计算机图形学

曲线与曲面建模

陶钧
taoj23@mail.sysu.edu.cn
中山大学 计算机学院
国家超级计算广州中心

曲线与曲面建模提纲

- 引言
- 贝塞尔曲线及曲面
- B样条曲线及曲面

讲义中部分图片来自我的导师Prof. Ching-Kuang Shene的讲义

曲线与曲面建模

此前，我们讨论过的几何建模方式主要有两种

- 解析形式及分段线性（piecewise linear）形式
- 以曲线为例，解析形式包括
  - 显示（explicit）表示
  - 隐式（implicit）表示
  - 参数（parametric）表示
- 显示表示主要问题
  - 与坐标轴相关
  - 会出现斜率无穷大的情况
  - 不便于计算机编程

$y = x^2+5x+3 \longrightarrow y=f(x)$

(explicit curve)

$(x-x_c)^2 + (y-y_c)^2-r^2=0 \longrightarrow g(x,y)=0$

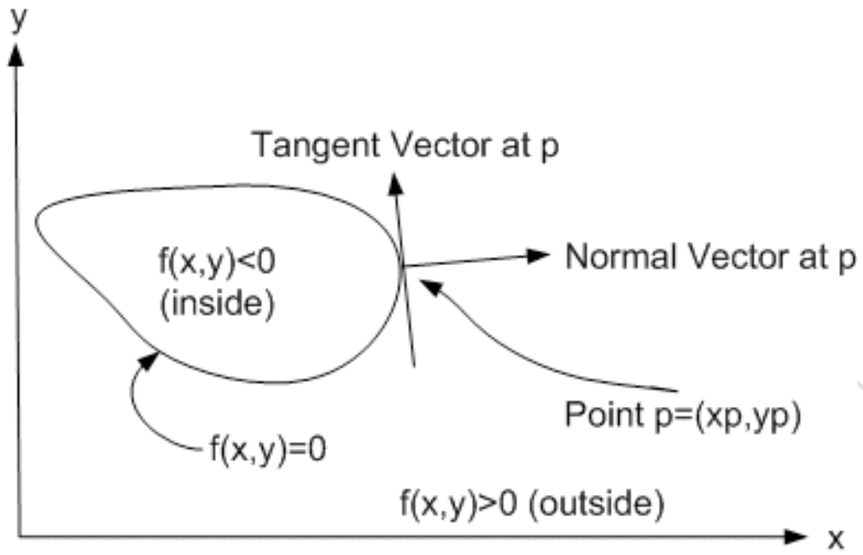
(implicit curve)

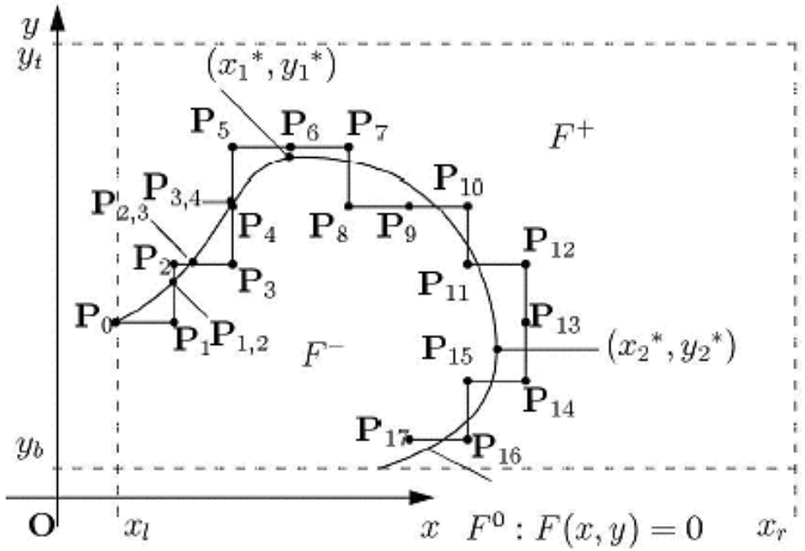
$$x = x_c + r \cdot \cos\theta \qquad y = y_c + r \cdot \sin\theta \longrightarrow \begin{cases} x = x(t) \\ y = y(t) \end{cases}$$

(parametric curve)

曲线与曲面建模

- 此前，我们讨论过的几何建模方式主要有两种
  - 解析形式及分段线性（piecewise linear）形式
  - 隐式表示
    - 对于给定点容易决定是否在曲线/曲面上
    - 但对于给定曲线/曲面，难以对其进行采样

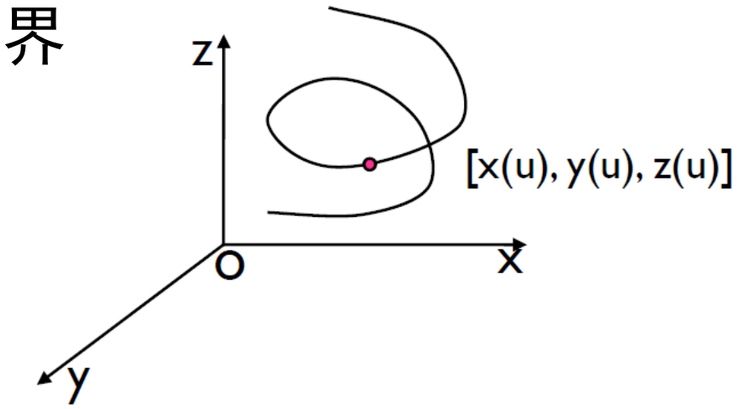




曲线与曲面建模

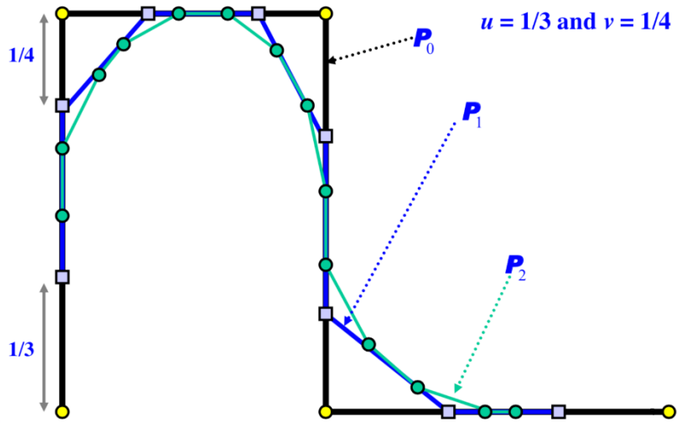
此前，我们讨论过的几何建模方式主要有两种

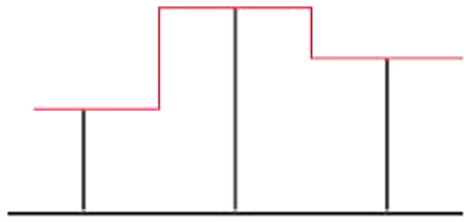
- 解析形式及分段线性（piecewise linear）形式
- 参数表示
  - 满足几何不变形的要求
  - 有更大的自由度来控制曲线、曲面的形状
  - 对曲线、曲面进行变化，可直接对其参数方程进行几何变换
  - 便于处理斜率无穷大的情形
  - 规格化的参数变量$t \in [0,1]$，不必额外参数定义边界
  - 易于用矢量和矩阵表示几何分量
- 如何找出需要建模的几何形状的参数表示?



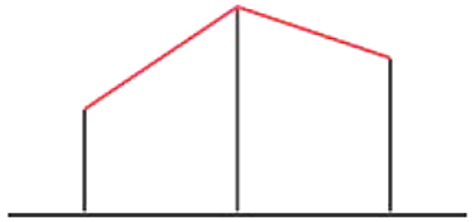
曲线与曲面建模

- 此前，我们讨论过的几何建模方式主要有两种
  - 解析形式及分段线性（piecewise linear）形式
  - 分段线性在采样点间通过线性插值得到完整曲线/曲面
    - 假设采样点之间通过直线/平面连接
  - 为产生外表更光滑的曲线，只能增加采样点的数量
    - 如对曲线的corner cutting算法
  - 通过采样点更易构造需要表示的几何形状，但如何使其更平滑？





NN interpolation

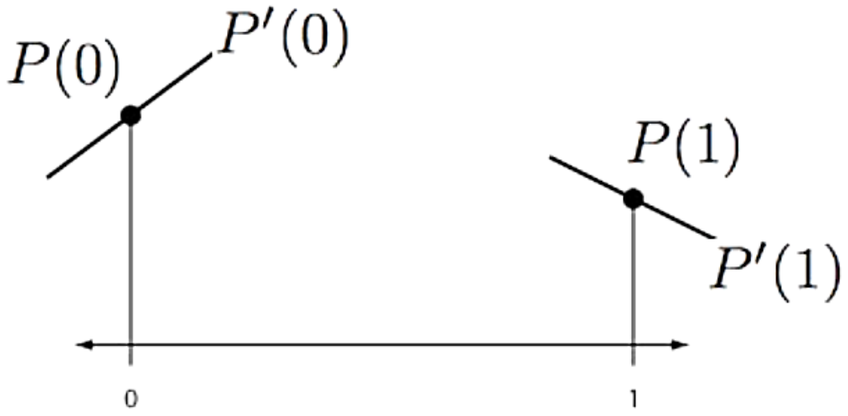


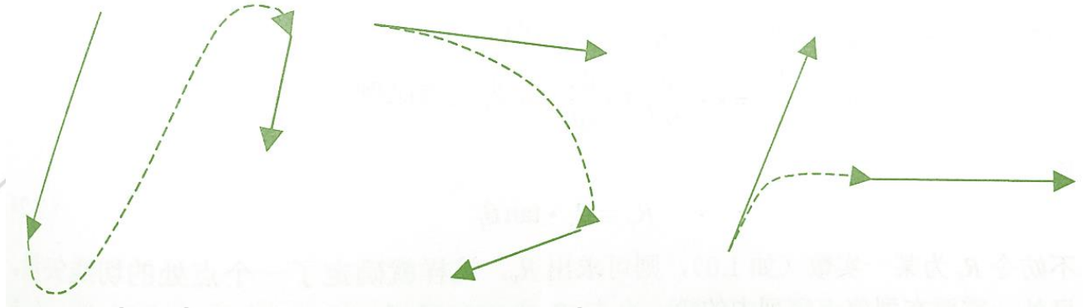
Linear interpolation

Smooth curve

曲线与曲面建模

- 此前，我们讨论过的几何建模方式主要有两种
  - 解析形式及分段线性（piecewise linear）形式
  - 使用分线性插值能得到更光滑的曲线
  - 如，cubic Hermite interpolation
    - 给定两个端点$P(0), P(1)$
    - 及端点上的斜率$P'(0), P'(1)$
    - 假设端点间通过三次多项式表示的曲线连接
      - $P(t) = at^3 + bt^2 + ct + d$
      - $P'(t) = 3at^2 + 2bt + c$
      - $P(0) = d$
      - $P(1) = a + b + c + d$
      - $P'(0) = c$
      - $P'(1) = 3a + 2b + c$





曲线与曲面建模提纲

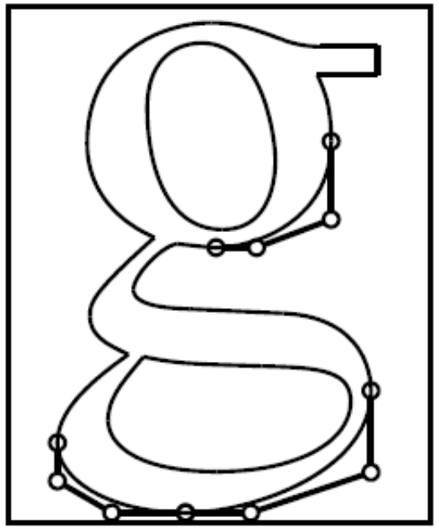
引言

贝塞尔曲线及曲面

B样条曲线及曲面

贝塞尔曲线及曲面

- 贝塞尔曲线及曲面（Bézier curve and surface)
  - 计算机图形学中常用的参数式曲线/曲面表现形式
  - 1971年，由法国雷诺汽车公司的Pierre Bézier提出的构造样条曲线和曲面的方法
  - 比较直观地表示给定的条件与曲线形状的关系，使用户能方便地通过修改参数来改变曲线的形状和阶次



