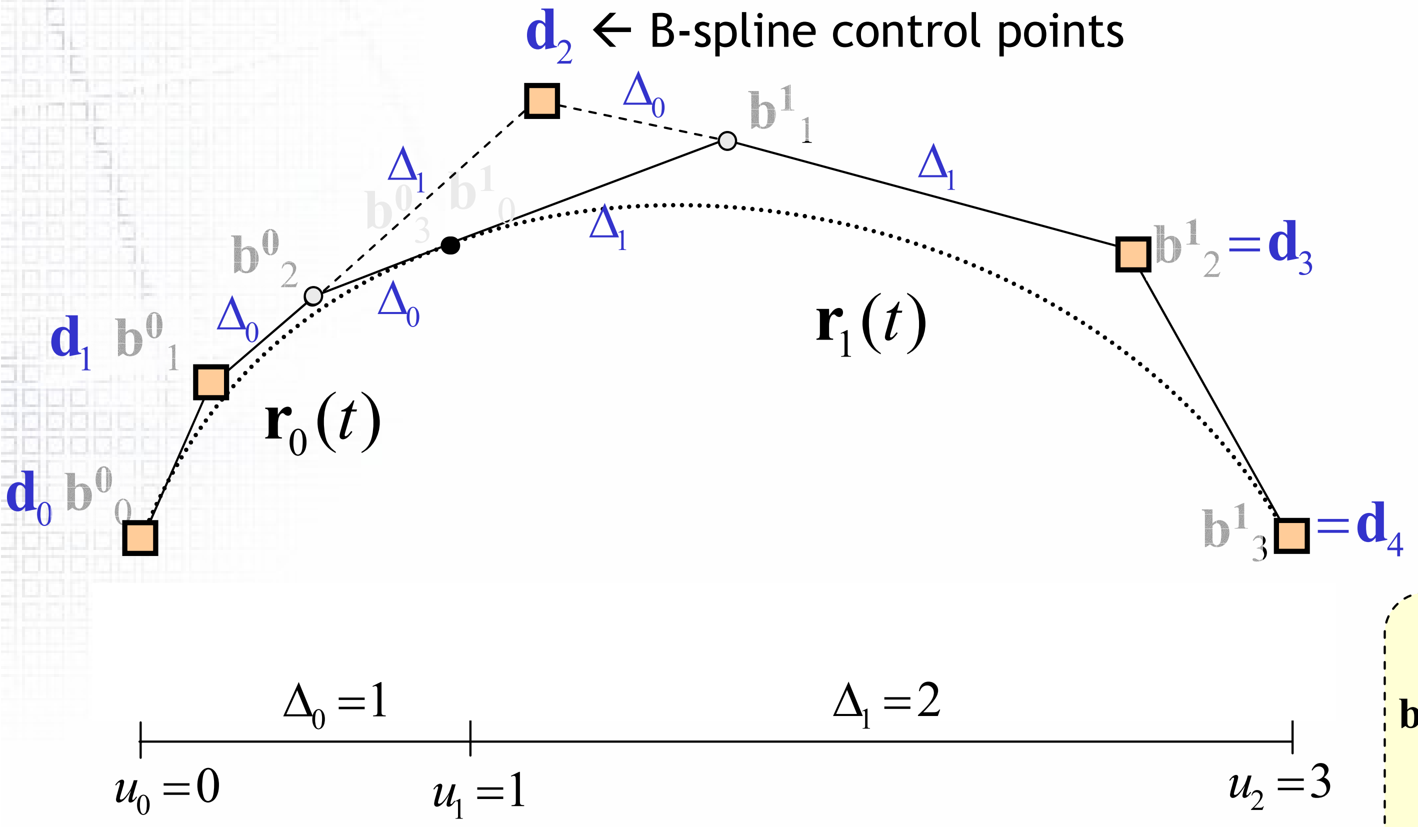
C^1 continuity condition

$$\mathbf{b}^0_3 = \mathbf{b}^1_0$$

$$\mathbf{b}^0_3 = \mathbf{b}^1_0 = \frac{\Delta_1}{\Delta_0 + \Delta_1}\mathbf{b}^0_2 + \frac{\Delta_0}{\Delta_0 + \Delta_1}\mathbf{b}^1_1$$

2.3.1.3 Geometric meanings of cubic B-spline curve (1)

- 'Cubic' B-spline curve consist of 'cubic' Bezier curves, which are connected with the C^2 continuity condition



Assign new global parameter u to jointed curve

C^2 continuity condition

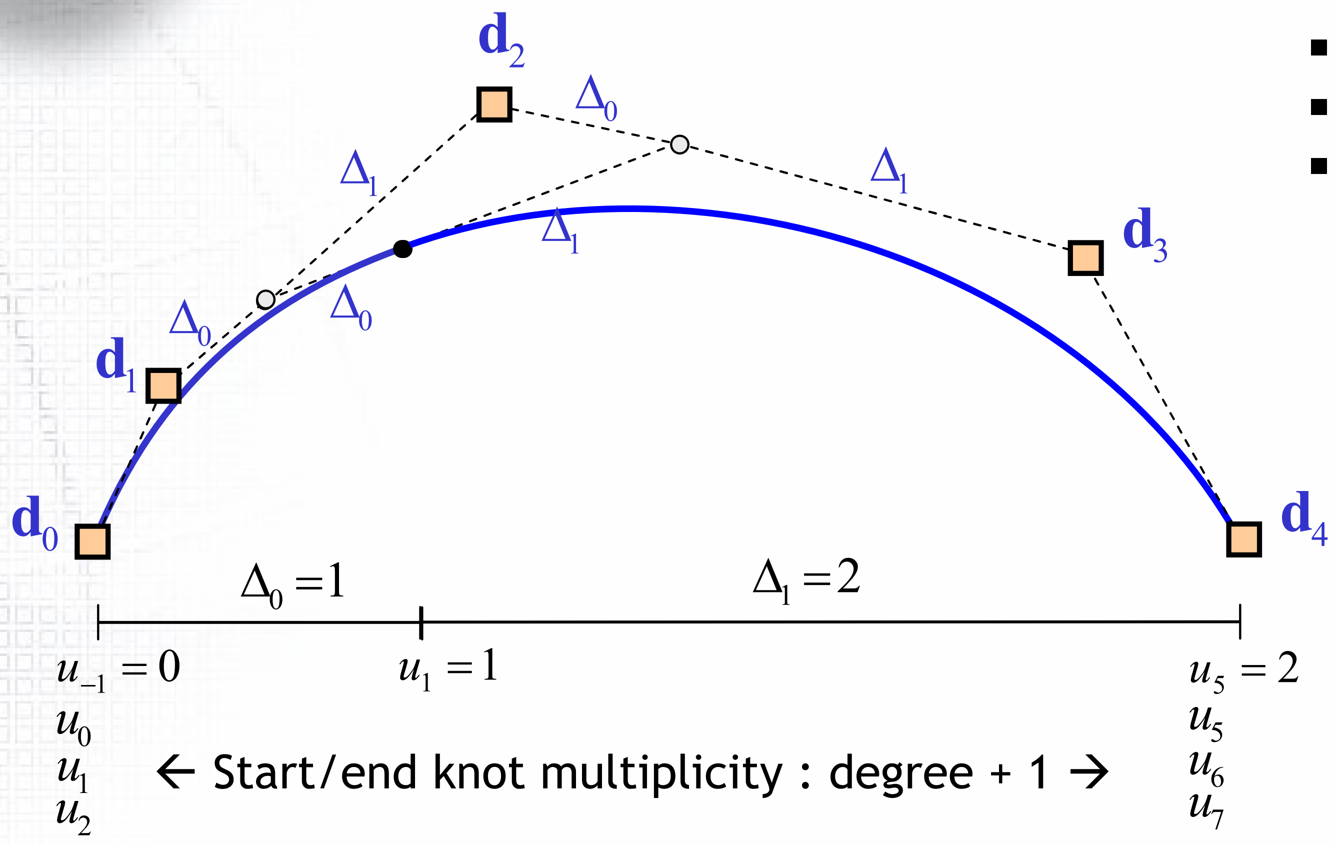
$$\mathbf{b}_3^0 = \mathbf{b}_0^1$$

$$\mathbf{b}_3^0 = \mathbf{b}_0^1 = \frac{\Delta_1}{\Delta_0 + \Delta_1}\mathbf{b}_2^0 + \frac{\Delta_0}{\Delta_0 + \Delta_1}\mathbf{b}_1^1$$

$$\mathbf{b}_2^0 = \{\frac{\Delta_1}{\Delta_0 + \Delta_1}\}\mathbf{d}_1 + \{\frac{\Delta_0}{\Delta_0 + \Delta_1}\}\mathbf{d}_2$$

$$\mathbf{b}_1^1 = \{\frac{\Delta_1}{\Delta_0 + \Delta_1}\}\mathbf{d}_2 + \{\frac{\Delta_0}{\Delta_0 + \Delta_1}\}\mathbf{d}_3$$

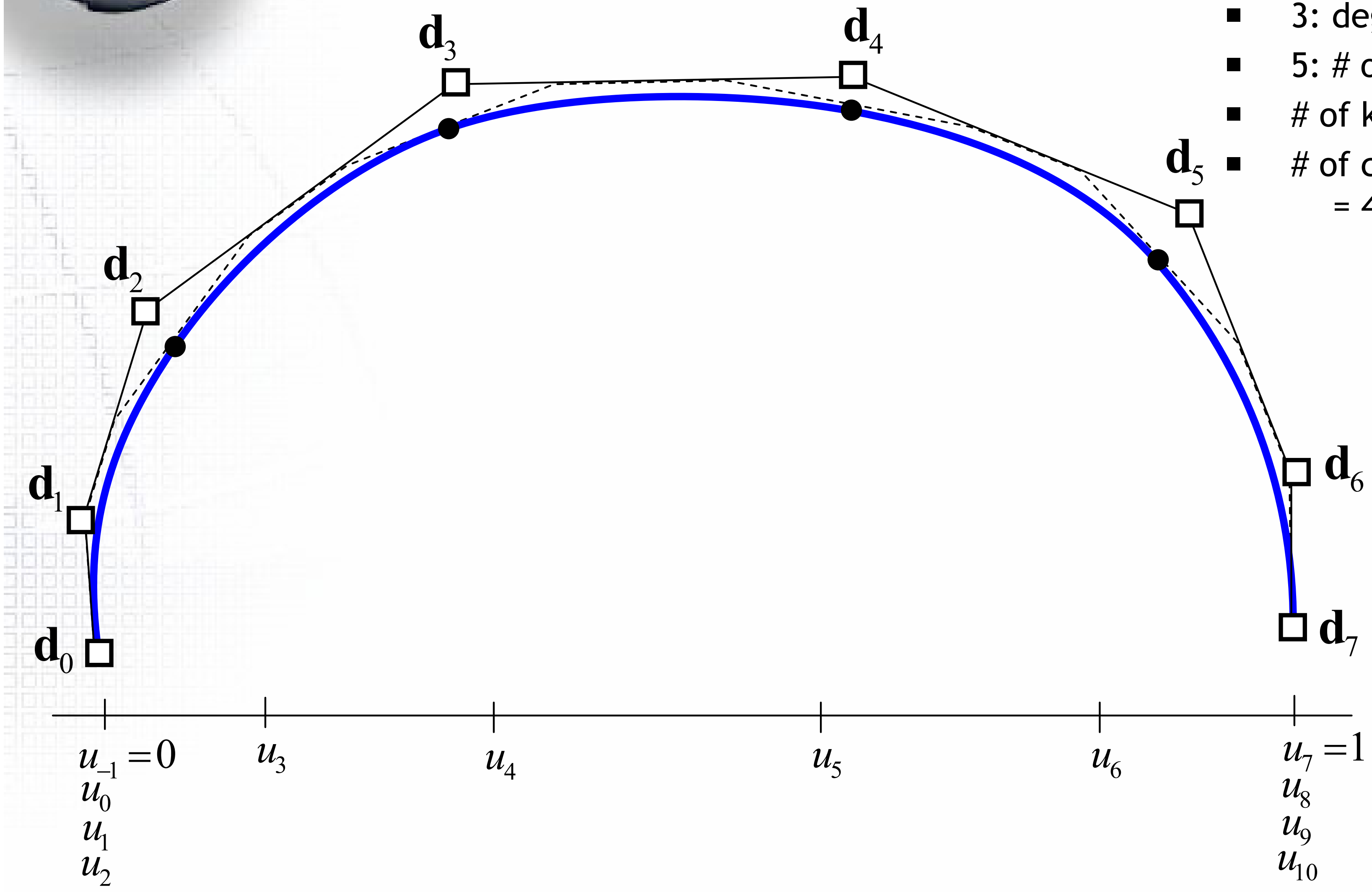
2.3.1.3 Geometric meanings of cubic B-spline curve (2)



- n: degree
- S: # of Bezier curve segments
- # of knot = (S-1) + 2(n+1)
- # of control points
  = 4 + (S-1) = (n+1) + (S-1)

$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u) + \mathbf{d}_4 N_4^3(u)$$

2.3.1.3 Geometric meanings of cubic B-spline curve (3)



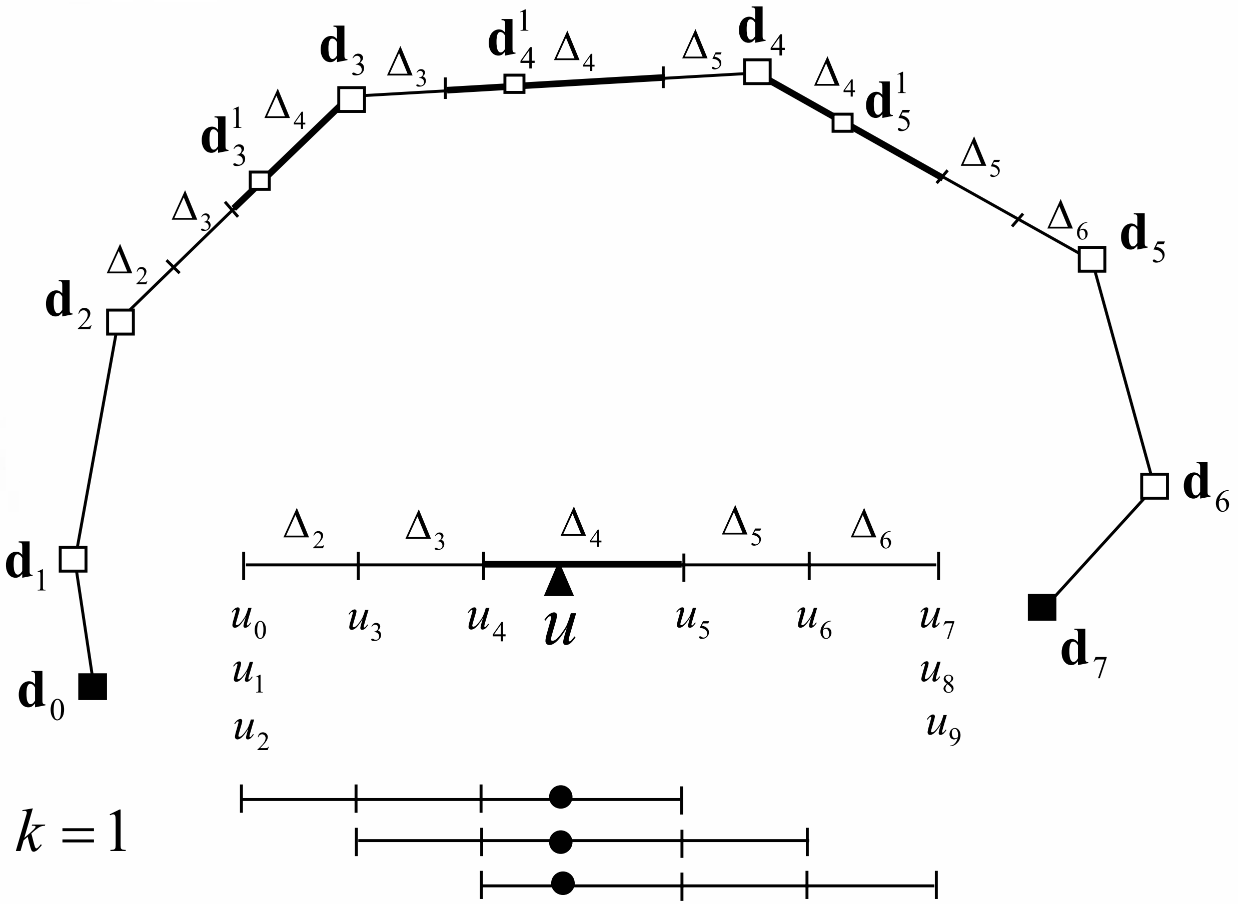
- 3: degree
- 5: # of Bezier curve segments
- # of knot = (5-1) + 2(3+1)
- # of control points
  = 4 + (5-1) = (3+1) + (5-1)

$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u) +$$

$$\mathbf{d}_4 N_4^3(u) + \mathbf{d}_5 N_5^3(u) + \mathbf{d}_6 N_6^3(u) + \mathbf{d}_7 N_7^3(u)$$

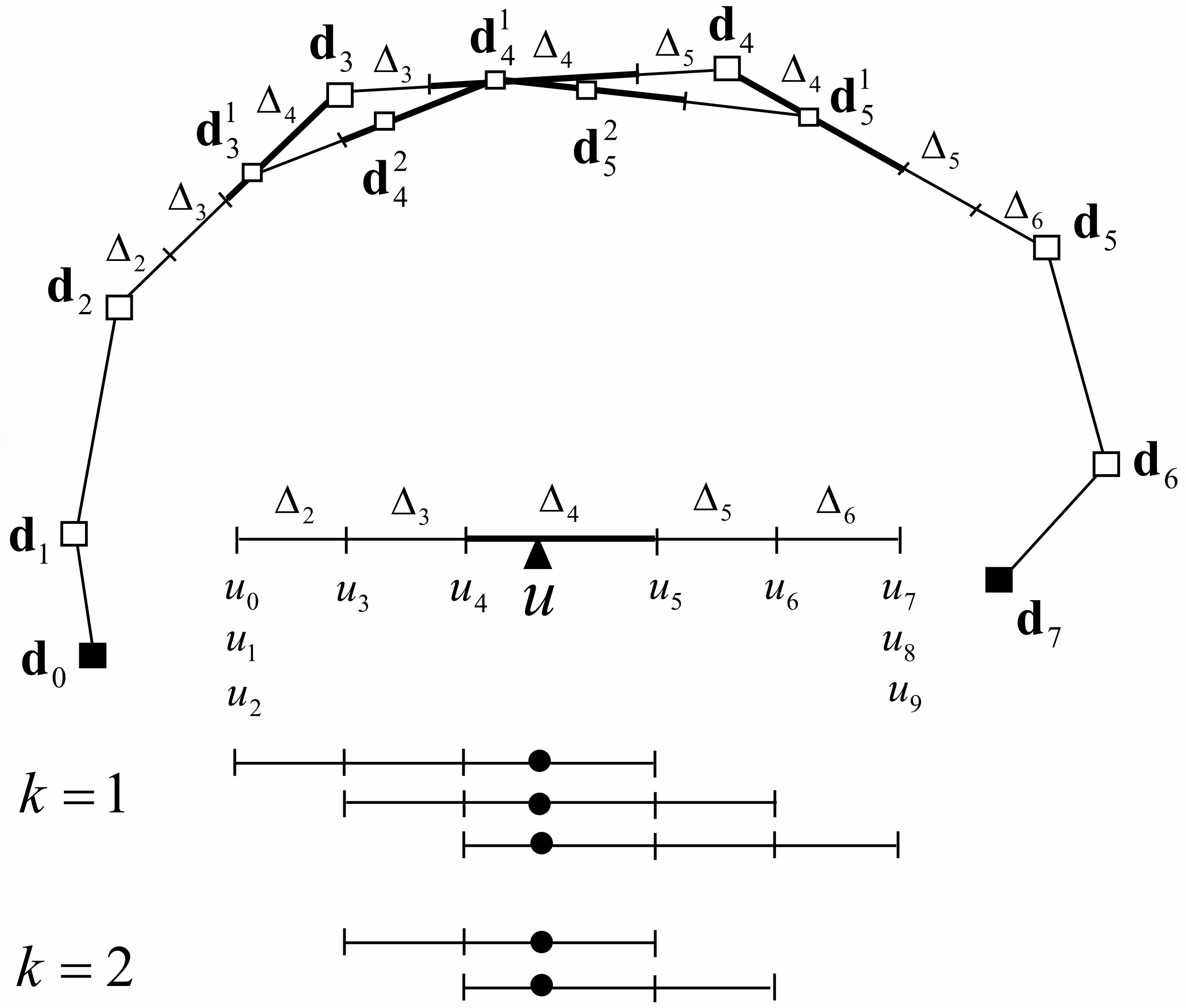
2.3.2 de Boor algorithm

2.3.2.1 de Boor Algorithm (1)



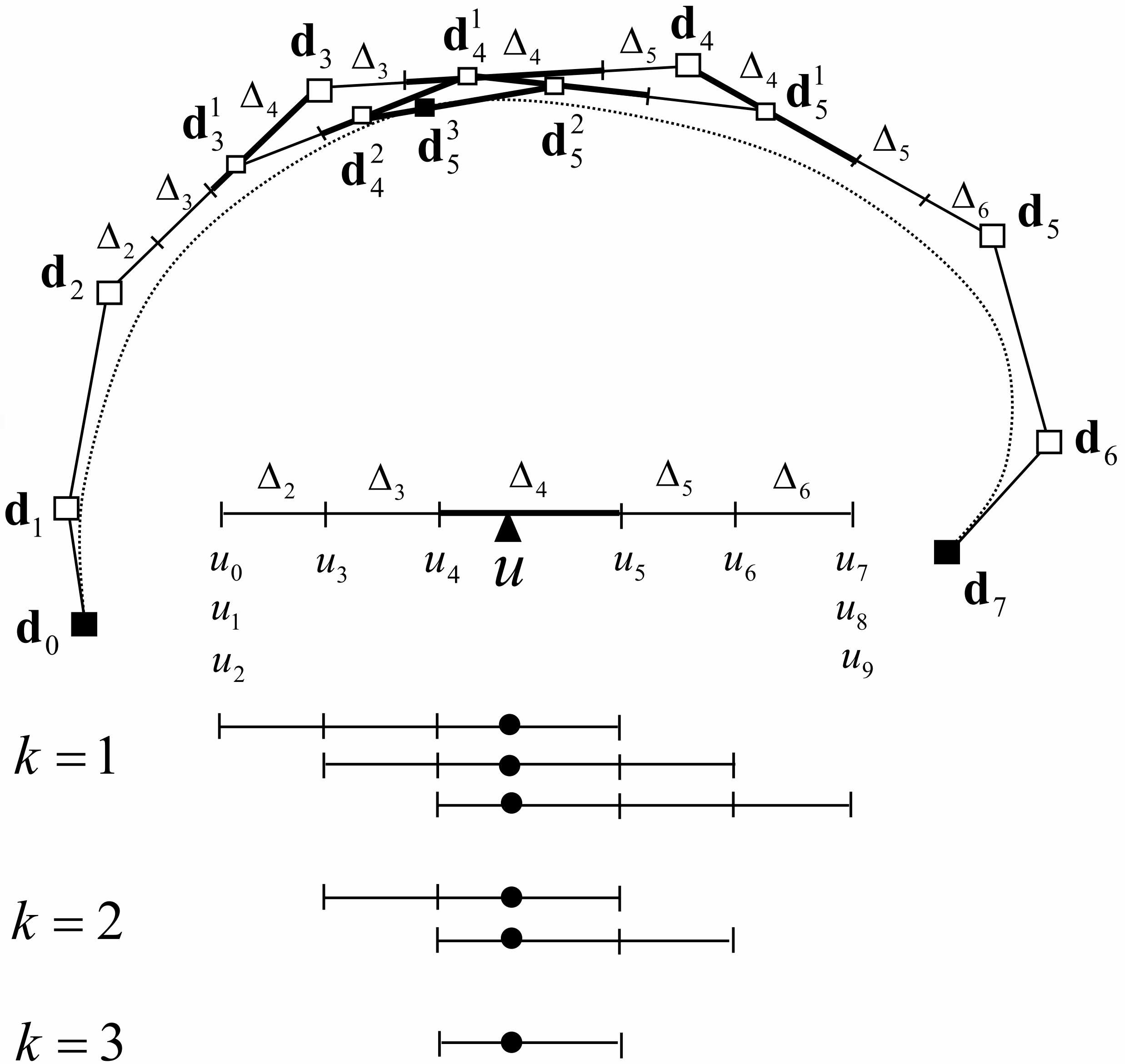
- Linear Interpolation **비율이** t:(1-t)**로 일정했던** de Casteljau algorithm**에 비하여** de Boor algorithm**에서는** Linear Interpolation **비율이 변한다**
- **이는** B-spline curve **를 구성하는** Bezier curve segment**의 매개변수 간격이 서로 다르기 때문이다**

2.3.2.1 de Boor Algorithm (2)



$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \cdots + \mathbf{d}_n N_n^3(u)$$

2.3.2.1 de Boor Algorithm (3)



$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \cdots + \mathbf{d}_n N_n^3(u)$$

2.3.2.2. Relationship between de Boor algorithm & B-spline curves

- de Boor 알고리즘 : “Constructive Approach”

Input: $\mathbf{d}_i$ (de Boor Points)

Processor: 구간별로 $\mathbf{d}_i$를 n번 순차적 ‘linear interpolation’

Output : n차 곡선상의 점

→ ‘B-spline function’(Cox-de Boor recurrence formula) 형태로 표현 됨

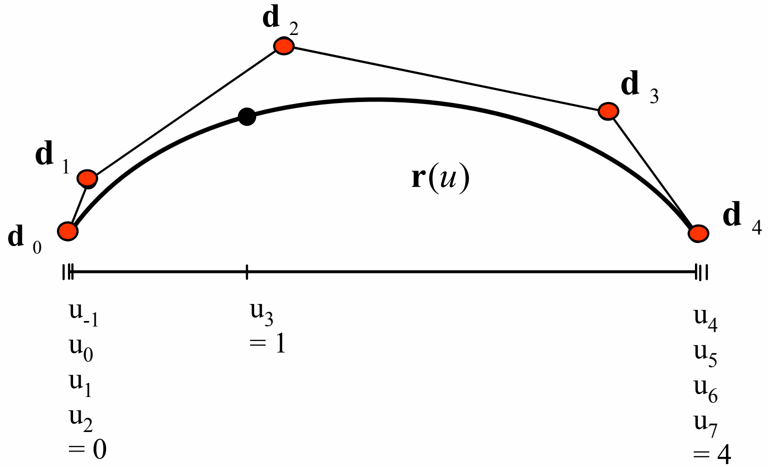
$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \cdots + \mathbf{d}_n N_n^3(u)$$

2.3.3 B-spline basis function

(Cox-de Boor recurrence formula)

2.3.3.1 Cox-de Boor Recurrence Formular (B-spline function) (1)

- 예: Cubic B-Spline 곡선



$$\mathbf{r}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \sum_{i=0}^{D-1} \mathbf{d}_i N_i^n(u)$$

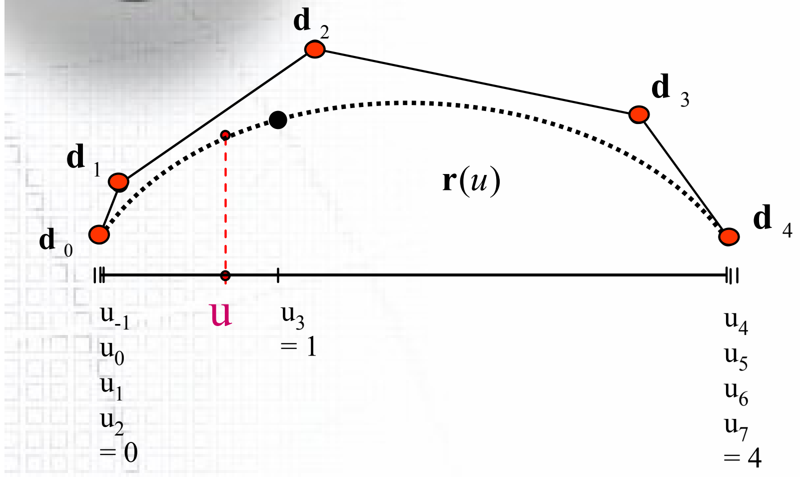
$$= \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u) + \mathbf{d}_4 N_4^3(u)$$

- Cox-de Boor Recurrence Formula (B-spline function)

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

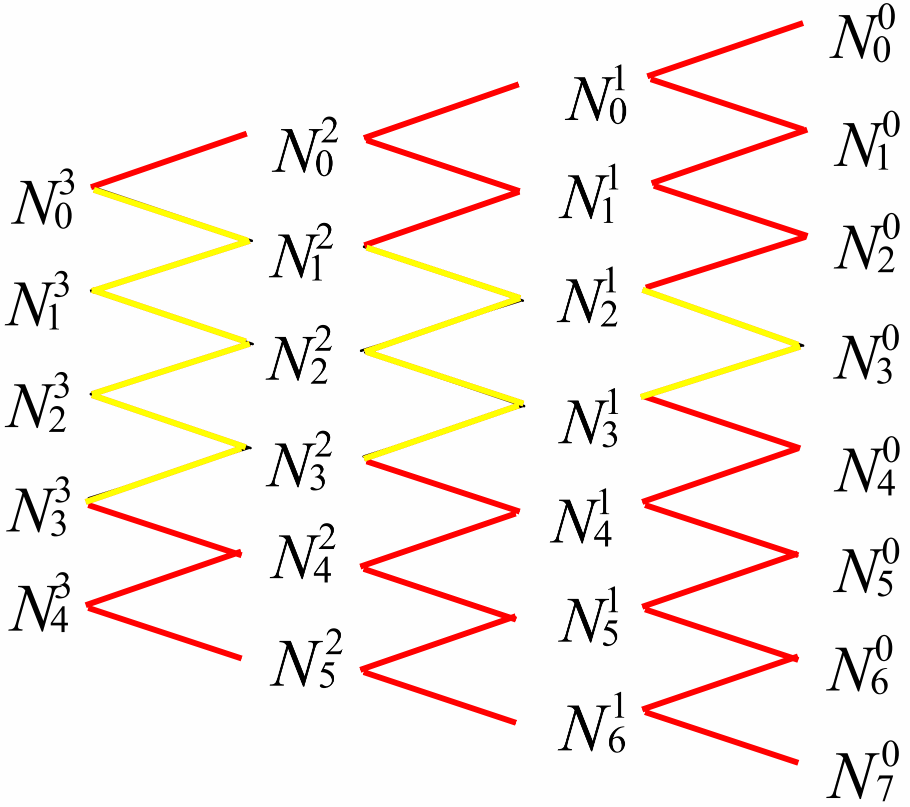
$$N_i^0(u) = \begin{cases} 1 & \text{if } \; u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}$$

2.3.3.1 Cox-de Boor Recurrence Formular (B-spline function) (2)



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

$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}$$



N_0^3 N_1^3 N_2^3 N_3^3 N_4^3

N_0^2 N_1^2 N_2^2 N_3^2 N_4^2 N_5^2

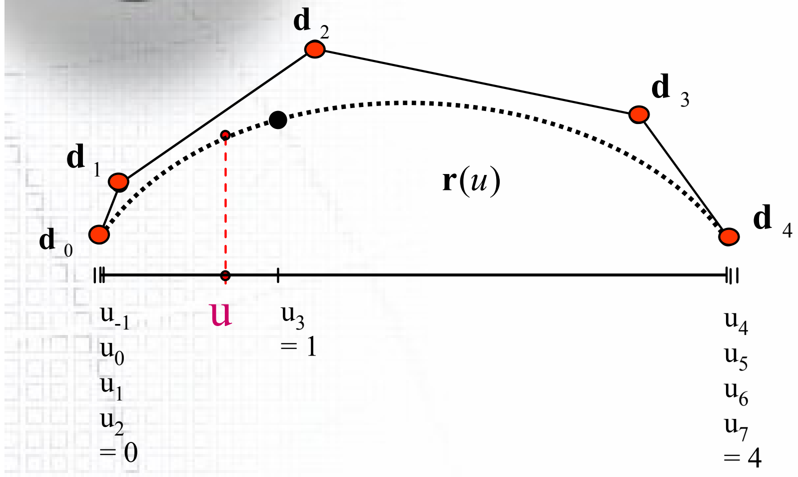
N_0^1 N_1^1 N_2^1 N_3^1 N_4^1 N_5^1 N_6^1

N_0^0	0	$u_{-1} \le u < u_0$
N_1^0	0	$u_0 \le u < u_1$
N_2^0	0	$u_1 \le u < u_2$
N_3^0	1	$u_2 \le u < u_3$
N_4^0	0	$u_3 \le u < u_4$
N_5^0	0	$u_4 \le u < u_5$
N_6^0	0	$u_5 \le u < u_6$
N_7^0	0	$u_6 \le u < u_7$

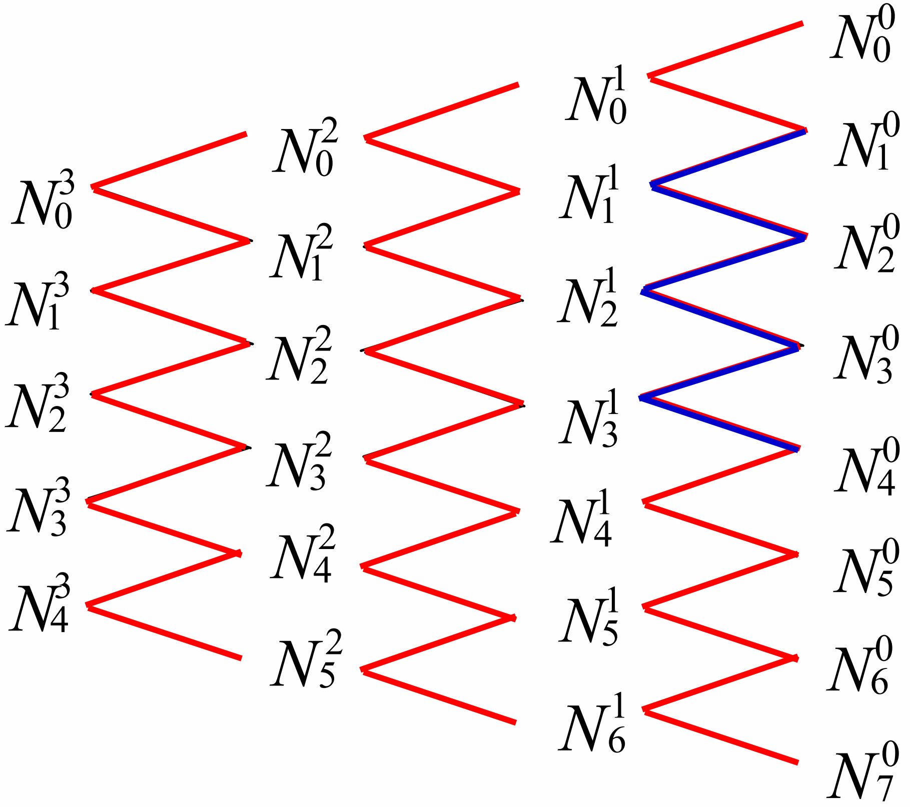
2.3.3.1 Cox-de Boor Recurrence Formular (B-spline function) (2)

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

$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_{i-1} \leq u < u_i \\ 0 & \text{else} \end{cases}$$

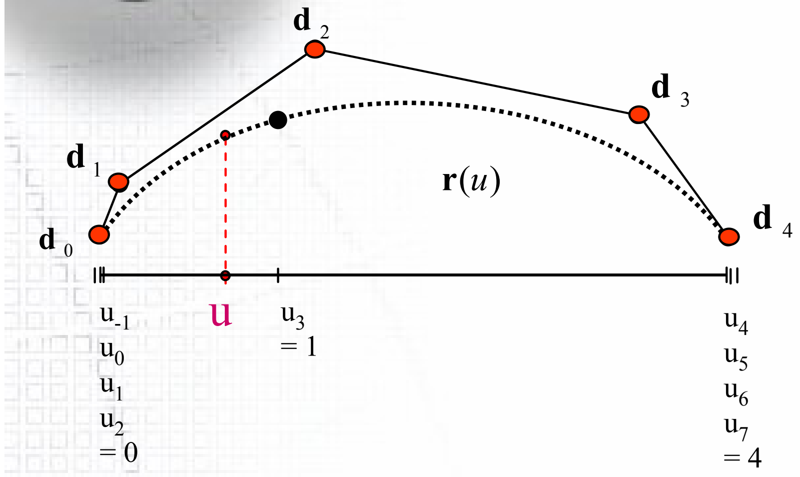


$N_1^1 = 0$ $\quad N_2^1 = \frac{u_2 - u}{u_3 - u_2}$ $\quad N_3^1 = \frac{u - u_2}{u_3 - u_2}$



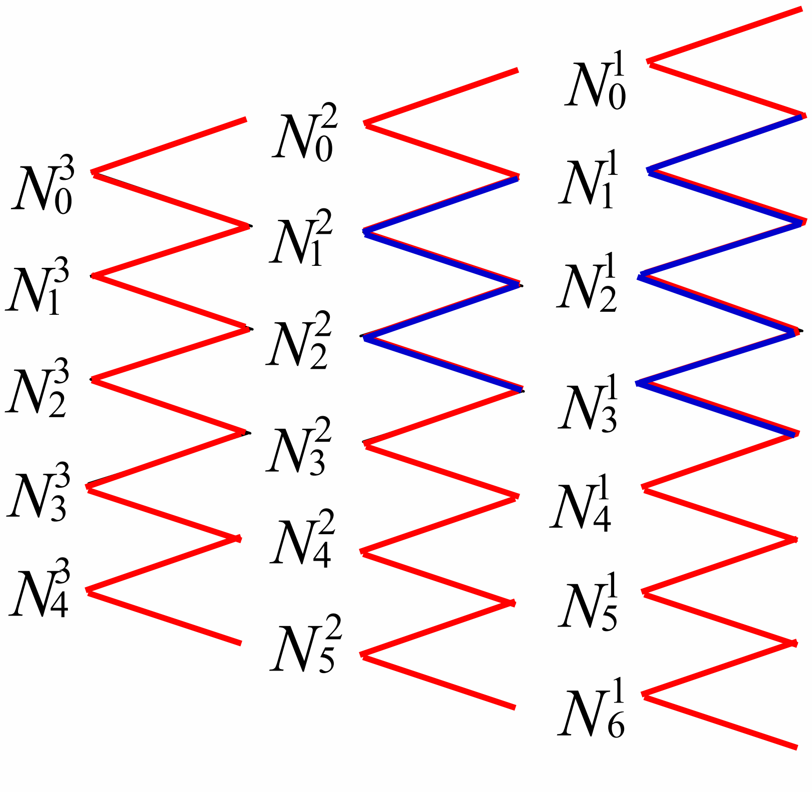
N_0^0	0	$u_{-1} \leq u < u_0$
N_1^0	0	$u_0 \leq u < u_1$
N_2^0	0	$u_1 \leq u < u_2$
N_3^0	1	$u_2 \leq u < u_3$
N_4^0	0	$u_3 \leq u < u_4$
N_5^0	0	$u_4 \leq u < u_5$
N_6^0	0	$u_5 \leq u < u_6$
N_7^0	0	$u_6 \leq u < u_7$

2.3.3.1 Cox-de Boor Recurrence Formular (B-spline function) (3)



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

$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}$$



N_0^0	0	$u_{-1} \le u < u_0$
N_1^0	0	$u_0 \le u < u_1$
N_2^0	0	$u_1 \le u < u_2$
N_3^0	1	$u_2 \le u < u_3$
N_4^0	0	$u_3 \le u < u_4$
N_5^0	0	$u_4 \le u < u_5$
N_6^0	0	$u_5 \le u < u_6$
N_7^0	0	$u_6 \le u < u_7$

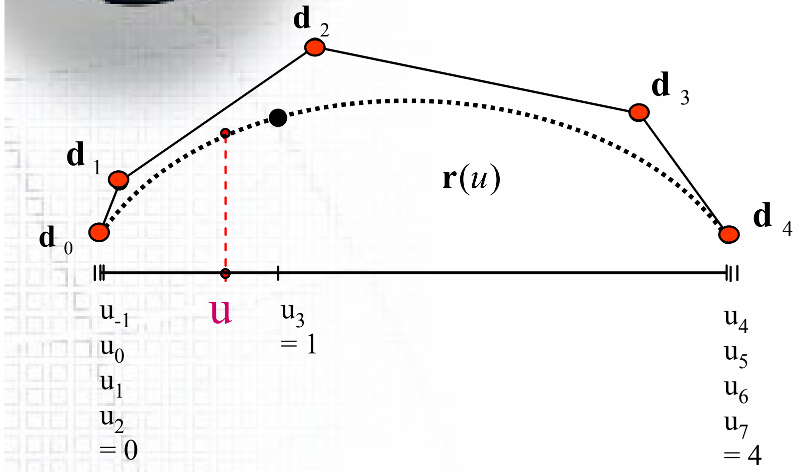
$$N_1^1 = 0 \qquad N_2^1 = \frac{u_2 - u}{u_3 - u_2} \qquad N_3^1 = \frac{u - u_2}{u_3 - u_2}$$

$$N_1^2 = \frac{u_3 - u}{u_3 - u_1} N_2^1 = \frac{u_3 - u}{u_3 - u_1} \cdot \frac{u_2 - u}{u_3 - u_2}$$

$$N_2^2 = \frac{u - u_1}{u_3 - u_1} N_2^1 + \frac{u_4 - u}{u_4 - u_2} N_3^1$$

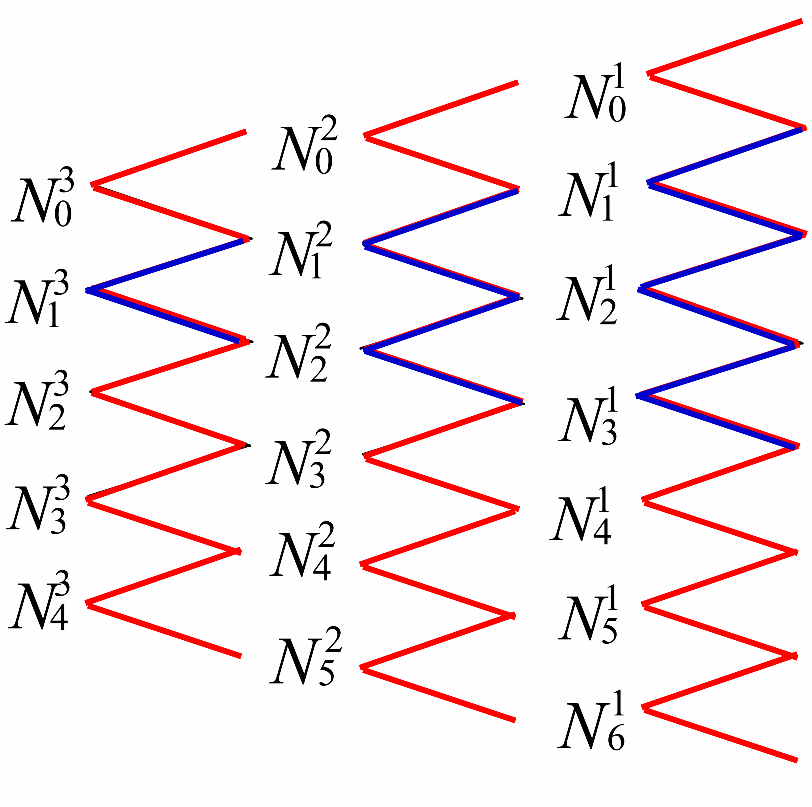
$$= \frac{u - u_1}{u_3 - u_1} \cdot \frac{u_2 - u}{u_3 - u_2} + \frac{u_4 - u}{u_4 - u_2} \cdot \frac{u - u_2}{u_3 - u_2}$$

2.3.3.1 Cox-de Boor Recurrence Formular (B-spline function) (4)



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

$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}$$



N_0^0	0	$u_{-1} \le u < u_0$
N_1^0	0	$u_0 \le u < u_1$
N_2^0	0	$u_1 \le u < u_2$
N_3^0	1	$u_2 \le u < u_3$
N_4^0	0	$u_3 \le u < u_4$
N_5^0	0	$u_4 \le u < u_5$
N_6^0	0	$u_5 \le u < u_6$
N_7^0	0	$u_6 \le u < u_7$

$$N_1^1 = 0 \qquad N_2^1 = \frac{u_2 - u}{u_3 - u_2} \qquad N_3^1 = \frac{u - u_2}{u_3 - u_2}$$

$$N_1^2 = \frac{u_3 - u}{u_3 - u_1} N_2^1 = \frac{u_3 - u}{u_3 - u_1} \cdot \frac{u_2 - u}{u_3 - u_2}$$

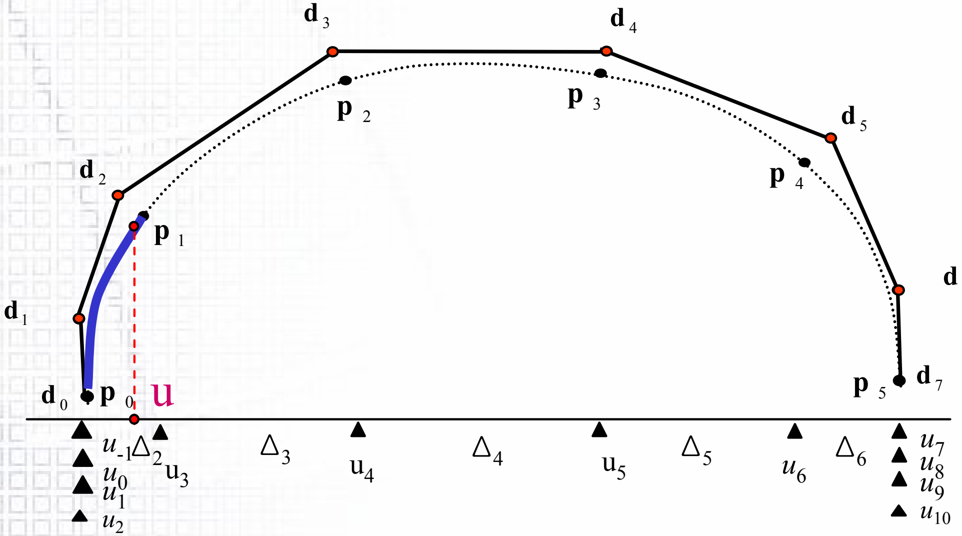
$$N_2^2 = \frac{u - u_1}{u_3 - u_1} N_2^1 + \frac{u_4 - u}{u_4 - u_2} N_3^1$$

$$= \frac{u - u_1}{u_3 - u_1} \cdot \frac{u_2 - u}{u_3 - u_2} + \frac{u_4 - u}{u_4 - u_2} \cdot \frac{u - u_2}{u_3 - u_2}$$

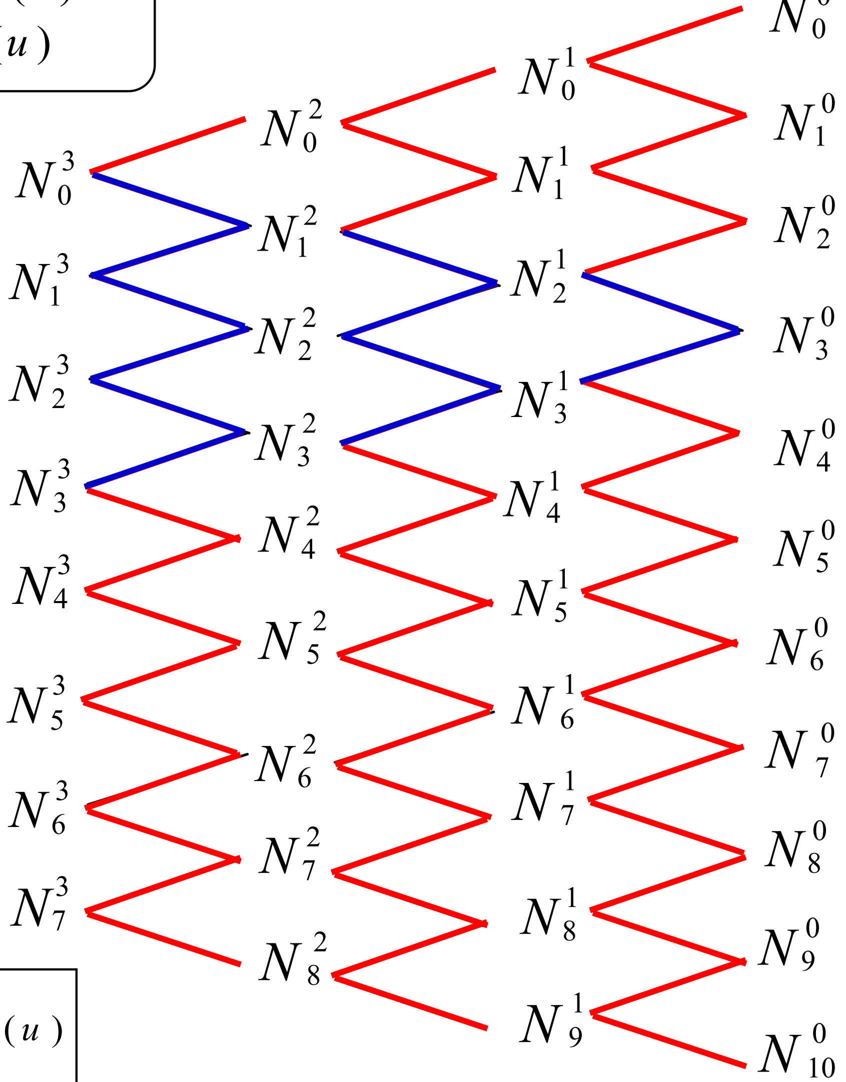
$$N_1^3 = \frac{u - u_0}{u_3 - u_2} N_1^2 + \frac{u_4 - u}{u_4 - u_1} N_2^2 = \frac{u - u_0}{u_3 - u_2} \cdot \frac{u_3 - u}{u_3 - u_1} \cdot \frac{u_2 - u}{u_3 - u_2} + \frac{u_4 - u}{u_4 - u_1} \cdot \frac{u - u_1}{u_3 - u_1} \cdot \frac{u_2 - u}{u_3 - u_2} + \frac{u_4 - u}{u_4 - u_1} \cdot \frac{u_4 - u}{u_4 - u_2} \cdot \frac{u - u_2}{u_3 - u_2}$$

2.3.3.2 B-Spline curves (1)

$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u) + \mathbf{d}_4 N_4^3(u) + \mathbf{d}_5 N_5^3(u) + \mathbf{d}_6 N_6^3(u) + \mathbf{d}_7 N_7^3(u)$$



$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u)$$



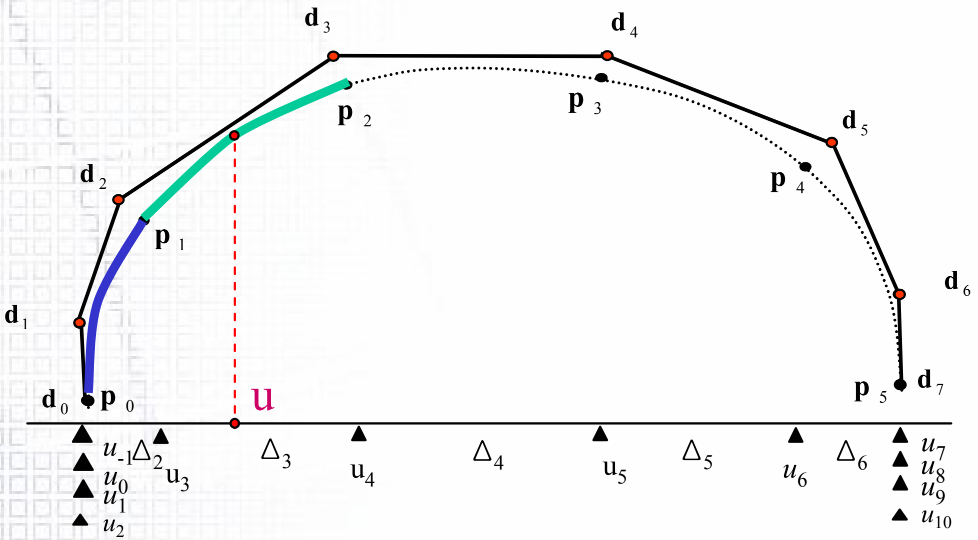
N_0^0	0	$u_{-1} \le u < u_0$
N_1^0	0	$u_0 \le u < u_1$
N_2^0	0	$u_1 \le u < u_2$
N_3^0	1	$u_2 \le u < u_3$
N_4^0	0	$u_3 \le u < u_4$
N_5^0	0	$u_4 \le u < u_5$
N_6^0	0	$u_5 \le u < u_6$
N_7^0	0	$u_6 \le u < u_7$
N_8^0	0	$u_7 \le u < u_8$
N_9^0	0	$u_8 \le u < u_9$
N_{10}^0	0	$u_9 \le u < u_{10}$

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

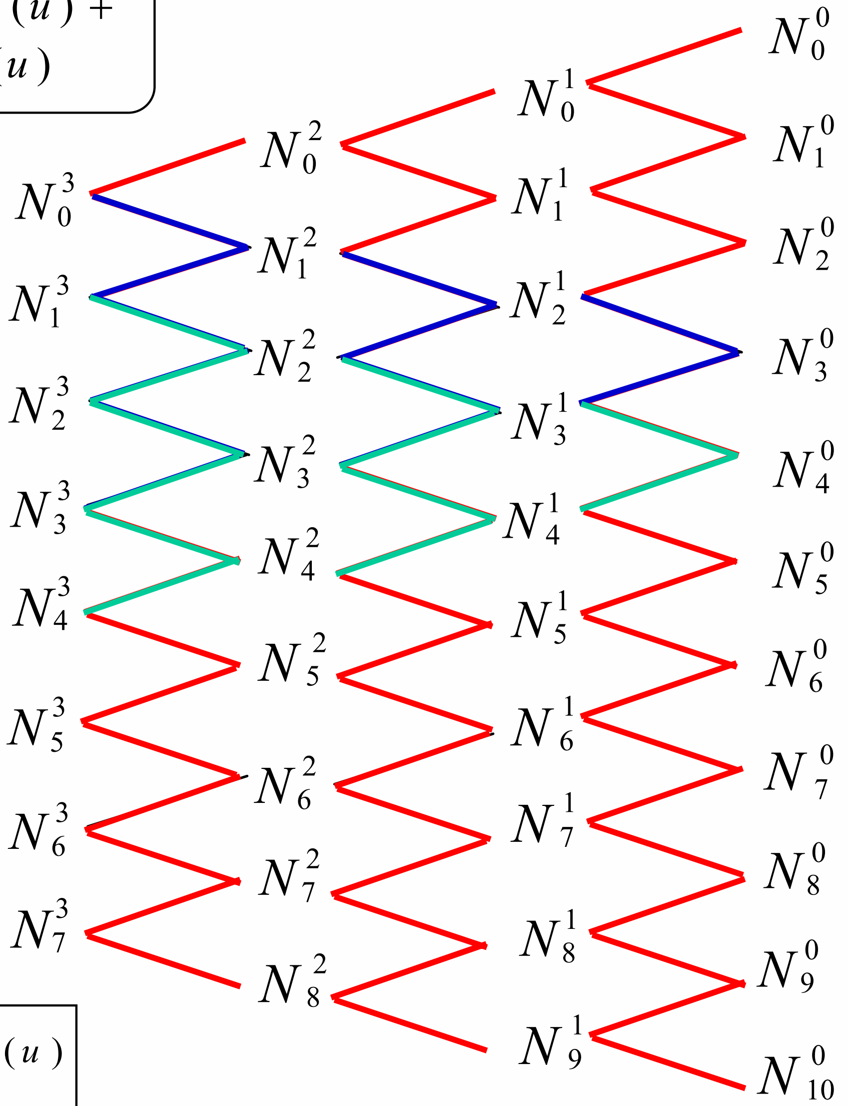
$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}$$

2.3.3.2 B-Spline curves (2)

$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u) + \mathbf{d}_4 N_4^3(u) + \mathbf{d}_5 N_5^3(u) + \mathbf{d}_6 N_6^3(u) + \mathbf{d}_7 N_7^3(u)$$



$$\mathbf{r}(u) = \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u) + \mathbf{d}_4 N_4^3(u)$$



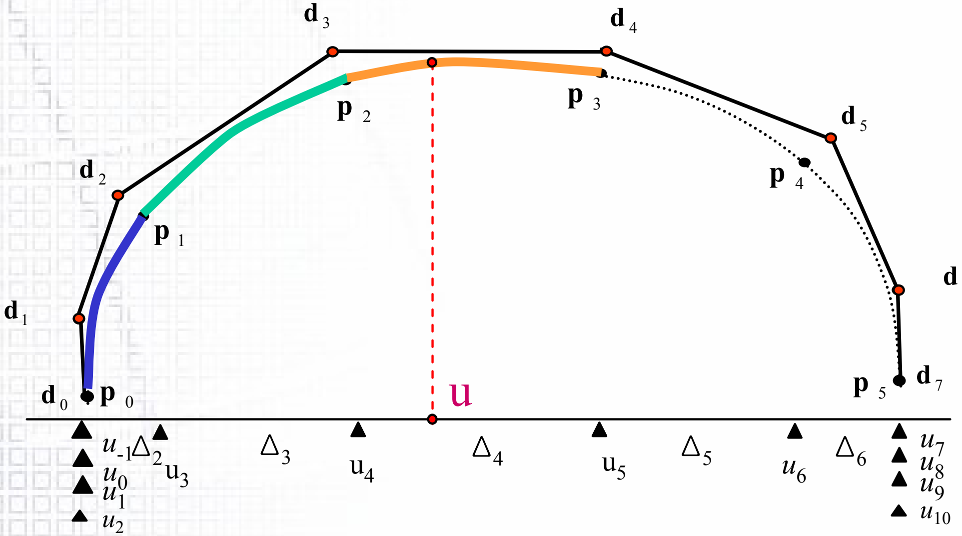
0	$u_{-1} \le u < u_0$
0	$u_0 \le u < u_1$
0	$u_1 \le u < u_2$
0	$u_2 \le u < u_3$
1	$u_3 \le u < u_4$
0	$u_4 \le u < u_5$
0	$u_5 \le u < u_6$
0	$u_6 \le u < u_7$
0	$u_7 \le u < u_8$
0	$u_8 \le u < u_9$
0	$u_9 \le u < u_{10}$

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

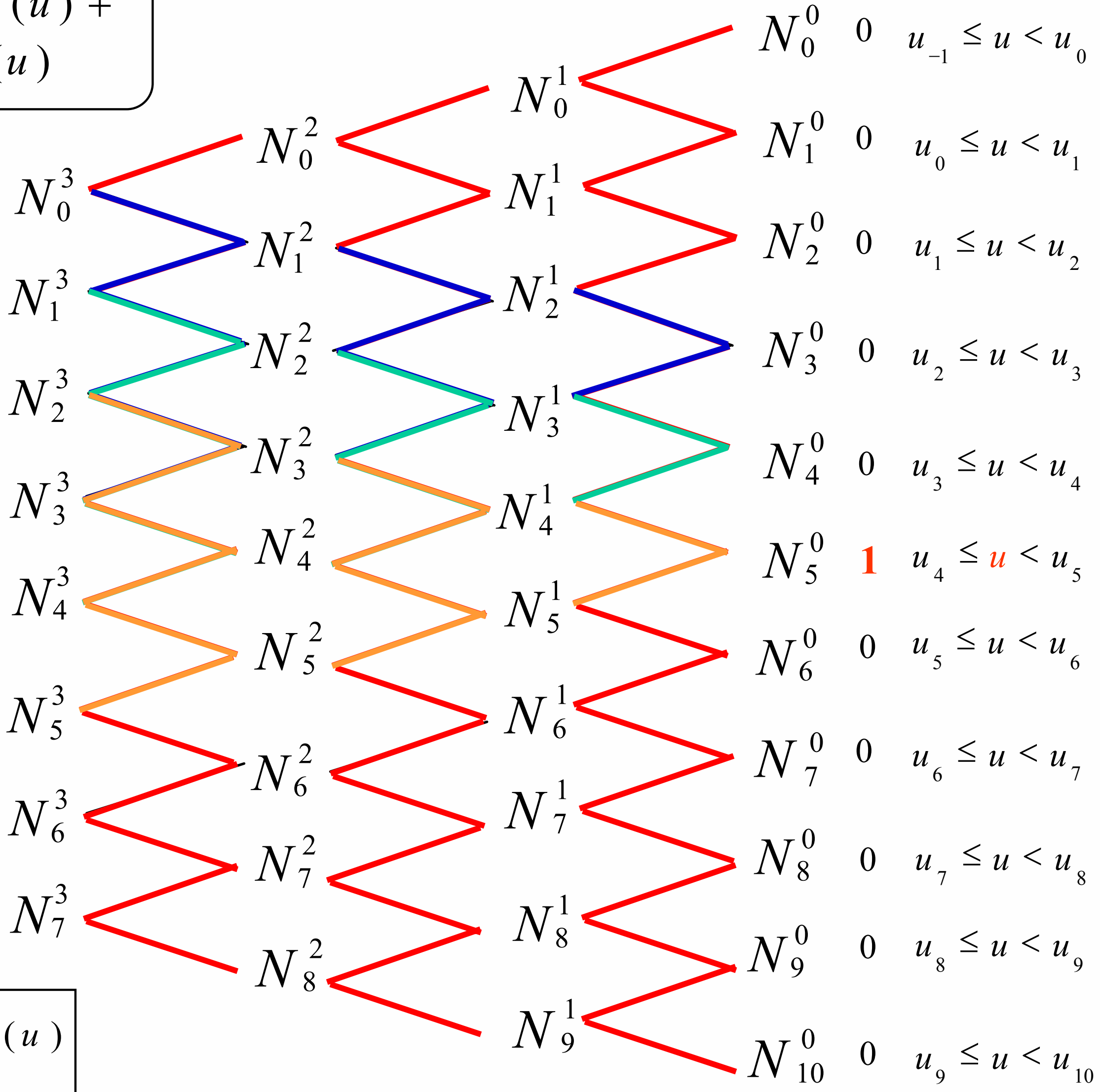
$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}$$

2.3.3.2 B-Spline curves (3)

$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u) + \mathbf{d}_4 N_4^3(u) + \mathbf{d}_5 N_5^3(u) + \mathbf{d}_6 N_6^3(u) + \mathbf{d}_7 N_7^3(u)$$



$$\mathbf{r}(u) = \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u) + \mathbf{d}_4 N_4^3(u) + \mathbf{d}_5 N_5^3(u)$$

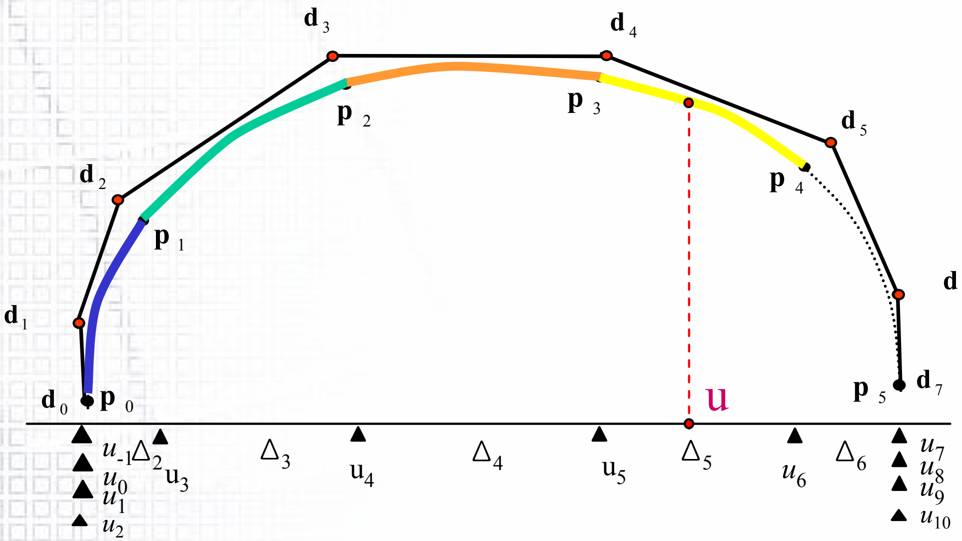


$$N_i^n(u) = \frac{u - u_{i-1}}{u_{i+n-1} - u_{i-1}} N_i^{n-1}(u) + \frac{u_{i+n} - u}{u_{i+n} - u_i} N_{i+1}^{n-1}(u)$$

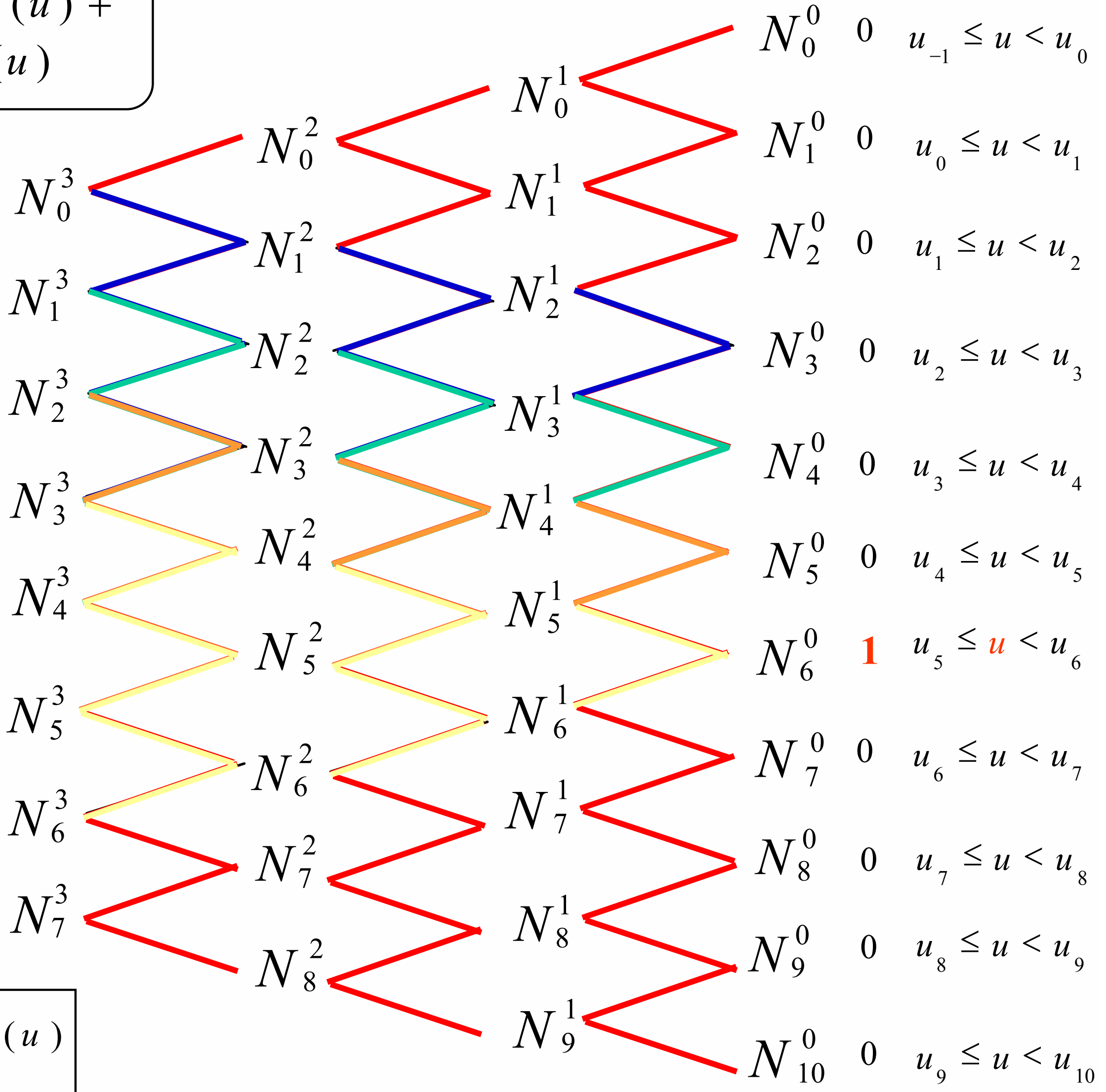
$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}$$

2.3.3.2 B-Spline curves (4)

$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u) + \mathbf{d}_4 N_4^3(u) + \mathbf{d}_5 N_5^3(u) + \mathbf{d}_6 N_6^3(u) + \mathbf{d}_7 N_7^3(u)$$



$$\mathbf{r}(u) = \mathbf{d}_3 N_3^3(u) + \mathbf{d}_4 N_4^3(u) + \mathbf{d}_5 N_5^3(u) + \mathbf{d}_6 N_6^3(u)$$



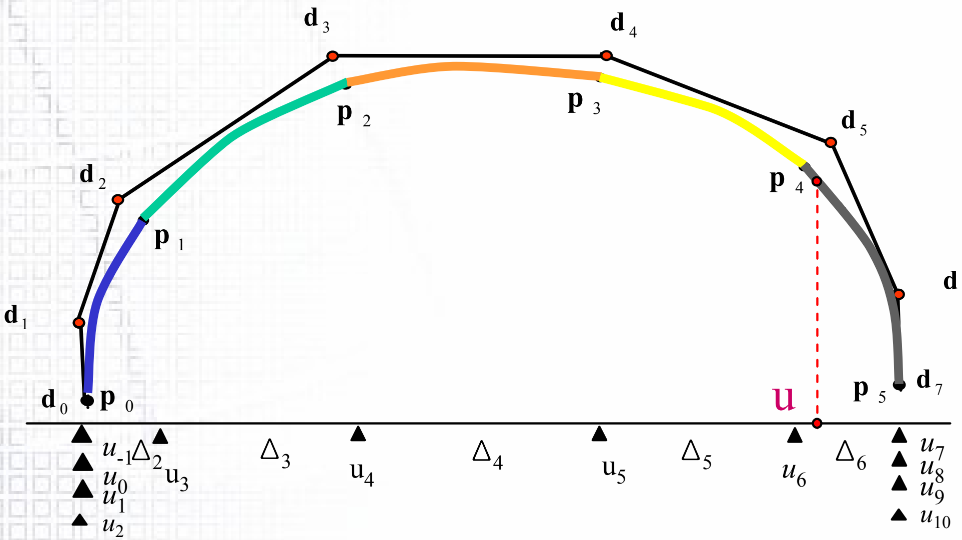
N_0^0	0	$u_{-1} \le u < u_0$
N_1^0	0	$u_0 \le u < u_1$
N_2^0	0	$u_1 \le u < u_2$
N_3^0	0	$u_2 \le u < u_3$
N_4^0	0	$u_3 \le u < u_4$
N_5^0	0	$u_4 \le u < u_5$
N_6^0	1	$u_5 \le u < u_6$
N_7^0	0	$u_6 \le u < u_7$
N_8^0	0	$u_7 \le u < u_8$
N_9^0	0	$u_8 \le u < u_9$
N_{10}^0	0	$u_9 \le u < u_{10}$

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

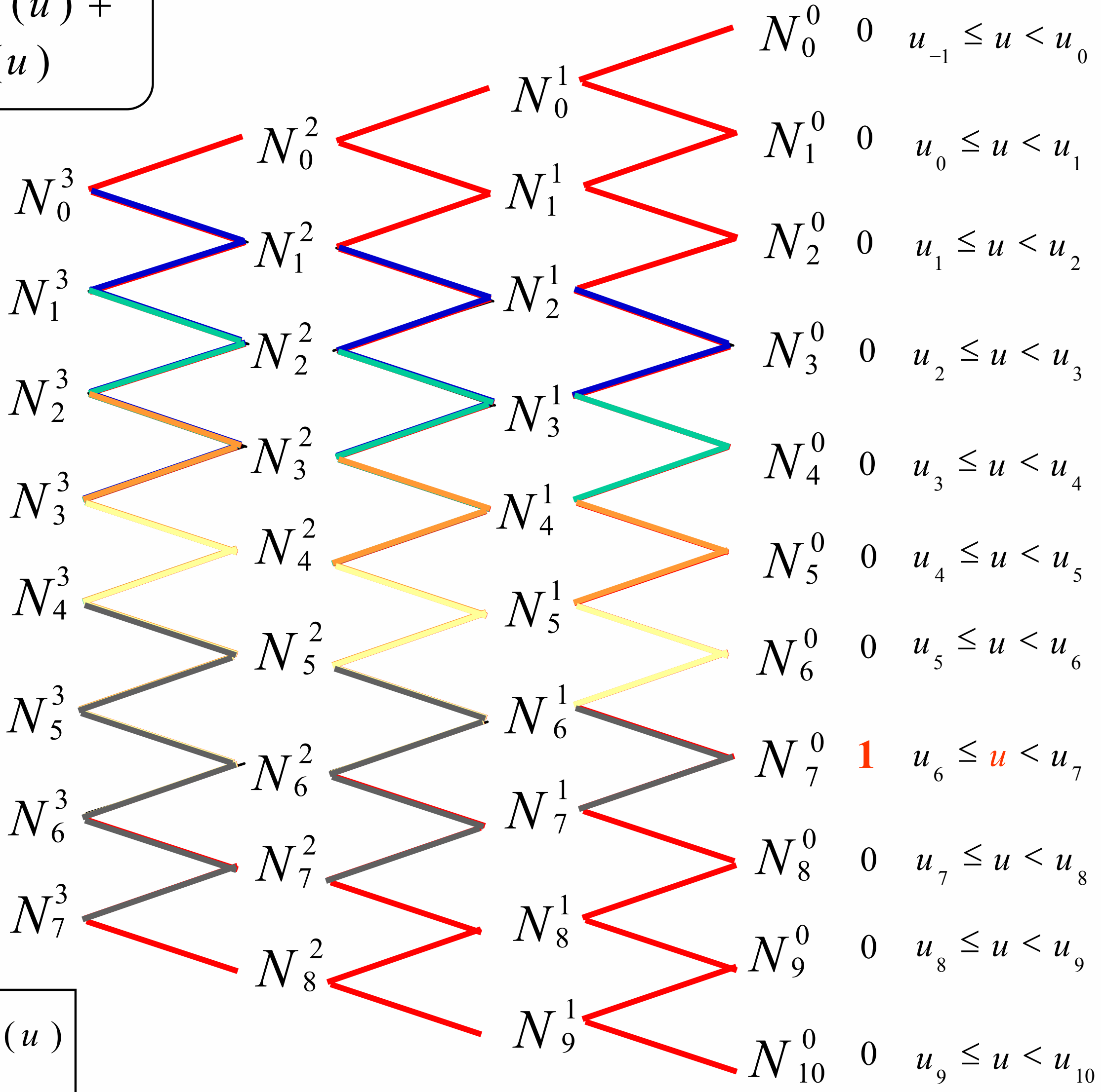
$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}$$

2.3.3.2 B-Spline curves (5)

$$\mathbf{r}(u) = \mathbf{d}_0 N_0^3(u) + \mathbf{d}_1 N_1^3(u) + \mathbf{d}_2 N_2^3(u) + \mathbf{d}_3 N_3^3(u) + \mathbf{d}_4 N_4^3(u) + \mathbf{d}_5 N_5^3(u) + \mathbf{d}_6 N_6^3(u) + \mathbf{d}_7 N_7^3(u)$$



$$\mathbf{r}(u) = \mathbf{d}_4 N_4^3(u) + \mathbf{d}_5 N_5^3(u) + \mathbf{d}_6 N_6^3(u) + \mathbf{d}_7 N_7^3(u)$$



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

$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u_{i-1} \le u < u_i \\ 0 & \text{else} \end{cases}$$

2.3.3.3 Relationship between de Boor algorithm & B-spline curves

- de Boor 알고리즘 : "Constructive Approach"

  Input: $\mathbf{d}_i$ (de Boor Points)

  Processor: 구간별로 $\mathbf{d}_i$를 n번 순차적 'linear interpolation'

  Output : n차 곡선상의 점
  →'B-spline function'(Cox-de Boor recurrence formula) 형태로 표현 됨

- B-spline 곡선식: "B-spline function evaluation Approach"

  Input: $\mathbf{d}_i$ (de Boor Points)

  Processor: 공간 상의 점 $\mathbf{d}_i$와 B-spline function을 "blending"하여 함수 값을 계산하면 곡선상의 점을 구할 수 있음

  Output: B-spline function과 $\mathbf{d}_i$의 혼합 함수 형태로 표현

2.3.3.4 Sample code of Cubic B-spline Curve (1)

```
#ifndef __CubicBSpline_h__
#define __CubicBSpline_h__

#include "vector.h“

class CubicBSplineCurve {
public:
    Vector* m_ControlPoint;   int m_nControlPoint;
    double* m_Knot; int m_nKnot;
    int m_nDegree;

    CubicBSplineCurve();
    ~CubicBSplineCurve();

    void SetControlPoint(Vector* pControlPoint, int nControlPoint);
    void SetKnot(double* pKnot, int nKnot);
    Vector CalcPoint(double u);
    double N(int d, int i, double u);                // B-spline basis function
};
#endif
```

2.3.3.4 Sample code of Cubic B-spline Curve (2)

```
CubicBSplineCurve::CubicBSplineCurve () {
   m_ControlPoint = 0;        m_Knot = 0;
   m_nControlPoint = 0;       m_nKnot = 0;       int m_nDegree =3;
}
CubicBSplineCurve::~CubicBSplineCurve () {
   if(m_ControlPoint)  delete[] m_ControlPoint;
   if(m_Knot)  delete[] m_Knot;
}
void CubicBSplineCurve::SetControlPoint(Vector* pControlPoint, int nControlPoint) {
   m_ControlPoint = new Vector[nControlPoint];
   for(int i=0; i < nControlPoint; i++) {
      m_ControlPoint[i] = pControlPoint[i];
   }
}
void CubicBSplineCurve::SetKnot(double* pKnot, int nKnot){
   m_Knot = new double[nKnot];
   for(int i=0; i < nKnot; i++) {
     m_Knot[i] = pKnot[i];
   }
}
```

2.3.3.4 Sample code of Cubic B-spline Curve (3)

```
Vector CubicBSplineCurve::CalcPoint(double u)
{
    Vector PointOnCurve(0,0,0);
    if ( t < m_Knot[0] || t > m_Knot[m_nKnot-1] ) {
        return PointOnCurve;
    }
    for(int i = 0; i < m_nControlPoint; i++){
        PointOnCurve  =  PointOnCurve  +  m_ControlPoint[i] * N(m_nDegree, i, u);
    }
    return PointOnCurve;
}
```