贝塞尔曲线及曲面

贝塞尔曲线

– 由调和函数（harmonic functions）根据控制点（control points）插值生成

– N阶贝塞尔曲线可表示为

$$\boldsymbol{C}(u) = \sum_{i=0}^{n} B_{n,i}(u)\boldsymbol{P}_i$$

- 其中u为参数，$B_{n,i}(u)$为伯恩斯坦（Bernstein）基函数

$$B_{n,i}(u) = \frac{n!}{i!\,(n-i)!}u^i(1-u)^{n-i}$$

- 注意，对任意给定的$u, B_{n,i}(u) > 0$，都有$\sum B_{n,i}(u) = 1$（partition of unity）

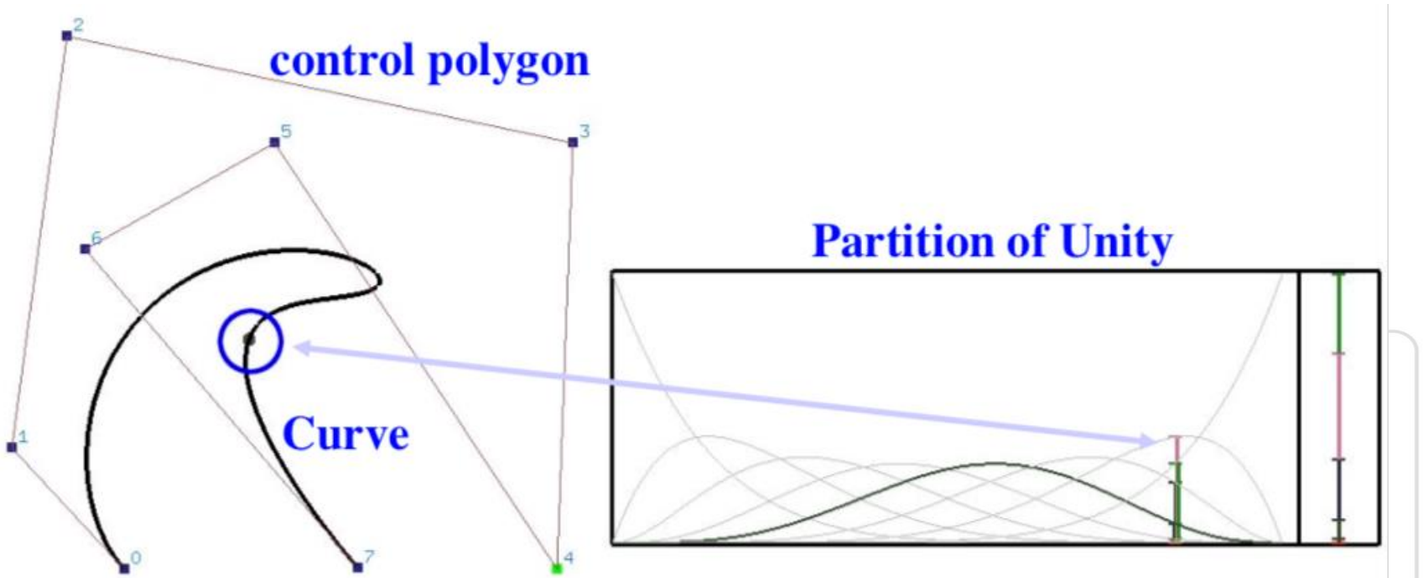
贝塞尔曲线及曲面

贝塞尔曲线

– N阶贝塞尔曲线可表示为

$$\boldsymbol{C}(u) = \sum_{i=0}^{n} B_{n,i}(u)\boldsymbol{P}_i$$

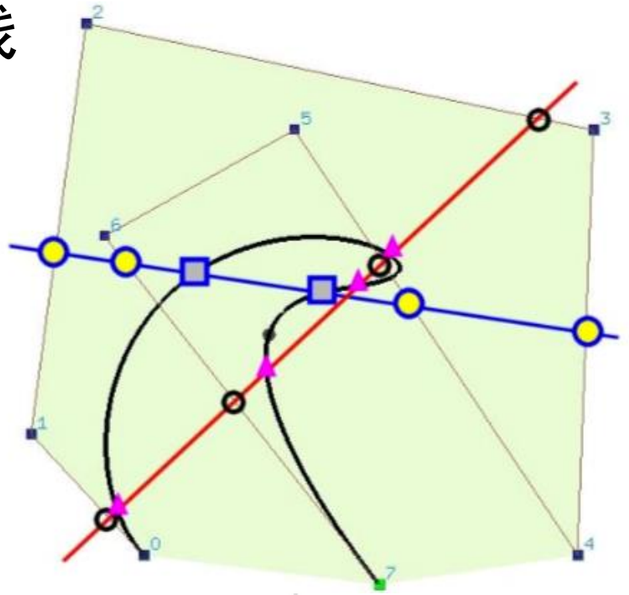
- $\boldsymbol{P}_0$，$\boldsymbol{P}_1$，…，$\boldsymbol{P}_n$为控制点，其形成的形状称为control polygon/polyline



贝塞尔曲线

– 贝塞尔曲线具有以下重要性质

- Convex hull property：贝塞尔曲线位于其控制点所定义的凸包中
- Variation diminishing property：对于平面内的贝塞尔曲线而言，任意直线与贝塞尔曲线的相交次数不超过其与control polyline相交的次数
- Affine invariance：贝塞尔曲线在仿射变换下保持不变，意味着我们可以对控制点进行变换，从而得到变换后的贝塞尔曲线

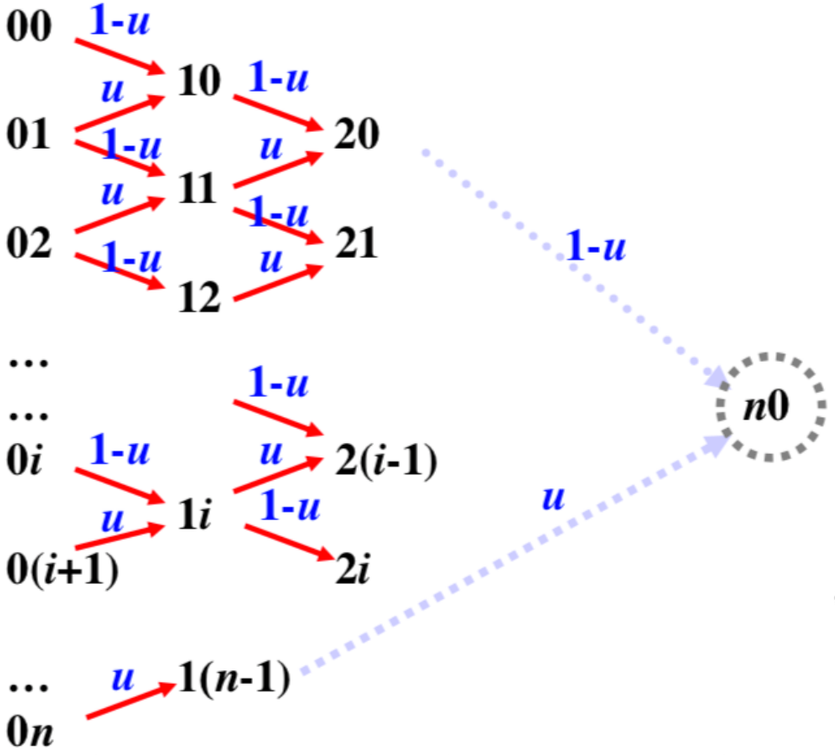


贝塞尔曲线及曲面

- De Casteljau's algorithm
  - 高效且稳定的计算方法
  - 对于给定u，计算贝塞尔曲线上的点$\boldsymbol{C}(u)$
  - 将$n+1$个控制点$\boldsymbol{P}_0$，$\boldsymbol{P}_1$，$\boldsymbol{P}_2$，$\cdots$，$\boldsymbol{P}_n$记为$\boldsymbol{00}, \boldsymbol{01}, \boldsymbol{02}, \cdots, \boldsymbol{0n}$（列0）
  - 对于相邻的两个控制点$\boldsymbol{0i}$及$\boldsymbol{0(i+1)}$，通过线性插值计算$\boldsymbol{1i}$
    - $\boldsymbol{1i} = (1-u)\cdot\boldsymbol{0i} + u\cdot\boldsymbol{0(i+1)}$
  - 按这一方式，可从$n+1$个控制点中产生n个新点
    - $\boldsymbol{10}, \boldsymbol{11}, \cdots, \boldsymbol{1(n-1)}$（列1）
  - 类似地，从列1的n个点$\boldsymbol{10}, \boldsymbol{11}, \cdots, \boldsymbol{1(n-1)}$中，可产生$n$-1个新点
    - $\boldsymbol{20}, \boldsymbol{21}, \cdots, \boldsymbol{2(n-2)}$（列2）
    - $\boldsymbol{2i} = (1-u)\cdot\boldsymbol{1i} + u\cdot\boldsymbol{1(i+1)}$

贝塞尔曲线及曲面

- De Casteljau's algorithm
  - 重复这一过程，第k次循环产生$n-k+1$个点$\boldsymbol{k0}, \boldsymbol{k1}, \cdots, \boldsymbol{k(n-k)}$
    - $\boldsymbol{ki} = (1-u)\cdot(\boldsymbol{k-1})\boldsymbol{i} + u\cdot(\boldsymbol{k-1})(\boldsymbol{i+1})$
  - 将此过程重复n次，则最后的到的点$\boldsymbol{n0}$即为$\boldsymbol{C}(u)$