2.3.3.4 Sample code of Cubic B-spline Curve (4)

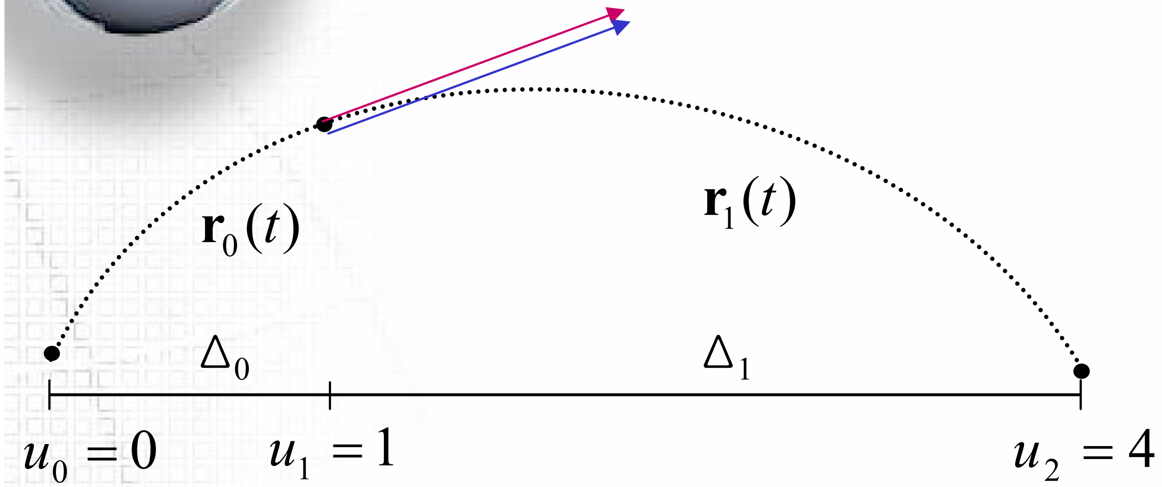
```
double CubicBSplineCurve:: N(int d, int i, double u) {
    // Find Span k
    // U i-1 <= U < U i  → k = i

    if( d == 0 ) {
        // return 0 or 1;
    } else {
        // return Cox de-Boor recurrence formula
    }
}
```

2.3.4 C^1 and C^2 continuity condition

2.3.4.1 1st Derivatives of Cubic Bezier Curves at Junction point
2.3.4.2 C^1 continuity condition of composite curves
2.3.4.3 2nd Derivatives of Cubic Bezier Curves
2.3.4.4 C^2 continuity condition of composite curves

Advanced
Ship
Design
Automation
Laboratory

2.3.4.1 1st Derivatives of Cubic Bezier Curves at Junction point



$$t = \frac{u - u_i}{u_{i+1} - u_i} = \frac{u - u_i}{\Delta_i}$$

t는 [0,1] 구간의
국부매개변수('local parameter')

$$\frac{d\mathbf{r}(u(t))}{du} = \frac{d\mathbf{r}_i(t)}{dt}\frac{dt}{du} = \frac{1}{\Delta_i}\frac{d\mathbf{r}_i(t)}{dt}$$

$\frac{d\mathbf{r}\ (u)}{du}$ 의 $u_0 \le \mathrm{u} \le u_1$ 에서의 미분 값

$$t = \frac{u - u_0}{u_1 - u_0} = \frac{u - u_0}{\Delta_0}$$

t는 [0,1] 구간의
국부 매개 변수

$$\frac{d\mathbf{r}\ (u)}{du} = \frac{d\ \mathbf{r}_0(u(t))}{dt}\frac{dt}{du} = \frac{1}{\Delta_0}\frac{d\ \mathbf{r}_0(t)}{dt}$$

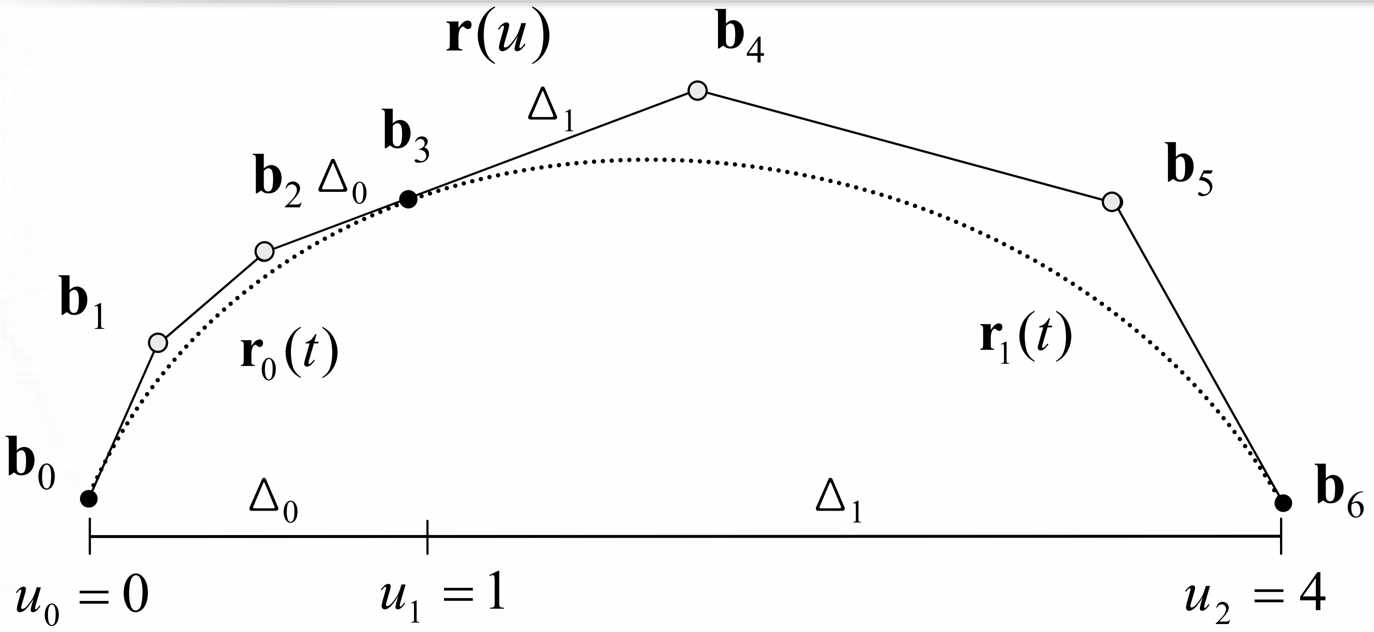
$\frac{d\mathbf{r}\ (u)}{du}$ 의 $u_1 \le \mathrm{u} \le u_2$ 에서의 미분 값

$$t = \frac{u - u_1}{u_2 - u_1} = \frac{u - u_1}{\Delta_1}$$

t는 [0,1] 구간의
국부 매개 변수

$$\frac{d\mathbf{r}\ (u)}{du} = \frac{d\ \mathbf{r}_1(t)}{dt}\frac{dt}{du} = \frac{1}{\Delta_1}\frac{d\ \mathbf{r}_1(t)}{dt}$$

2.3.4.2 C¹ continuity condition of composite curves



$\mathbf{r}(u = u_1) = \mathbf{r}_0(t = 1) = \mathbf{r}_1(t = 0)$ 연결 점에서 C¹ 조건을 만족 해야 하므로

$$\left.\begin{aligned} \left.\frac{d\mathbf{r}(u)}{du}\right|_{u_1=1} &= \frac{1}{\Delta_0}\left.\frac{d\mathbf{r}_0(t)}{dt}\right|_{t=1} = \frac{1}{\Delta_0}3(\mathbf{b}_3-\mathbf{b}_2) \\ &= \frac{1}{\Delta_1}\left.\frac{d\mathbf{r}_1(t)}{dt}\right|_{t=0} = \frac{1}{\Delta_1}\cdot 3(\mathbf{b}_4-\mathbf{b}_3) \end{aligned}\right\}$$

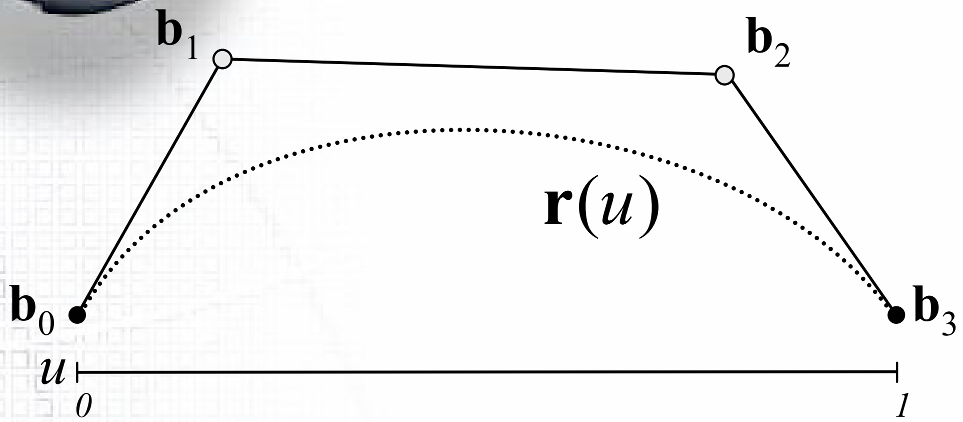
$$(\mathbf{b}_3-\mathbf{b}_2):(\mathbf{b}_4-\mathbf{b}_3) = \Delta_0 : \Delta_1$$

⇓

$$\mathbf{b}_3 = \frac{\Delta_1}{\Delta}\mathbf{b}_2 + \frac{\Delta_0}{\Delta}\mathbf{b}_4$$

parameter *u*를 시간이라고 생각하면, 1차 미분 계수는 곡선상을 지나는 점의 속도라고 생각할 수 있다.
연결점 $\mathbf{b}_3$에서 1차 미분 계수가 연속이라면 그 점에서 속도가 연속이어야 한다는 의미이다.
그러므로 시간 간격이 Δ_0 에서 Δ_1 으로 변하면 즉, 시간 간격이 변하면,
그 거리도 비례하여 변하여야 연결점에서 속도가 연속이다.!!!

2.3.4.3 2nd Derivatives of Cubic Bezier Curves



a) u 가 0 에서 1까지 변할 때

n차 Bezier곡선 2차미분

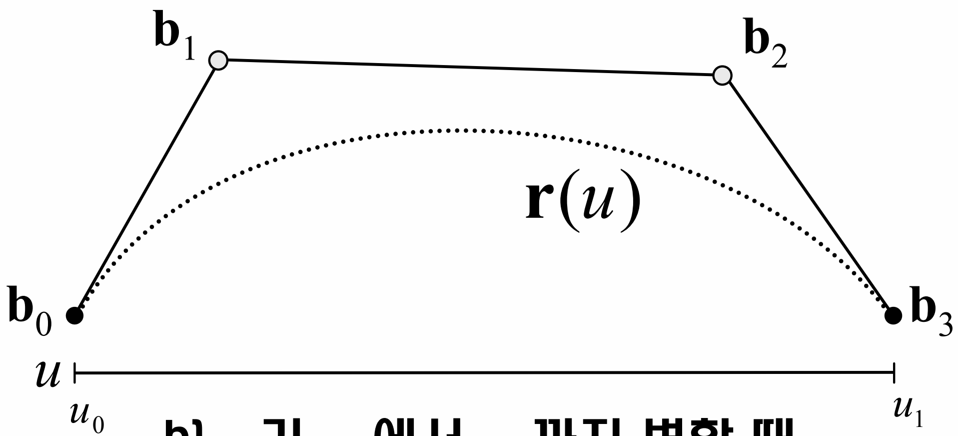
$$\frac{d^2\mathbf{r}(u)}{du^2}=n(n-1)\sum_{i=0}^{n-2}(\mathbf{b}_{i+2}-2\mathbf{b}_{i+1}+\mathbf{b}_i)B_i^{n-2}$$

3차 Bezier곡선 2차미분

$$\frac{d^2\mathbf{r}(u)}{du^2}=3(3-1)\sum_{i=0}^{1}(\mathbf{b}_{i+2}-2\mathbf{b}_{i+1}+\mathbf{b}_i)B_i^1(u)$$

$u=1$일 때

$$\frac{d^2\mathbf{r}(1)}{du^2}=3(3-1)(\mathbf{b}_3-2\mathbf{b}_2+\mathbf{b}_1)$$



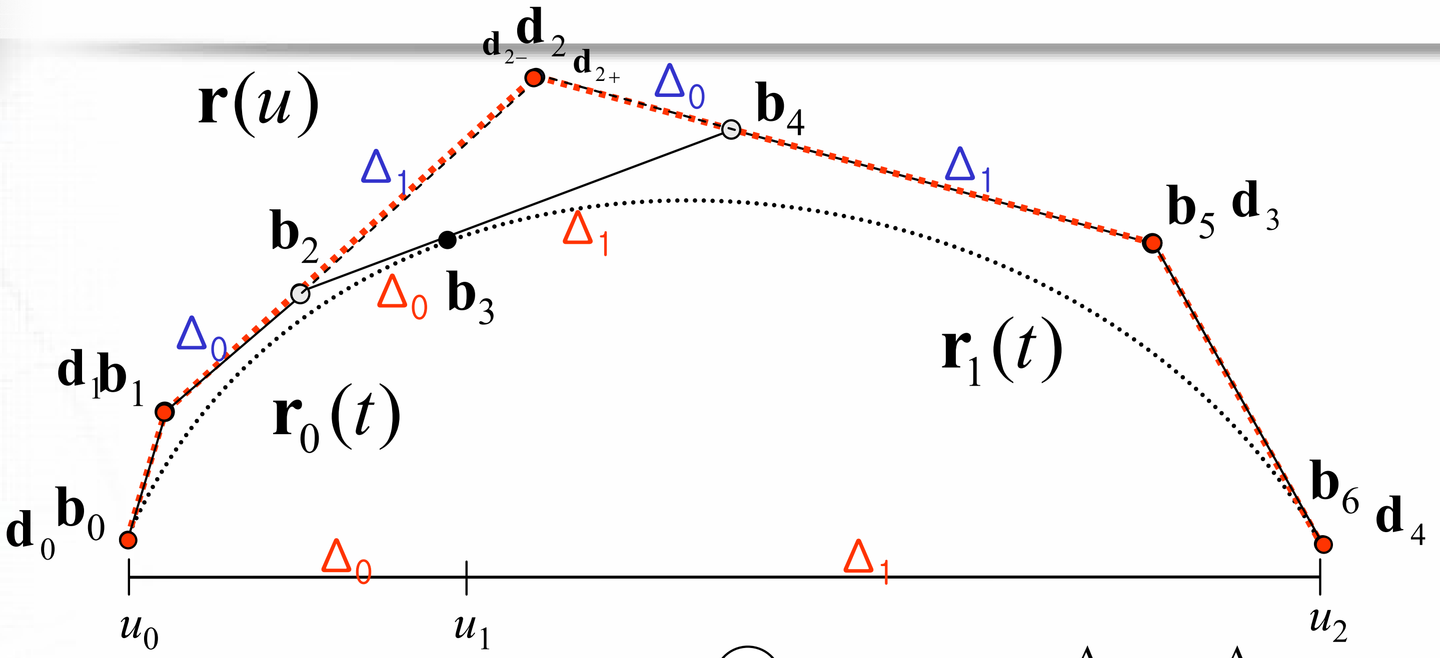
b) u 가 u_0 에서 u_1 까지 변할 때

$$\frac{d^2\mathbf{r}(u(t))}{du^2}=\frac{1}{(\Delta)^2}\frac{d^2\mathbf{r}(t)}{dt^2}\qquad(\Delta=u_1-u_0)$$

$u=u_1$ 일 때

$$\frac{d^2\mathbf{r}(u_1)}{du^2}=\frac{1}{(\Delta)^2}\frac{d^2\mathbf{r}(1)}{dt^2}=\frac{1}{(\Delta)^2}3(3-1)(\mathbf{b}_3-2\mathbf{b}_2+\mathbf{b}_1)$$

2.3.4.4. C² continuity condition of composite curves



① 연결점 $\mathbf{b}_3$에서 C^2조건

$$\frac{d^2\mathbf{r}(u_{1-})}{du^2}=\frac{1}{(\Delta_0)^2}\frac{d^2\mathbf{r}_0(1)}{dt^2}=\frac{1}{(\Delta_0)^2}3(3-1)(\mathbf{b}_3-2\mathbf{b}_2+\mathbf{b}_1)$$

$$\frac{d^2\mathbf{r}(u_{1+})}{du^2}=\frac{1}{(\Delta_1)^2}\frac{d^2\mathbf{r}_1(0)}{dt^2}=\frac{1}{(\Delta_1)^2}3(3-1)(\mathbf{b}_5-2\mathbf{b}_4+\mathbf{b}_3)$$

② $\dfrac{d^2\mathbf{r}(u_{1-})}{du^2}=\dfrac{d^2\mathbf{r}(u_{1+})}{du^2}$ 이어야 하므로

$\dfrac{6}{(\Delta_0)^2}(\mathbf{b}_3-2\mathbf{b}_2+\mathbf{b}_1)=\dfrac{6}{(\Delta_0)^2}(\mathbf{b}_5-2\mathbf{b}_4+\mathbf{b}_3)$이다.

③ 그리고 C^1조건($\mathbf{b}_3=\dfrac{\Delta_1}{\Delta}\mathbf{b}_2+\dfrac{\Delta_0}{\Delta}\mathbf{b}_4$)을 대입 하여

정리하면

$$\Rightarrow -\frac{\Delta_1}{\Delta_0}\mathbf{b}_1+\frac{\Delta}{\Delta_0}\mathbf{b}_2=\frac{\Delta}{\Delta_1}\mathbf{b}_4-\frac{\Delta_0}{\Delta_1}\mathbf{b}_5$$

④ 좌변을 $\mathbf{d}_{2-}=-\dfrac{\Delta_1}{\Delta_0}\mathbf{b}_1+\dfrac{\Delta}{\Delta_0}\mathbf{b}_2$ 라 하면

$$\mathbf{b}_2=\frac{\Delta_1}{\Delta}\mathbf{b}_1+\frac{\Delta_0}{\Delta}\mathbf{d}_{2-}$$

⑤ 우변을 $\mathbf{d}_{2+}=\dfrac{\Delta}{\Delta_1}\mathbf{b}_4-\dfrac{\Delta_0}{\Delta_1}\mathbf{b}_5$ 라 하면

$$\mathbf{b}_4=\frac{\Delta_1}{\Delta}\mathbf{d}_{2+}+\frac{\Delta_1}{\Delta}\mathbf{b}_5$$

⑥ 즉,($\mathbf{d}_{2-}=\mathbf{d}_{2+}=\mathbf{d}_2$)인 점이 존재하면 C² 조건을 만족한다.

⑦ 연결점에서 C^2조건

$$-\frac{\Delta_1}{\Delta_0}\mathbf{b}_1+\frac{\Delta}{\Delta_0}\mathbf{b}_2=\frac{\Delta}{\Delta_1}\mathbf{b}_4-\frac{\Delta_0}{\Delta_1}\mathbf{b}_5$$

$$ratio(\mathbf{b}_1,\mathbf{b}_2,d_2)=ratio(d_2,\mathbf{b}_4,\mathbf{b}_5)=\frac{\Delta_0}{\Delta_1}$$

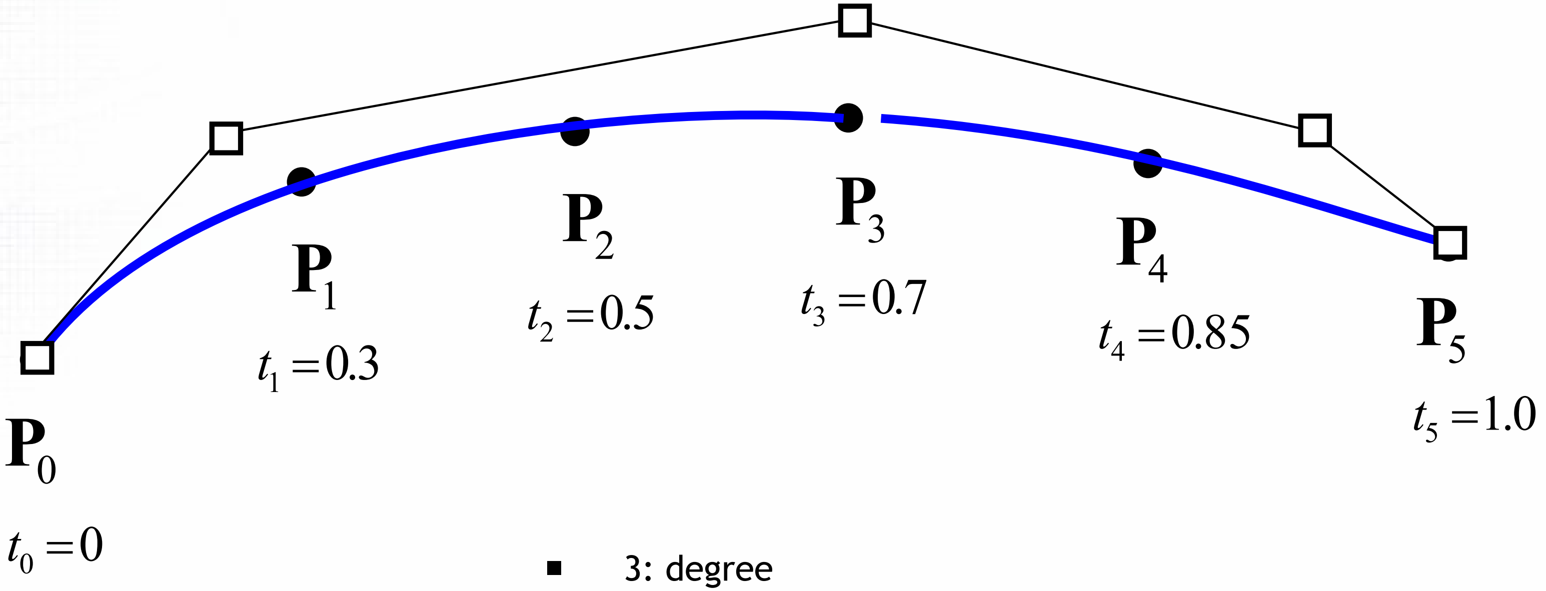
2.3.5 B-spline curve Interpolation

Advanced
Ship
Design
Automation
Laboratory

2.3.5.1 Determine # of Bezier curve segment & Knot value (1)

- Given: fitting points $\mathbf{P}_i$ and corresponding parameter t_i
  where, $i = 0,1,\ldots,m$ and $t_0 = 0,\ t_m = 1,$

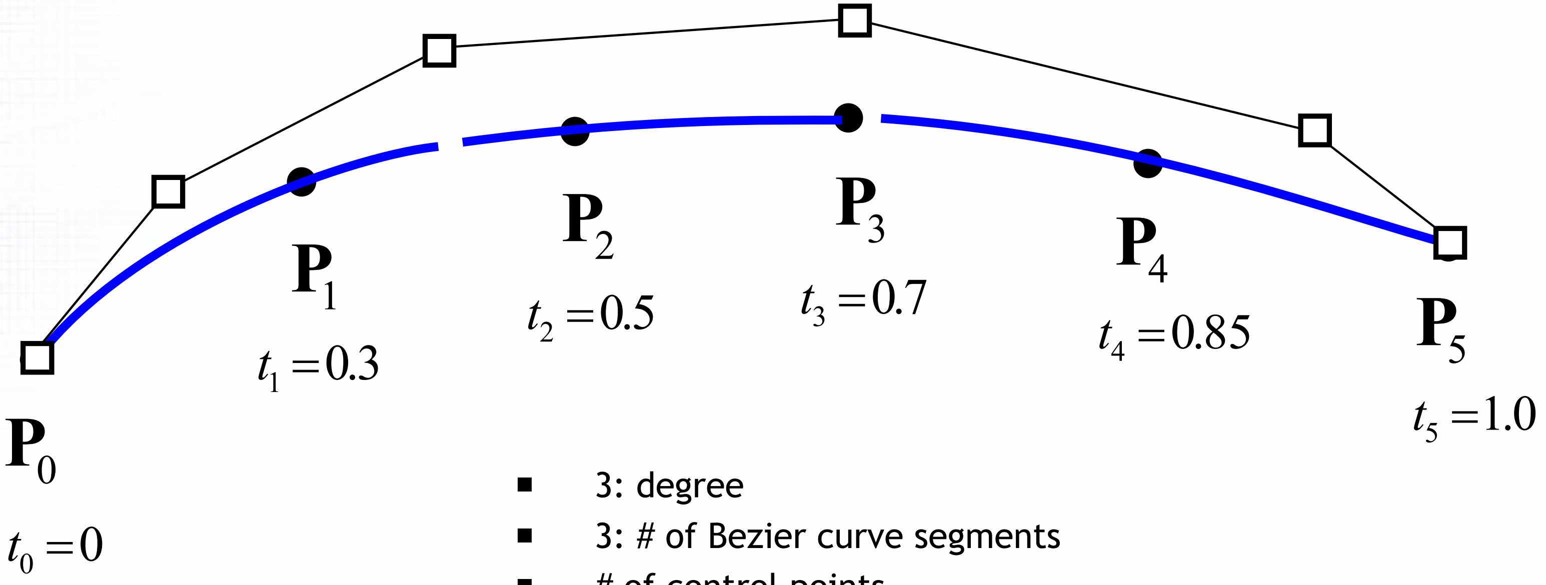
- First, determine # of Bezier curve segment and its knots



- 3: degree
- 2: # of Bezier curve segments
- # of control points
  = 4 + (2-1) = 5

2.3.5.1 Determine # of Bezier curve segment & Knot value (2)

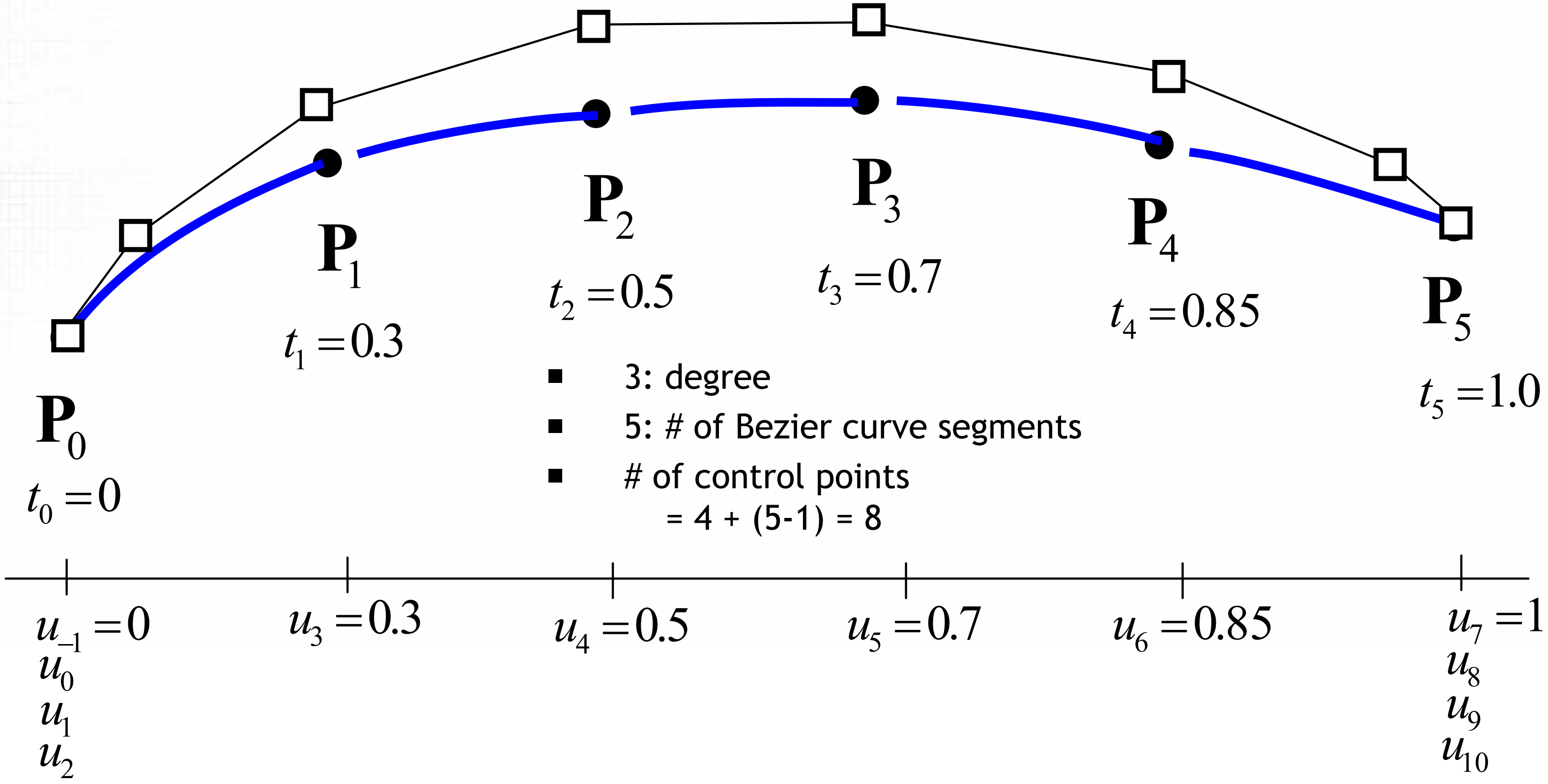
- Given: fitting points $\mathbf{P}_i$ and corresponding parameter t_i
  where, $i = 0,1,\ldots,m$ and $t_0 = 0,\ t_m = 1,$
- First, determine # of Bezier curve segment and its knots



- 3: degree
- 3: # of Bezier curve segments
- # of control points
  = 4 + (3-1) = 6
- How we determine Knots ?
  (= start / end points of each
  cubic Bezier curve)

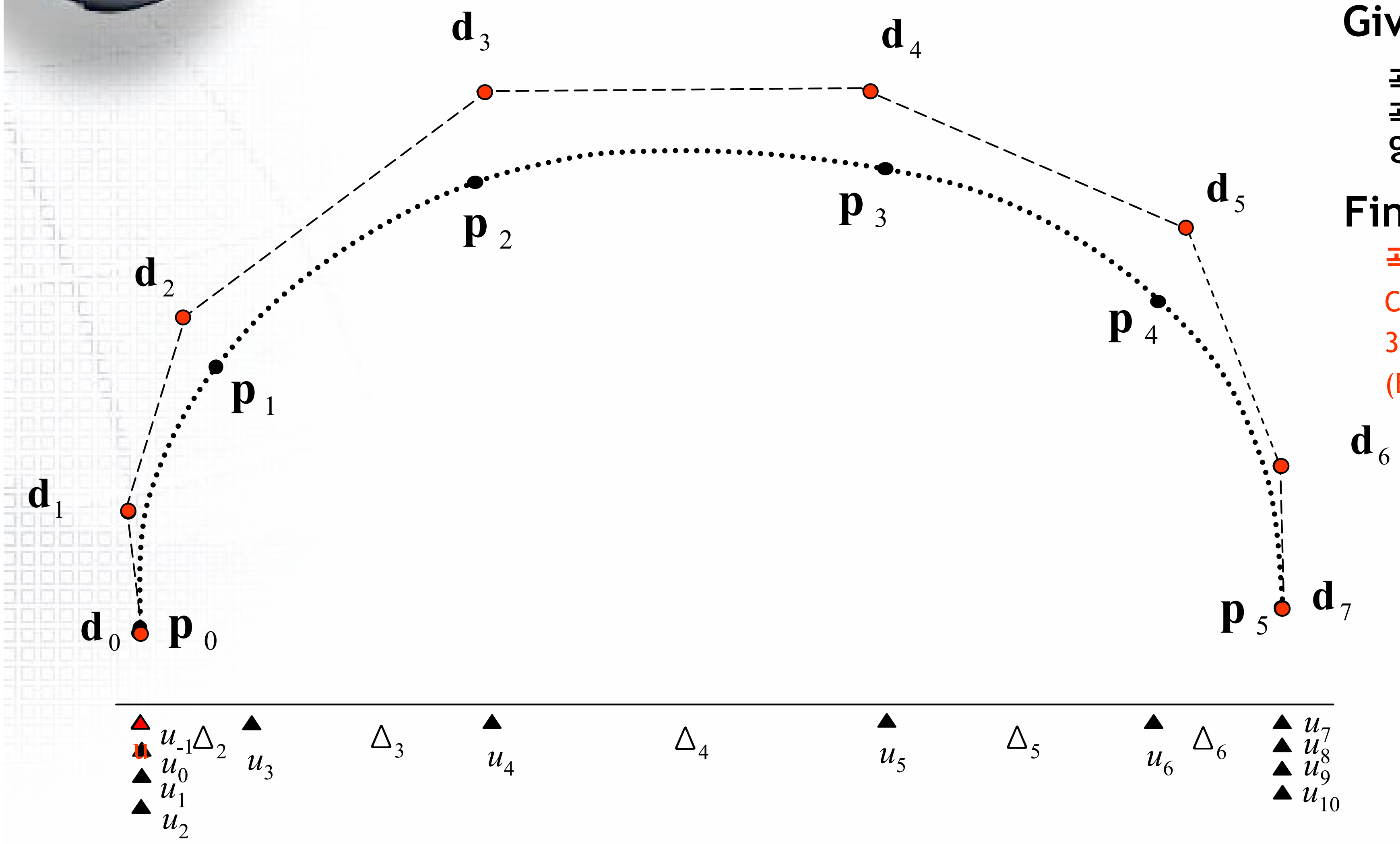
2.3.5.1 Determine # of Bezier curve segment & Knot value (3)

- Given: fitting points $\mathbf{P}_i$ and corresponding parameter t_i
  where, $i = 0,1,\ldots,m$ and $t_0 = 0,\ t_m = 1,$
- ① determine # of Bezier curve segment to be (# of fitting point -1)
- ② We can determine knots to be the same as the parameters t_i
- ③ How about the B-spline control points ?



- 3: degree
- 5: # of Bezier curve segments
- # of control points
  = 4 + (5-1) = 8

2.3.5.2 Problem definition of cubic B-spline curve interpolation



Given:

곡선 상의 점 $\mathbf{p}_i , t_i$

곡선의 놋트 u_j

양끝단의 접선 벡터 $\mathbf{t}_0, \mathbf{t}_1$

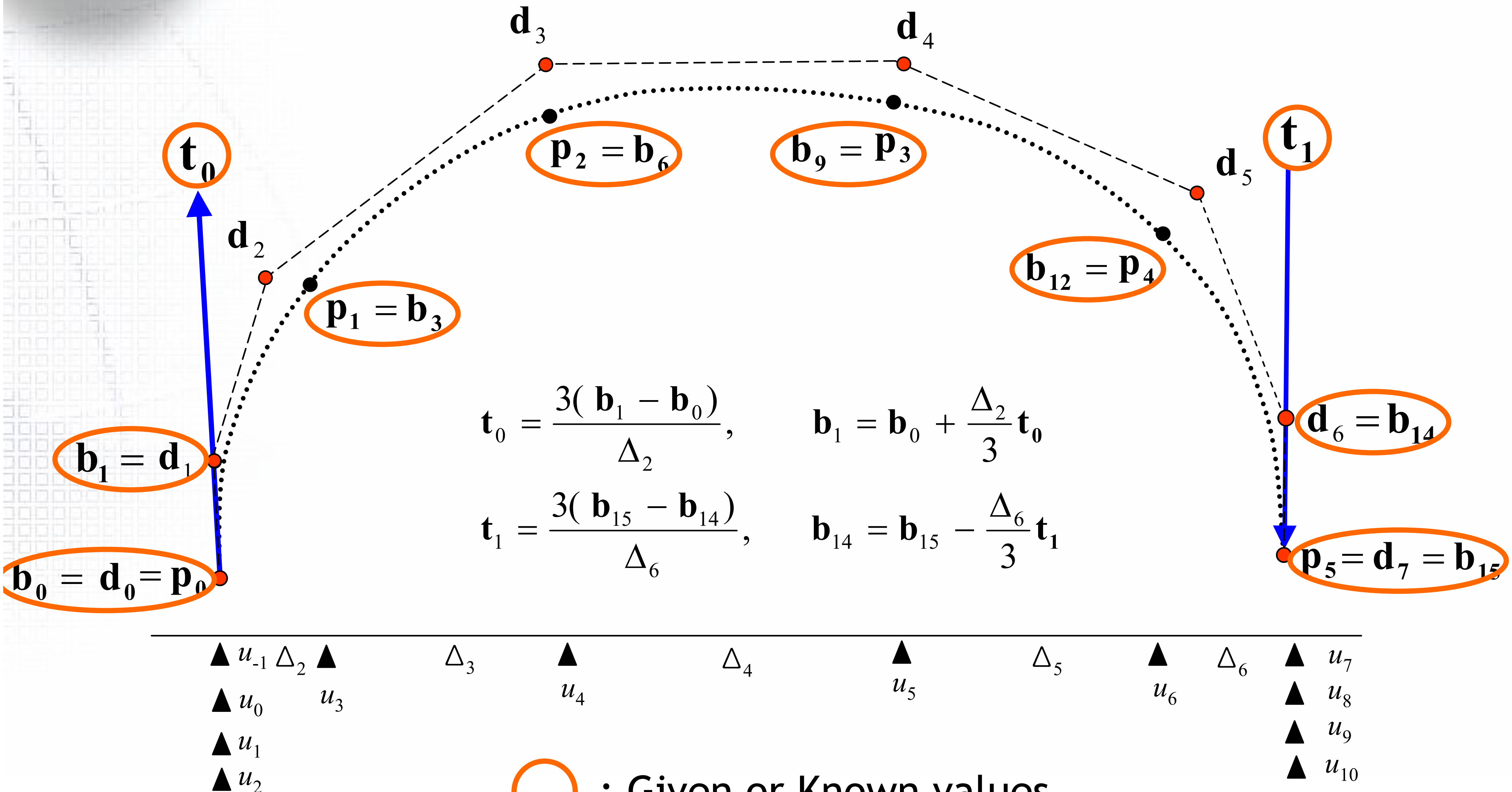
Find:

곡선 상의 점 p_i을 지나고
C^2 연속 조건을 만족하는
3차 B-Spline 곡선
(B-Spline 조정점)

- 3: degree
- 5: # of Bezier curve segments
- # of knot = (5-1) + 2(3+1)
- # of control points
  = 4 + (5-1) = (3+1) + (5-1)

가정 : 각 곡선 세그먼트는 3차 Bezier Curve 이다.
연결점에서는 C^1, C^2 연속조건을 만족한다.

2.3.5.3 Determine **Bezier end control points** by end tangent vectors

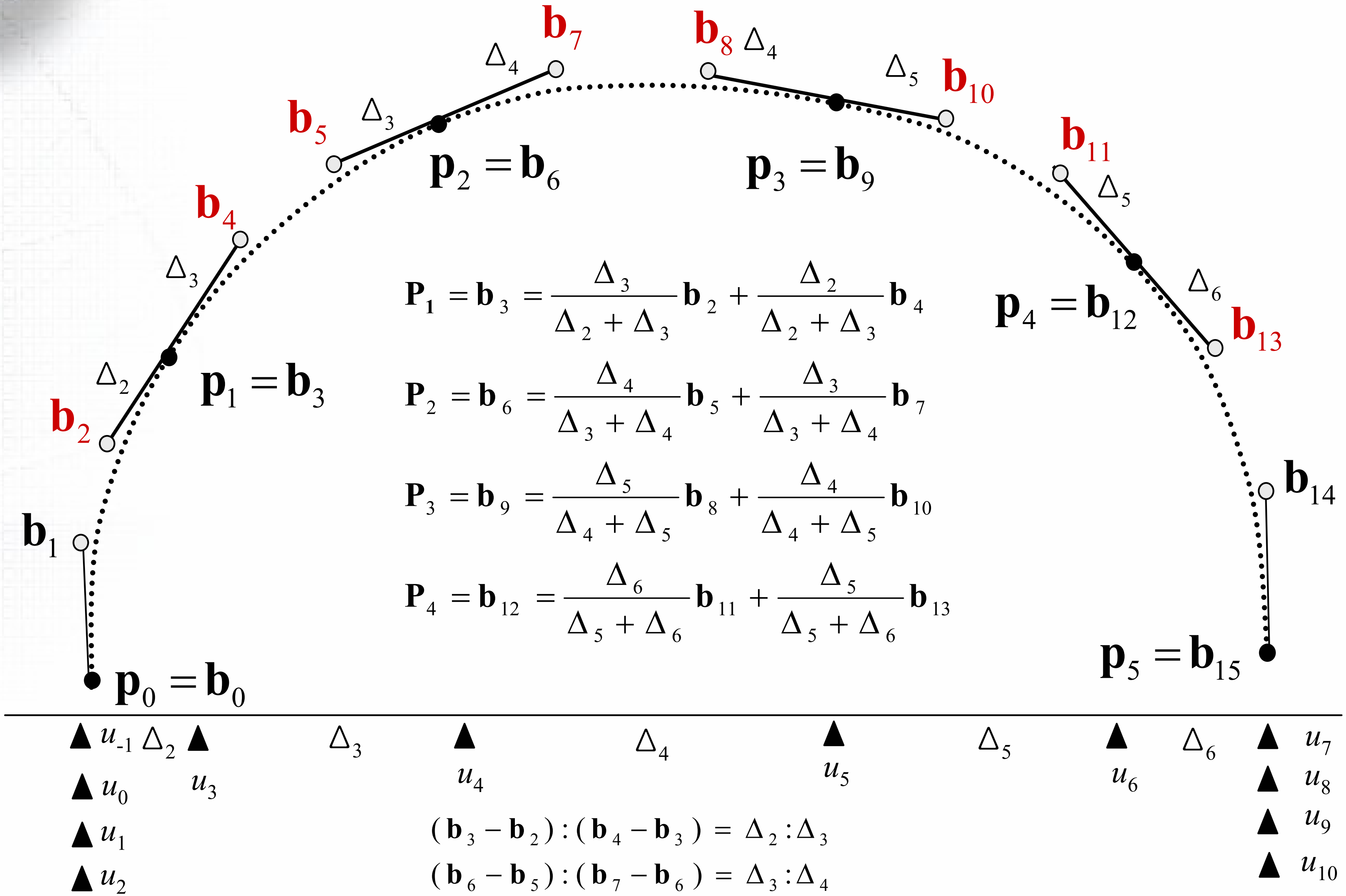


$$\mathbf{t}_0 = \frac{3(\mathbf{b}_1 - \mathbf{b}_0)}{\Delta_2}, \qquad \mathbf{b}_1 = \mathbf{b}_0 + \frac{\Delta_2}{3}\mathbf{t}_0$$

$$\mathbf{t}_1 = \frac{3(\mathbf{b}_{15} - \mathbf{b}_{14})}{\Delta_6}, \qquad \mathbf{b}_{14} = \mathbf{b}_{15} - \frac{\Delta_6}{3}\mathbf{t}_1$$

○ : Given or Known values

2.3.5.4 Determine **Bezier control points** by C^1 continuity condition



$$\mathbf{P}_1 = \mathbf{b}_3 = \frac{\Delta_3}{\Delta_2 + \Delta_3}\mathbf{b}_2 + \frac{\Delta_2}{\Delta_2 + \Delta_3}\mathbf{b}_4$$

$$\mathbf{P}_2 = \mathbf{b}_6 = \frac{\Delta_4}{\Delta_3 + \Delta_4}\mathbf{b}_5 + \frac{\Delta_3}{\Delta_3 + \Delta_4}\mathbf{b}_7$$

$$\mathbf{P}_3 = \mathbf{b}_9 = \frac{\Delta_5}{\Delta_4 + \Delta_5}\mathbf{b}_8 + \frac{\Delta_4}{\Delta_4 + \Delta_5}\mathbf{b}_{10}$$

$$\mathbf{P}_4 = \mathbf{b}_{12} = \frac{\Delta_6}{\Delta_5 + \Delta_6}\mathbf{b}_{11} + \frac{\Delta_5}{\Delta_5 + \Delta_6}\mathbf{b}_{13}$$

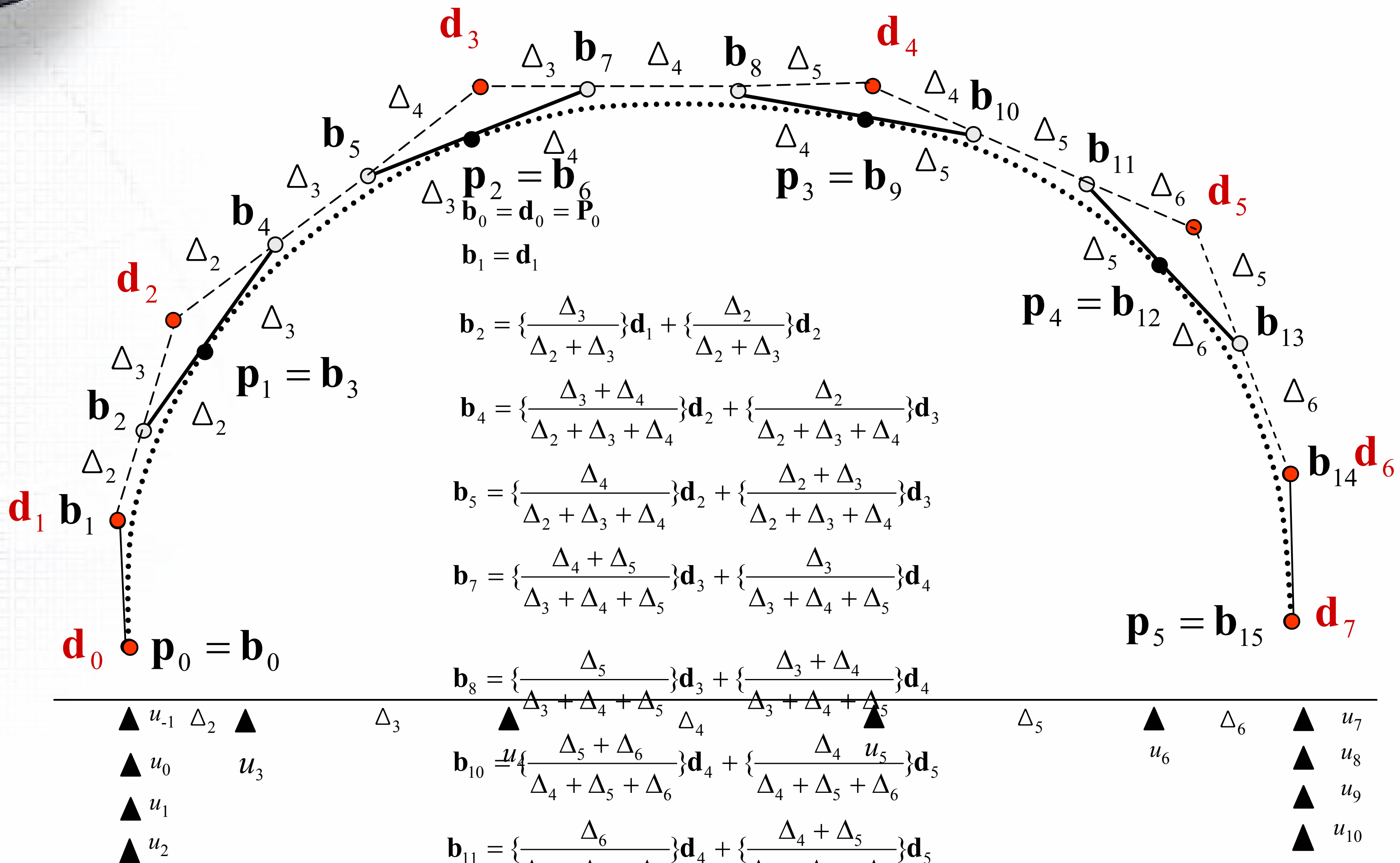
$$(\mathbf{b}_3 - \mathbf{b}_2) : (\mathbf{b}_4 - \mathbf{b}_3) = \Delta_2 : \Delta_3$$

$$(\mathbf{b}_6 - \mathbf{b}_5) : (\mathbf{b}_7 - \mathbf{b}_6) = \Delta_3 : \Delta_4$$

$$(\mathbf{b}_9 - \mathbf{b}_8) : (\mathbf{b}_{10} - \mathbf{b}_9) = \Delta_4 : \Delta_5$$

$$(\mathbf{b}_{12} - \mathbf{b}_{11}) : (\mathbf{b}_{13} - \mathbf{b}_{12}) = \Delta_5 : \Delta_6$$

2.3.5.5 Determine B-spline control points by C^2 continuity condition (1)



$$\mathbf{b}_0 = \mathbf{d}_0 = \mathbf{P}_0$$

$$\mathbf{b}_1 = \mathbf{d}_1$$

$$\mathbf{b}_2 = \{\frac{\Delta_3}{\Delta_2+\Delta_3}\}\mathbf{d}_1 + \{\frac{\Delta_2}{\Delta_2+\Delta_3}\}\mathbf{d}_2$$

$$\mathbf{b}_4 = \{\frac{\Delta_3+\Delta_4}{\Delta_2+\Delta_3+\Delta_4}\}\mathbf{d}_2 + \{\frac{\Delta_2}{\Delta_2+\Delta_3+\Delta_4}\}\mathbf{d}_3$$

$$\mathbf{b}_5 = \{\frac{\Delta_4}{\Delta_2+\Delta_3+\Delta_4}\}\mathbf{d}_2 + \{\frac{\Delta_2+\Delta_3}{\Delta_2+\Delta_3+\Delta_4}\}\mathbf{d}_3$$

$$\mathbf{b}_7 = \{\frac{\Delta_4+\Delta_5}{\Delta_3+\Delta_4+\Delta_5}\}\mathbf{d}_3 + \{\frac{\Delta_3}{\Delta_3+\Delta_4+\Delta_5}\}\mathbf{d}_4$$

$$\mathbf{b}_8 = \{\frac{\Delta_5}{\Delta_3+\Delta_4+\Delta_5}\}\mathbf{d}_3 + \{\frac{\Delta_3+\Delta_4}{\Delta_3+\Delta_4+\Delta_5}\}\mathbf{d}_4$$

$$\mathbf{b}_{10} = \{\frac{\Delta_5+\Delta_6}{\Delta_4+\Delta_5+\Delta_6}\}\mathbf{d}_4 + \{\frac{\Delta_4}{\Delta_4+\Delta_5+\Delta_6}\}\mathbf{d}_5$$

$$\mathbf{b}_{11} = \{\frac{\Delta_6}{\Delta_4+\Delta_5+\Delta_6}\}\mathbf{d}_4 + \{\frac{\Delta_4+\Delta_5}{\Delta_4+\Delta_5+\Delta_6}\}\mathbf{d}_5$$

$$\mathbf{b}_{13} = \{\frac{\Delta_6}{\Delta_5+\Delta_6}\}\mathbf{d}_5 + \{\frac{\Delta_5}{\Delta_5+\Delta_6}\}\mathbf{d}_6$$

$$\mathbf{b}_{14} = \mathbf{d}_6$$

$$\mathbf{b}_{15} = \mathbf{d}_7 = \mathbf{p}_5$$

2.3.5.5 Determine B-spline control points by C^2 continuity condition (2)

C^1, C^2 조건을 이용하여 P_i에 관한 식 유도

$$\mathbf{P}_1 = \mathbf{b}_3 = \frac{\Delta_3}{\Delta_2 + \Delta_3}\mathbf{b}_2 + \frac{\Delta_2}{\Delta_2 + \Delta_3}\mathbf{b}_4$$

$$\mathbf{P}_2 = \mathbf{b}_6 = \frac{\Delta_4}{\Delta_3 + \Delta_4}\mathbf{b}_5 + \frac{\Delta_3}{\Delta_3 + \Delta_4}\mathbf{b}_7$$

$$\mathbf{P}_3 = \mathbf{b}_9 = \frac{\Delta_5}{\Delta_4 + \Delta_5}\mathbf{b}_8 + \frac{\Delta_4}{\Delta_4 + \Delta_5}\mathbf{b}_{10}$$

$$\mathbf{P}_4 = \mathbf{b}_{12} = \frac{\Delta_6}{\Delta_5 + \Delta_6}\mathbf{b}_{11} + \frac{\Delta_5}{\Delta_5 + \Delta_6}\mathbf{b}_{13}$$

$$\mathbf{b}_0 = \mathbf{d}_0 = \mathbf{P}_0$$

$$\mathbf{b}_1 = \mathbf{d}_1$$

$$\mathbf{b}_2 = \{\frac{\Delta_3}{\Delta_2 + \Delta_3}\}\mathbf{d}_1 + \{\frac{\Delta_2}{\Delta_2 + \Delta_3}\}\mathbf{d}_2$$

$$\mathbf{b}_4 = \{\frac{\Delta_3 + \Delta_4}{\Delta_2 + \Delta_3 + \Delta_4}\}\mathbf{d}_2 + \{\frac{\Delta_2}{\Delta_2 + \Delta_3 + \Delta_4}\}\mathbf{d}_3$$

$$\mathbf{b}_5 = \{\frac{\Delta_4}{\Delta_2 + \Delta_3 + \Delta_4}\}\mathbf{d}_2 + \{\frac{\Delta_2 + \Delta_3}{\Delta_2 + \Delta_3 + \Delta_4}\}\mathbf{d}_3$$

$$\mathbf{b}_7 = \{\frac{\Delta_4 + \Delta_5}{\Delta_3 + \Delta_4 + \Delta_5}\}\mathbf{d}_3 + \{\frac{\Delta_3}{\Delta_3 + \Delta_4 + \Delta_5}\}\mathbf{d}_4$$

$$\mathbf{b}_8 = \{\frac{\Delta_5}{\Delta_3 + \Delta_4 + \Delta_5}\}\mathbf{d}_3 + \{\frac{\Delta_3 + \Delta_4}{\Delta_3 + \Delta_4 + \Delta_5}\}\mathbf{d}_4$$

$$\mathbf{b}_{10} = \{\frac{\Delta_5 + \Delta_6}{\Delta_4 + \Delta_5 + \Delta_6}\}\mathbf{d}_4 + \{\frac{\Delta_4}{\Delta_4 + \Delta_5 + \Delta_6}\}\mathbf{d}_5$$

$$\mathbf{b}_{11} = \{\frac{\Delta_6}{\Delta_4 + \Delta_5 + \Delta_6}\}\mathbf{d}_4 + \{\frac{\Delta_4 + \Delta_5}{\Delta_4 + \Delta_5 + \Delta_6}\}\mathbf{d}_5$$

$$\mathbf{b}_{13} = \{\frac{\Delta_6}{\Delta_5 + \Delta_6}\}\mathbf{d}_5 + \{\frac{\Delta_5}{\Delta_5 + \Delta_6}\}\mathbf{d}_6$$

$$\mathbf{b}_{14} = \mathbf{d}_6$$

$$\mathbf{b}_{15} = \mathbf{d}_7 = \mathbf{p}_5$$

2.3.5.5 Determine B-spline control points
by C^2 continuity condition (3)

$$\mathbf{P}_1 = \frac{1}{(\Delta_2+\Delta_3)(\Delta_2+\Delta_3+\Delta_4)}[(\Delta_3)^2(\Delta_2+\Delta_3+\Delta_4)/(\Delta_2+\Delta_3)\mathbf{d}_1$$
$$+\{\Delta_2\Delta_3(\Delta_2+\Delta_3+\Delta_4)+\Delta_2(\Delta_2+\Delta_3)(\Delta_3+\Delta_4)\}/(\Delta_2+\Delta_3)\mathbf{d}_2+(\Delta_2)^2\mathbf{d}_3]$$
$$=\alpha_1\mathbf{d}_1+\beta_1\mathbf{d}_2+\gamma_1\mathbf{d}_3$$

$$\alpha_i = \frac{(\Delta_{i+2})^2}{(\Delta_i+\Delta_{i+1}+\Delta_{i+2})(\Delta_{i+1}+\Delta_{i+2})}$$
$$\beta_i = \{\frac{\Delta_{i+2}(\Delta_i+\Delta_{i+1})}{(\Delta_i+\Delta_{i+1}+\Delta_{i+2})}+\frac{\Delta_{i+1}(\Delta_{i+2}+\Delta_{i+3})}{(\Delta_{i+1}+\Delta_{i+2}+\Delta_{i+3})}\}/(\Delta_{i+1}+\Delta_{i+2})$$
$$\gamma_i = \frac{(\Delta_{i+1})^2}{(\Delta_{i+1}+\Delta_{i+2}+\Delta_{i+3})(\Delta_{i+1}+\Delta_{i+2})}$$

$$\mathbf{P}_2 = \frac{1}{(\Delta_3+\Delta_4)(\Delta_3+\Delta_4+\Delta_5)}[(\Delta_4)^2\mathbf{d}_2+\{\Delta_4(\Delta_2+\Delta_3)+$$
$$\Delta_3(\Delta_4+\Delta_5)\}\mathbf{d}_3+(\Delta_3)^2\mathbf{d}_4]=\alpha_2\mathbf{d}_2+\beta_2\mathbf{d}_3+\gamma_2\mathbf{d}_4$$

$$\mathbf{P}_3 = \frac{1}{(\Delta_4+\Delta_5)(\Delta_3+\Delta_4+\Delta_5)}[(\Delta_5)^2\mathbf{d}_3+\{\Delta_5(\Delta_3+\Delta_4)(\Delta_4+\Delta_5+\Delta_6)$$
$$+\Delta_4(\Delta_5+\Delta_6)(\Delta_3+\Delta_4+\Delta_5)\}/(\Delta_4+\Delta_5+\Delta_6)\mathbf{d}_4+(\Delta_4)^2(\Delta_3+\Delta_4+\Delta_5)$$
$$/(\Delta_4+\Delta_5+\Delta_6)\mathbf{d}_5]=\alpha_3\mathbf{d}_3+\beta_3\mathbf{d}_4+\gamma_3\mathbf{d}_5$$

$$\mathbf{P}_4 = \frac{1}{(\Delta_5+\Delta_6)(\Delta_4+\Delta_5+\Delta_6)}[(\Delta_6)^2\mathbf{d}_4+$$
$$\{\Delta_6(\Delta_4+\Delta_5)+\Delta_5\Delta_6(\Delta_4+\Delta_5+\Delta_6)\}\mathbf{d}_5$$
$$+(\Delta_5)^2(\Delta_4+\Delta_5+\Delta_6)\mathbf{d}_6]=\alpha_4\mathbf{d}_4+\beta_4\mathbf{d}_5+\gamma_4\mathbf{d}_6$$

주어진 것 구해야 하는 것

$$\begin{bmatrix}\mathbf{p}_0\\\mathbf{t}_0\\\mathbf{p}_1\\\mathbf{p}_2\\\mathbf{p}_3\\\mathbf{p}_4\\\mathbf{t}_1\\\mathbf{p}_5\end{bmatrix}=\begin{bmatrix}1&0&0&0&0&0&0&0\\ \frac{-3}{\Delta_2}&\frac{3}{\Delta_2}&0&0&0&0&0&0\\ 0&\alpha_1&\beta_1&\gamma_1&0&0&0&0\\ 0&0&\alpha_2&\beta_2&\gamma_2&0&0&0\\ 0&0&0&\alpha_3&\beta_3&\gamma_3&0&0\\ 0&0&0&0&\alpha_4&\beta_4&\gamma_4&0\\ 0&0&0&0&0&0&\frac{-3}{\Delta_6}&\frac{3}{\Delta_6}\\ 0&0&0&0&0&0&0&1\end{bmatrix}\begin{bmatrix}\mathbf{d}_0\\\mathbf{d}_1\\\mathbf{d}_2\\\mathbf{d}_3\\\mathbf{d}_4\\\mathbf{d}_5\\\mathbf{d}_6\\\mathbf{d}_7\end{bmatrix}$$

2.3.5.6 Tridiagonal matrix 해법을 이용한 B-spline 곡선 조정점($\mathbf{d}_i$) 결정(1)

$$\underbrace{\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{t}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \\ \mathbf{t}_1 \\ \mathbf{p}_5 \end{bmatrix}}_{=\mathbf{D}\ \text{주어진 것}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-3}{\Delta_2} & \frac{3}{\Delta_2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha_1 & \beta_1 & \gamma_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_2 & \beta_2 & \gamma_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_3 & \beta_3 & \gamma_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha_4 & \beta_4 & \gamma_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-3}{\Delta_6} & \frac{3}{\Delta_6} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{=\mathbf{A}\ \text{계산할 수 있는 것}} \underbrace{\begin{bmatrix} \mathbf{d}_0 \\ \mathbf{d}_1 \\ \mathbf{d}_2 \\ \mathbf{d}_3 \\ \mathbf{d}_4 \\ \mathbf{d}_5 \\ \mathbf{d}_6 \\ \mathbf{d}_7 \end{bmatrix}}_{=\mathbf{X}\ \text{구해야 하는 것}}$$

$$\mathbf{D} = \mathbf{A}\mathbf{X}$$

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{D}$$

그런데 행렬 A가 Tri-diagonal matrix**이므로 간단하게 A^{-1}를 계산할 수 있음**

2.3.5.6 Tridiagonal matrix 해법을 이용한 B-spline 곡선 조정점($\mathbf{d}_i$) 결정(2)

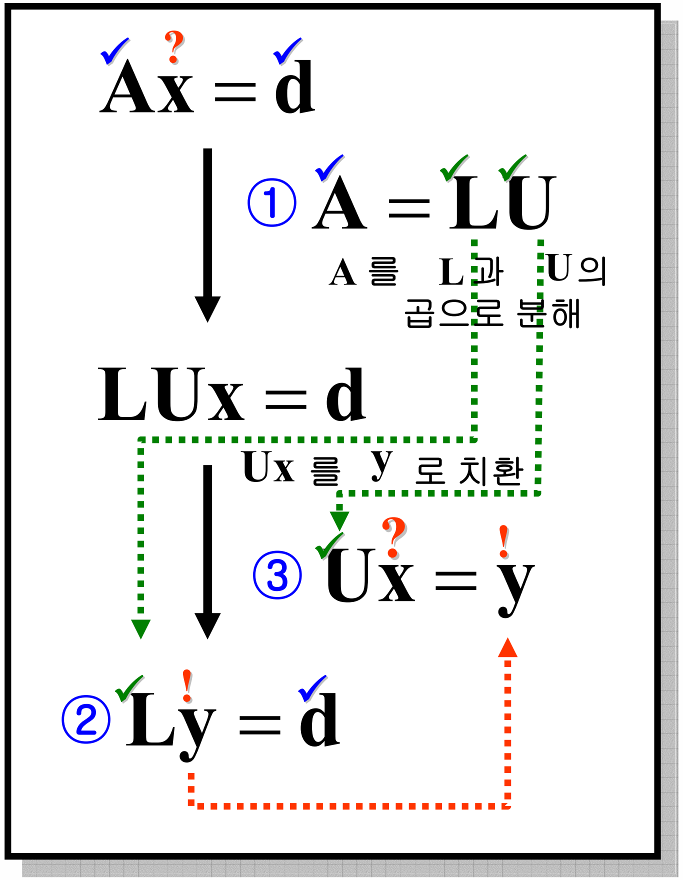
Tridiagonal matrix

대각 성분과 그 위/아래, 좌/우 성분만 0이 아닌 값이고, 나머지 성분은 0 값인 행렬
즉, 대각 성분을 중심으로 3개의 성분만 0이 아닌 값 ➡ Tri + Diagonal

$$\begin{bmatrix} b_0 & c_0 & 0 & & & & \\ a_1 & b_1 & c_1 & 0 & & & \\ 0 & a_2 & b_2 & c_2 & 0 & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

$$\mathbf{A} \qquad \mathbf{x} = \mathbf{d}$$

$\mathbf{A}$와 $\mathbf{d}$를 알고 있을 때, $\mathbf{x}$ 구하기



$$\overset{\checkmark}{\mathbf{A}}\overset{?}{\mathbf{x}} = \overset{\checkmark}{\mathbf{d}}$$

① $\overset{\checkmark}{\mathbf{A}} = \overset{\checkmark}{\mathbf{L}}\overset{\checkmark}{\mathbf{U}}$

$$\mathbf{LUx} = \mathbf{d}$$

③ $\overset{\checkmark}{\mathbf{U}}\overset{?}{\mathbf{x}} = \overset{!}{\mathbf{y}}$

② $\overset{\checkmark}{\mathbf{L}}\overset{!}{\mathbf{y}} = \overset{\checkmark}{\mathbf{d}}$

① $\mathbf{A}$를 $\mathbf{L}$과 $\mathbf{U}$의 곱으로 분해

② $\mathbf{Ly} = \mathbf{d}$를 만족하는 $\mathbf{y}$구하기

③ $\mathbf{Ux} = \mathbf{y}$를 만족하는 $\mathbf{x}$를 구하면, 곧 $\mathbf{Ax} = \mathbf{d}$를 만족하는 $\mathbf{x}$를 구하는 것임

2.3.5.6 Tridiagonal matrix 해법을 이용한 B-spline 곡선 조정점($\mathbf{d}_i$) 결정(3)

① $\overset{\checkmark}{\mathbf{A}} = \overset{\checkmark}{\mathbf{L}}\overset{\checkmark}{\mathbf{U}}$

$$\begin{bmatrix} b_0 & c_0 & 0 & & & & \\ a_1 & b_1 & c_1 & 0 & & & \\ 0 & a_2 & b_2 & c_2 & 0 & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & 0 & a_n & b_n \end{bmatrix} = \begin{bmatrix} \beta_0 & 0 & & & & & \\ \alpha_1 & \beta_1 & 0 & & & & \\ 0 & \alpha_2 & \beta_2 & 0 & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & 0 & \alpha_{n-1} & \beta_{n-1} & 0 \\ & & & & 0 & \alpha_n & \beta_n \end{bmatrix} \begin{bmatrix} 1 & \gamma_1 & 0 & & & & \\ 0 & 1 & \gamma_2 & 0 & & & \\ & 0 & 1 & \gamma_3 & 0 & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & 0 & 1 & \gamma_n \\ & & & & & 0 & 1 \end{bmatrix}$$

$$\mathbf{A} \qquad = \qquad \mathbf{L} \qquad \mathbf{U}$$

$$\overset{\checkmark}{b}_0 = \overset{\checkmark}{\beta}_0 \qquad \overset{\checkmark}{c}_0 = \overset{\checkmark}{\beta}_0\overset{\checkmark}{\gamma}_1$$

$$\overset{\checkmark}{a}_1 = \overset{\checkmark}{\alpha}_1 \qquad \overset{\checkmark}{b}_1 = \overset{\checkmark}{\alpha}_1\overset{\checkmark}{\gamma}_1 + \overset{\checkmark}{\beta}_1 \qquad \overset{\checkmark}{c}_1 = \beta_1\gamma_2$$

$$a_2 = \alpha_2 \qquad b_2 = \alpha_2\gamma_2 + \beta_2 \qquad c_2 = \beta_2\gamma_3$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$a_{n-1} = \alpha_{n-1} \qquad b_{n-1} = \alpha_{n-1}\gamma_{n-1} + \beta_{n-1} \qquad c_{n-1} = \beta_{n-1}\gamma_n$$

$$a_n = \alpha_n \qquad b_n = \alpha_n\gamma_n + \beta_n$$

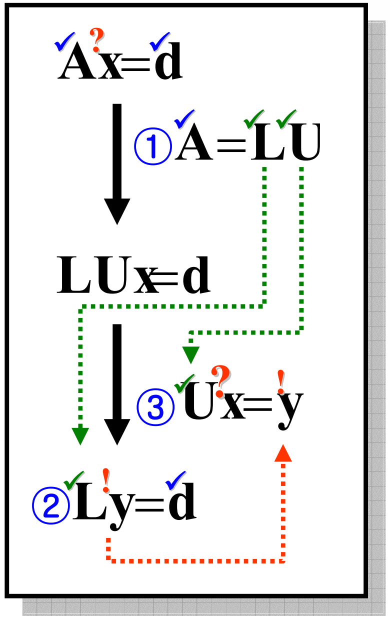
$$\alpha_i = a_i \qquad i = 1, \cdots, n$$

$$\gamma_{i+1} = \frac{c_i}{\beta_i} \qquad i = 0, \cdots, n-1$$

$$\beta_{i+1} = b_{i+1} - \alpha_{i+1}\gamma_{i+1}$$

$$i = 0, \cdots, n-1$$

with $\beta_0 = b_0$



$\overset{\checkmark}{\mathbf{A}}\overset{?}{\mathbf{x}}=\overset{\checkmark}{\mathbf{d}}$

①$\overset{\checkmark}{\mathbf{A}}=\overset{\checkmark}{\mathbf{L}}\overset{\checkmark}{\mathbf{U}}$

$\mathbf{LUx=d}$

③$\overset{\checkmark}{\mathbf{U}}\overset{?}{\mathbf{x}}=\overset{!}{\mathbf{y}}$

②$\overset{\checkmark}{\mathbf{L}}\overset{!}{\mathbf{y}}=\overset{\checkmark}{\mathbf{d}}$

2.3.5.6 Tridiagonal matrix 해법을 이용한 B-spline 곡선 조정점($\mathbf{d}_i$) 결정(4)

$$② \overset{\checkmark}{\mathbf{L}}\overset{!}{\mathbf{y}} = \overset{\checkmark}{\mathbf{d}}$$

$$\begin{bmatrix} \beta_0 & 0 & & & & & \\ \alpha_1 & \beta_1 & 0 & & & & \\ 0 & \alpha_2 & \beta_2 & 0 & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & 0 & \alpha_{n-1} & \beta_{n-1} & 0 \\ & & & & 0 & \alpha_n & \beta_n \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

$$\mathbf{L} \qquad \mathbf{y} = \mathbf{d}$$

$$\overset{\checkmark}{\beta_0}\overset{!}{y_0} = \overset{\checkmark}{d_0}$$

$$\overset{\checkmark}{\alpha_1}\overset{!}{y_0} + \overset{\checkmark}{\beta_1}\overset{!}{y_1} = \overset{\checkmark}{d_1}$$

$$\alpha_2 y_1 + \beta_2 y_2 = d_2$$

$$\vdots$$

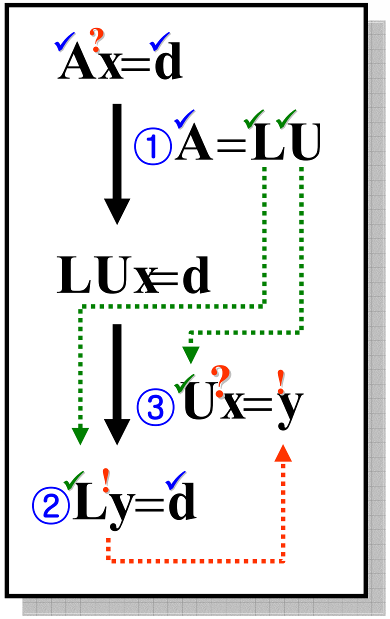
$$\alpha_{n-1} y_{n-2} + \beta_{n-1} y_{n-1} = d_{n-1}$$

$$\alpha_n y_{n-1} + \beta_n y_n = d_n$$

Forward substitution

$$y_i = \frac{d_i - \alpha_i y_{i-1}}{\beta_i} \qquad i = 1, \cdots, n$$

$$\text{with} \quad y_0 = \frac{d_0}{\beta_0}$$



$\overset{\checkmark}{\mathbf{A}}\overset{?}{\mathbf{x}}=\overset{\checkmark}{\mathbf{d}}$

① $\overset{\checkmark}{\mathbf{A}}=\overset{\checkmark}{\mathbf{L}}\overset{\checkmark}{\mathbf{U}}$

$\mathbf{LUx}=\mathbf{d}$

③ $\overset{\checkmark}{\mathbf{U}}\overset{?}{\mathbf{x}}=\overset{!}{\mathbf{y}}$

② $\overset{\checkmark}{\mathbf{L}}\overset{!}{\mathbf{y}}=\overset{\checkmark}{\mathbf{d}}$

2.3.5.6 Tridiagonal matrix 해법을 이용한 B-spline 곡선 조정점($\mathbf{d}_i$) 결정(5)

$$③\ \overset{\checkmark}{\mathbf{U}}\overset{?}{\mathbf{x}} = \overset{!}{\mathbf{y}}$$

$$\begin{bmatrix} 1 & \gamma_1 & 0 & & & & \\ 0 & 1 & \gamma_2 & 0 & & & \\ & 0 & 1 & \gamma_3 & 0 & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & 0 & 1 & \gamma_n \\ & & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}$$

$$\mathbf{U} \qquad \mathbf{x} \quad = \quad \mathbf{y}$$

$$x_0 + \gamma_0 x_1 = y_0$$

$$x_1 + \gamma_1 x_2 = y_1$$

$$x_2 + \gamma_2 x_3 = y_2$$

$$\vdots$$

$$\overset{?}{x}_{n-1} + \overset{\checkmark}{\gamma}_{n-1}\overset{?}{x}_n = \overset{!}{y}_{n-1}$$

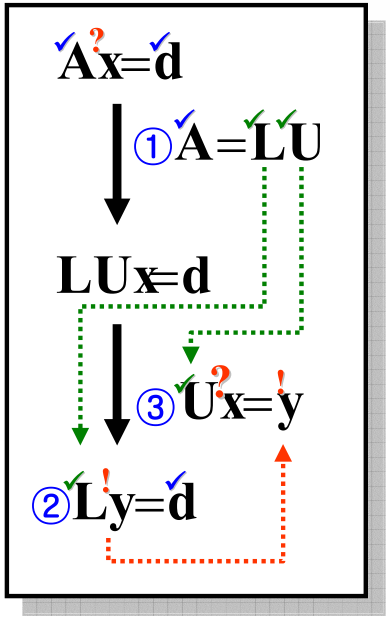
$$\overset{?}{x}_n = \overset{!}{y}_n$$

Backward substitution

$$x_i = y_i - \gamma_{i+1} x_{i+1}$$

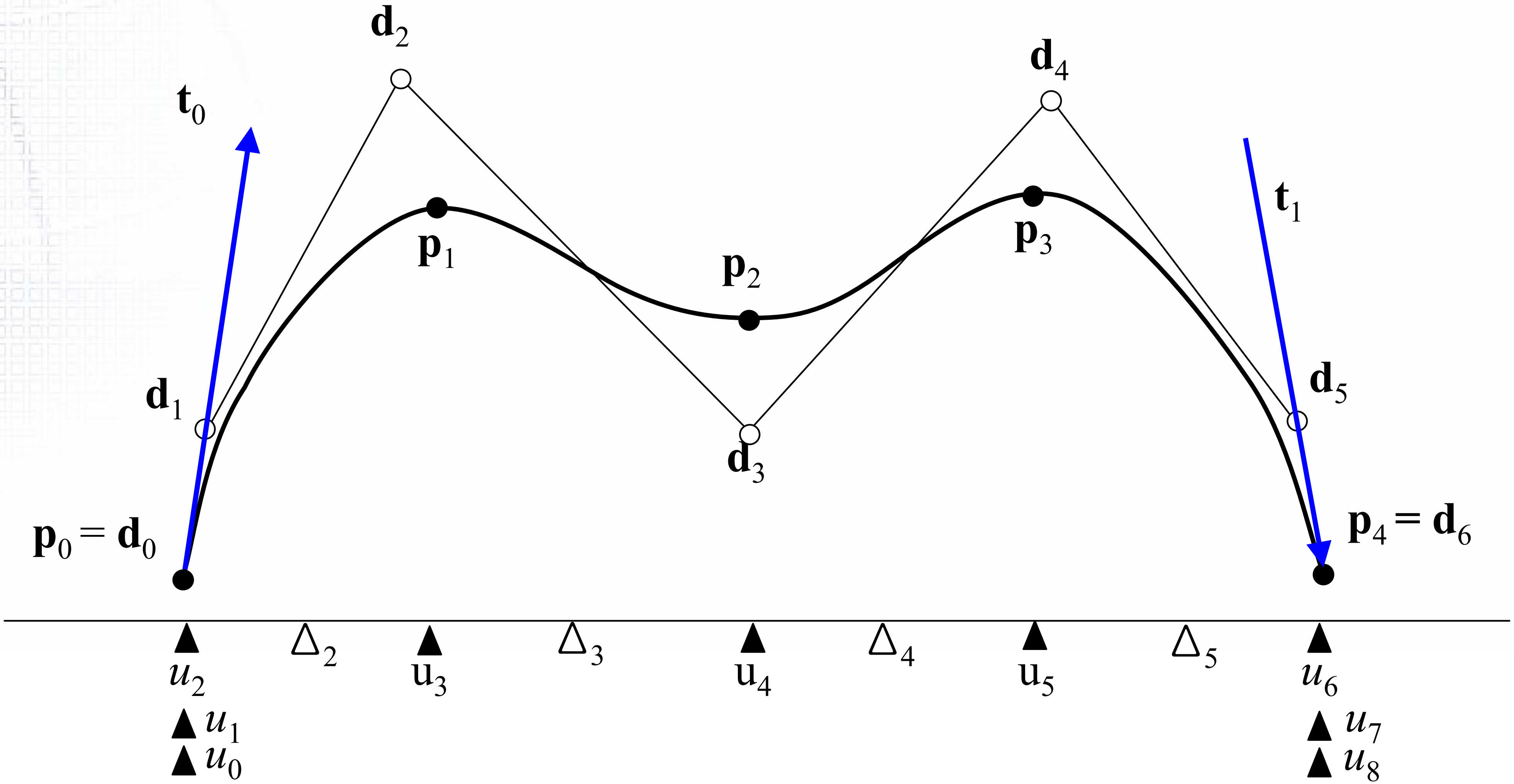
$$i = n-1, \cdots, 0$$

$$\text{with} \quad x_n = y_n$$

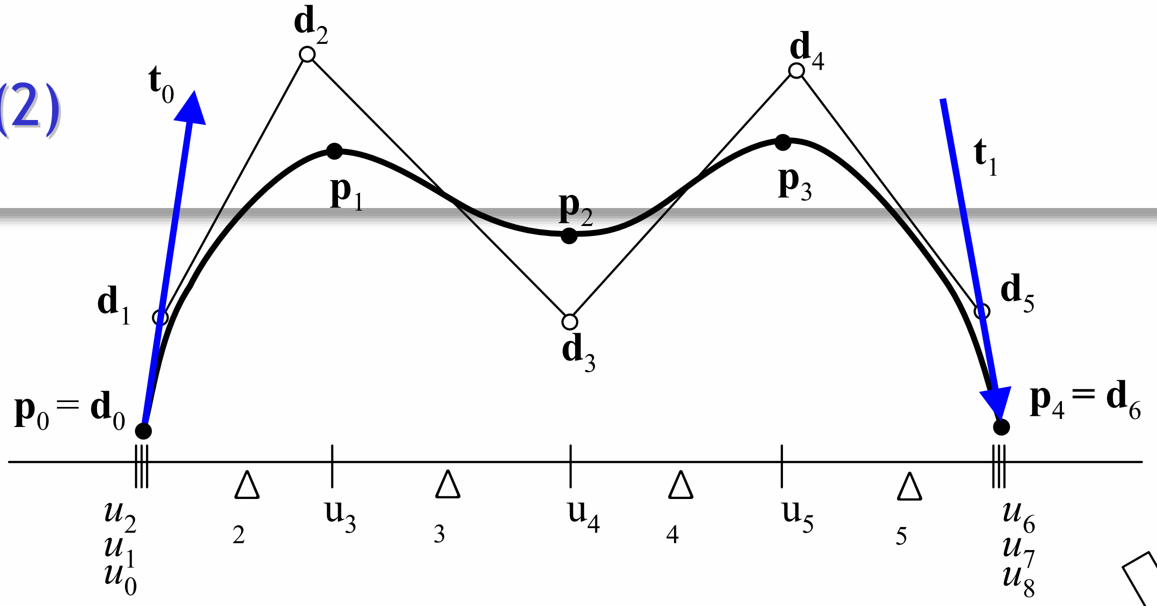


2.3.5.7 Bessel End Condition (1)

- B-spline curve interpolation에서 양끝점에서의 접선벡터 t_0, t_1이 주어지지 않았을 때,
  (1) 곡선의 양끝의 연속된 세 점으로부터 2차 곡선(quadratic curve)을 생성하고,
  (2) 생성된 2차 곡선의 양 끝점에서의 1차 미분값을 우리가 생성하고자 하는
  B-spline curve의 양 끝점에서의 접선 벡터로 가정하는 방법