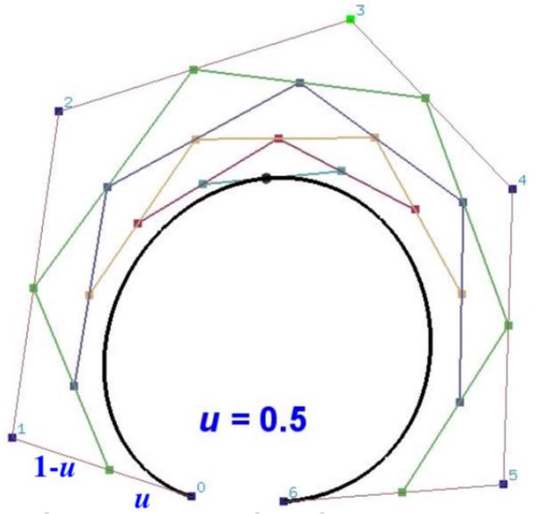
贝塞尔曲线及曲面

De Casteljau's algorithm

– 几何意义

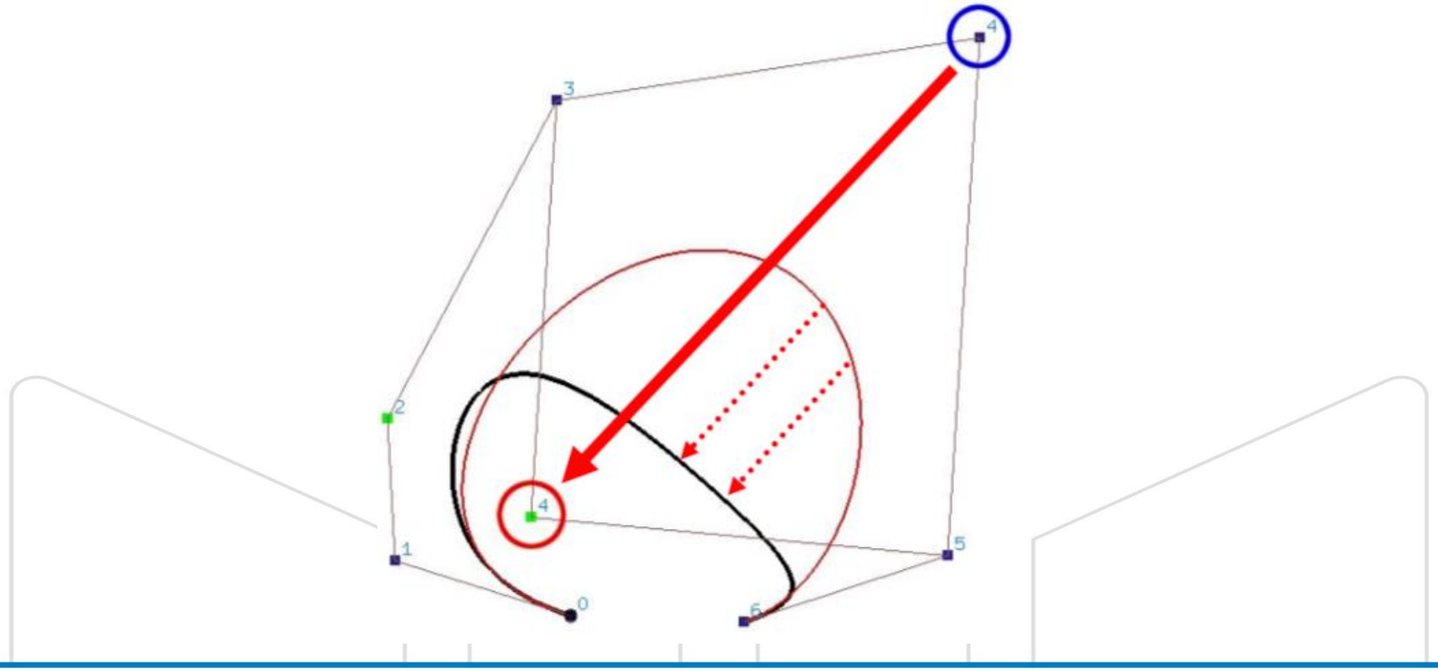
- 以$1-u$及u为参数，对两个点A, B进行线性插值得到的点为AB连线上的一点，其距离A与B的比值为$u:1-u$

- De Casteljau算法可视为，通过线性插值，从最原始的控制点一步步靠近曲线的过程（de Casteljau net）



移动控制点

- 移动控制点可改变贝塞尔曲线的形状
- 将控制点向一个方向移动可以导致贝塞尔曲线整体向该方向移动
  - 造成全局改变，即原曲线上的任意一点都会发生变化
    - 由贝塞尔曲线表达式$\boldsymbol{C}(u)=\sum_{i=0}^{n}B_{n,i}(u)\boldsymbol{P}_i$易知



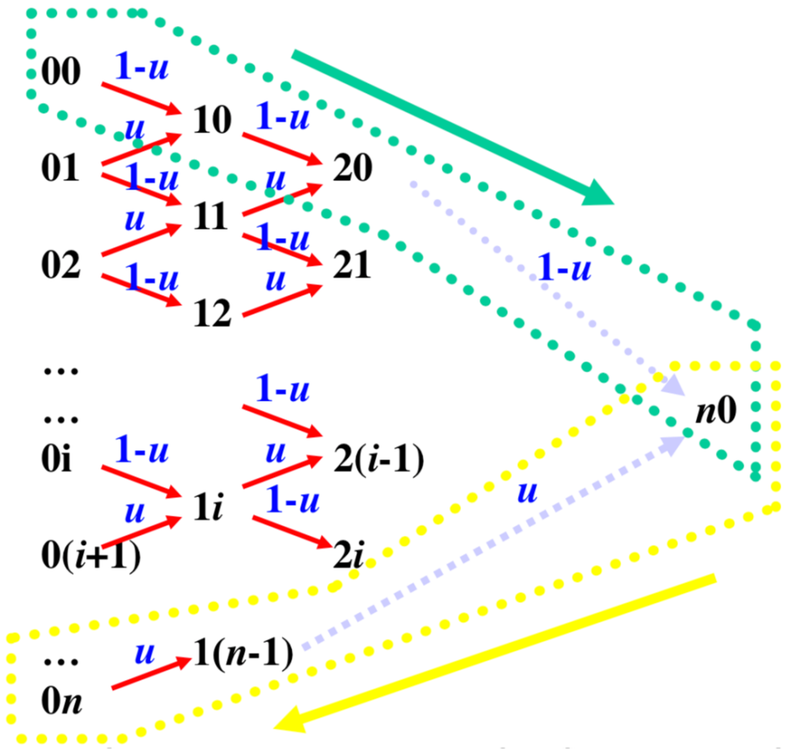
贝塞尔曲线及曲面

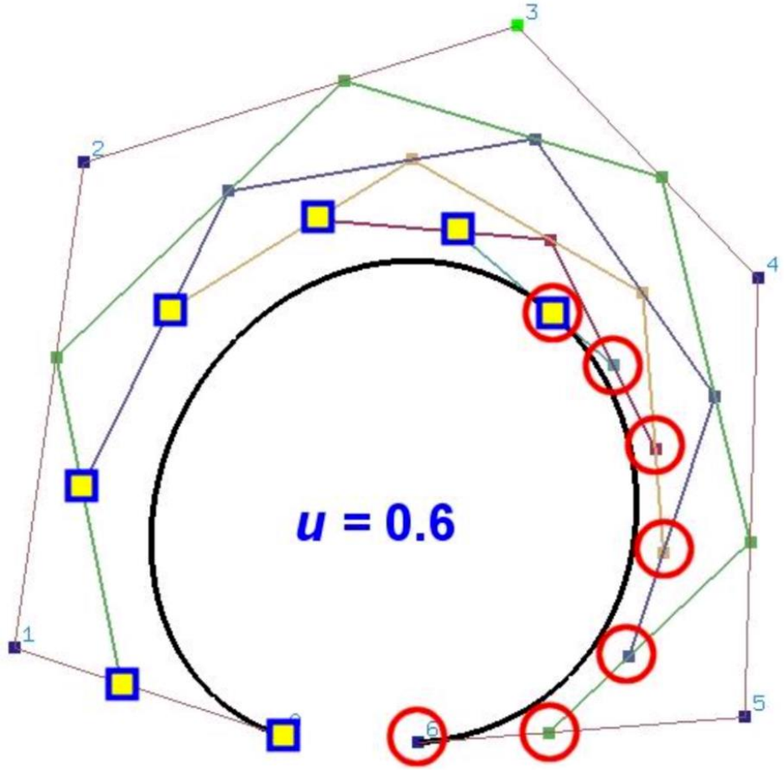
- 贝塞尔曲线细分
  - 控制点影响范围的全局化不利于细节改动
  - 对给定u进行贝塞尔曲线意味着将曲线$\boldsymbol{C}(u)$拆分成两部分，每一部分是一条独立的贝塞尔曲线
    - 即$\boldsymbol{C}(u)$拆分为$[0,u]$上的曲线$\boldsymbol{C}_1(u)$及$[u,1]$上的曲线$\boldsymbol{C}_2(u)$
    - 因此，我们需要找到两组控制点，每组控制点对一条拆分后的曲线
  - 如何计算这两组控制点？

贝塞尔曲线细分

– 从de Casteljau算法中找出这两组控制点

- $[0,u]$上的曲线$\boldsymbol{C}_1(u)$控制点为绿色框内计算全过程中以$1-u$为系数的点
- $[u,1]$上的曲线$\boldsymbol{C}_2(u)$控制点为黄色框内计算全过程中以u为系数的点





贝塞尔曲线及曲面

Degree elevation

- 提升贝塞尔曲线的阶次（degree）但不改变其形状
- 由于贝塞尔曲线的阶次提高，其控制点也需要相应发生改变
  - n阶贝塞尔曲线需要$n+1$个控制点
  - $n+1$阶贝塞尔曲线需要$n+2$个控制点
- 令原控制点为$\boldsymbol{P}_0, \boldsymbol{P}_1, \cdots, \boldsymbol{P}_n$，提升后控制点为$\boldsymbol{Q}_0, \boldsymbol{Q}_1, \cdots, \boldsymbol{Q}_{n+1}$
- 显然，对于端点有$\boldsymbol{P}_0 = \boldsymbol{Q}_0$且$\boldsymbol{P}_n = \boldsymbol{Q}_{n+1}$
- 其他中间控制点$\boldsymbol{Q}_1, \boldsymbol{Q}_2, \cdots, \boldsymbol{Q}_n$可由以下式子给出

$$\boldsymbol{Q}_i = \frac{i}{n+1}\boldsymbol{P}_{i-1} + (1 - \frac{i}{n+1})\boldsymbol{P}_i$$

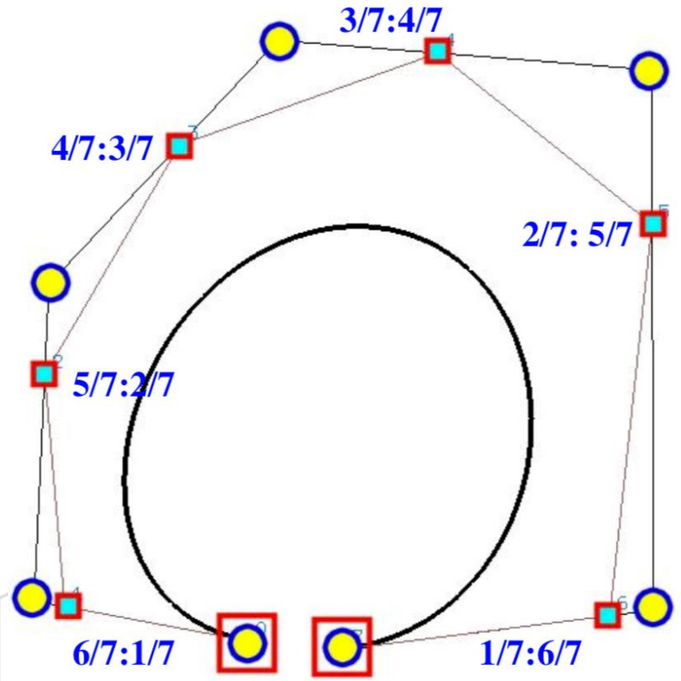
Degree elevation

– 可见，$\boldsymbol{Q}_i$为线段$\boldsymbol{P}_{i-1}\ \boldsymbol{P}_i$上一点，将$\boldsymbol{P}_{i-1}\ \boldsymbol{P}_i$分为两段，其比值为