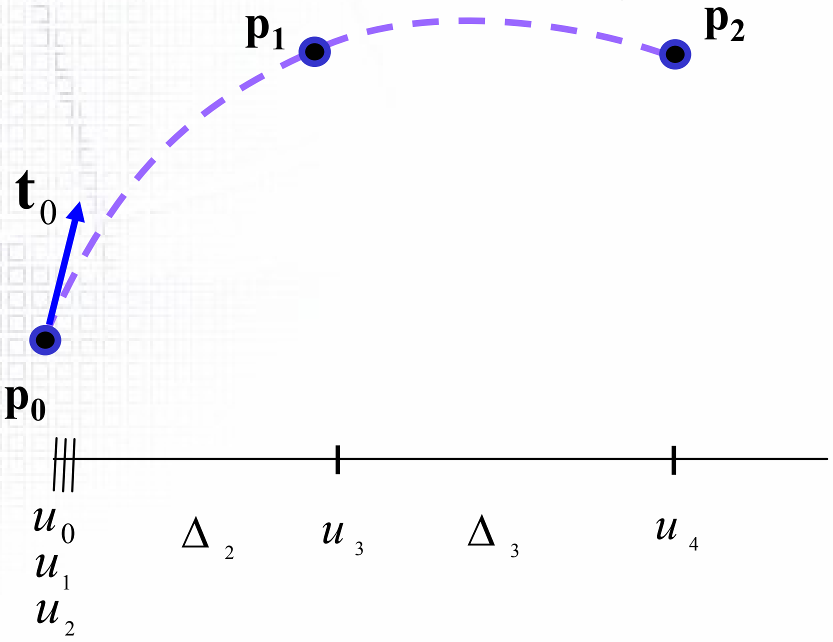


2.3.5.7 Bessel End Condition(2)

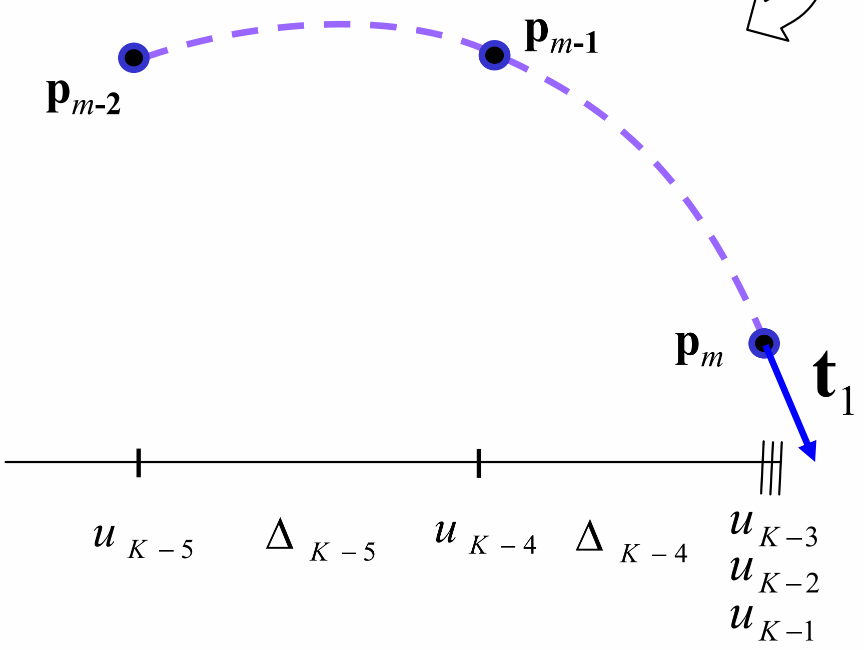


곡선의 시작 부분

곡선의 끝 부분



$$\mathbf{t}_s = \left(-\frac{2\Delta_2+\Delta_3}{\Delta_2(\Delta_2+\Delta_3)}\mathbf{p}_0 + \frac{(\Delta_2+\Delta_3)}{\Delta_2\Delta_3}\mathbf{p}_1 - \frac{\Delta_2}{\Delta_3(\Delta_2+\Delta_3)}\mathbf{p}_2\right)$$



$$\mathbf{t}_e = \left(\frac{\Delta_{K-4}}{\Delta_{K-5}(\Delta_{K-5}+\Delta_{K-4})}\mathbf{p}_{m-2} - \frac{(\Delta_{K-5}+\Delta_{K-4})}{\Delta_{K-5}\Delta_{K-4}}\mathbf{p}_{m-1} + \frac{(2\Delta_{K-4}+\Delta_{K-5})}{(\Delta_{K-5}+\Delta_{K-4})\Delta_{K-4}}\mathbf{p}_m\right)$$

2.3.5.8 Sample code of Cubic B-spline Curve (1)

```
#ifndef __CubicBSpline_h__
#define __CubicBSpline_h__

#include "vector.h“

class CubicBSplineCurve {
public:
    Vector* m_ControlPoint;   int m_nControlPoint;
    double* m_Knot; int m_nKnot;  int m_nDegree;

    .........

    void SetControlPoint(Vector* pControlPoint, int nControlPoint);
    void SetKnot(double* pKnot, int nKnot);
    Vector CalcPoint(double u);
    double N(int d, int i, double u);
    void Interpolate(Vector *pFittingPoint, int nFittingPoint);
    void Parameterization(int nType, Vector* FittingPoint, int nPoint, double* t);
};
#endif
```

2.3.5.8 Sample code of Cubic B-spline Curve (2)

```
void CubicBSplineCurve::Interpolate(Vector *pFittingPoint, int nFittingPoint)
{
  // Generate Knot
    if(m_Knot) delete[] m_Knot;
    m_nKnot = (m_nFittingPoint - 2) + 2*(3+1);
    m_Knot = new double [m_nKnot];
    // Use Chord length or Centripetal method
    ……

//---------------------------------------
// Generate Matrix : (L+1) * (L+1)
   int L = m_nFittingPoint + 1;              // (L+1)*(L+1) size Matrix

  // Fill rhs
    Vector* rhs = new Vector[L+1];
    for(i = 1; i <= L-1 ; i++) rhs[i] = pFittingPoint[i-1];

  // Bessel End condition
    rhs[0] = rhs[1]; rhs[L] = rhs[L-1];
    rhs[1] = StartTangentByBesselEndCondition;   rhs[L-1] = EndTangentByBesselEndCondition;
```

2.3.5.8 Sample code of Cubic B-spline Curve (3)

```
    double* alpha = new double[L+1];
    double* beta = new double[L+1];
    double* gamma = new double[L+1];
    double* up = new double[L+1];
    double* low = new double[L+1];
    if(m_ControlPoint) delete[] m_ControlPoint;
    m_nControlPoint = L+1;
    m_ControlPoint = new Vector[m_nControlPoint];
  // Fill alpha, beta, gamma
    ……
  // Solve LU system
    l_u_system(alpha, beta, gamma, L, up, low);
    solve_system(up, low, gamma, L, rhs, m_ControlPoint);

//------------------------------
// Release memory
  delete[] rhs;   delete[] alpha;   delete[] beta;   delete[] gamma;   delete[] up;   delete[] low;
}
```

Ch 3. 곡면(Surfaces)

3.1 Parametric Surfaces

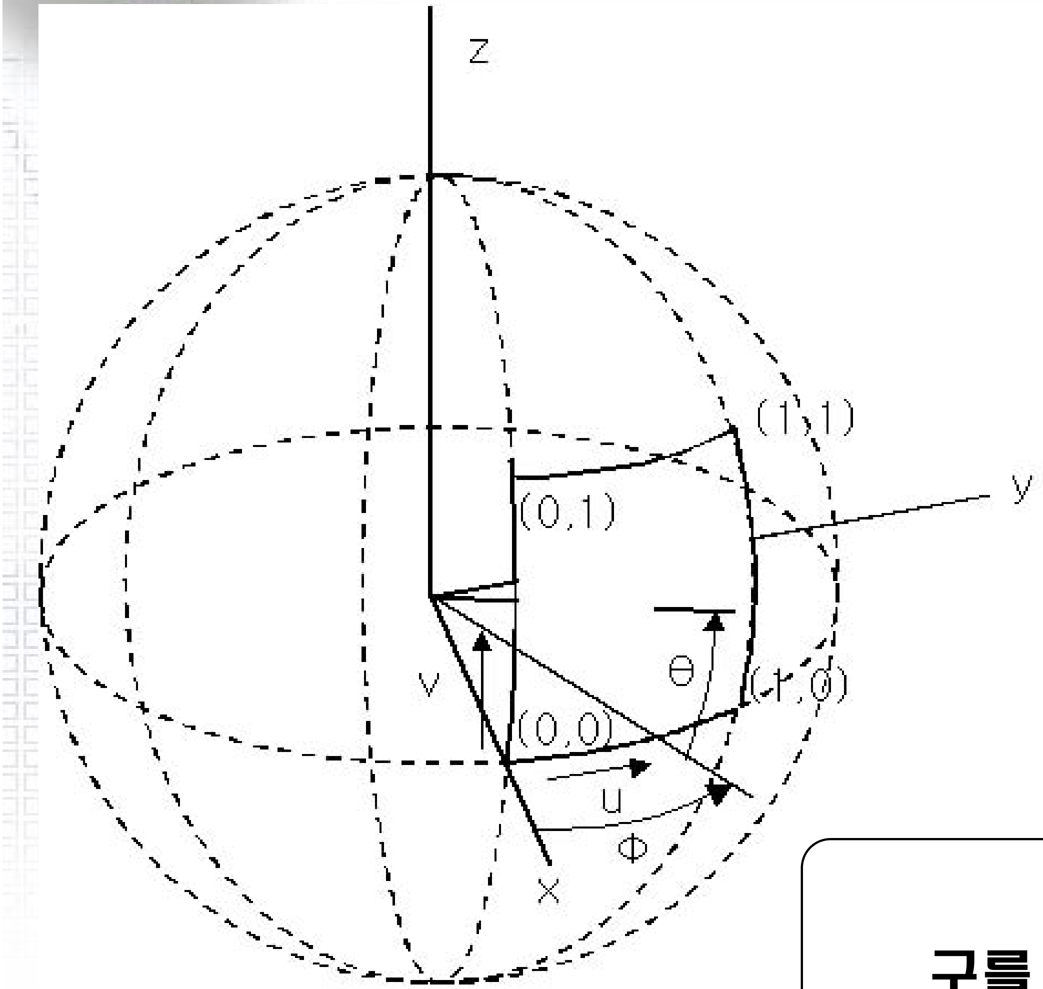
3.2 Bezier Surfaces

3.3 B-spline surfaces

3.1 Parametric Surfaces

Advanced
Ship
Design
Automation
Laboratory

2.1 Parametric Surfaces



	곡면
양함수	$z = \pm\sqrt{d^2 - x^2 - y^2}$
음함수	$x^2 + y^2 + z^2 = d^2$
매개변수	$x = d\cos\phi\cos\theta$ $y = d\sin\phi\cos\theta$ $z = d\sin\theta$
	$\mathbf{r} = r(x(\phi,\theta), y(\phi,\theta), z(\phi,\theta))$

구를 2개의 매개변수 (ϕ, θ) 로 표현하면 간단히 수식화 할 수 있다.

3.2 Bezier surfaces

3.2.1 Generation of Bezier surfaces by de Casteljau algorithm

3.2.2 Generation of Bezier surfaces by tensor-product approach

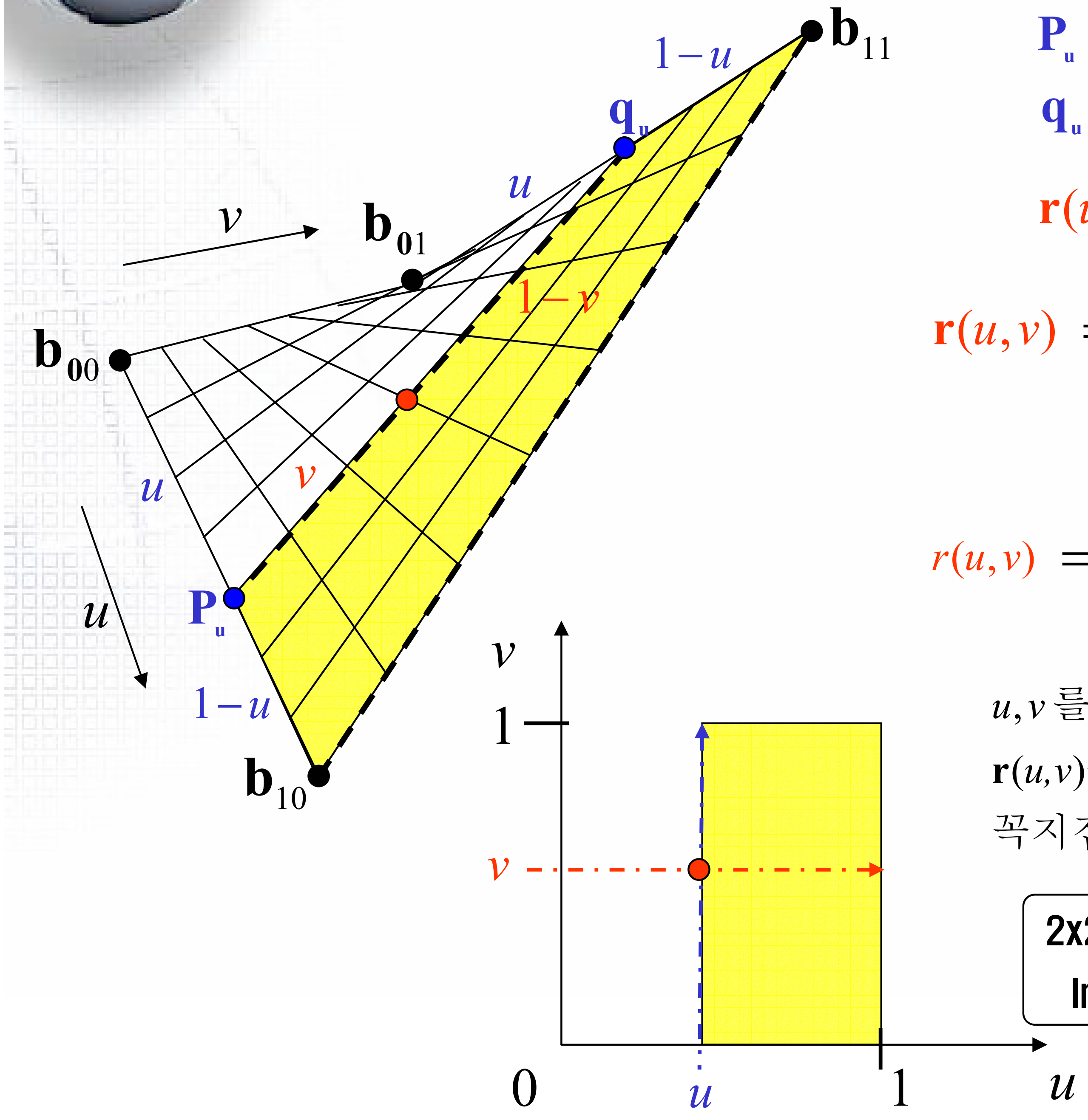
3.2.1 Generation of Bezier surfaces by de Casteljau algorithm

3.2.1.1 Bi-linear Bezier Surface Patch

3.2.1.2 Bi-quadratic Bezier Surface Patch

3.2.1.3 Bi-Cubic Bezier Surface Patch

3.2.1.1 Given : 2x2 Bezier control point
Find : Bi-linear Bezier Surface Patch
방법: u,v 방향으로 'de Casteljau algorithm' 적용



$$\mathbf{P}_u = (1-u)\mathbf{b}_{00} + u\,\mathbf{b}_{10}$$

$$\mathbf{q}_u = (1-u)\mathbf{b}_{01} + u\,\mathbf{b}_{11}$$

$$\mathbf{r}(u,v) = (1-v)\,\mathbf{P}_u + v\,\mathbf{q}_u$$

$$\mathbf{r}(u,v) = (1-v)\,(1-u)\mathbf{b}_{00} + (1-v)\;u\;\mathbf{b}_{10} + v\,(1-u)\mathbf{b}_{01} + v\,u\;\mathbf{b}_{11}$$

$$r(u,v) = \begin{bmatrix}(1-v) & v\end{bmatrix}\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{10}\\ \mathbf{b}_{01} & \mathbf{b}_{11}\end{bmatrix}\begin{bmatrix}(1-u)\\ u\end{bmatrix}$$

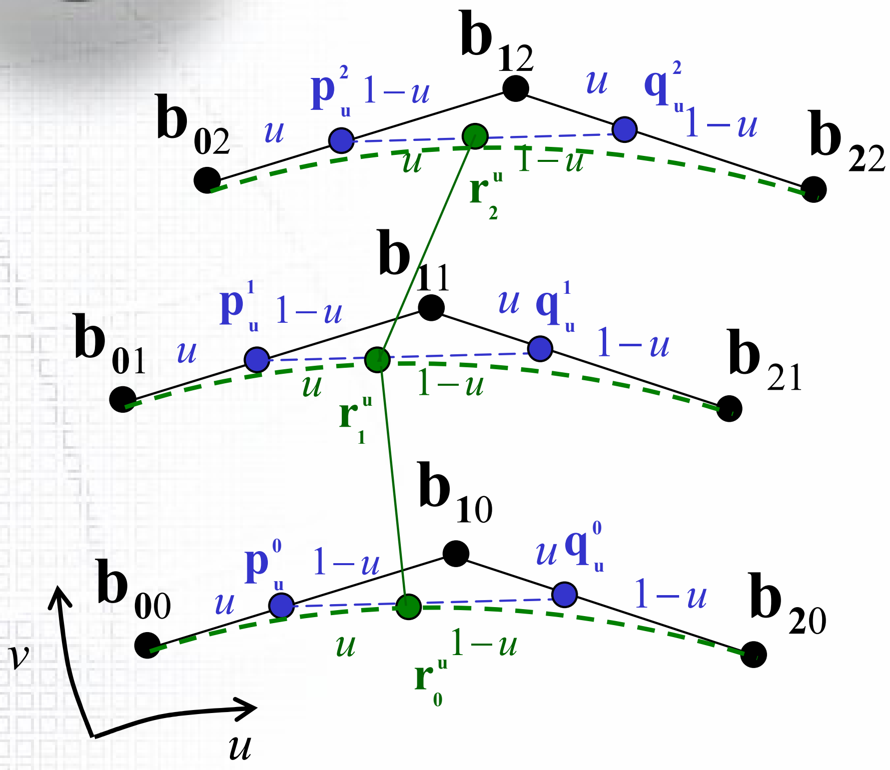
u,v를 0에서 1까지 증가하며 $\mathbf{r}(u,v)$를 계산하면 점 $\mathbf{b}_{00},\mathbf{b}_{10},\mathbf{b}_{01},\mathbf{b}_{11}$ 을 꼭지점으로 하는 곡면을 얻을 수 있다.

2x2개의 Bezier 조정점을 이용하여 Bi-linear Interpolation 으로 곡면식을 구할 수 있다.

3.2.1.2 Given : 3x3 Bezier control point

Find : Bi-quadratic Bezier Surface Patch

방법: de Casteljau algorithm



$$\mathbf{P}^0_u = (1-u)\mathbf{b}_{00} + u\,\mathbf{b}_{10}$$
$$\mathbf{q}^0_u = (1-u)\mathbf{b}_{10} + u\,\mathbf{b}_{20}$$

$$\mathbf{P}^1_u = (1-u)\mathbf{b}_{01} + u\,\mathbf{b}_{11}$$
$$\mathbf{q}^1_u = (1-u)\mathbf{b}_{11} + u\,\mathbf{b}_{21}$$

$$\mathbf{P}^2_u = (1-u)\mathbf{b}_{02} + u\,\mathbf{b}_{12}$$
$$\mathbf{q}^2_u = (1-u)\mathbf{b}_{12} + u\,\mathbf{b}_{22}$$

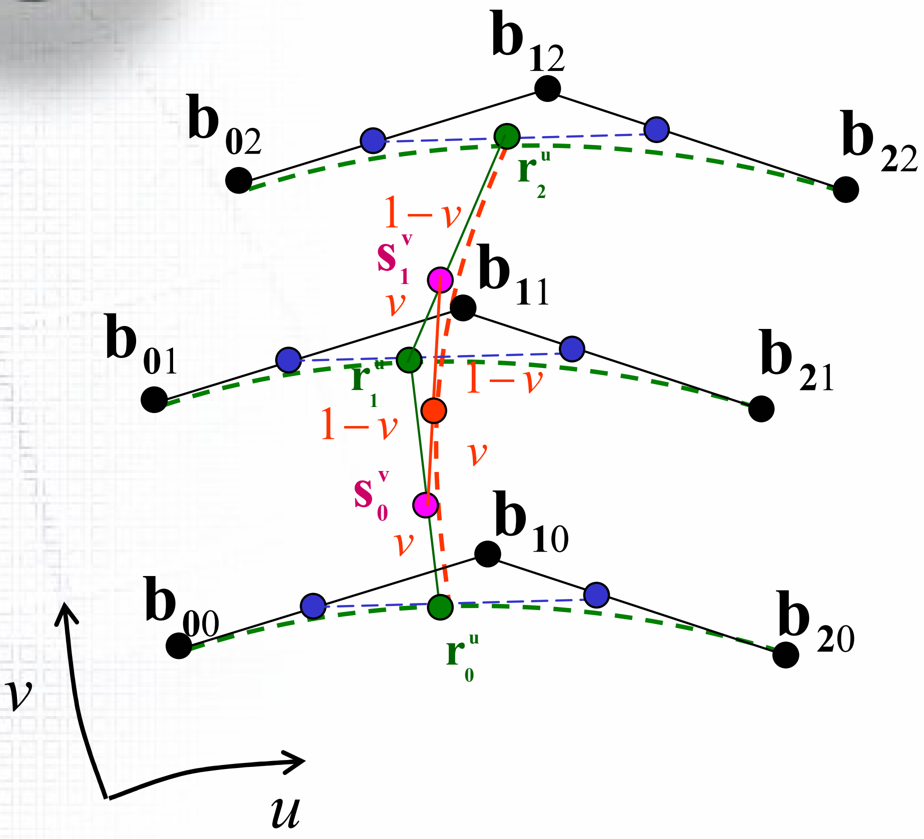
$$\mathbf{r}^u_0 = (1-u)\,\mathbf{p}^0_u + u\,\mathbf{q}^0_u$$

$$\mathbf{r}^u_1 = (1-u)\,\mathbf{p}^1_u + u\,\mathbf{q}^1_u$$
$$\mathbf{r}^u_2 = (1-u)\,\mathbf{p}^2_u + u\,\mathbf{q}^2_u$$

$$\begin{bmatrix} \mathbf{r}^u_0 \\ \mathbf{r}^u_1 \\ \mathbf{r}^u_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{10} & \mathbf{b}_{20} \\ \mathbf{b}_{01} & \mathbf{b}_{11} & \mathbf{b}_{21} \\ \mathbf{b}_{02} & \mathbf{b}_{12} & \mathbf{b}_{22} \end{bmatrix} \begin{bmatrix} (1-u)^2 \\ 2(1-u)u \\ u^2 \end{bmatrix}$$

$$\mathbf{r}^u_0 = (1-u)^2\mathbf{b}_{00} + 2(1-u)u\,\mathbf{b}_{10} + u^2\mathbf{b}_{20}$$
$$\mathbf{r}^u_1 = (1-u)^2\mathbf{b}_{01} + 2(1-u)u\,\mathbf{b}_{11} + u^2\mathbf{b}_{21}$$
$$\mathbf{r}^u_2 = (1-u)^2\mathbf{b}_{02} + 2(1-u)u\,\mathbf{b}_{12} + u^2\mathbf{b}_{22}$$

3.2.1.2 Given : 3x3 Bezier control point
Find : Bi-quadratic Bezier Surface Patch
방법: de Casteljau algorithm



$$\mathbf{s}^{v}_{0} = (1-v)\ \mathbf{r}^{u}_{0} + v\ \mathbf{r}^{u}_{1}$$

$$\mathbf{s}^{v}_{1} = (1-v)\ \mathbf{r}^{u}_{1} + v\ \mathbf{r}^{u}_{2}$$

$$r(u,v) = (1-v)\ \mathbf{s}^{v}_{0} + v\ \mathbf{s}^{v}_{1}$$

$$r(u,v) = (1-v)^2\ \mathbf{r}^{u}_{0} + 2(1-v)v\ \mathbf{r}^{u}_{1} + v^2\ \mathbf{r}^{u}_{2}$$

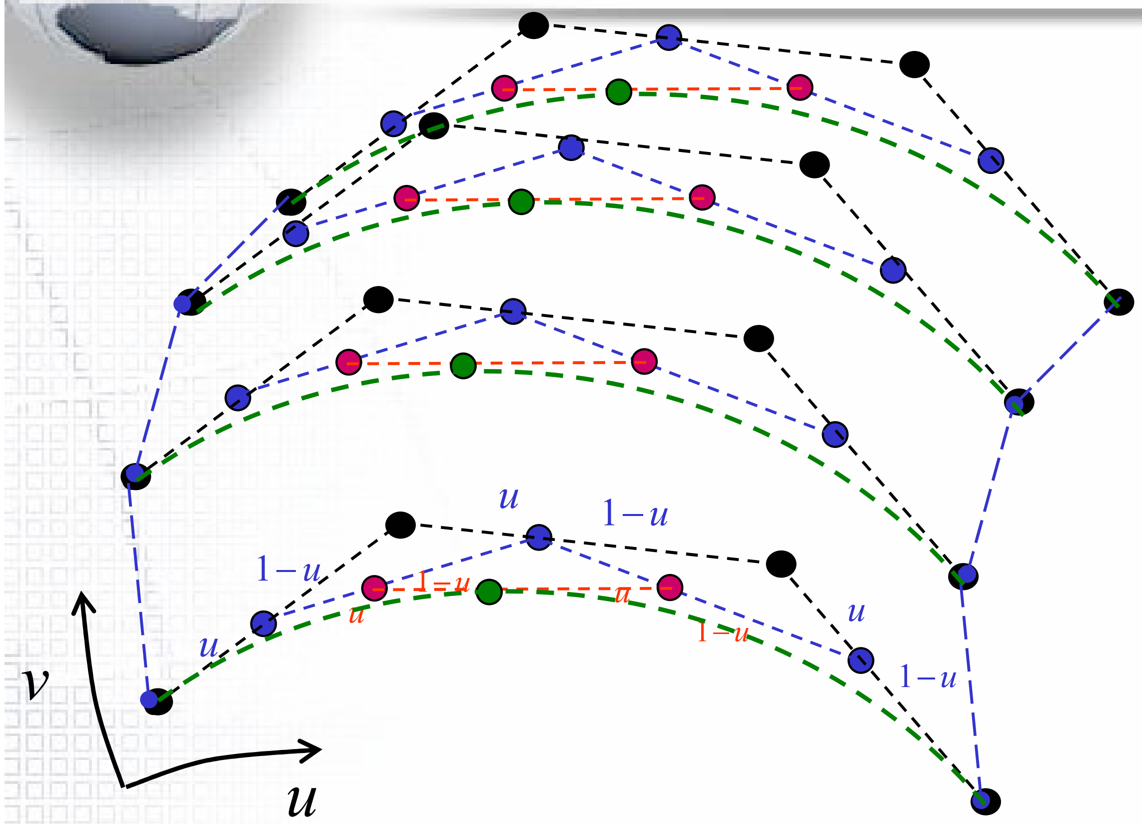
$$r(u,v) = \begin{bmatrix}(1-v)^2 & 2(1-v)v & v^2\end{bmatrix}\begin{bmatrix}\mathbf{r}^{u}_{0}\\ \mathbf{r}^{u}_{1}\\ \mathbf{r}^{u}_{2}\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{r}^{u}_{0}\\ \mathbf{r}^{u}_{1}\\ \mathbf{r}^{u}_{2}\end{bmatrix} = \begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{10} & \mathbf{b}_{20}\\ \mathbf{b}_{01} & \mathbf{b}_{11} & \mathbf{b}_{21}\\ \mathbf{b}_{02} & \mathbf{b}_{12} & \mathbf{b}_{22}\end{bmatrix}\begin{bmatrix}(1-u)^2\\ 2(1-u)u\\ u^2\end{bmatrix}$$

$$r(u,v) = \begin{bmatrix}(1-v)^2 & 2(1-v)v & v^2\end{bmatrix}\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{10} & \mathbf{b}_{20}\\ \mathbf{b}_{01} & \mathbf{b}_{11} & \mathbf{b}_{21}\\ \mathbf{b}_{02} & \mathbf{b}_{12} & \mathbf{b}_{22}\end{bmatrix}\begin{bmatrix}(1-u)^2\\ 2(1-u)u\\ u^2\end{bmatrix}$$

3x3개의 조정점을 이용하여 Bi-Quadratic Bezier Patch를 구할 수 있다.

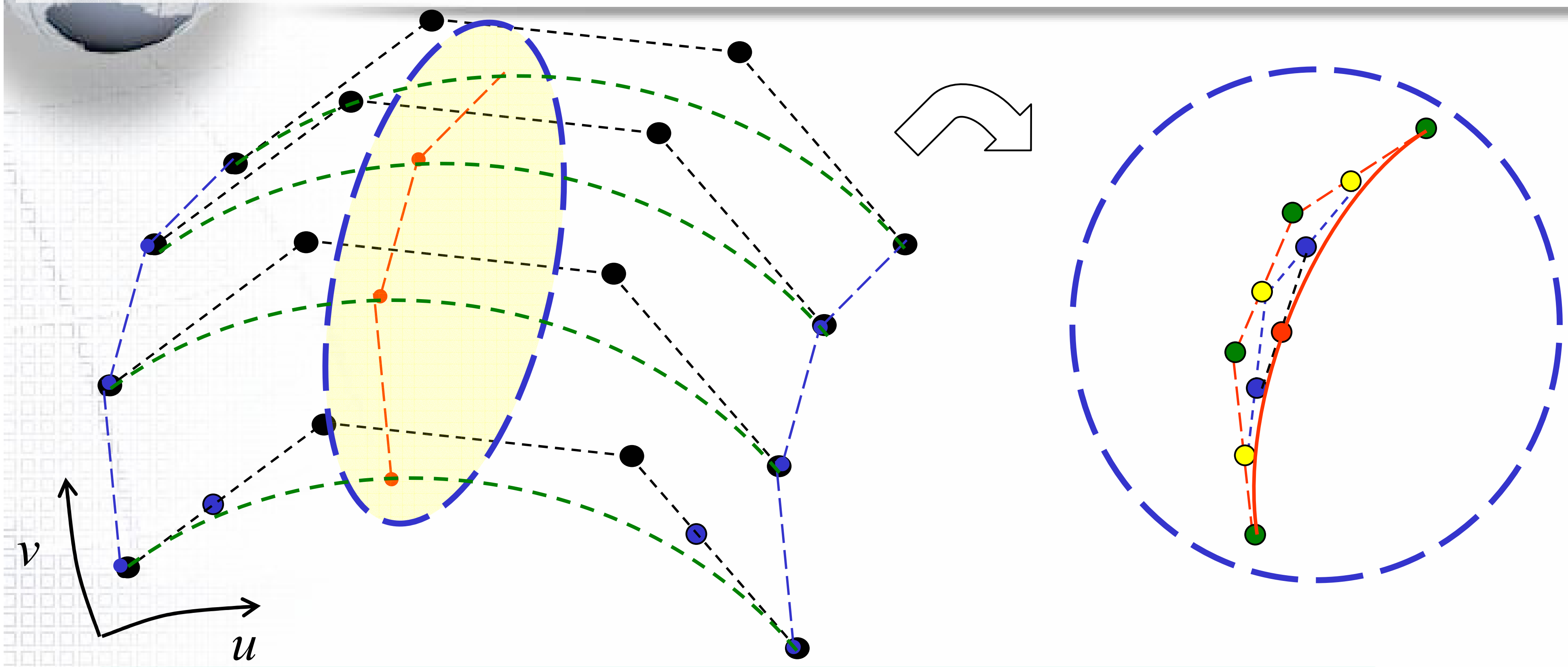
3.2.1.3 Given : 4x4 Bezier control point
Find : Bi-Cubic Bezier Surface Patch by de Casteljau algorithm



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

4x4개의 조정점을 이용하여 Bi-Cubic Bezier Patch를 구할 수 있다.

3.2.1.3 Given : 4x4 Bezier control point
Find : Bi-Cubic Bezier Surface Patch by de Casteljau algorithm



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

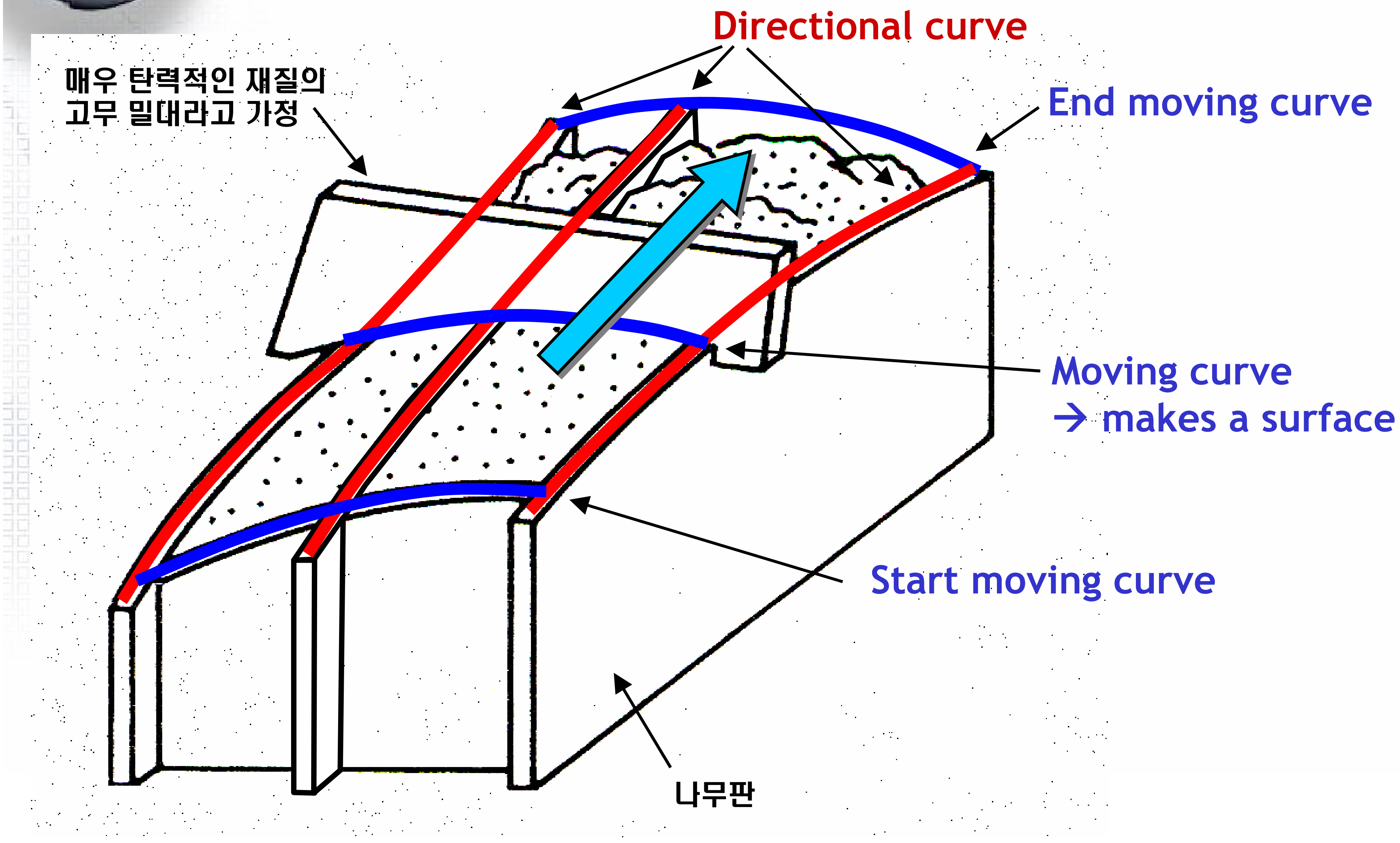
4x4개의 조정점을 이용하여 Bi-Cubic
Bezier Patch를 구할 수 있다.

3.2.2 Generation of Bezier surfaces by tensor-product approach

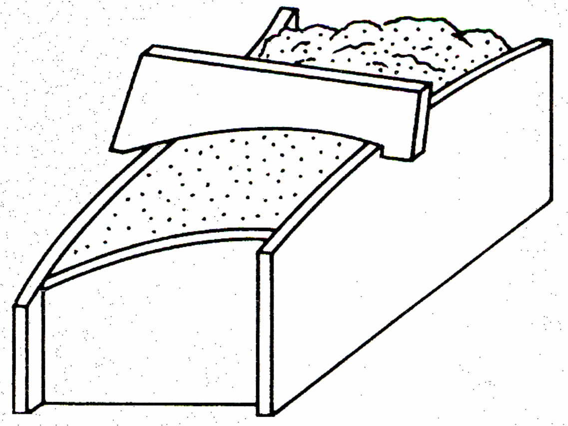
3.2.2.1 Tensor-product approach
3.2.2.2 Tensor-product biquadratic Bezier surface
3.2.2.3 Tensor-product bicubic Bezier surface

Advanced
Ship
Design
Automation
Laboratory

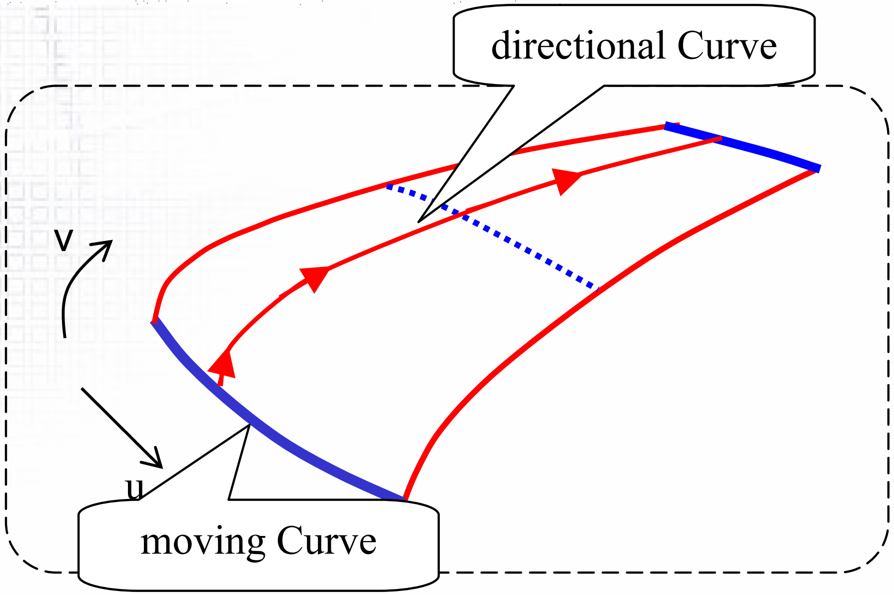
3.2.2.1 Tensor product approach (1)



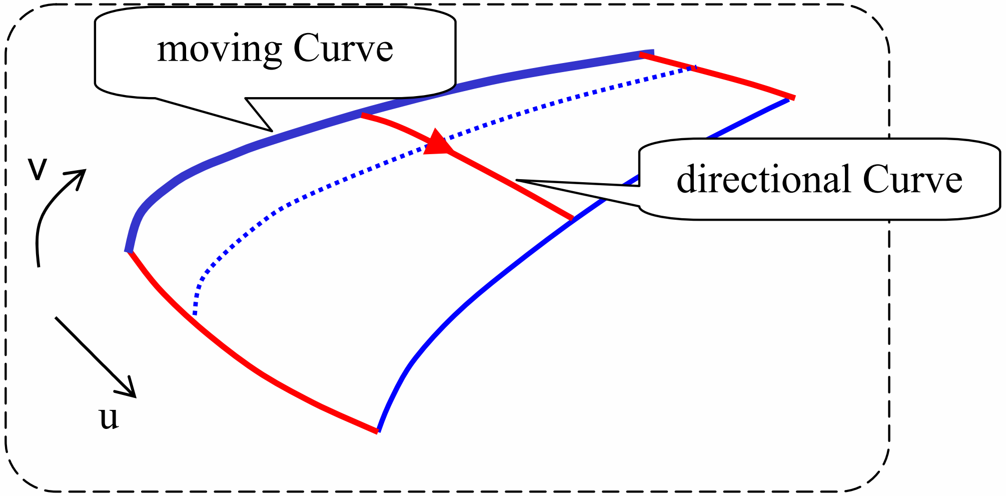
3.2.2.1 Tensor product approach (2)



- moving curve가 일정한 차수의 Bezier curve 이고, moving curve의Bezier control points의 궤적을 나타내는 directional curve도 Bezier curve일 때, 이러한 방법으로 생성되는 곡면을 "Tensor product Bezier surface" 라고 한다.



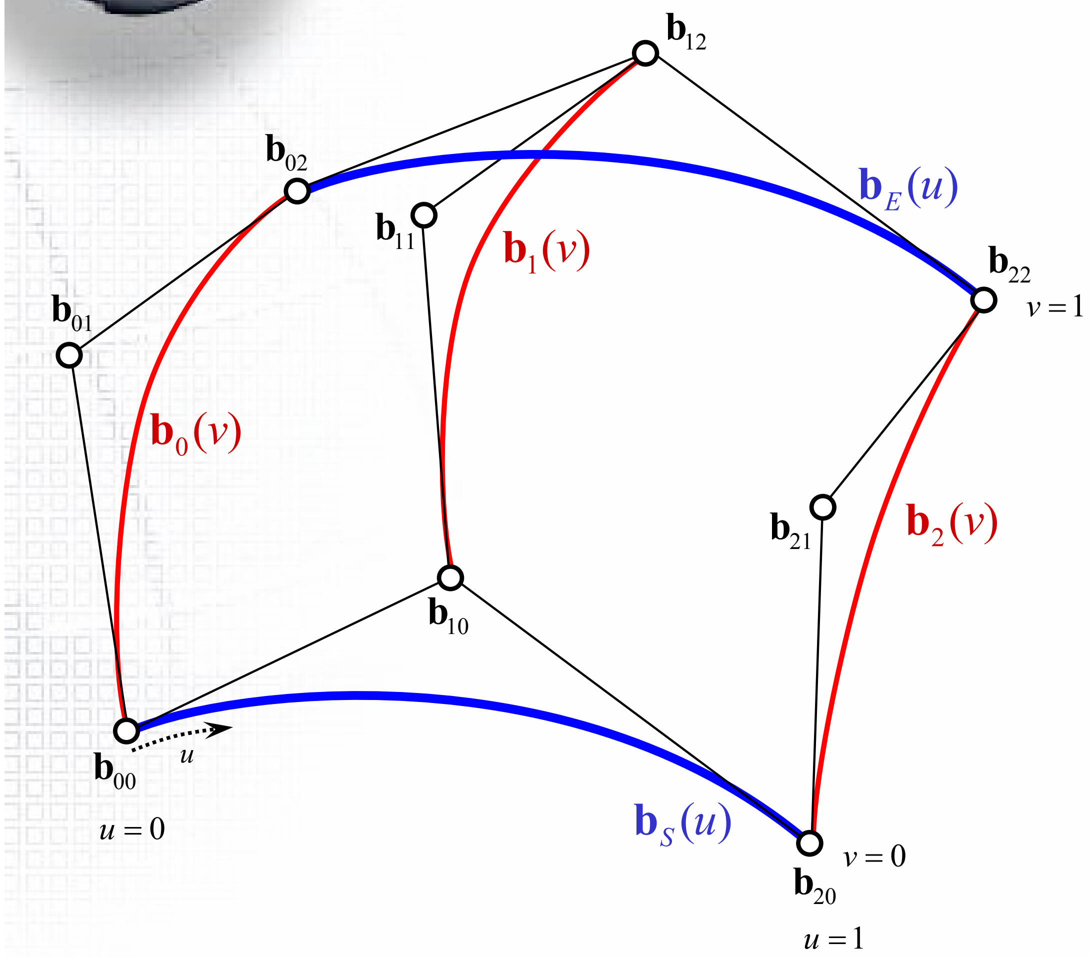
$\mathbf{r}(u)$ 곡선을 v 방향으로 sweeping



$\mathbf{r}(v)$ 곡선을 u 방향으로 sweeping

* CAGD, Ch14.3, Tensor Product Approach

3.2.2.2 Tensor-product biquadratic Bezier surface (1)



- Given 3x3 Points $\mathbf{b}_{ij}$,
- Generate start/end moving curves and directional curves in quadratic Bezier form

$$\mathbf{b}_E(u)=\mathbf{b}_{02}B_0^2(u)+\mathbf{b}_{12}B_1^2(u)+\mathbf{b}_{22}B_2^2(u)$$

$$\mathbf{b}_S(u)=\mathbf{b}_{00}B_0^2(u)+\mathbf{b}_{10}B_1^2(u)+\mathbf{b}_{20}B_2^2(u)$$

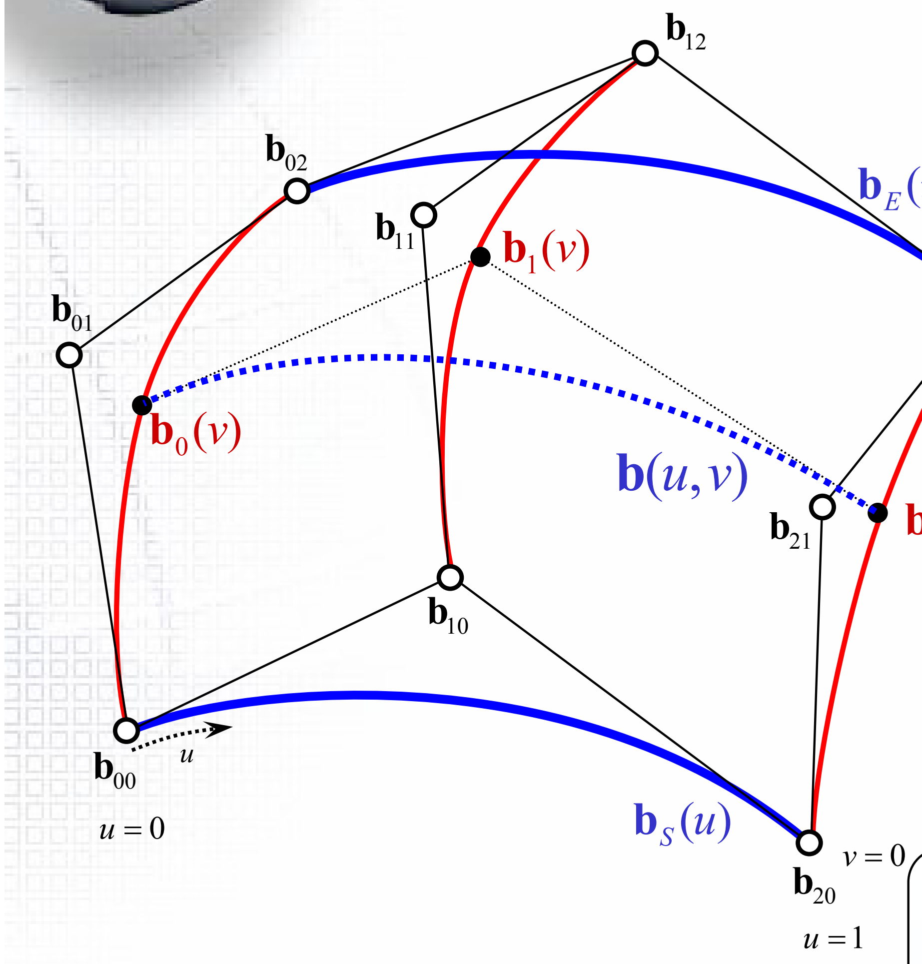
$$\mathbf{b}_0(v)=\mathbf{b}_{00}B_0^2(v)+\mathbf{b}_{01}B_1^2(v)+\mathbf{b}_{02}B_2^2(v)$$

$$\mathbf{b}_1(v)=\mathbf{b}_{10}B_0^2(v)+\mathbf{b}_{11}B_1^2(v)+\mathbf{b}_{12}B_2^2(v)$$

$$\mathbf{b}_2(v)=\mathbf{b}_{20}B_0^2(v)+\mathbf{b}_{21}B_1^2(v)+\mathbf{b}_{22}B_2^2(v)$$

$$\begin{bmatrix}\mathbf{b}_0(v)\\ \mathbf{b}_1(v)\\ \mathbf{b}_2(v)\end{bmatrix}=\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02}\\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12}\\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22}\end{bmatrix}\begin{bmatrix}B_0^2(v)\\ B_1^2(v)\\ B_2^2(v)\end{bmatrix}$$

3.2.2.2 Tensor-product biquadratic Bezier surface (2)



- Given 3x3 Points $\mathbf{b}_{ij}$,
- Moving curve can be represented in the following form:

$$\mathbf{b}(u,v) = \mathbf{b}_0(v)B_0^2(u) + \mathbf{b}_1(v)B_1^2(u) + \mathbf{b}_2(v)B_2^2(u)$$

$$= \begin{bmatrix} B_0^2(u) & B_1^2(u) & B_2^2(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_0(v) \\ \mathbf{b}_1(v) \\ \mathbf{b}_2(v) \end{bmatrix}$$

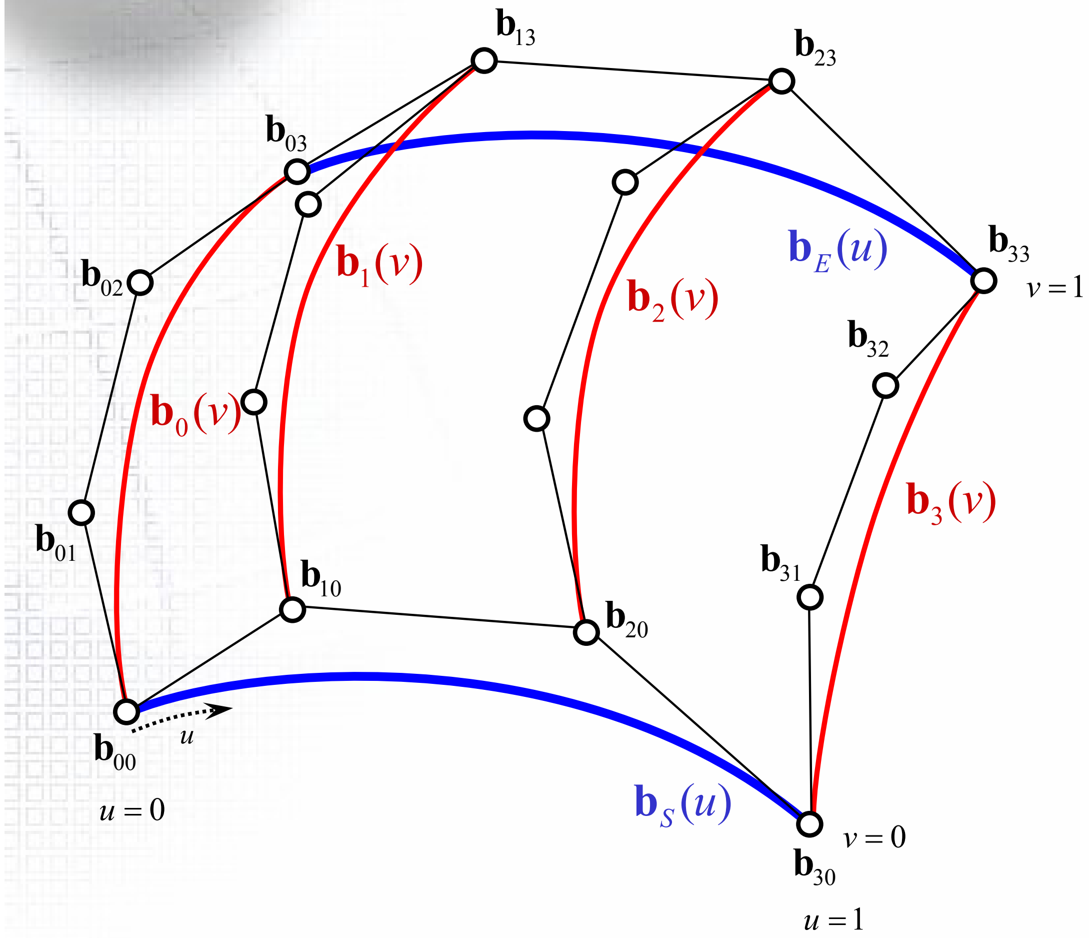
$$\begin{bmatrix} \mathbf{b}_0(v) \\ \mathbf{b}_1(v) \\ \mathbf{b}_2(v) \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} \\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} \\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} \end{bmatrix} \begin{bmatrix} B_0^2(v) \\ B_1^2(v) \\ B_2^2(v) \end{bmatrix}$$

$$\mathbf{b}(u,v) = \begin{bmatrix} B_0^2(u) & B_1^2(u) & B_2^2(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} \\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} \\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} \end{bmatrix} \begin{bmatrix} B_0^2(v) \\ B_1^2(v) \\ B_2^2(v) \end{bmatrix}$$

$$= \sum_{j=0}^{2} \sum_{i=0}^{2} \mathbf{b}_{ij} B_i^2(u) B_j^2(v)$$

Bezier surface control points

3.2.2.3 Tensor-product bicubic Bezier surface (1)



- Given 4x4 Points $\mathbf{b}_{ij}$,
- Generate start/end moving curves and directional curves in cubic Bezier form

$$\mathbf{b}_E(u)=\mathbf{b}_{03}B_0^3(u)+\mathbf{b}_{13}B_1^3(u)+\mathbf{b}_{23}B_2^3(u)+\mathbf{b}_{33}B_3^3(u)$$

$$\mathbf{b}_S(u)=\mathbf{b}_{00}B_0^3(u)+\mathbf{b}_{10}B_1^3(u)+\mathbf{b}_{20}B_2^3(u)+\mathbf{b}_{30}B_3^3(u)$$

$$\mathbf{b}_0(v)=\mathbf{b}_{00}B_0^3(v)+\mathbf{b}_{01}B_1^3(v)+\mathbf{b}_{02}B_2^3(v)+\mathbf{b}_{03}B_3^3(v)$$

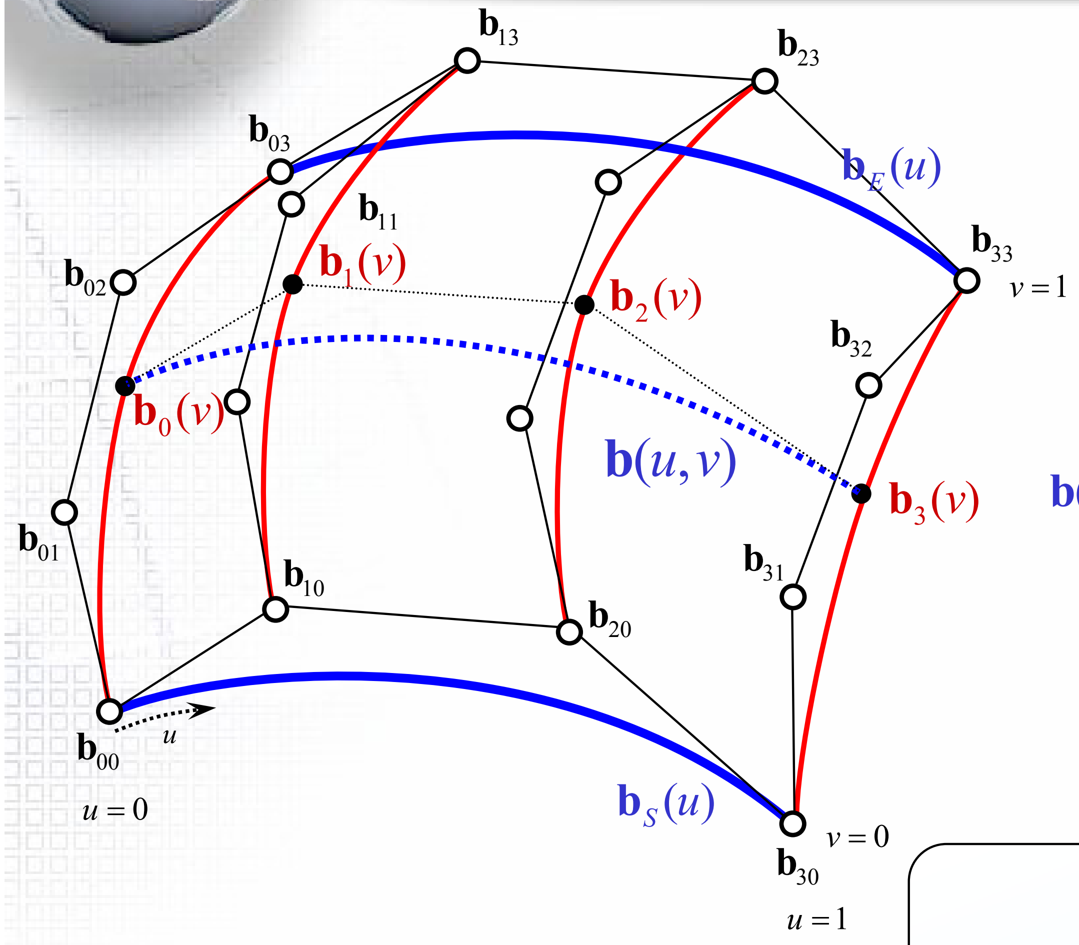
$$\mathbf{b}_1(v)=\mathbf{b}_{10}B_0^3(v)+\mathbf{b}_{11}B_1^3(v)+\mathbf{b}_{12}B_2^3(v)+\mathbf{b}_{13}B_3^3(v)$$

$$\mathbf{b}_2(v)=\mathbf{b}_{20}B_0^3(v)+\mathbf{b}_{21}B_1^3(v)+\mathbf{b}_{22}B_2^3(v)+\mathbf{b}_{23}B_3^3(v)$$

$$\mathbf{b}_3(v)=\mathbf{b}_{30}B_0^3(v)+\mathbf{b}_{31}B_1^3(v)+\mathbf{b}_{32}B_2^3(v)+\mathbf{b}_{33}B_3^3(v)$$

$$\begin{bmatrix}\mathbf{b}_0(v)\\ \mathbf{b}_1(v)\\ \mathbf{b}_2(v)\\ \mathbf{b}_3(v)\end{bmatrix}=\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} & \mathbf{b}_{03}\\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} & \mathbf{b}_{13}\\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} & \mathbf{b}_{23}\\ \mathbf{b}_{30} & \mathbf{b}_{31} & \mathbf{b}_{32} & \mathbf{b}_{33}\end{bmatrix}\begin{bmatrix}B_0^3(v)\\ B_1^3(v)\\ B_2^3(v)\\ B_3^3(v)\end{bmatrix}$$

3.2.2.3 Tensor-product bicubic Bezier surface (2)



- Given 4x4 Points $\mathbf{b}_{ij}$,
- Moving curve can be represented in the following form:

$$\mathbf{b}(u,v)=\boxed{\mathbf{b}_0(v)}B_0^3(u)+\boxed{\mathbf{b}_1(v)}B_1^3(u)+\boxed{\mathbf{b}_2(v)}B_2^3(u)+\boxed{\mathbf{b}_3(v)}B_3^3(u)$$

$$=\begin{bmatrix}B_0^3(u) & B_1^3(u) & B_2^3(u) & B_3^3(u)\end{bmatrix}\begin{bmatrix}\mathbf{b}_0(v)\\ \mathbf{b}_1(v)\\ \mathbf{b}_2(v)\\ \mathbf{b}_3(v)\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{b}_0(v)\\ \mathbf{b}_1(v)\\ \mathbf{b}_2(v)\\ \mathbf{b}_3(v)\end{bmatrix}=\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} & \mathbf{b}_{03}\\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} & \mathbf{b}_{13}\\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} & \mathbf{b}_{23}\\ \mathbf{b}_{30} & \mathbf{b}_{31} & \mathbf{b}_{32} & \mathbf{b}_{33}\end{bmatrix}\begin{bmatrix}B_0^3(v)\\ B_1^3(v)\\ B_2^3(v)\\ B_3^3(v)\end{bmatrix}$$

$$\mathbf{b}(u,v)=\begin{bmatrix}B_0^3(u) & B_1^3(u) & B_2^3(u) & B_3^3(u)\end{bmatrix}\begin{bmatrix}\mathbf{b}_{00} & \mathbf{b}_{01} & \mathbf{b}_{02} & \mathbf{b}_{03}\\ \mathbf{b}_{10} & \mathbf{b}_{11} & \mathbf{b}_{12} & \mathbf{b}_{13}\\ \mathbf{b}_{20} & \mathbf{b}_{21} & \mathbf{b}_{22} & \mathbf{b}_{23}\\ \mathbf{b}_{30} & \mathbf{b}_{31} & \mathbf{b}_{32} & \mathbf{b}_{33}\end{bmatrix}\begin{bmatrix}B_0^3(v)\\ B_1^3(v)\\ B_2^3(v)\\ B_3^3(v)\end{bmatrix}$$

$$=\sum_{j=0}^{3}\sum_{i=0}^{3}\mathbf{b}_{ij}B_i^3(u)B_j^3(v)$$

3.3 B-spline surfaces

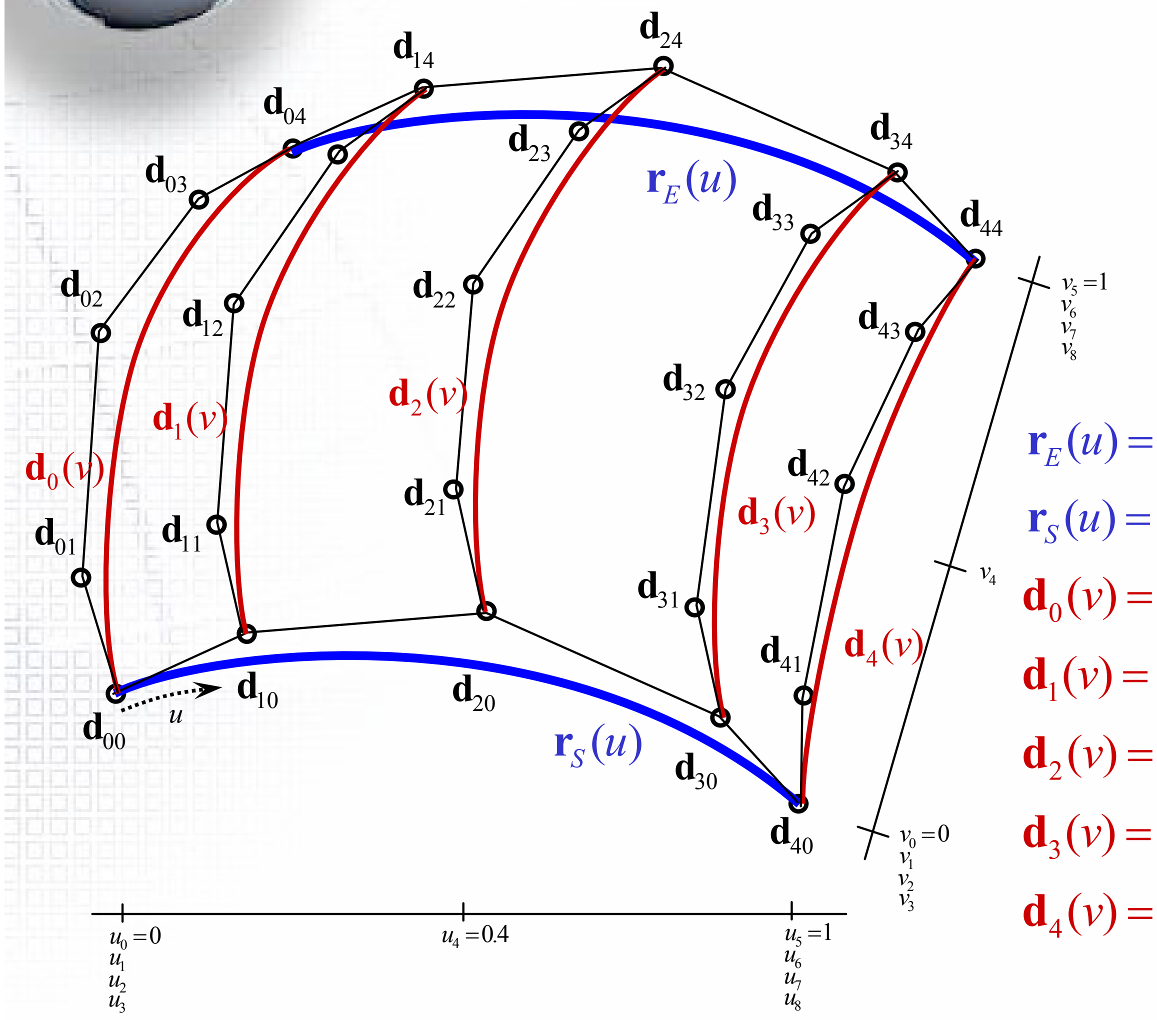
3.3.1 Generation of B-spline surfaces by tensor-product approach

3.3.2 B-spline surface Interpolation

3.3.1 Generation of B-spline surfaces by tensor-product approach

3.3.1.1 Tensor-product bicubic B-spline surface

3.3.1.1 Tensor-product bicubic B-spline surface (1)



- Given 5x5 Points $\mathbf{d}_{ij}$, u-knots, v-knots, u-degree(=3), v-degree(=3),
- Generate start/end moving curves and directional curves in cubic B-spline form:

$$\mathbf{r}_E(u)=\mathbf{d}_{04}N_0^3(u)+\mathbf{d}_{14}N_1^3(u)+\mathbf{d}_{24}N_2^3(u)+\mathbf{d}_{34}N_3^3(u)+\mathbf{d}_{44}N_4^3(u)$$

$$\mathbf{r}_S(u)=\mathbf{d}_{00}N_0^3(u)+\mathbf{d}_{10}N_1^3(u)+\mathbf{d}_{20}N_2^3(u)+\mathbf{d}_{30}N_3^3(u)+\mathbf{d}_{40}N_4^3(u)$$

$$\mathbf{d}_0(v)=\mathbf{d}_{00}N_0^3(v)+\mathbf{d}_{01}N_1^3(v)+\mathbf{d}_{02}N_2^3(v)+\mathbf{d}_{03}N_3^3(v)+\mathbf{d}_{04}N_4^3(v)$$

$$\mathbf{d}_1(v)=\mathbf{d}_{10}N_0^3(v)+\mathbf{d}_{11}N_1^3(v)+\mathbf{d}_{12}N_2^3(v)+\mathbf{d}_{13}N_3^3(v)+\mathbf{d}_{14}N_4^3(v)$$

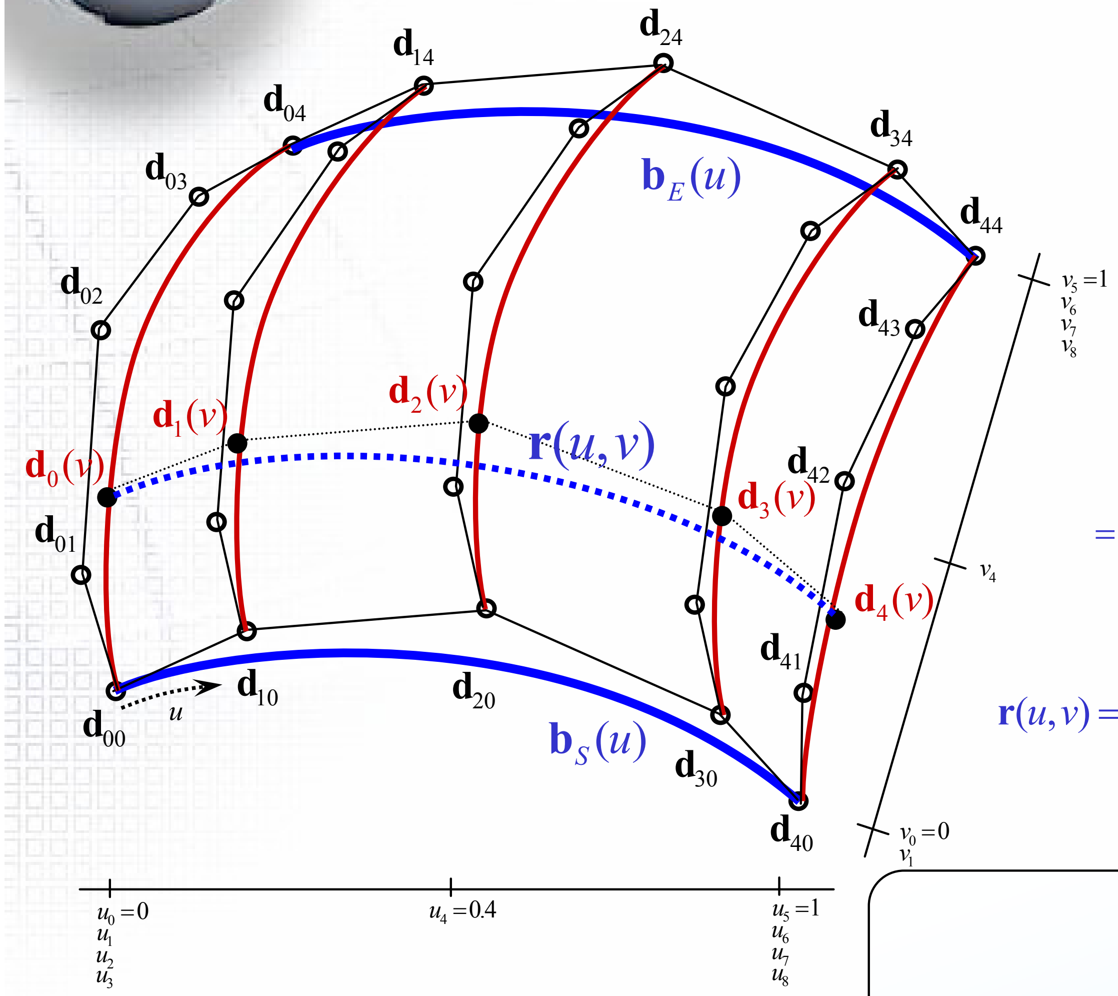
$$\mathbf{d}_2(v)=\mathbf{d}_{20}N_0^3(v)+\mathbf{d}_{21}N_1^3(v)+\mathbf{d}_{22}N_2^3(v)+\mathbf{d}_{23}N_3^3(v)+\mathbf{d}_{24}N_4^3(v)$$

$$\mathbf{d}_3(v)=\mathbf{d}_{30}N_0^3(v)+\mathbf{d}_{31}N_1^3(v)+\mathbf{d}_{32}N_2^3(v)+\mathbf{d}_{33}N_3^3(v)+\mathbf{d}_{34}N_4^3(v)$$

$$\mathbf{d}_4(v)=\mathbf{d}_{40}N_0^3(v)+\mathbf{d}_{41}N_1^3(v)+\mathbf{d}_{42}N_2^3(v)+\mathbf{d}_{43}N_3^3(v)+\mathbf{d}_{44}N_4^3(v)$$

$$\begin{bmatrix}\mathbf{d}_0(v)\\ \mathbf{d}_1(v)\\ \mathbf{d}_2(v)\\ \mathbf{d}_3(v)\\ \mathbf{d}_4(v)\end{bmatrix}=\begin{bmatrix}\mathbf{d}_{00}&\mathbf{d}_{01}&\mathbf{d}_{02}&\mathbf{d}_{03}&\mathbf{d}_{04}\\ \mathbf{d}_{10}&\mathbf{d}_{11}&\mathbf{d}_{12}&\mathbf{d}_{13}&\mathbf{d}_{14}\\ \mathbf{d}_{20}&\mathbf{d}_{21}&\mathbf{d}_{22}&\mathbf{d}_{23}&\mathbf{d}_{24}\\ \mathbf{d}_{30}&\mathbf{d}_{31}&\mathbf{d}_{32}&\mathbf{d}_{33}&\mathbf{d}_{34}\\ \mathbf{d}_{40}&\mathbf{d}_{41}&\mathbf{d}_{42}&\mathbf{d}_{43}&\mathbf{d}_{44}\end{bmatrix}\begin{bmatrix}N_0^3(v)\\ N_1^3(v)\\ N_2^3(v)\\ N_3^3(v)\\ N_4^3(v)\end{bmatrix}$$

3.3.1.1 Tensor-product bicubic B-spline surface (2)



- Given 5x5 Points $\mathbf{d}_{ij}$,
  u-knots, v-knots,
  u-degree(=3), v-degree(=3),
- Moving curve can be represented in the following form:

$$=\begin{bmatrix} N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{d}_0(v) \\ \mathbf{d}_1(v) \\ \mathbf{d}_2(v) \\ \mathbf{d}_3(v) \\ \mathbf{d}_4(v) \end{bmatrix}$$

$$\mathbf{r}(u,v)=\mathbf{d}_0(v)N_0^3(u)+\mathbf{d}_1(v)N_1^3(u)+\mathbf{d}_2(v)N_2^3(u)+\mathbf{d}_3(v)N_3^3(u)+\mathbf{d}_4(v)N_4^3(u)$$

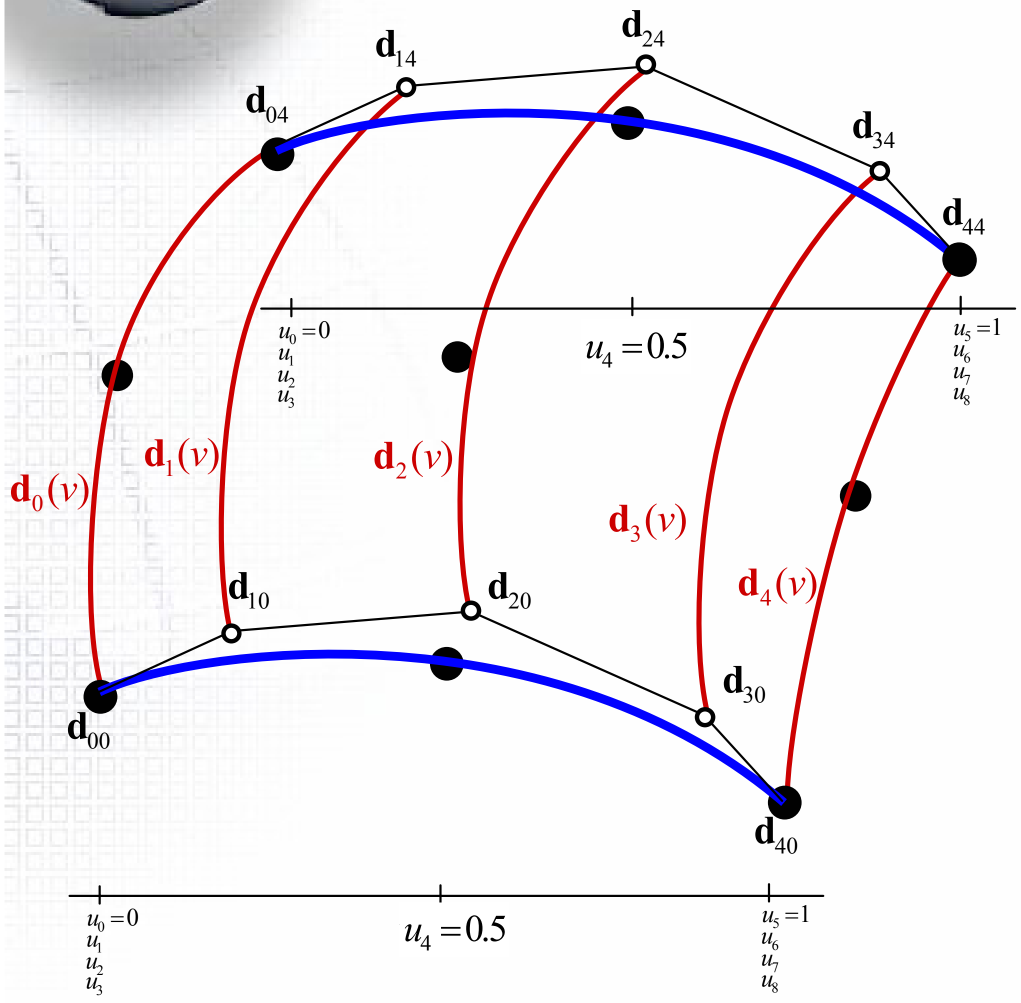
$$\begin{bmatrix} \mathbf{d}_0(v) \\ \mathbf{d}_1(v) \\ \mathbf{d}_2(v) \\ \mathbf{d}_3(v) \\ \mathbf{d}_4(v) \end{bmatrix} = \begin{bmatrix} \mathbf{d}_{00} & \mathbf{d}_{01} & \mathbf{d}_{02} & \mathbf{d}_{03} & \mathbf{d}_{04} \\ \mathbf{d}_{10} & \mathbf{d}_{11} & \mathbf{d}_{12} & \mathbf{d}_{13} & \mathbf{d}_{14} \\ \mathbf{d}_{20} & \mathbf{d}_{21} & \mathbf{d}_{22} & \mathbf{d}_{23} & \mathbf{d}_{24} \\ \mathbf{d}_{30} & \mathbf{d}_{31} & \mathbf{d}_{32} & \mathbf{d}_{33} & \mathbf{d}_{34} \\ \mathbf{d}_{40} & \mathbf{d}_{41} & \mathbf{d}_{42} & \mathbf{d}_{43} & \mathbf{d}_{44} \end{bmatrix} \begin{bmatrix} N_0^3(v) \\ N_1^3(v) \\ N_2^3(v) \\ N_3^3(v) \\ N_4^3(v) \end{bmatrix}$$

$$\mathbf{r}(u,v)=\begin{bmatrix} N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{d}_{00} & \mathbf{d}_{01} & \mathbf{d}_{02} & \mathbf{d}_{03} & \mathbf{d}_{04} \\ \mathbf{d}_{10} & \mathbf{d}_{11} & \mathbf{d}_{12} & \mathbf{d}_{13} & \mathbf{d}_{14} \\ \mathbf{d}_{20} & \mathbf{d}_{21} & \mathbf{d}_{22} & \mathbf{d}_{23} & \mathbf{d}_{24} \\ \mathbf{d}_{30} & \mathbf{d}_{31} & \mathbf{d}_{32} & \mathbf{d}_{33} & \mathbf{d}_{34} \\ \mathbf{d}_{40} & \mathbf{d}_{41} & \mathbf{d}_{42} & \mathbf{d}_{43} & \mathbf{d}_{44} \end{bmatrix} \begin{bmatrix} N_0^3(v) \\ N_1^3(v) \\ N_2^3(v) \\ N_3^3(v) \\ N_4^3(v) \end{bmatrix}$$

$$=\sum_{j=0}^{5}\sum_{i=0}^{5}\mathbf{d}_{ij}N_i^3(u)N_j^3(v)$$

3.3.2 B-spline surfaces Interpolation

3.3.2.1 B-spline surface interpolation 과정 (1)



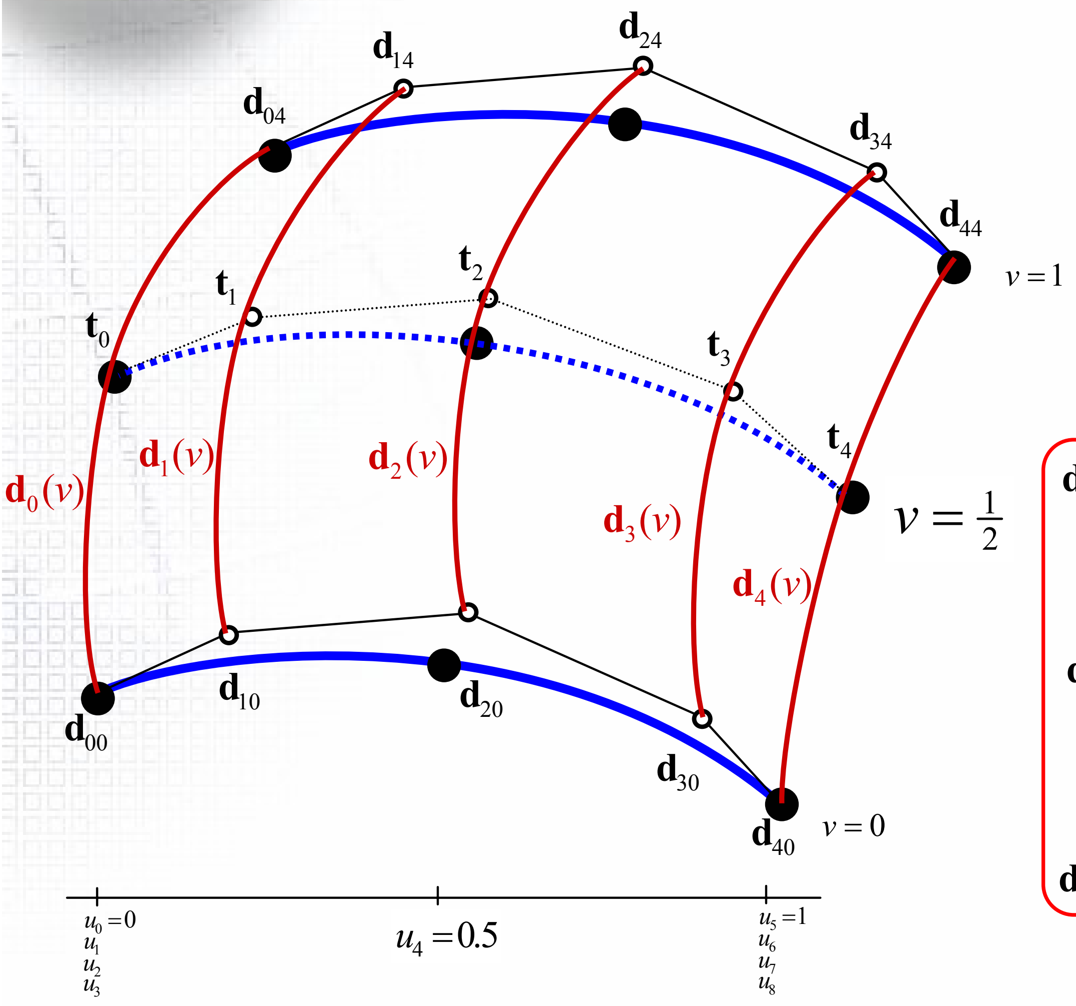
- Given 3x3 fitting points
- (1) interpolate start/end moving B-spline curves with same u-knots, which are swept along directional curves

$$\mathbf{d}_0(1)=\mathbf{d}_{04} \quad \mathbf{d}_1(1)=\mathbf{d}_{14} \quad \mathbf{d}_2(1)=\mathbf{d}_{24} \quad \mathbf{d}_3(1)=\mathbf{d}_{34} \quad \mathbf{d}_4(1)=\mathbf{d}_{44}$$

$$\mathbf{r}(u,v)=\begin{bmatrix} N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u) \end{bmatrix}\begin{bmatrix} \mathbf{d}_0(v) \\ \mathbf{d}_1(v) \\ \mathbf{d}_2(v) \\ \mathbf{d}_3(v) \\ \mathbf{d}_4(v) \end{bmatrix}$$

$$\mathbf{d}_0(0)=\mathbf{d}_{00} \quad \mathbf{d}_1(0)=\mathbf{d}_{10} \quad \mathbf{d}_2(0)=\mathbf{d}_{20} \quad \mathbf{d}_3(0)=\mathbf{d}_{30} \quad \mathbf{d}_4(0)=\mathbf{d}_{40}$$