$$\left(1-\frac{i}{n+1}\right):\frac{i}{n+1}$$

– 对于一条6阶贝塞尔曲线，其degree elevation过程中，新产生的控制点将原control polyline上线段分为

$6:1, 5:2, 4:3, 3:4, 2:5, 1:6$



贝塞尔曲线及曲面

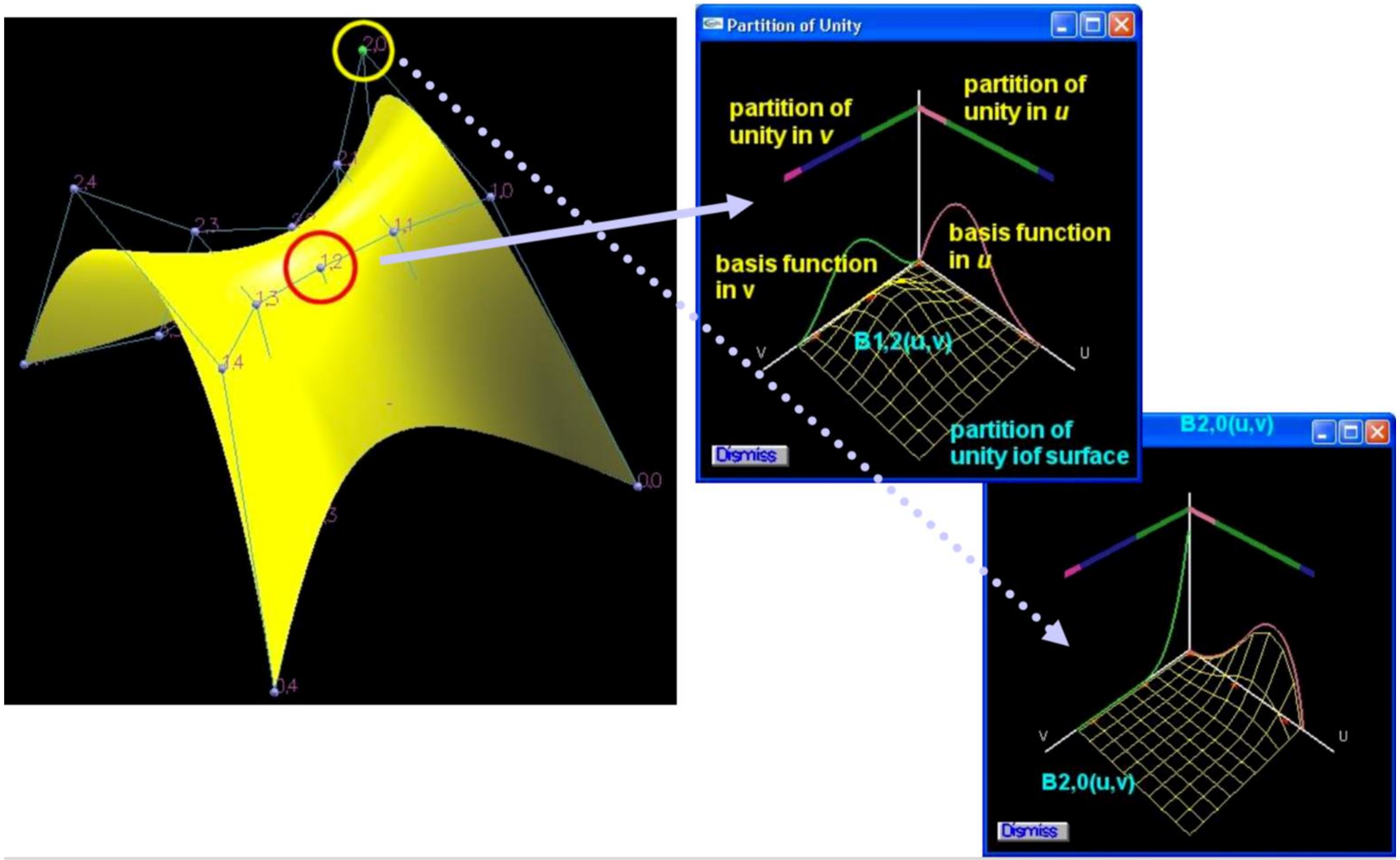
- 贝塞尔曲面
  - 贝塞尔曲线由控制点插值得到
  - 贝塞尔曲面由贝塞尔曲线插值得到
  - 贝塞尔曲面$\boldsymbol{S}(u,v)$(其中$u,v \in [0,1]$)由一个网格的控制点$\boldsymbol{P}_{i,j}$(其中$i \in [0,m], j \in [0,n]$)决定
    - 即$\boldsymbol{S}(u,v)$由$m+1$行$n+1$列控制点决定

$$\boldsymbol{S}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_{m,i}(u)B_{n,j}(v)\boldsymbol{P}_{i,j} = \sum_{i=0}^{m} B_{m,i}(u)\sum_{j=0}^{n} B_{n,j}(v)\boldsymbol{P}_{i,j} = \sum_{i=0}^{m} B_{m,i}(u)\boldsymbol{C}_i(v)$$

    - $B_{m,i}(u)B_{n,j}(v)$称为贝塞尔曲面的基函数，对任意给定u,v满足其和为1（partition of unity）
    - 贝塞尔曲面的degree为(m,n)

贝塞尔曲线及曲面

- 贝塞尔曲面



贝塞尔曲线及曲面

- 使用De Casteljau算法计算贝塞尔曲面
  - De Casteljau算法可扩展至贝塞尔曲面
  - 令$\boldsymbol{Q}_i(v)=\sum_{j=0}^{n}B_{n,j}(v)\boldsymbol{P}_{i,j}$，则贝塞尔曲面可写作

$$\boldsymbol{S}(u,v)=\sum_{i=0}^{m}B_{m,i}(u)\sum_{j=0}^{n}B_{n,j}(v)\boldsymbol{P}_{i,j}=\sum_{i=0}^{m}B_{m,i}(u)\,\boldsymbol{Q}_i(v)$$

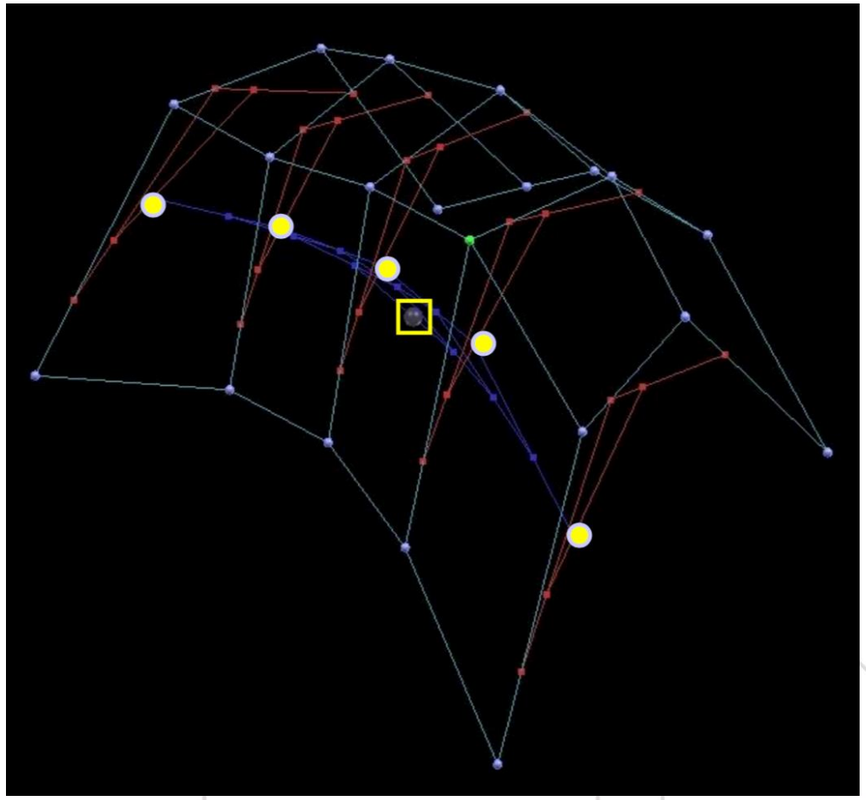
  - 因此，对于给定的(u,v)计算贝塞尔曲面上一点的过程可分解为
    - 对于给定v及m组控制点$\boldsymbol{P}_{i,j}$，使用de Casteljau算法计算$\boldsymbol{Q}_0(v),\boldsymbol{Q}_1(v),\cdots,\boldsymbol{Q}_m(v)$
    - 对于控制点$\boldsymbol{Q}_0(v),\boldsymbol{Q}_1(v),\cdots,\boldsymbol{Q}_m(v)$及给定$u$，使用de Casteljau算法计算$\boldsymbol{S}(u,v)$

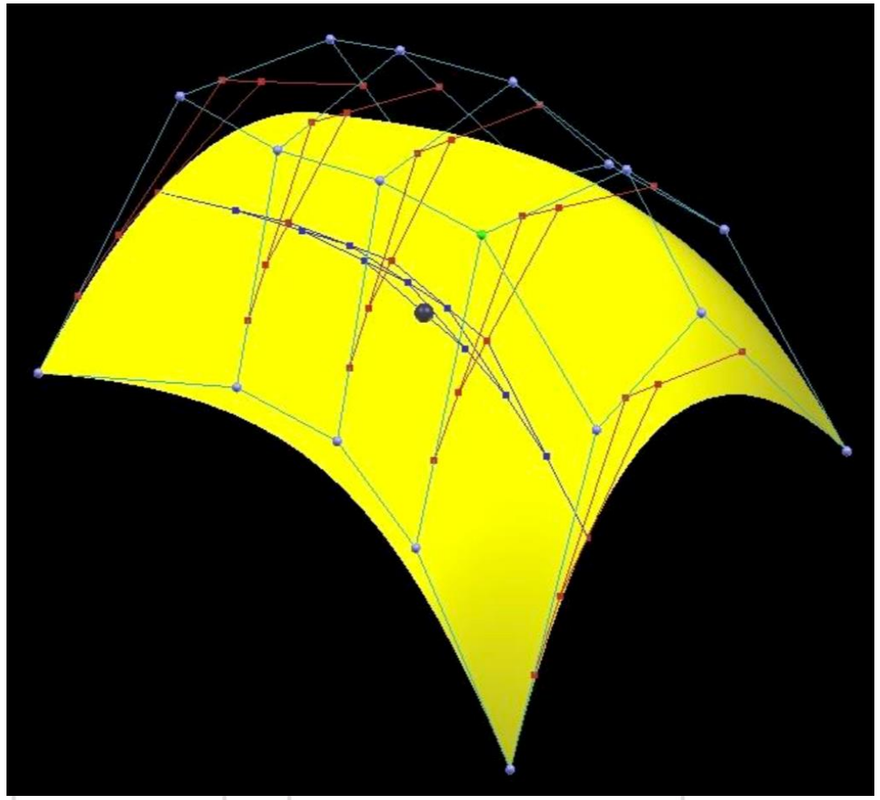
贝塞尔曲线及曲面

使用De Casteljau算法计算贝塞尔曲面

– 对于degree为$(4,3)$的贝塞尔曲面

- 首先使用de Casteljau算法计算5个$\boldsymbol{Q}_i(v)$（黄色圆点）
- 对于5个$\boldsymbol{Q}_i(v)$，使用de Casteljau算法计算$\boldsymbol{S}(u,v)$（黄色方框）



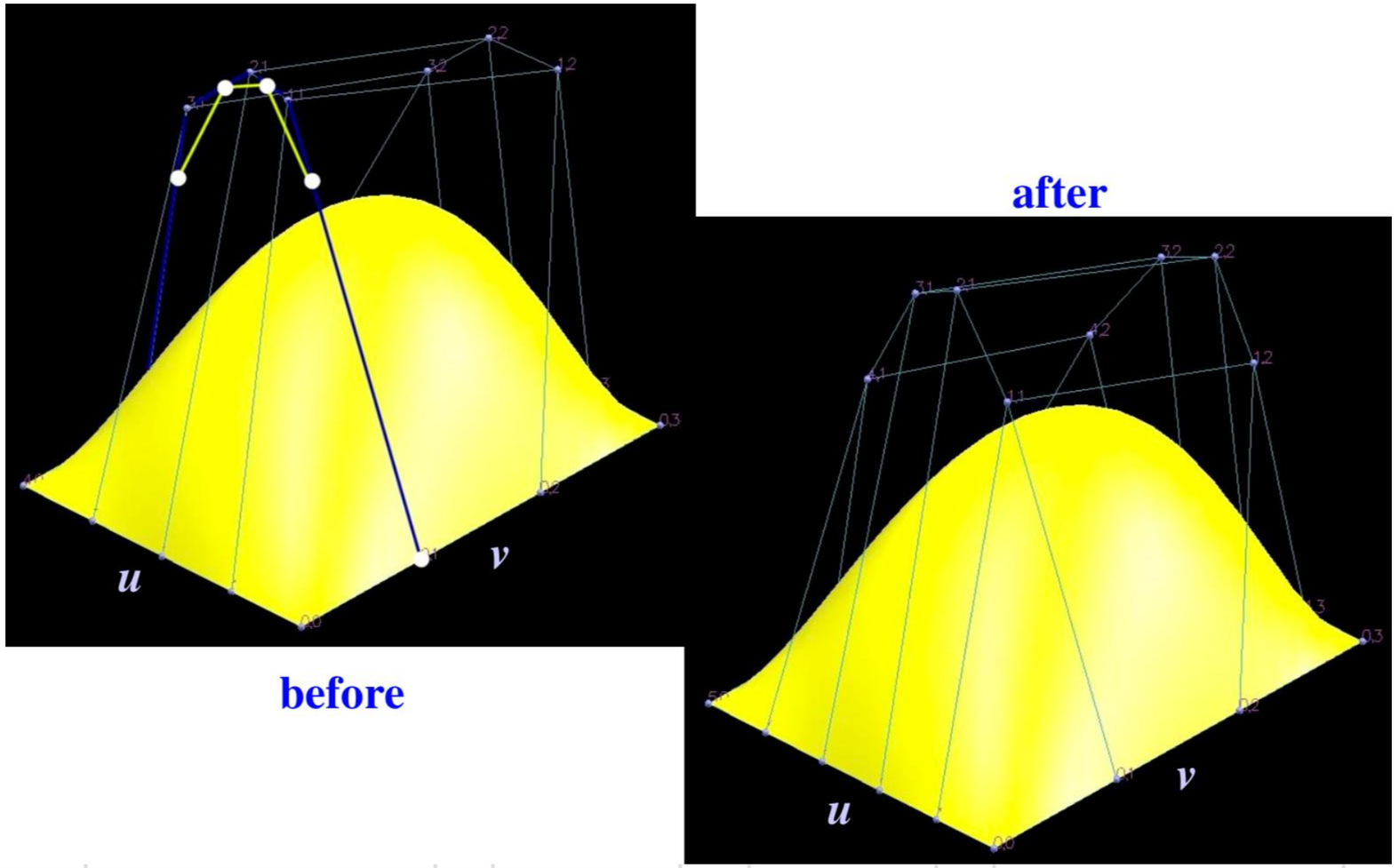


贝塞尔曲线及曲面

- 贝塞尔曲面
  - 同样，移动贝塞尔曲面的控制点将会产生全局影响
  - 因此，我们同样需要对贝塞尔曲面进行细分及degree elevation
  - 计算方法与贝塞尔曲线的细分及degree elevation相同
  - 细分：沿v将贝塞尔曲面$\boldsymbol{S}(u,v)$细分为两个曲面$\boldsymbol{S}_1(u,v) \in [0,1] \times [0,v]$及$\boldsymbol{S}_2(u,v) \in [0,1] \times [v,1]$
    - 将网格上每列的控制点按$\boldsymbol{Q}_i(v)$细分成两组控制点
  - Degree elevation：提升贝塞尔曲线的degree (m,n)
    - 如果需要将m增加至$m+1$，则需要增加一行控制点，即对于每列控制点，使用贝塞尔曲线的degree elevation方法，增加一个控制点
    - 如果需要将n增加至$n+1$，则需要增加一列控制点，即对于每行控制点，使用贝塞尔曲线的degree elevation方法，增加一个控制点

贝塞尔曲面

– Degree elevation



贝塞尔曲线及曲面

- 使用OpenGL evaluator绘制贝塞尔曲线及曲面
  - 如需绘制曲面则使用**glMap2d()**

```
glEnable(GL_MAP1_VERTEX_3); /*activatemapping*/

glMap1d(GL_MAP1_VERTEX_3,
        0.0, /* first u, i.e., left end */
        1.0, /* last u, i.e., right end */
        3, /* a 3-value gap */
        n, /* n+1 control points */
        &control_points[0][0]);

glBegin(GL_LINE_STRIP); /* draw a line strip */
for(i = 0; i <= 100; i++){ /*100segments*/
   glEvalCoord1d((GLdouble) (i*1.0)/100); /* gen points */
}
glEnd(); /* end of line strip */ }
```

曲线与曲面建模提纲

- 引言
- 贝塞尔曲线及曲面
- B样条曲线及曲面

B样条曲线及曲面

B样条曲线（B-Spline curve）

- 贝塞尔曲线每个控制点以固定方式影响整条曲线的形状
  - 对于给定u无法调整每个控制点的权重
  - 不支持局部修改和编辑
  - 曲线拼接时，很难满足几何连续条件
- B样条曲线是贝塞尔曲线的一般化形式由以下参数定义
  - Degree： p
  - $m+1$个节点（knots）组成的节点数组：$\{u_0, u_1, \cdots, u_m\}$
  - $n+1$个控制点： $\boldsymbol{P}_0, \boldsymbol{P}_1, \cdots, \boldsymbol{P}_n$
- 其中，n, m, p需满足fundamental identity: $m = n + p + 1$
- 半开区间$[u_i, u_{i+1}]$称为第i个knot span

B样条曲线及曲面

B样条曲线（B-Spline curve）

- 由$m+1$个节点组成的节点数组$\{u_0, u_1, \cdots, u_m\}$需满足
  - $0 = u_0 \leq u_1 \leq \cdots \leq u_m$（数组中$u_i$按非递减顺序排列）
  - u_i的值可能重复出现
  - 对于重复出现k次的节点u_i，我们称其重复度（multiplicity）为k
  - 如果一个节点的重复度为1，则称其为简单节点（simple knot）；否则，称其为重复节点（multiple knot）
  - 如果节点间等间距，则称其为均匀的，否则，称其为非均匀的

B样条曲线（B-Spline curve）

– B样条曲线的表达式为

$$\boldsymbol{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{P}_i$$

- 其中，$N_{i,p}(u)$为B样条曲线的第i个degree为p的 基函数

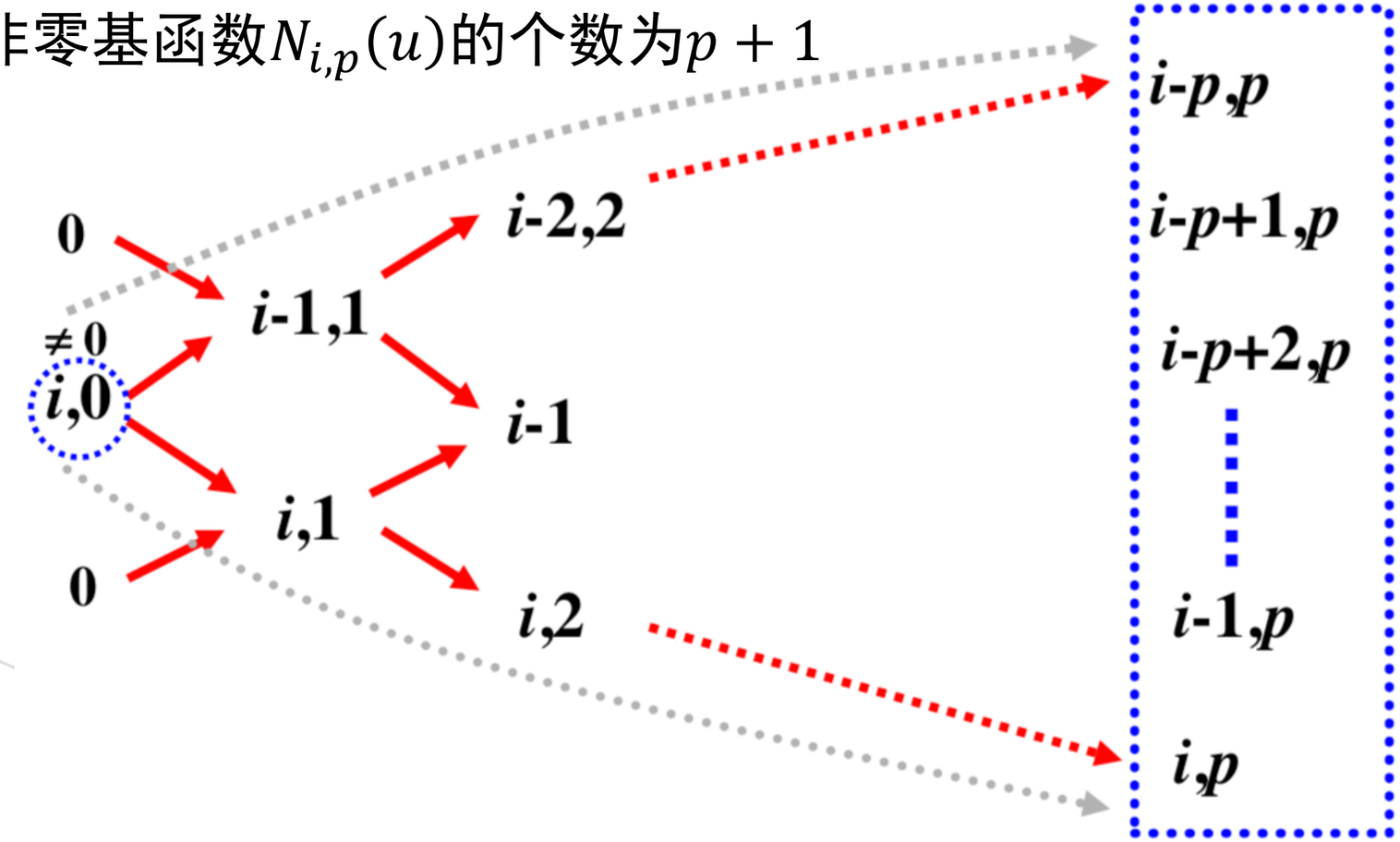
$$N_{i,0}(u) = \begin{cases} 1, & u_i \le u < u_{i+1} \\ 0, & otherwise \end{cases}$$

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

- 对于任意u，有$N_{i,p}(u) \ge 0$
- 对于给定p，所有$N_{i,p}(u)$之和为1（partition of unity）
- $N_{i,p}(u)$为以u为参数的p次多项式

B样条曲线及曲面

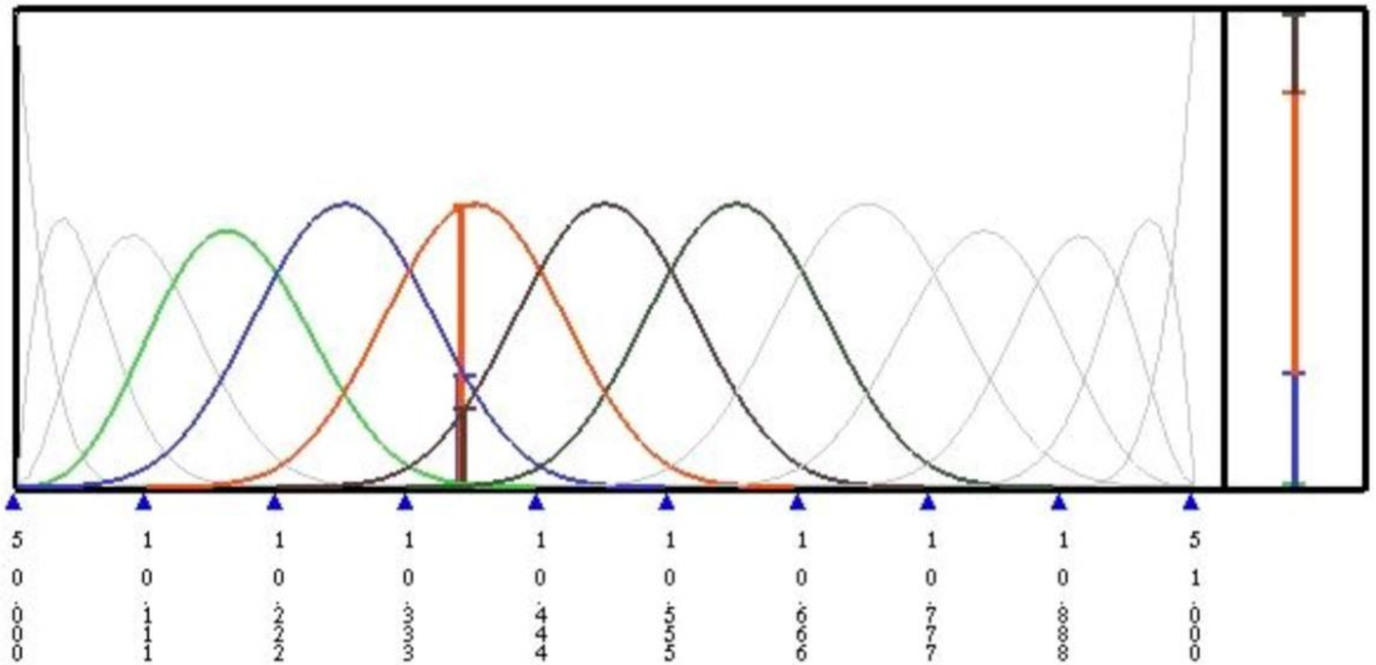
- B样条曲线（B-Spline curve）
  - $N_{i,p}(u)$的计算可描述为
    - 由图可见，非零基函数$N_{i,p}(u)$的个数为$p+1$