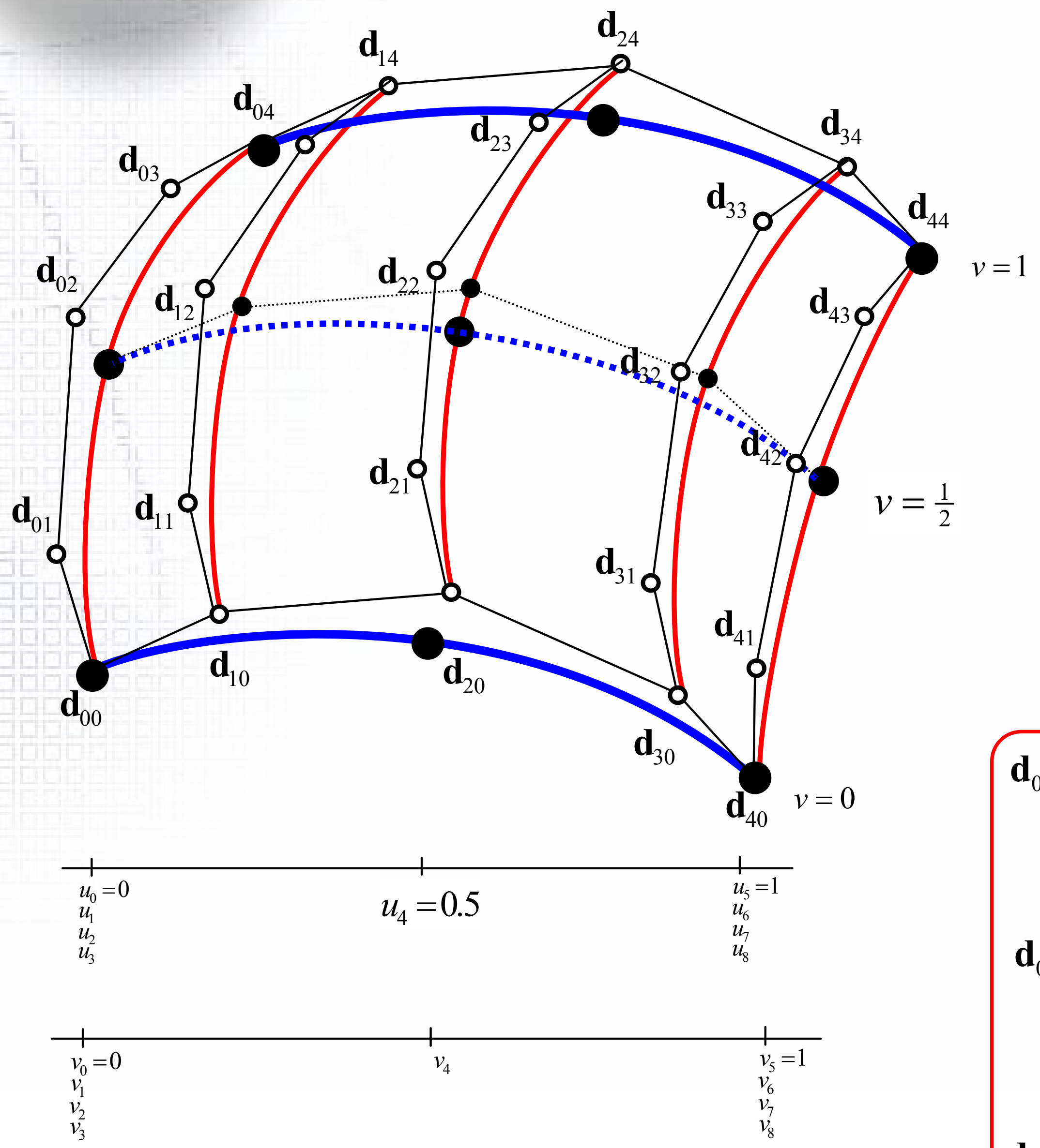
3.3.2.1 B-spline surface interpolation 과정 (2)



- Given 3x3 fitting points
- (1) interpolate start/end moving B-spline curves with same u-knots, which are swept along directional curves
- (2) interpolate moving curves at v=v* with same u-knots

$\mathbf{d}_0(1)=\mathbf{d}_{04}$	$\mathbf{d}_1(1)=\mathbf{d}_{14}$	$\mathbf{d}_2(1)=\mathbf{d}_{24}$	$\mathbf{d}_3(1)=\mathbf{d}_{34}$	$\mathbf{d}_4(1)=\mathbf{d}_{44}$
$\mathbf{d}_0(\frac{1}{2})=\mathbf{t}_0$	$\mathbf{d}_1(\frac{1}{2})=\mathbf{t}_1$	$\mathbf{d}_2(\frac{1}{2})=\mathbf{t}_2$	$\mathbf{d}_3(\frac{1}{2})=\mathbf{t}_3$	$\mathbf{d}_4(\frac{1}{2})=\mathbf{t}_4$
$\mathbf{d}_0(0)=\mathbf{d}_{00}$	$\mathbf{d}_1(0)=\mathbf{d}_{10}$	$\mathbf{d}_2(0)=\mathbf{d}_{20}$	$\mathbf{d}_3(0)=\mathbf{d}_{30}$	$\mathbf{d}_4(0)=\mathbf{d}_{40}$
$\mathbf{d}_0(v)$	$\mathbf{d}_1(v)$	$\mathbf{d}_2(v)$	$\mathbf{d}_3(v)$	$\mathbf{d}_4(v)$

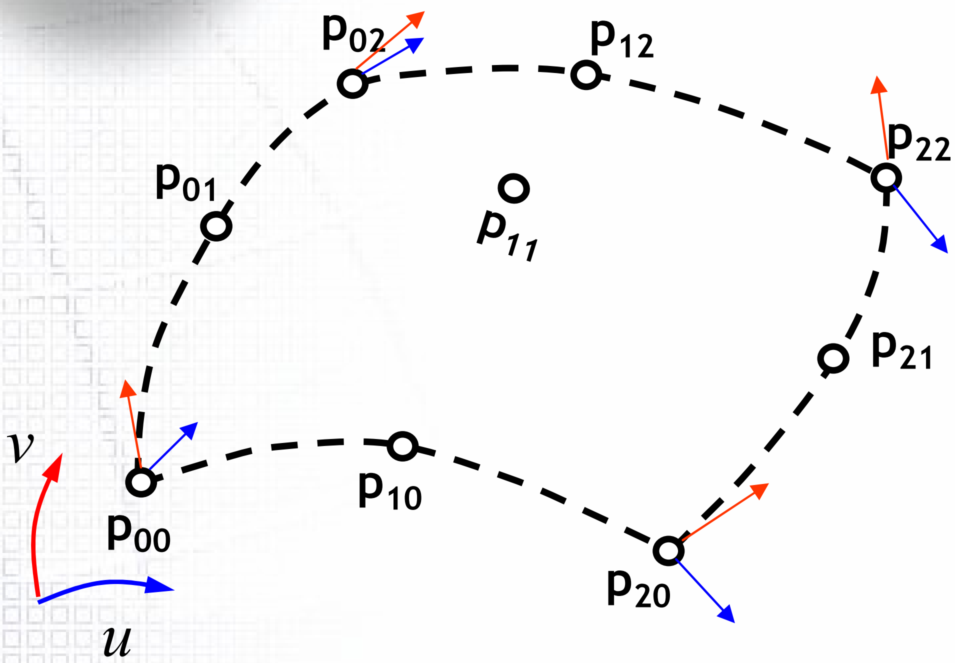
3.3.2.1 B-spline surface interpolation 과정 (3)



- Given 3x3 fitting points
- (1) interpolate start/end moving B-spline curves with same u-knots, which are swept along directional curves
- (2) interpolate moving curves at v=v* with same u-knots
- (3) generate directional B-spline curves with same v-knots by interpolating the control points of the moving B-spline curves
- The control points of the directional curves are the control points of final bicubic B-spline surface

$\mathbf{d}_0(1)=\mathbf{d}_{04}$	$\mathbf{d}_1(1)=\mathbf{d}_{14}$	$\mathbf{d}_2(1)=\mathbf{d}_{24}$	$\mathbf{d}_3(1)=\mathbf{d}_{34}$	$\mathbf{d}_4(1)=\mathbf{d}_{44}$
$\mathbf{d}_0(\frac{1}{2})=\mathbf{t}_0$	$\mathbf{d}_1(\frac{1}{2})=\mathbf{t}_1$	$\mathbf{d}_2(\frac{1}{2})=\mathbf{t}_2$	$\mathbf{d}_3(\frac{1}{2})=\mathbf{t}_3$	$\mathbf{d}_4(\frac{1}{2})=\mathbf{t}_4$
$\mathbf{d}_0(0)=\mathbf{d}_{00}$	$\mathbf{d}_1(0)=\mathbf{d}_{10}$	$\mathbf{d}_2(0)=\mathbf{d}_{20}$	$\mathbf{d}_3(0)=\mathbf{d}_{30}$	$\mathbf{d}_4(0)=\mathbf{d}_{40}$

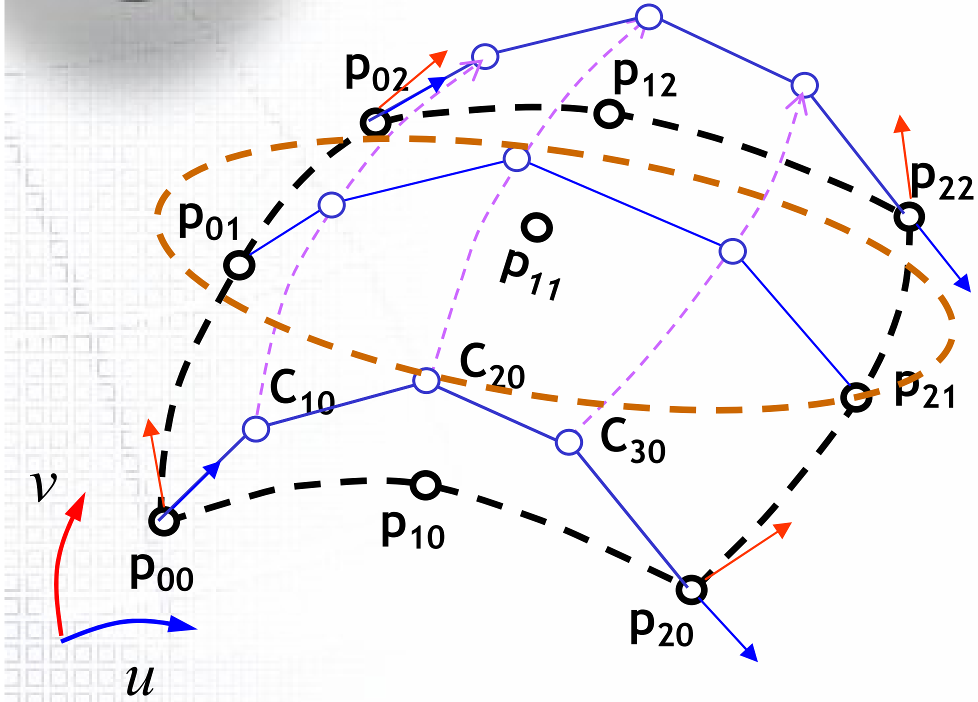
3.3.2.2 Given : 곡면상의 9개 점과, 4 꼭지점에서의 u,v방향의 접선벡터
Find: bicubic B-spline Surface(1)



$$\mathbf{r}(u,v) = \begin{bmatrix} N_0^3(u) & N_1^3(u) & N_2^3(u) & N_3^3(u) & N_4^3(u) \end{bmatrix} \begin{bmatrix} \mathbf{d}_{0,0} & \mathbf{d}_{1,0} & \mathbf{d}_{2,0} & \mathbf{d}_{3,0} & \mathbf{d}_{4,0} \\ \mathbf{d}_{0,1} & \mathbf{d}_{1,1} & \mathbf{d}_{2,1} & \mathbf{d}_{3,1} & \mathbf{d}_{4,1} \\ \mathbf{d}_{0,2} & \mathbf{d}_{1,2} & \mathbf{d}_{2,2} & \mathbf{d}_{3,2} & \mathbf{d}_{4,2} \\ \mathbf{d}_{0,3} & \mathbf{d}_{1,3} & \mathbf{d}_{2,3} & \mathbf{d}_{3,3} & \mathbf{d}_{4,3} \\ \mathbf{d}_{0,4} & \mathbf{d}_{1,4} & \mathbf{d}_{2,4} & \mathbf{d}_{3,4} & \mathbf{d}_{4,4} \end{bmatrix} \begin{bmatrix} N_0^3(v) \\ N_1^3(v) \\ N_2^3(v) \\ N_3^3(v) \\ N_4^3(v) \end{bmatrix}$$

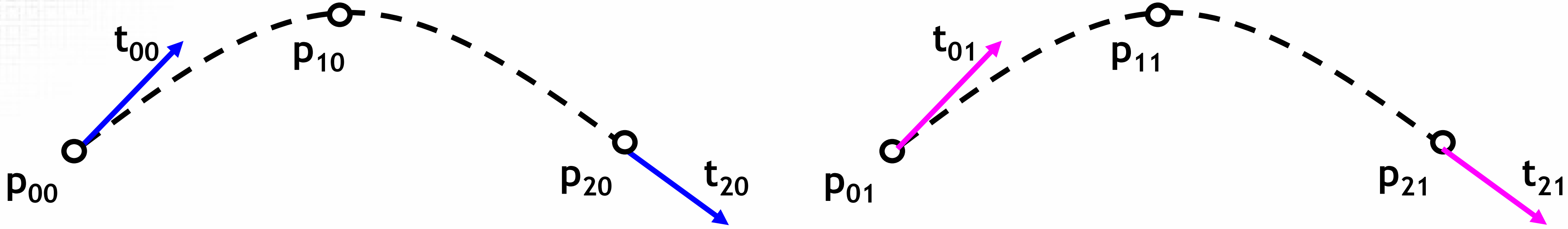
5x5 개의 조정점을 구하면 곡면을 생성할 수 있다.

3.3.2.2 Given : 곡면상의 9개 점과, 4 꼭지점에서의 u,v방향의 접선벡터
Find: bicubic B-spline Surface(2)



곡선상의 점($P_{i,j}$)과 접선벡터 ($t_{i,j}$) 으로부터 중간 조정점($C_{i,j}$)을 구한다.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\dfrac{3}{\Delta_s} & \dfrac{3}{\Delta_s} & 0 & 0 & 0 \\ 0 & \alpha & \beta & \gamma & 0 \\ 0 & 0 & 0 & -\dfrac{3}{\Delta_E} & \dfrac{3}{\Delta_E} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{C}_{00} \\ \mathbf{C}_{10} \\ \mathbf{C}_{20} \\ \mathbf{C}_{30} \\ \mathbf{C}_{40} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{00} \\ \mathbf{t}_{00} \\ \mathbf{P}_{10} \\ \mathbf{t}_{20} \\ \mathbf{P}_{20} \end{bmatrix}$$

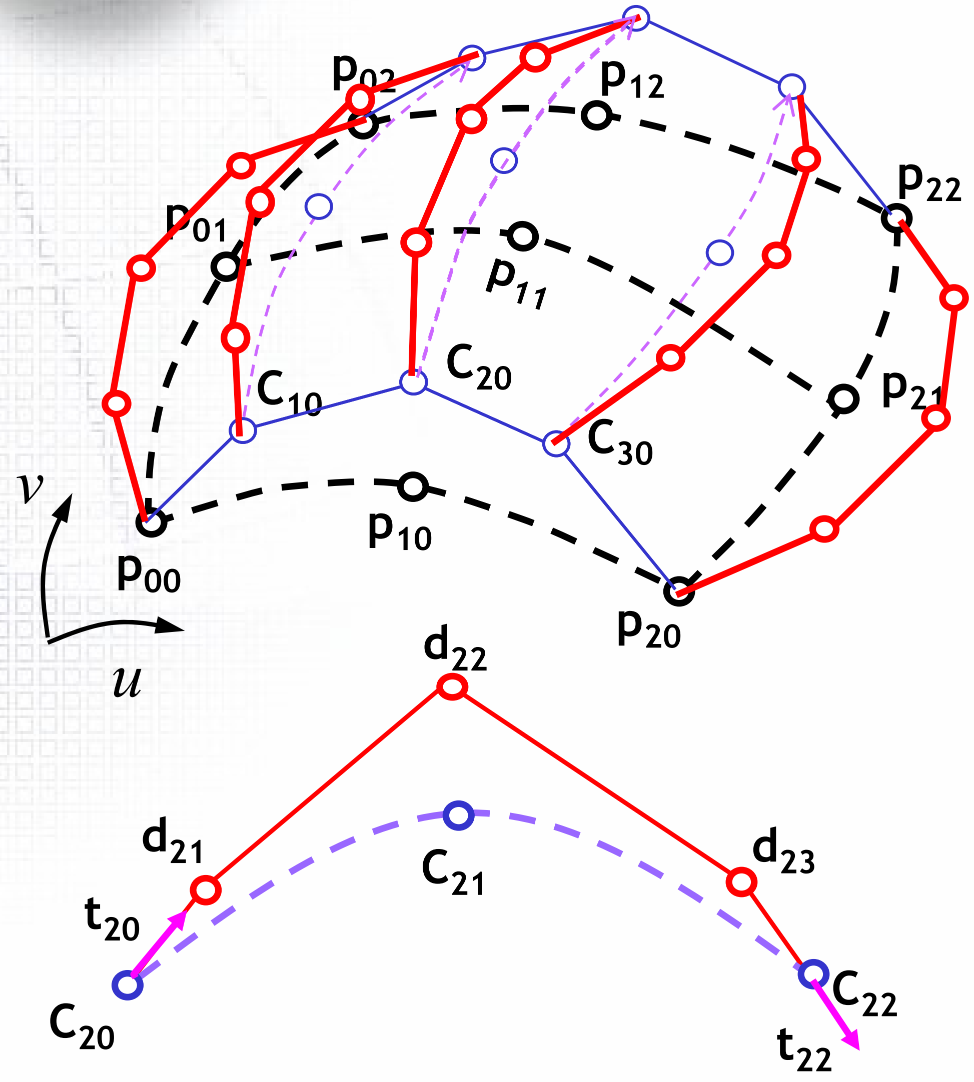


Bessel end condition:

곡선을 지나는 3점을 2차식으로 보간(interpolation)한 후, 곡선의 끝점에서의 1차미분값을 구하는 방법

Bessel end condition으로 접선벡터($t_{i,j}$)를 구한다

3.3.2.2 Given : 곡면상의 9개 점과, 4 꼭지점에서의 u,v방향의 접선벡터
Find: bicubic B-spline Surface(3)

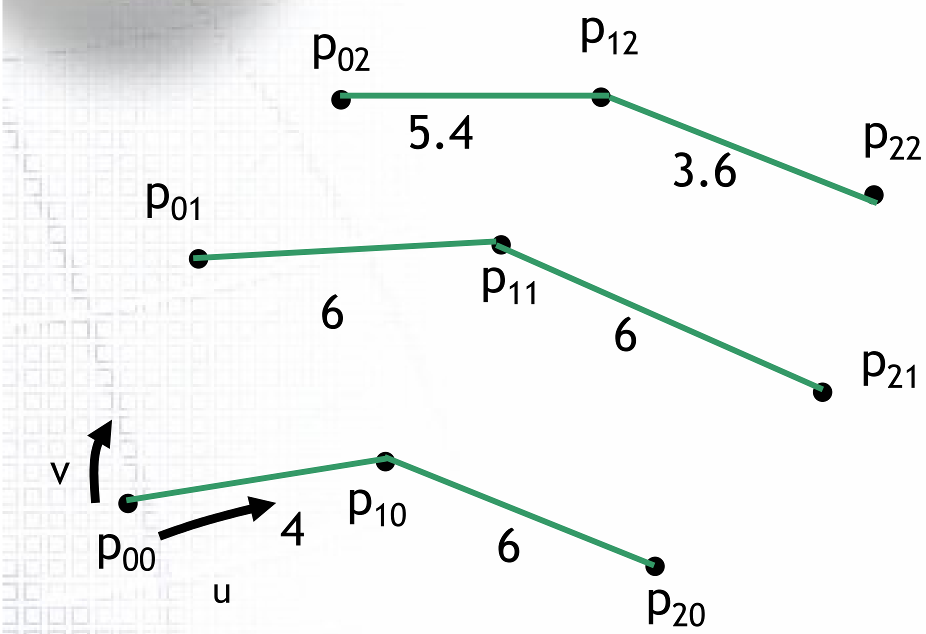


중간 조정점($C_{i,j}$)과 접선 벡터 ($t_{i,j}$) 으로부터 B-Spline조정점($d_{i,j}$)을 구한다.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{3}{\Delta_s} & \frac{3}{\Delta_s} & 0 & 0 & 0 \\ 0 & \alpha & \beta & \gamma & 0 \\ 0 & 0 & 0 & -\frac{3}{\Delta_E} & \frac{3}{\Delta_E} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{d}_{20} \\ \mathbf{d}_{21} \\ \mathbf{d}_{22} \\ \mathbf{d}_{23} \\ \mathbf{d}_{24} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{20} \\ \mathbf{t}_{20} \\ \mathbf{C}_{21} \\ \mathbf{t}_{22} \\ \mathbf{C}_{22} \end{bmatrix}$$

Bessel end condition으로 접선벡터($t_{i,j}$)를 구한다

[정리] 주어진 점을 보간하는 bicubic B-spline 곡면 생성 방법

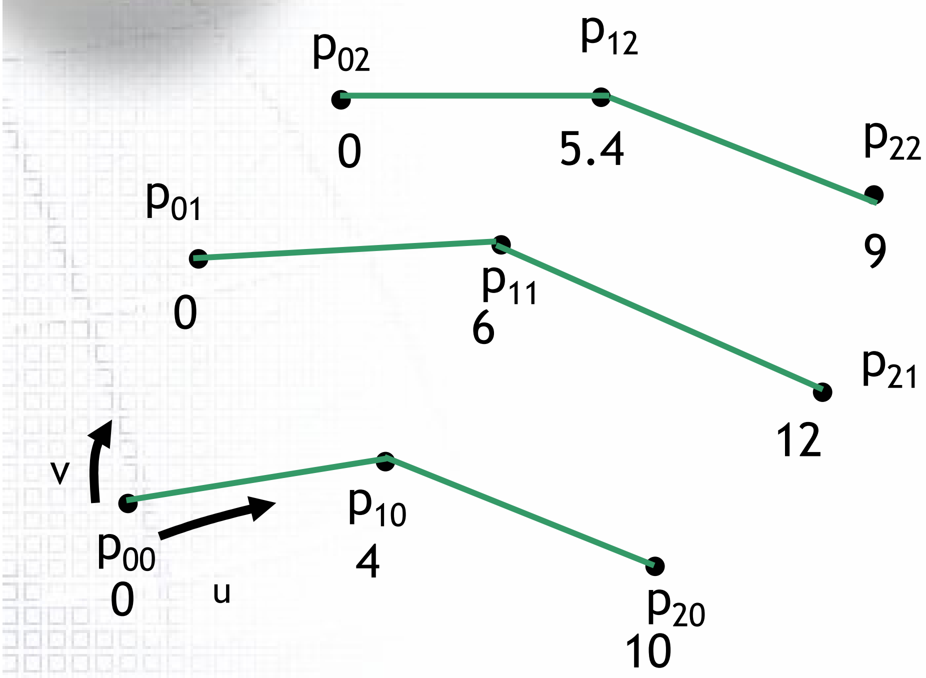


1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다

☐ Given

- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

[정리] 주어진 점을 보간하는 bicubic B-spline 곡면 생성 방법

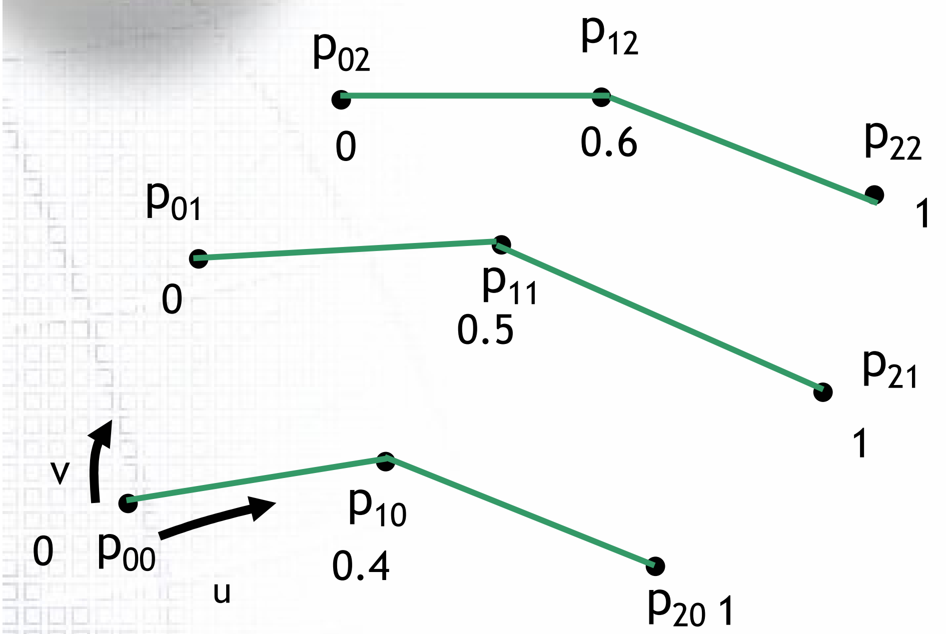


❑ Given

- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다

[정리] 주어진 점을 보간하는 bicubic B-spline 곡면 생성 방법

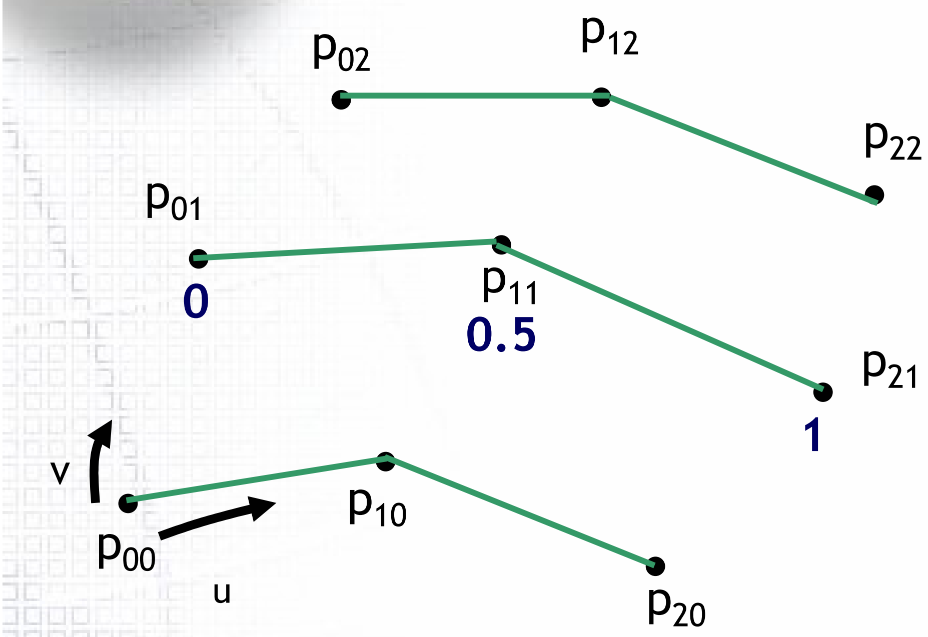


☐ Given

- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다
   - 마지막 점의 knot값으로 각 점의 knot값을 나누어 정규화된 knot값을 계산한다

[정리] 주어진 점을 보간하는 bicubic B-spline 곡면 생성 방법

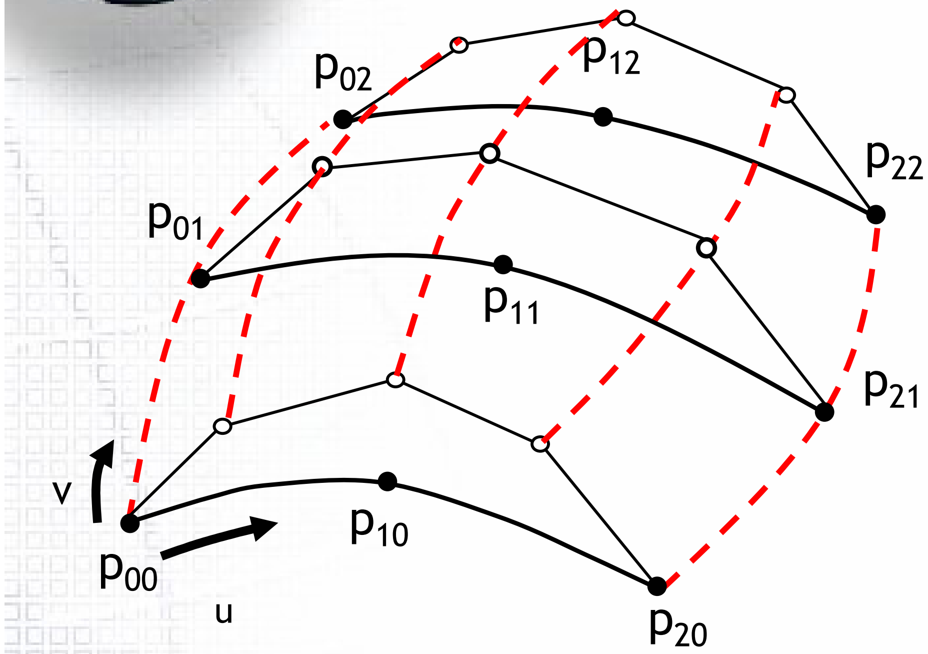


❑ Given

- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다
   - 마지막 점의 knot값으로 각 점의 knot값을 나누어 정규화된 knot값을 계산한다
   - 정규화된 knot값들을 v방향으로 평균하여 최종적인 u방향 knot값을 계산한다

[정리] 주어진 점을 보간하는 bicubic B-spline 곡면 생성 방법

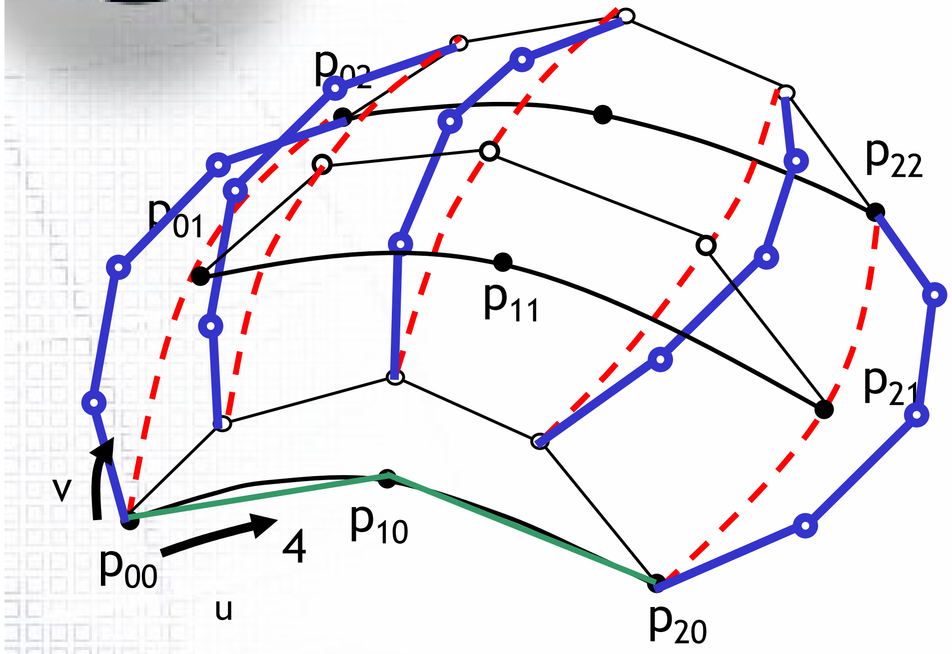


❑ Given

- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다
   - 마지막 점의 knot값으로 각 점의 knot값을 나누어 정규화된 knot값을 계산한다
   - 정규화된 knot값들을 v방향으로 평균하여 최종적인 u방향 knot값을 계산한다
2. u 방향 점들을 보간하는 B-spline 곡선을 계산
   - 최종적인 u방향 knot와 점들을 지나는 B-spline 곡선과 그 조정점을 계산한다
3. u 방향 B-spline 곡선의 조정점을 보간하는 v방향 B-spline 곡선을 계산
   - u방향 B-spline 곡선의 조점정에 대해서 1번과 같은 방법으로 v방향 knot를 결정한 후, 이를 이용하여 v방향 B-spline 곡선 (빨간점선) 을 생성한다. 이 곡선의 조정점 (파란점) 이 최종 B-spline 곡면의 조정점이다

[정리] 주어진 점을 보간하는 bicubic B-spline 곡면 생성 방법



❑ Given

- 곡면이 지나야할 점들의 좌표
- 점들은 사각형 grid 형태여야 함 (예, 2x2)

1. u 방향 knot를 결정
   - 주어진 점들의 u방향 거리를 계산한다
   - 계산한 거리를 각 점별로 누적한다. 이 거리를 곡면의 u방향 knot라고 부른다
   - 마지막 점의 knot값으로 각 점의 knot값을 나누어 정규화된 knot값을 계산한다
   - 정규화된 knot값들을 v방향으로 평균하여 최종적인 u방향 knot값을 계산한다

2. u 방향 점들을 보간하는 B-spline 곡선을 계산
   - 최종적인 u방향 knot와 점들을 지나는 B-spline 곡선과 그 조정점을 계산한다

3. u 방향 B-spline 곡선의 조정점을 보간하는 v방향 B-spline 곡선을 계산
   - u방향 B-spline 곡선의 조점정에 대해서 1번과 같은 방법으로 v방향 knot를 결정한 후, 이를 이용하여 v방향 B-spline 곡선 (빨간점선) 을 생성한다. 이 곡선의 조정점 (파란점) 이 최종 B-spline 곡면의 조정점이다

3.3.2.3 Finding Knot Sequences

B-Spline 곡선에서의
Knot 간격 예측

주어진 곡선상의 점들로부터 Chord Length를 구함

$$\frac{\Delta_i}{\Delta_{i+1}} = \sqrt{\frac{|p_{i-1}-p_{i-2}|}{|p_i-p_{i-1}|}}, (i=2,3,....,n+2)$$

곡선상의 점,접선벡터와 위의 Knot 간격을 이용하여 B-spline 조정점을 구할 수 있음

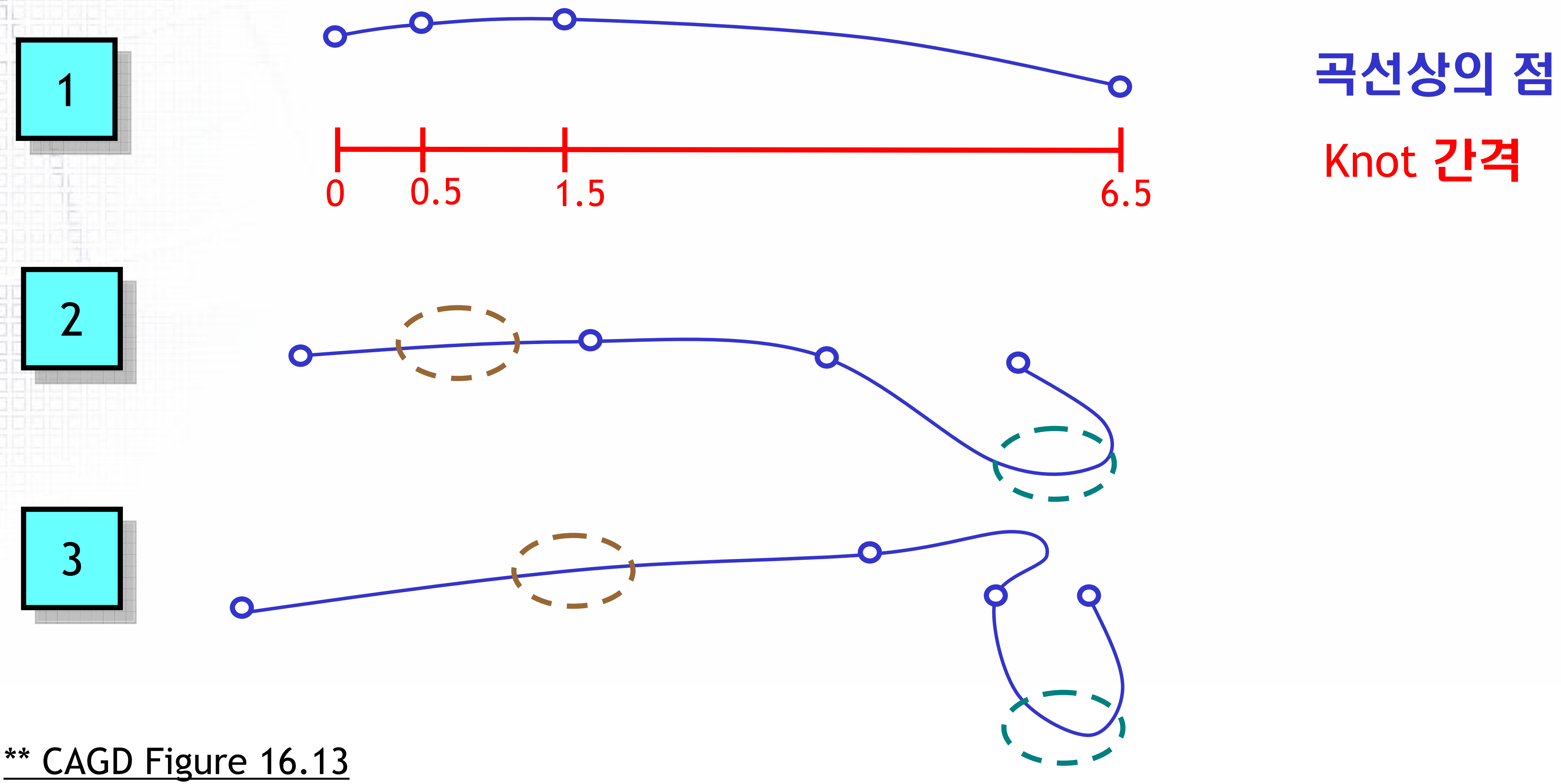
Tensor Product방식으로 정의된
Spline 곡면에서의 Knot 간격예측

주어지는 곡면상의 점이 GRID와 같이 균일하게 주어져야 이상적인 곡면재현 가능

즉, 주어진 곡선들의 Knot 간격이 모두 일정해야 함.