B样条曲线及曲面

B样条曲线（B-Spline curve）

- 图中显示的是当$u \in [0.333, 0.444)$时，B样条曲线的所有非零基函数
- 该B样条曲线的degree $p = 4$，由13个控制点（$n = 12$），18个节点（$m = 17$）所定义
  - 因此，其具有5个非零基函数



B样条曲线及曲面

- B样条曲线的strong convex hull property
  - 当$u \in [u_i, u_{i+1})$时，只有p个非零基函数：

$$N_{i,p}(u), N_{i-1,p}(u), \cdots, N_{i-p,p}(u)$$

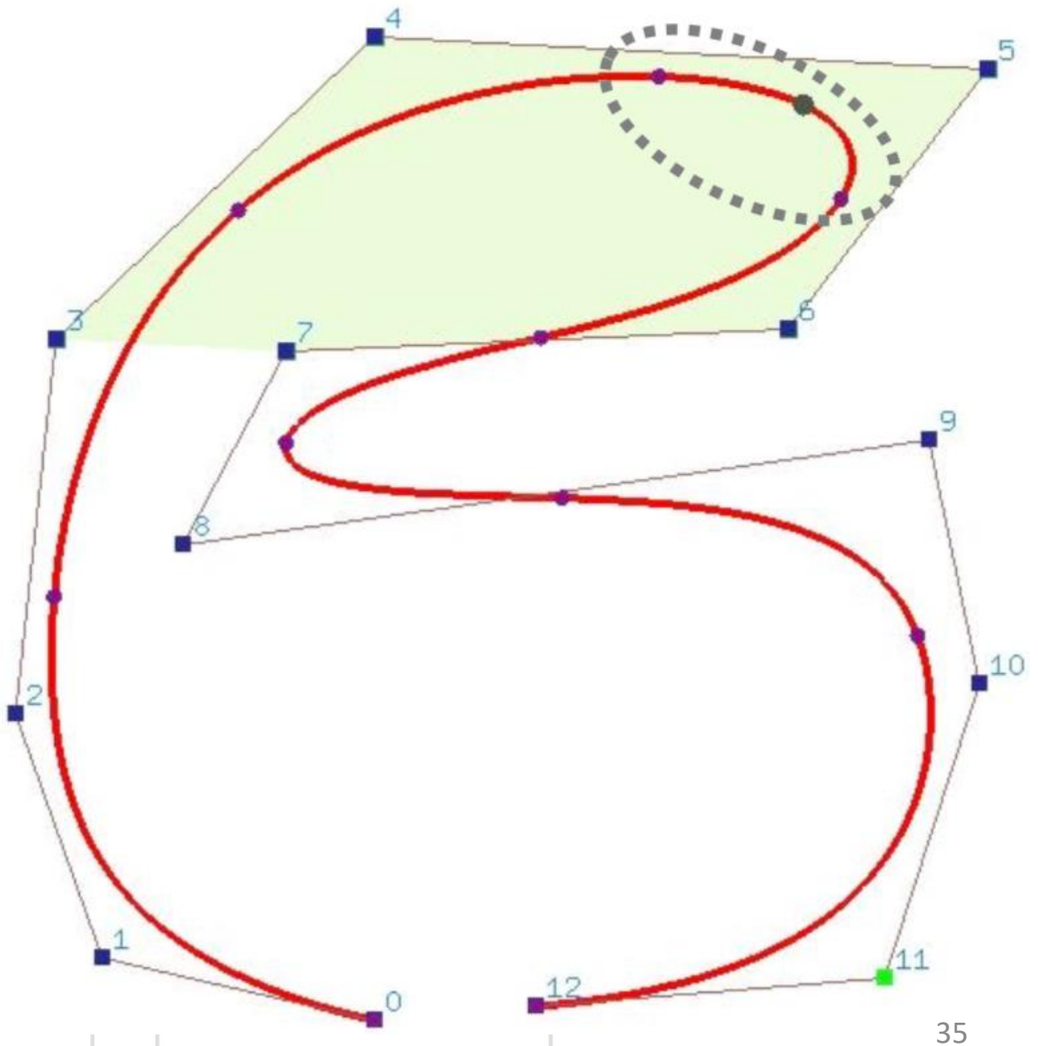
  - 因此，曲线$\boldsymbol{C}(u)$可写作

$$\boldsymbol{C}(u) = N_{i,p}(u)P_i + N_{i-1,p}(u)P_{i-1} + \cdots + N_{i-p,p}(u)P_{i-p}$$

  - 由于非零基函数$N_{i,p}(u)$之和为1（partition of unity），$\boldsymbol{C}(u)$位于P_i，$P_{i-1}, \cdots$，P_{i-p}所围的凸包中
    - Strong表示曲线段位于一个更小的凸包中

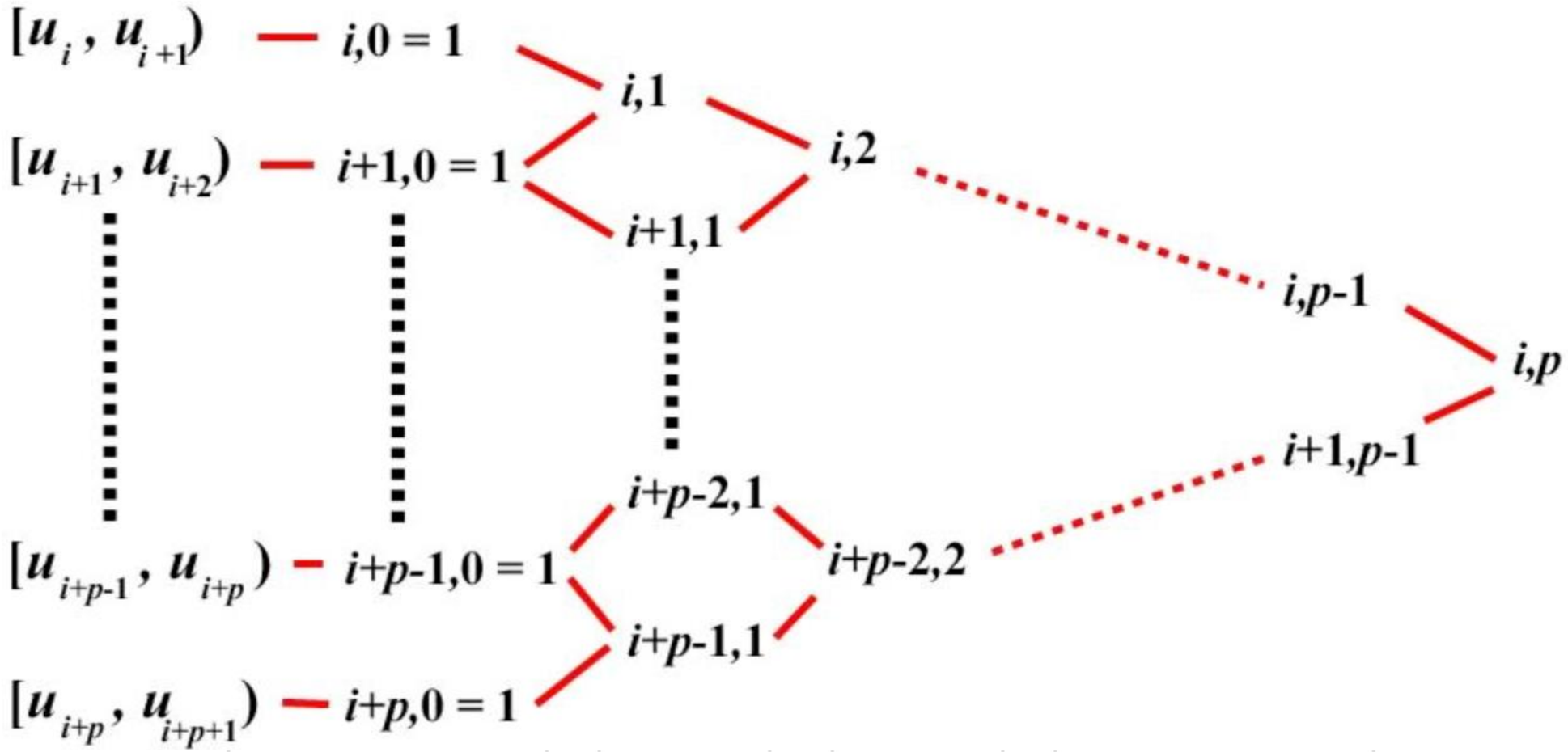
B样条曲线及曲面

- B样条曲线的strong convex hull property
  - 右图B样条曲线参数为
    - degree $p = 4$
    - 13个控制点（$n = 12$）
    - 18个节点（$m = 17$）
  - 当$u \in [u_7, u_8)$时
  - 非零基函数只有$N_{3,4}(u), N_{4,4}(u), N_{5,4}(u), N_{6,4}(u), N_{7,4}(u)$
  - 因此，$\boldsymbol{C}(u)$位于P_3, P_4, P_5, P_6, P_7所围的凸包中



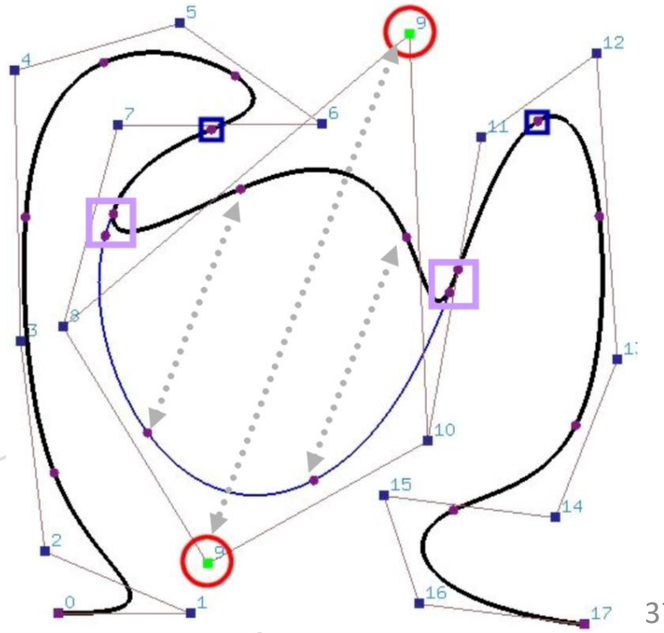
B样条曲线及曲面

- B样条曲线的local support property
  - 同样地， 基函数$N_{i,p}(u)$只在曲线段$[u_i, u_{i+p+1})$上为非零



B样条曲线的local support property

- 控制点P_i只对局部曲线段$[u_i, u_{i+p+1})$产生影响！
  - 由于P_i的系数$N_{i,p}(u)$只在$[u_i, u_{i+p+1})$上非零
  - 当P_i发生变化时，其只影响曲线段$[u_i, u_{i+p+1})$的形状
  - 而对于$[0, u_i)$及$[u_{i+p+1}, 1]$，由于$N_{i,p}(u) = 0$，即使移动P_i，曲线的形状也不受影响
  - 因此，改变P_i对曲线形状的影响是局部的！
- 对右图中所示$p = 4$的B样条曲线
- 改变控制点P_9只影响$[u_9, u_{9+4+1}) = [u_9, u_{14})$



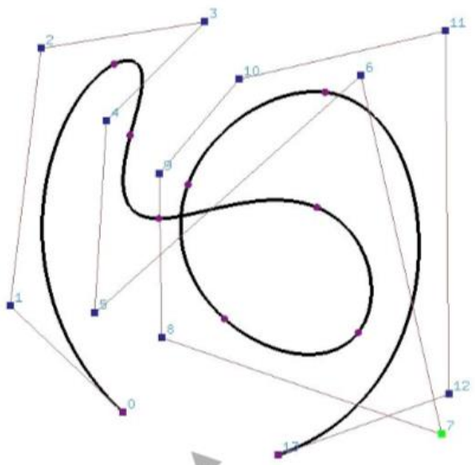
B样条曲线及曲面

B样条曲线的类型

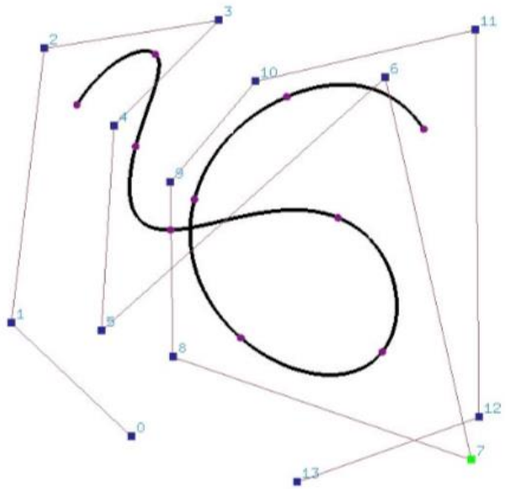
– 当degree为p的B样条曲线的前$p+1$个节点值为0，且后$p+1$个节点值为1时，曲线为clamped curve（曲线经过头尾两个控制点，且与第一段及最后一段控制折线相切）

– 否则，曲线为open curve（不一定经过首尾两个控制点或与控制折线相切）

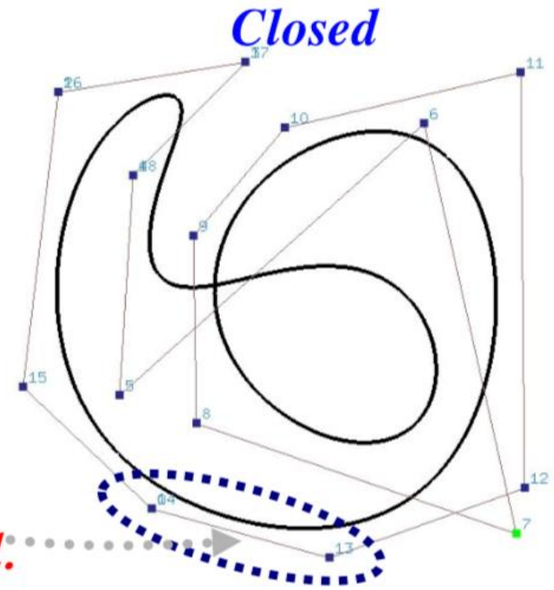
– Closed curve的构造方式更为复杂



Clamped



Open



Closed

Compare the open and closed version and you will see they are almost identical.

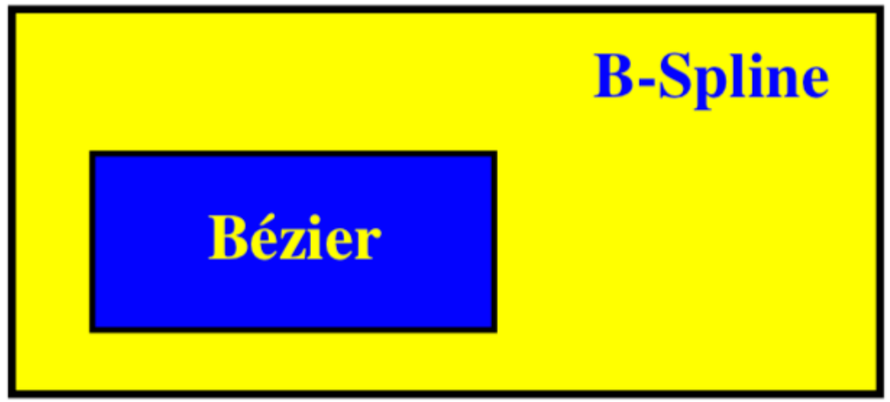
B样条曲线与贝塞尔曲线

– 没有内部节点的clamped B样条曲线为贝塞尔曲线

- 没有内部节点（internal knots）意味着曲线只有$2 \times (p+1)$个节点
- 前$p+1$个节点值为0，后$p+1$个节点值为1， $p+1$个控制点

– 在此情况下，对于任意i，都有$N_{i,p}(u) = B_{p,i}(u)$

- B样条曲线的系数缩减为贝塞尔曲线的系数！



B样条曲线及曲面

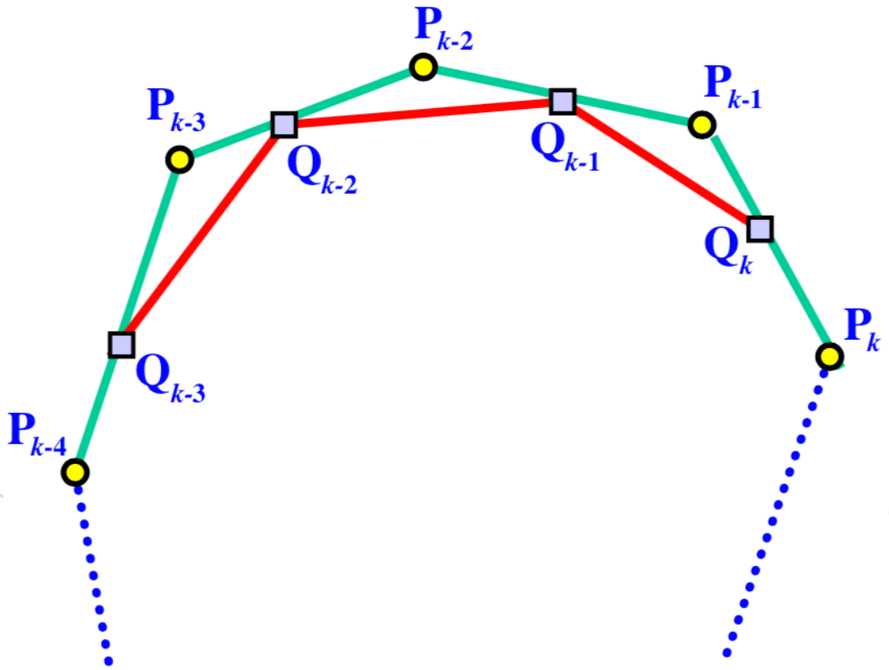
插入节点

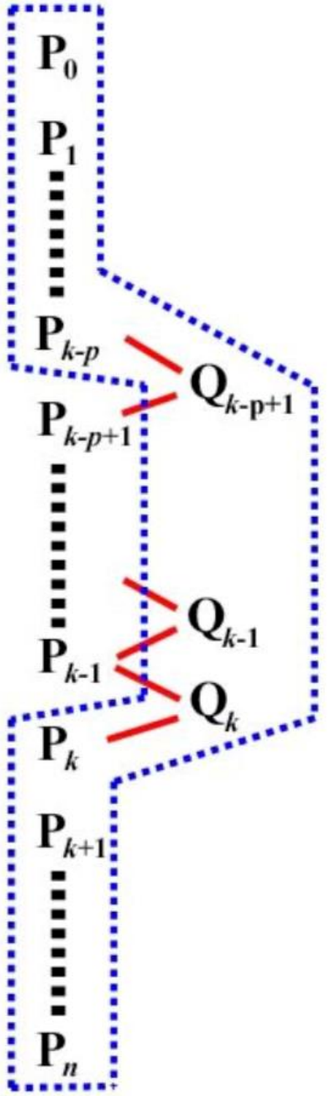
- 在节点数组中增加一个新的节点，而不改变曲线形状
  - 由于$m = n + p + 1$，当m增加1时，n也需要相应增加
  - 需要将源控制点中的$p - 1$个节点替换为新的p个节点（why？）
  - 插入节点是B样条曲线中最重要的研究之一
- 假设需要插入的新节点为$t \in (u_k, u_{k+1})$
  - 我们此后再讨论$t = u_k$（插入已有节点）的情况
  - 选取与t相关的$p + 1$个控制点为$P_k, P_{k-1}, P_{k-2}, \cdots, P_{k-p}$
  - 在其中每条线段$P_{j-1}P_j$上选取一个控制点Q_j
  - 将$P_{k-1}, P_{k-2}, \cdots, P_{k-p+1}$替换为$Q_k, Q_{k-1}, \cdots, Q_{k-p+1}$
    - 比照贝塞尔曲线的degree elevation

插入节点

– 假设要在$p = 4$的B样条曲线上插入$t \in (u_k, u_{k+1})$

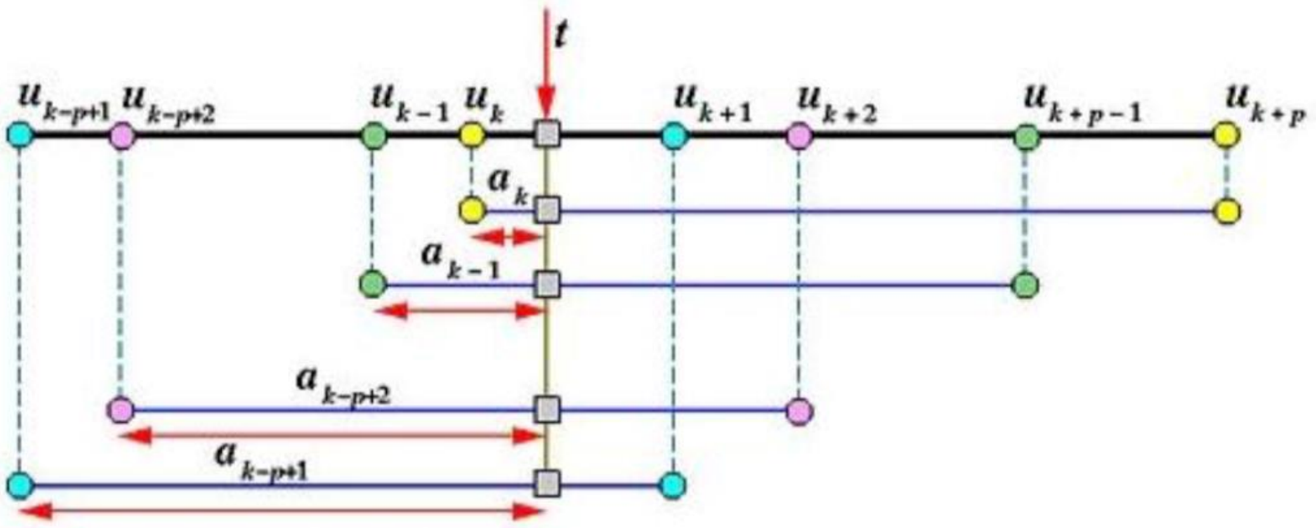
- 涉及的控制点为$P_k, P_{k-1}, P_{k-2}, P_{k-3}, P_{k-4}$
- 从这五个点组成的折线上计算新的控制点$Q_k, Q_{k-1}, Q_{k-2}, Q_{k-3}$
- 曲线新的控制点为$P_0, \cdots, P_{k-4}, Q_{k-3}, Q_{k-2}, Q_{k-1}, Q_k, P_k, \cdots, P_n$





B样条曲线及曲面

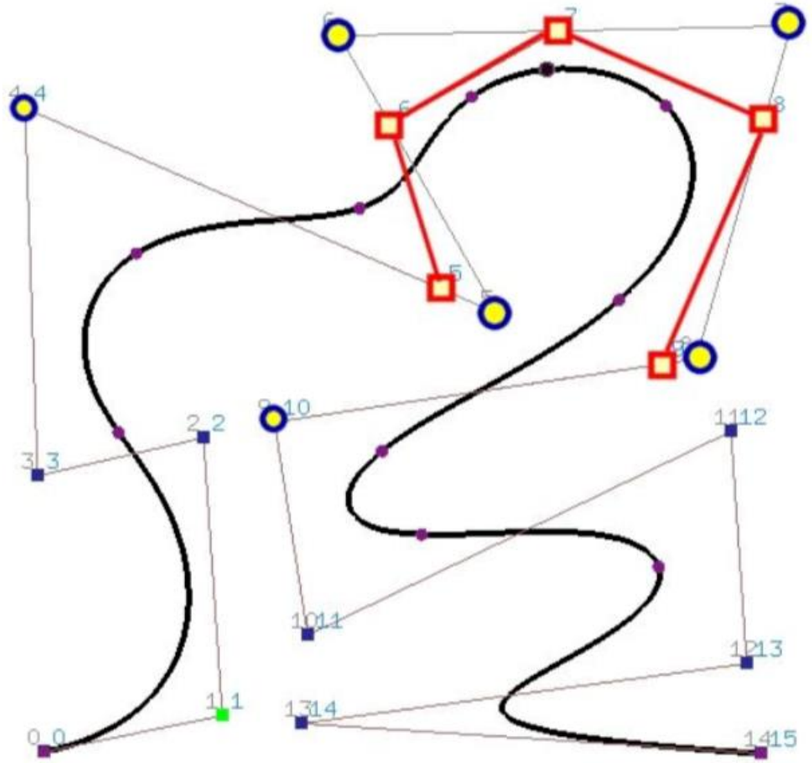
- 插入节点
  - 在线段$P_{i-1}P_i$上使用线性插值计算Q_i
    - $Q_i = (1-\alpha_i)P_{i-1} + \alpha_i P_i$
    - 其中，$\alpha_i = \frac{t-u_i}{u_{i+p}-u_i}$