3.3.2.4 Knot 간격 차이가 주는 영향

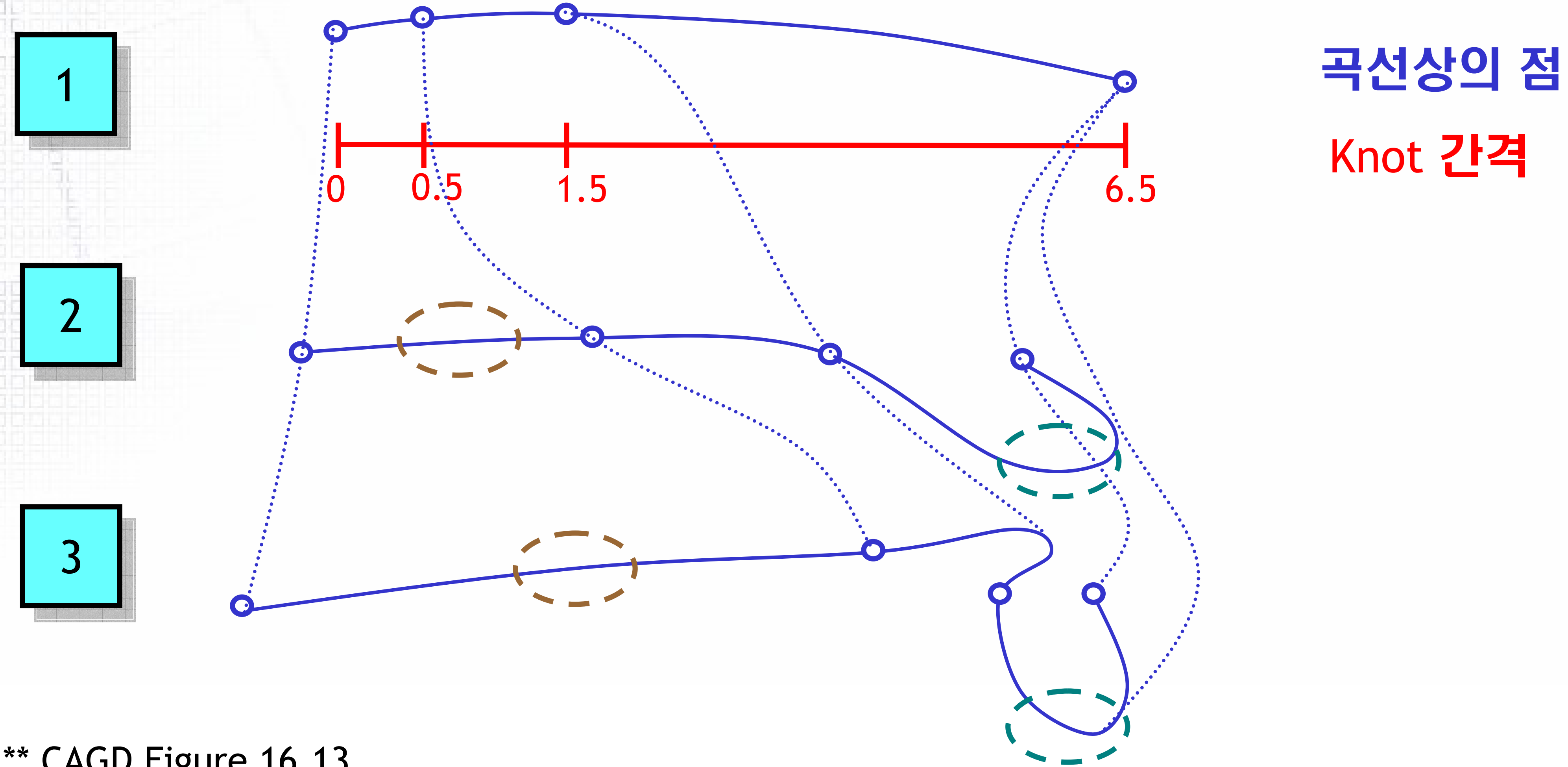
점과 점 사이의 knot 간격은 그 점 사이를 지나가는 데 걸리는 시간과 같은 개념이다.



** CAGD Figure 16.13

3.3.2.4 Knot 간격 차이가 주는 영향

점과 점 사이의 knot 간격은 그 점 사이를 지나가는 데 걸리는 시간과 같은 개념이다.



** CAGD Figure 16.13

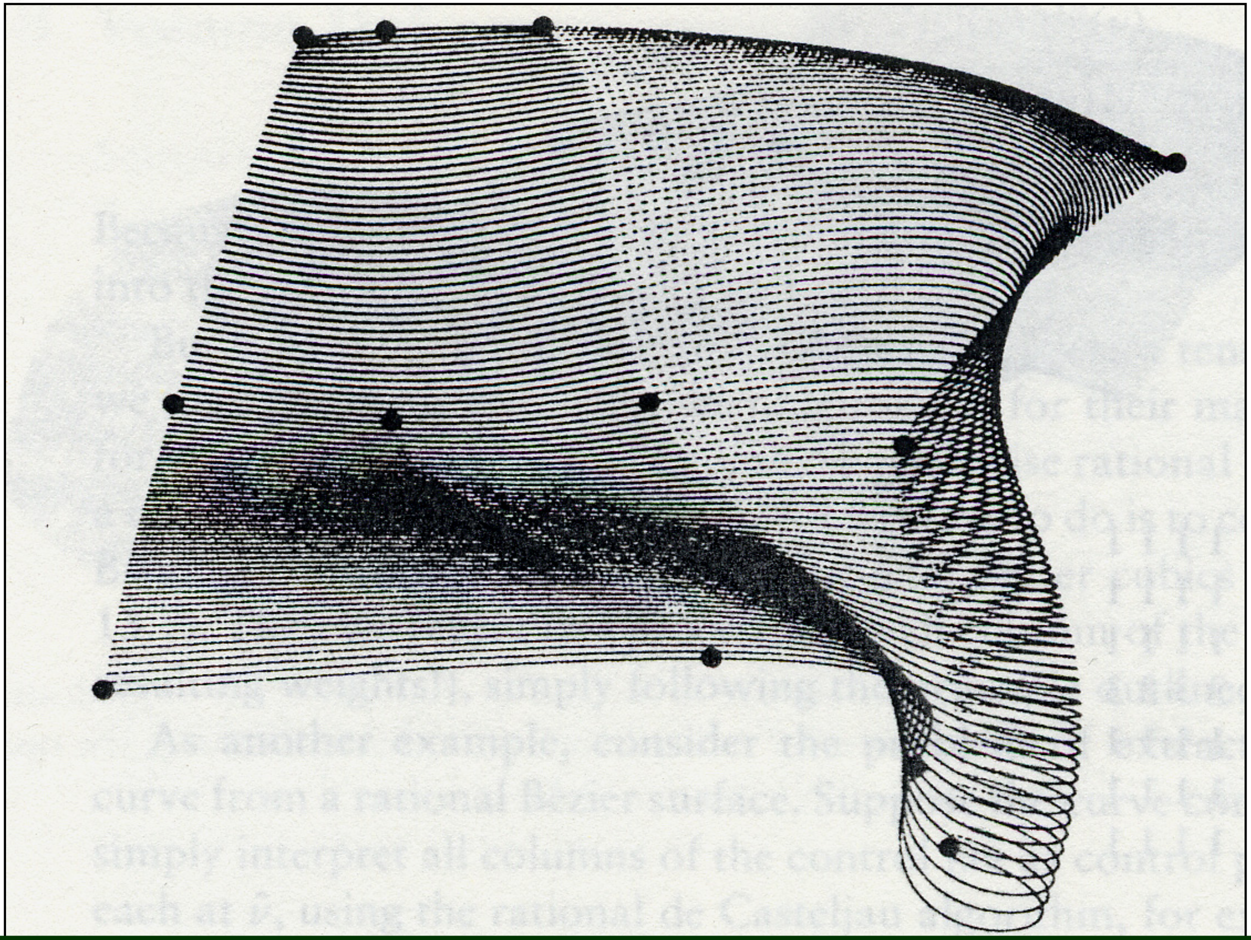
3.3.2.4 Knot 간격 차이가 주는 영향

점과 점 사이의 knot 간격은 그 점 사이를 지나가는 데 걸리는 시간과 같은 개념이다.

1

2

3



곡선상의 점

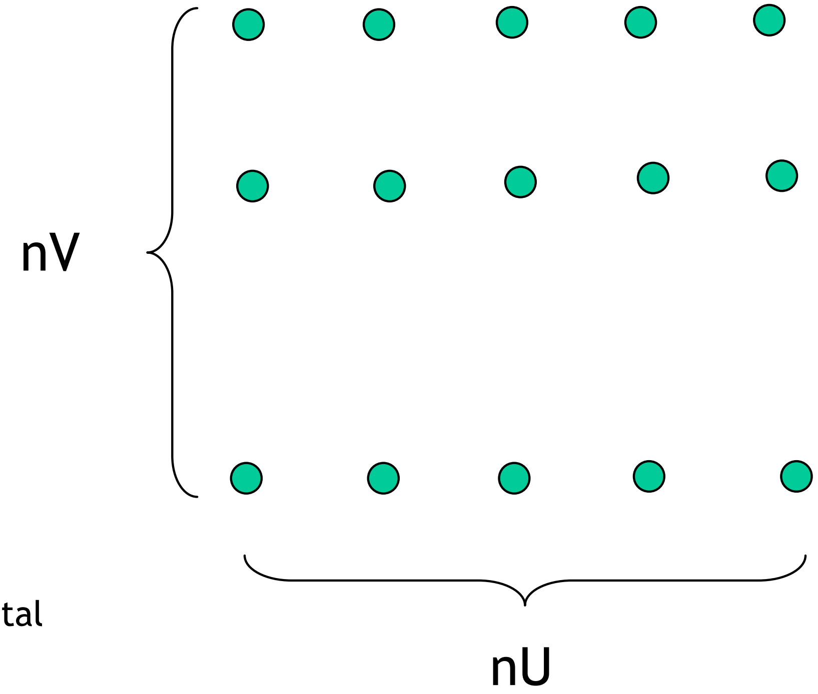
Knot 간격

** CAGD

그러므로 knot간격비가 비슷할 수록 곡면의 왜곡이 적다.

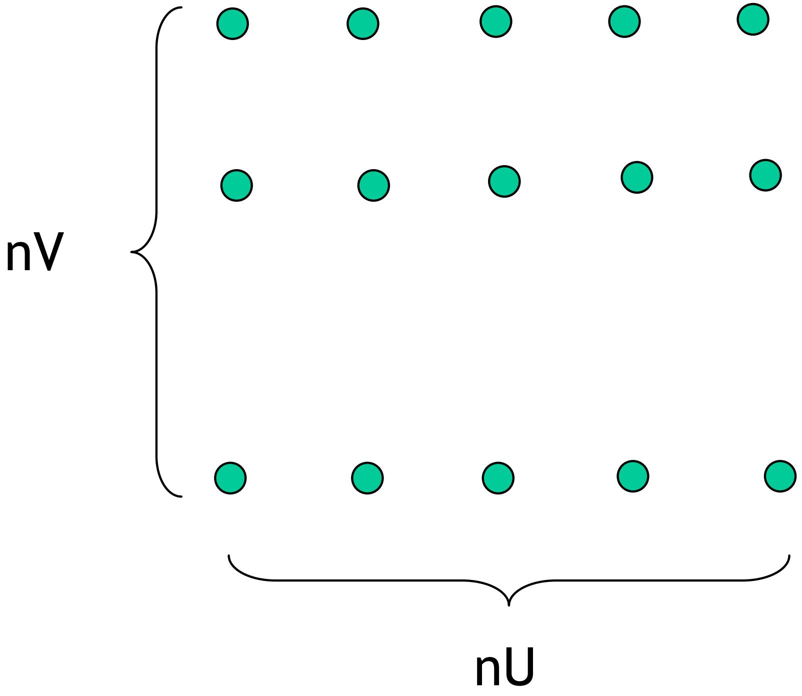
3.3.2.5 Sample code of bicubic B-spline Surface (1)

```
void BicubicBSplineSurface::Interpolate(Vector **pFittingPoint, int nU, int nV) {
   // Generate u-Knot
      if(m_UKnot) delete[] m_UKnot;
      m_nUKnot = (m_nU - 2) + 2*(3+1);
      m_UKnot = new double [m_nUKnot];

      // Initial u-Knot
      double** tmpUKnots;
      tmpUKnots = new double*[nV];
      for(int j = 0; j < nV; j++){
          tmpUKnots[j] = new double[nU];
          for(int i = 0; i < nU; i++){
                tmpUKnot[j][i] = ...;    // chord length or centripetal
          }
      }
      // generate average u-Knot
     for(int i = 0; i < nU; i++){
          m_UKnot[i] = 0;
          for(int j=0; j<nV; j++) {        m_UKnot[i] += tmpUKnot[j][i];       }
          m_UKnot[i] = m_UKnot[i] / nV;
   }
```



3.3.2.5 Sample code of bicubic B-spline Surface (2)

```
// Interpolate u-directional B-spline curve
CubicBsplineCurve* u_curve = new CubicBsplineCurve[nV];
for(int j = 0; j < nV; j++){
    u_curve[j].SetKnot( m_UKnot );
    u_curve[j].Interpolate(  pFittingPoint[j], nU );
}

// Generate v-directional Fitting Point
int nvFittingPoint = u_curve[0].m_nControlPoint;
Vector** vFittingPoint = new Vector [ nvFittingPoint ];
for(int j=0; j < nvFittingPoint; j++){
    vFittingPoint[j] = new Vector[ nV ];
    for( int i = 0; i < nV; i++){
          vFittingPoint[j][i] = u_curve[i].m_ControlPoint[j];
    }
}
……………………..
```



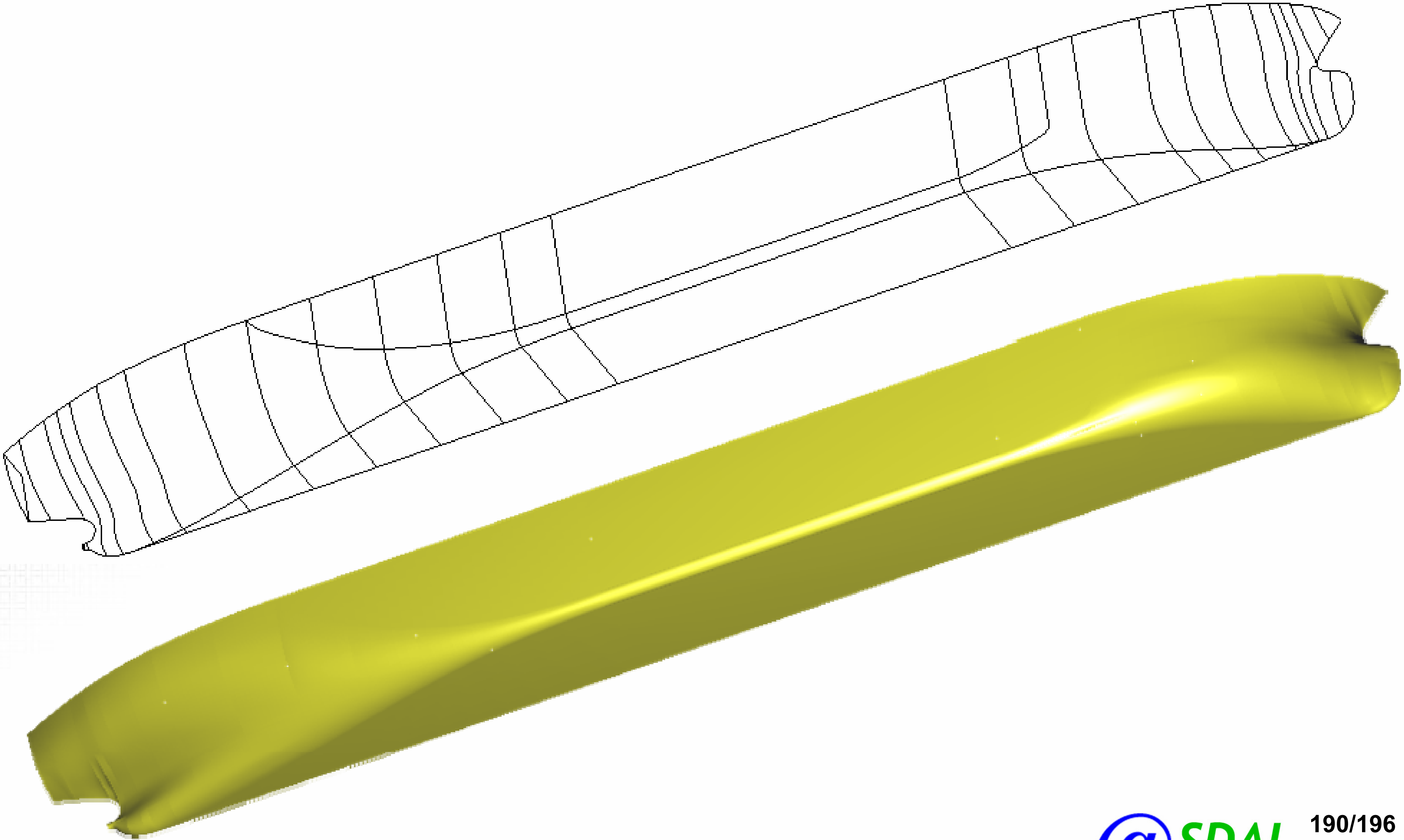
Ch 4. Term Project

Generation Ship hull surfaces
by interpolating given points

- Given: $\mathbf{P}_{i,j}$
- Find : $\mathbf{d}_{i,j}$

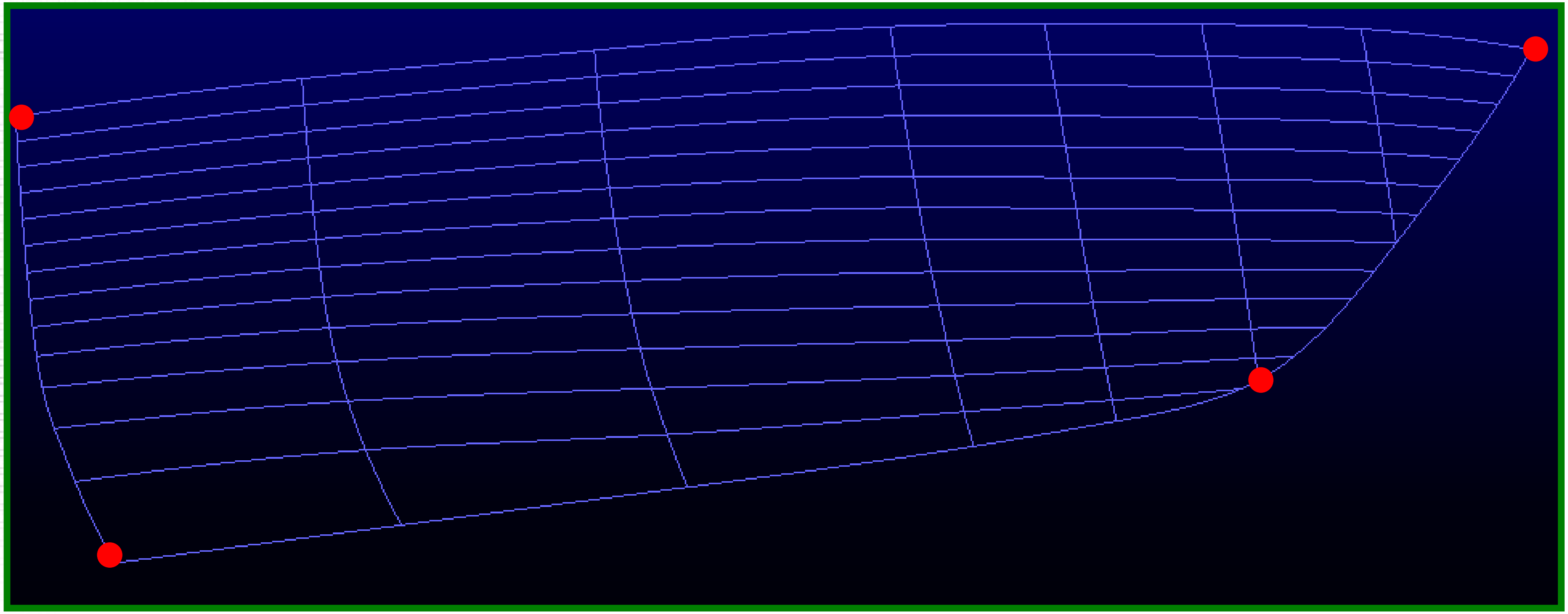
Advanced
Ship
Design
Automation
Laboratory

4.1 단일 B-spline 곡면 patch를 이용한 선형곡면 생성 프로그램 구현



4.2 간단한 요트 형상의 선수부 곡선그물망 형상

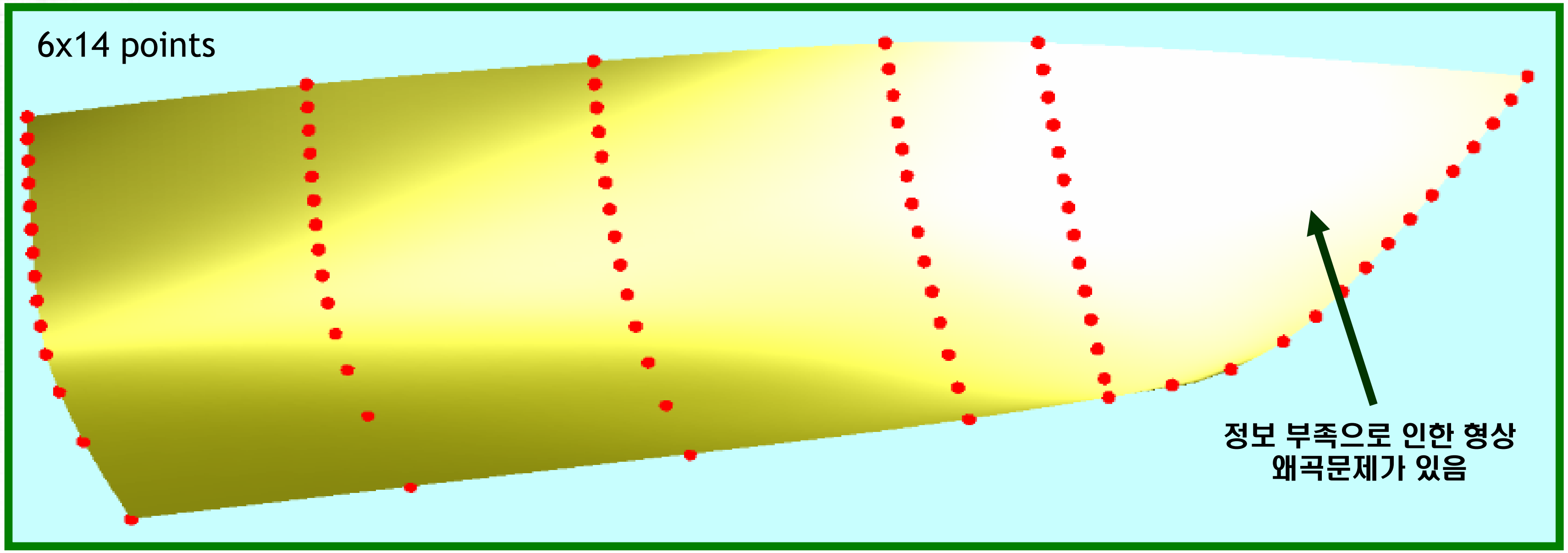
* 출처 : 서울대 조선해양공학과 2005년 2학년 교과목 『조선해양공학계획』 강좌 중에 학생들이 설계한 선형



사각형 패치의 꼭지점 결정

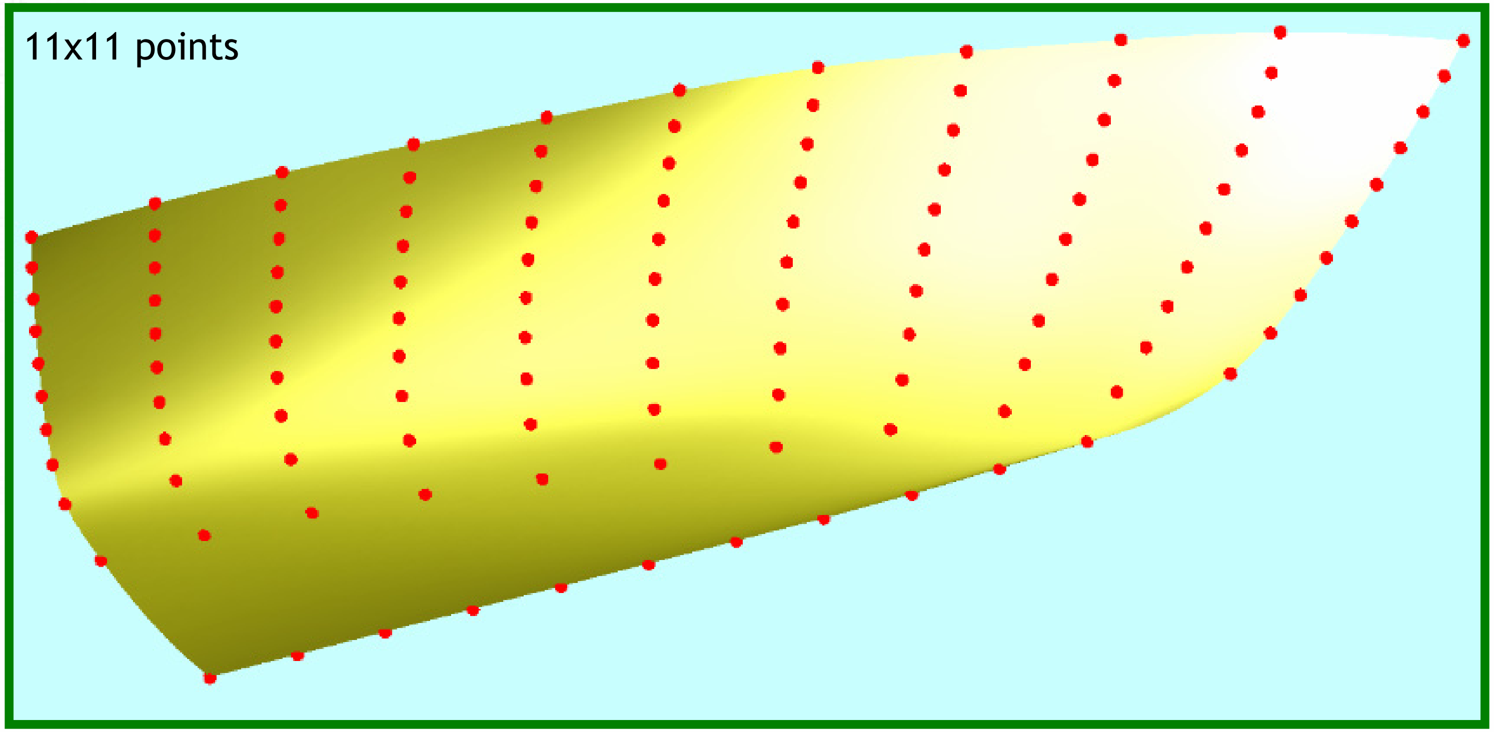
4.3 요트 형상의 곡선그물망으로부터 선형곡면 생성결과 (1)

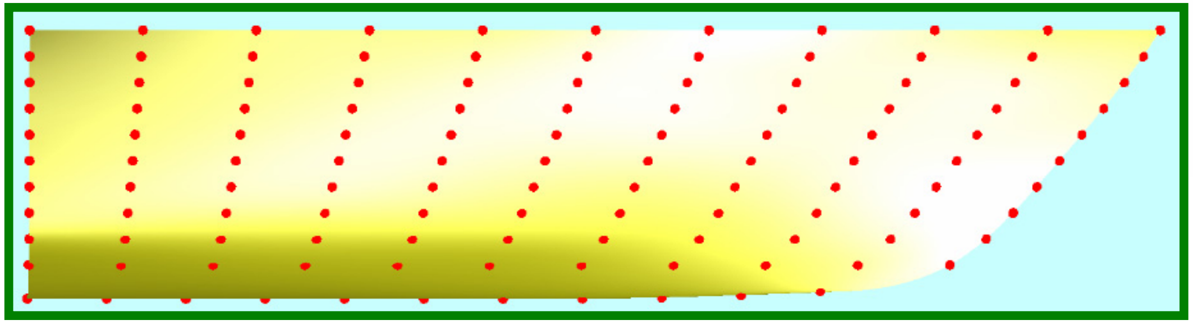
- Offset table **형식으로 점을 추출한 후,**
  이 점들로 부터 bicubic B-spline **선형곡면을 생성한 결과**



4.3 요트 형상의 곡선그물망으로부터 선형곡면 생성결과 (2)

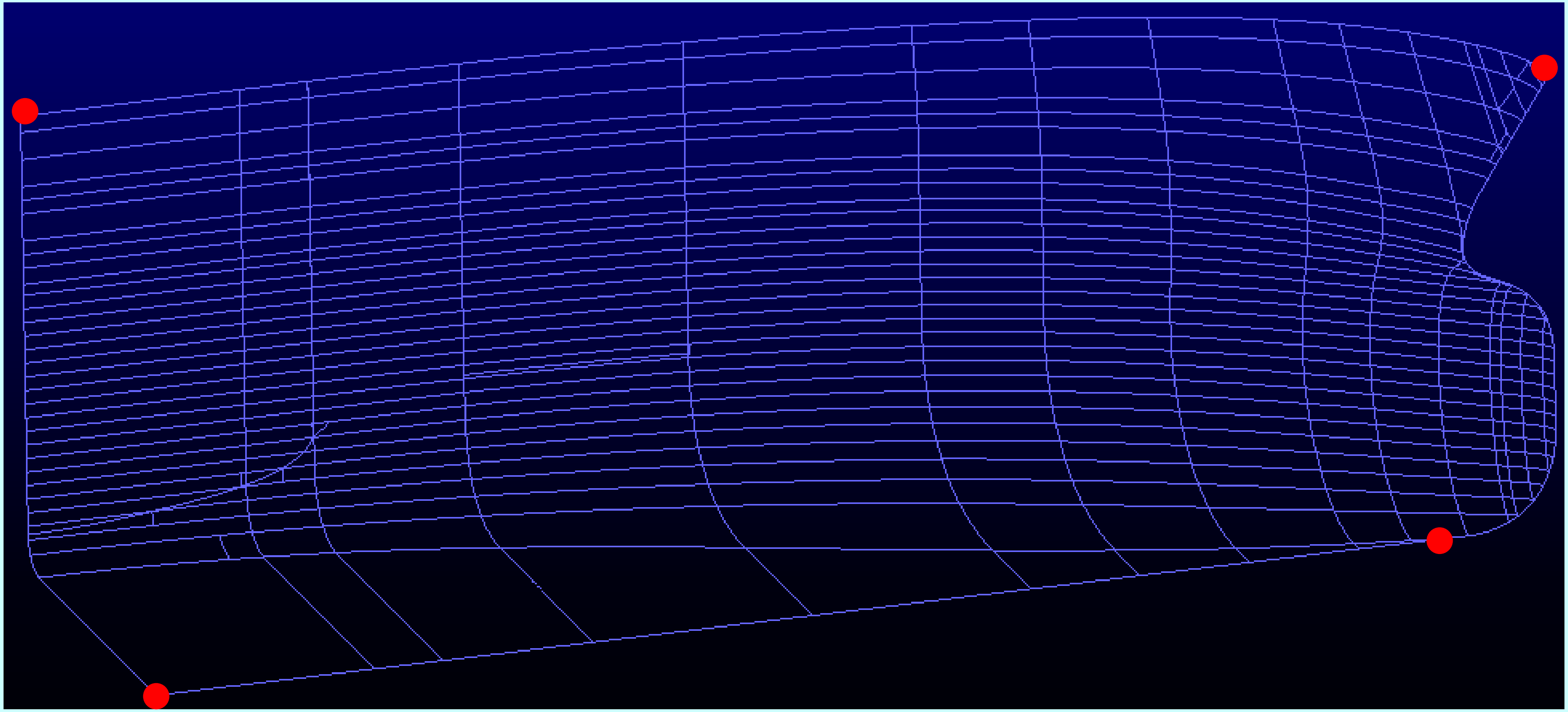
□ 점들의 x좌표 사이의 거리가 일정하도록 점을 추출한 후,
이 점들로부터 bicubic B-spline 선형곡면을 생성한 결과





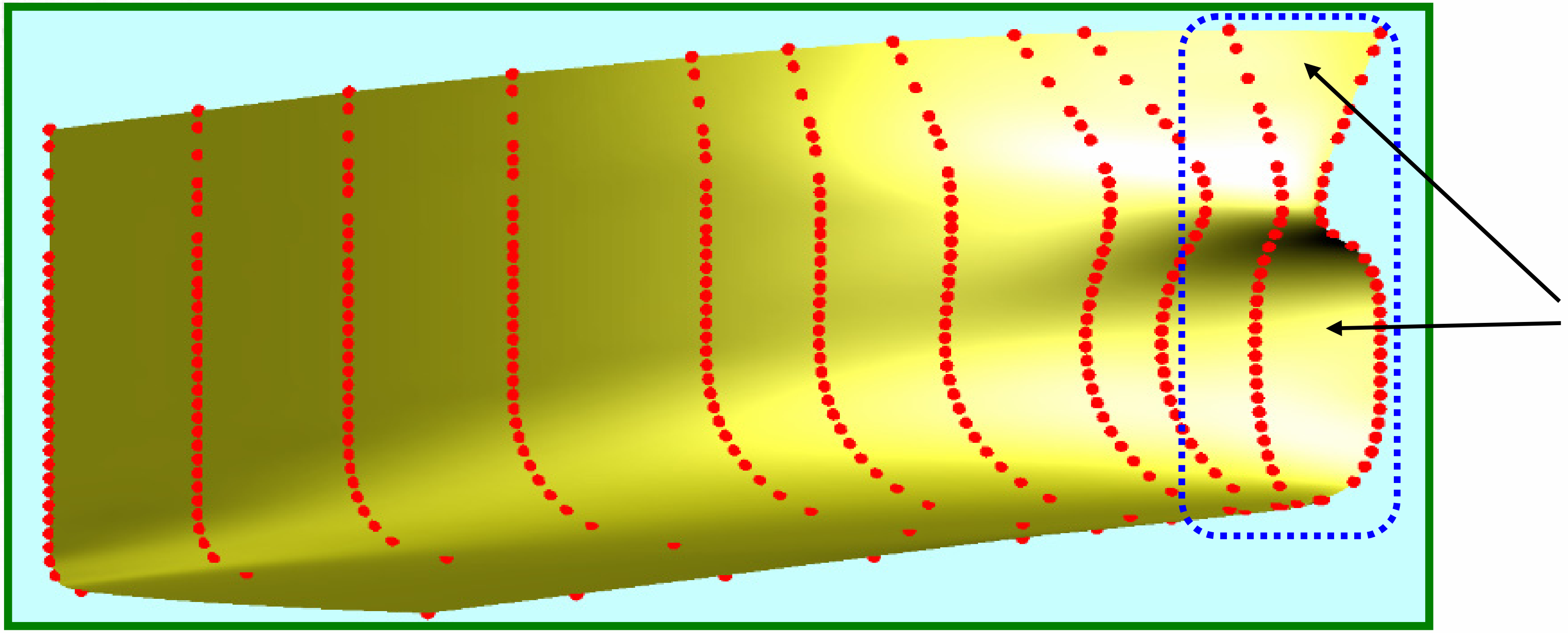
4.4 구상선수를 갖는 단축선의 선수부 곡선그물망 형상

선수부



4.5 구상선수부 곡선그물망으로부터 선수부 선형곡면 생성결과

□ 주어진 곡선그물망 이외의 보조선을 생성하여 곡선 보간에 적합한 점 data를 생성한 후, 점 data로부터 선수부 선형곡면을 생성한 결과



참고 문헌

- **이규열, 조두연, 노명일, 차주환, 전산선박설계**, 3th Ed., 2003.9
- G. Farin, Curves and Surfaces for CAGD, 5th Edi., Academic Press, 2002
- G. Farin and D. Hansford, The Essemtials of CAGD, A K Peters, 2000