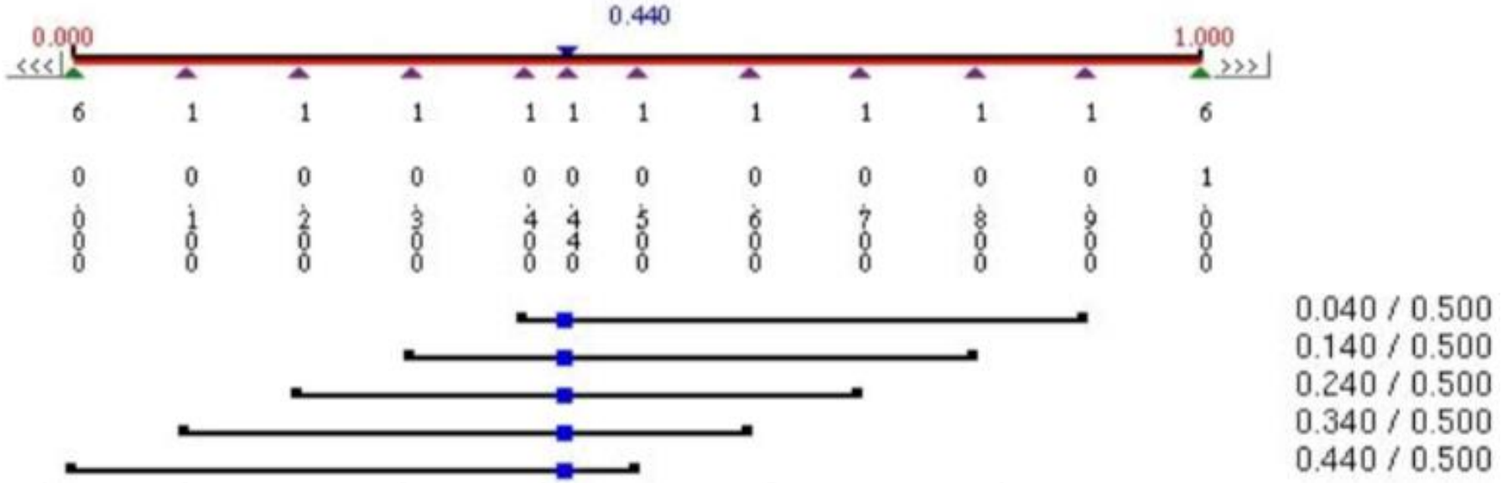


B样条曲线及曲面

插入节点

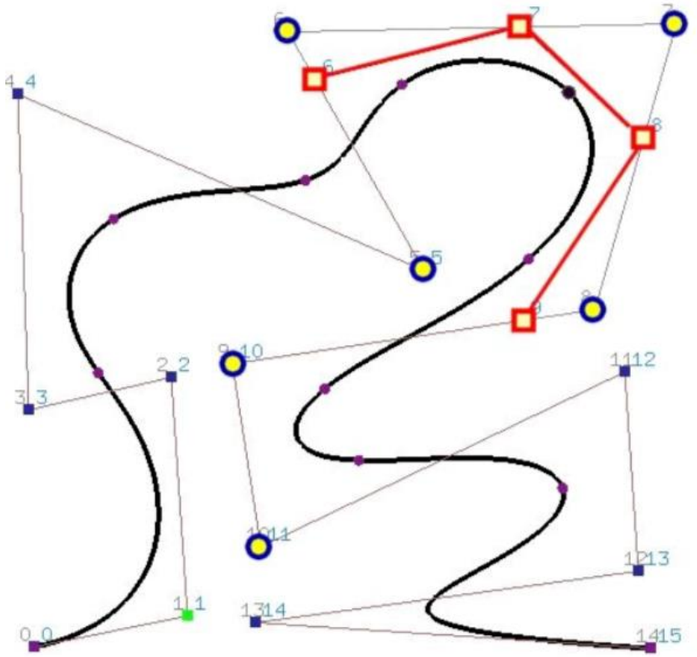
- 在$n = 14, p = 5$的B样条曲面上插入$t = 0.44 \in (0.4,0.5) = (u_9, u_{10})$
- 涉及的节点为$P_9, P_8, P_7, P_6, P_5, P_4$





B样条曲线及曲面

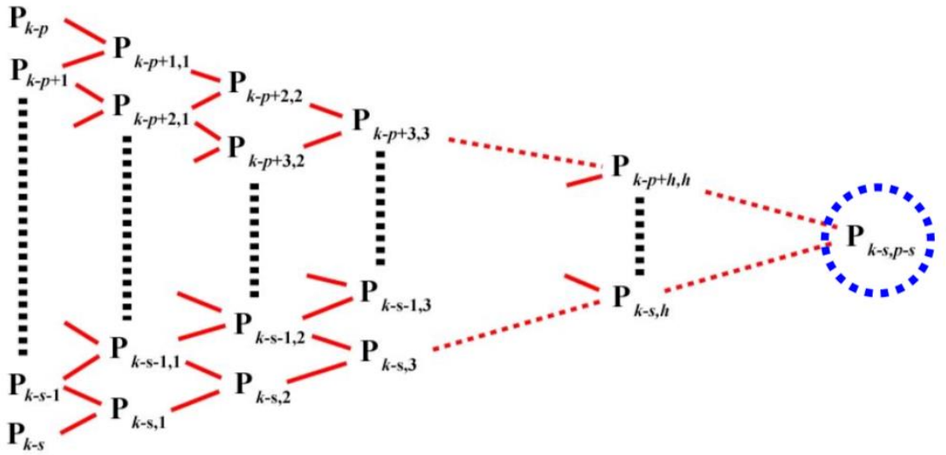
- 插入已有的简单节点（simple knot）
  - 插入$t = u_k \in [u_k, u_{k+1})$
    - 简单节点意味着u_k在节点数组中只出现过一次
  - 插入方式与此前完全相同，但计算更简单
    - $\alpha_k = \frac{t-u_k}{u_{k+p}-u_k} = 0$
    - $Q_k = (1-\alpha_k)P_{k-1} + \alpha_k P_k = P_{k-1}$
    - 因此，无需替换P_{k-1}
    - 曲线上新的控制点为$P_0, \cdots, P_{k-p}, Q_{k-p+1}, \cdots, Q_{k-1}, Q_k = P_{k-1}, P_k, \cdots, P_n$



  - 在$n = 14, p = 5$的B样条曲面上插入$t = 0.5 = u_{10}$
    - 涉及的控制点为$P_{10}, P_9, P_8, P_7, P_6, P_5$
    - 由于$t = u_{10}$，只需替换4个控制点：P_9, P_8, P_7, P_6
  - 插入已有的multiple knots与此类似，但计算量进一步减小

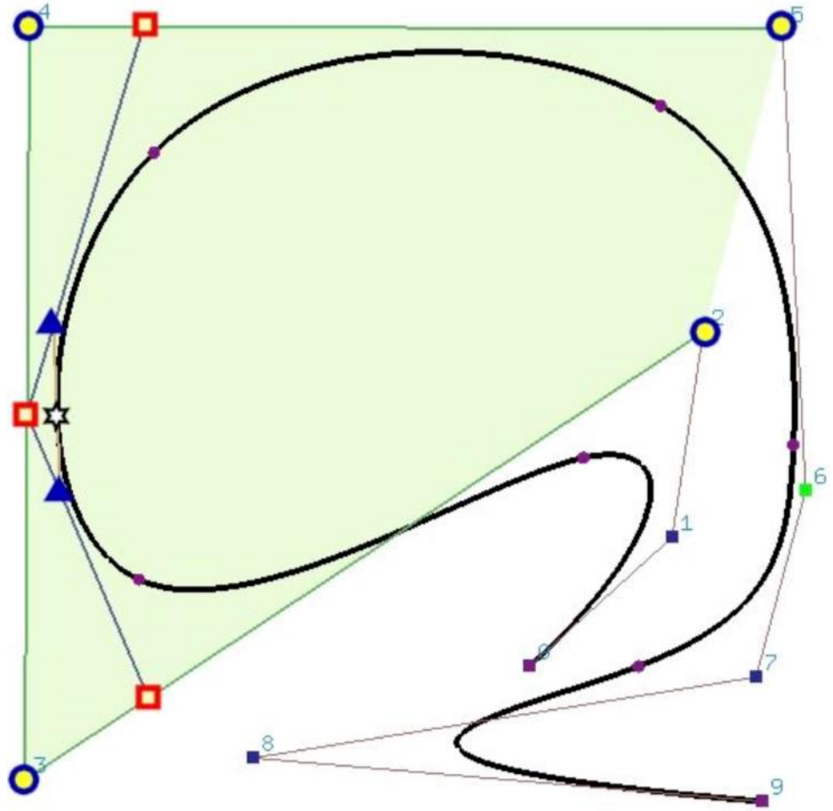
B样条曲线及曲面

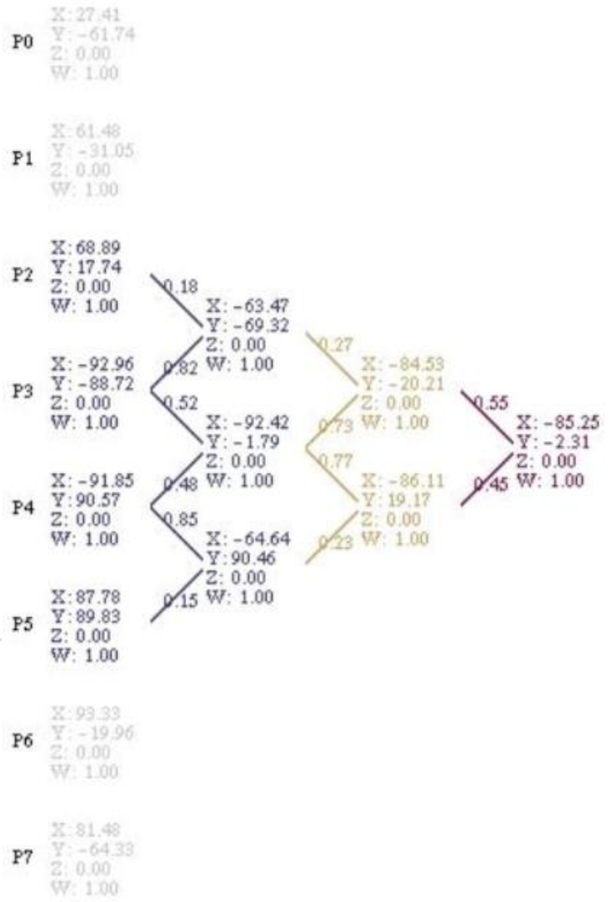
- B样条曲线计算：de Boor’s algorithm
  - 给定u，计算B样条曲线$\boldsymbol{C}(u)$上的对应点
  - De Boor’s algorithm使用插入算法计算$\boldsymbol{C}(u)$
    - 将u插入足够多次，直到u的multiplicity（出现次数）为p
    - 则最后计算的点为$\boldsymbol{C}(u)$
    - 如，$u \in (u_k, u_{k+1})$，则将u插入p次
    - 如，$u = u_k$，则插入$p - s$次，其中s为u_k的multicplicity



B样条曲线及曲面

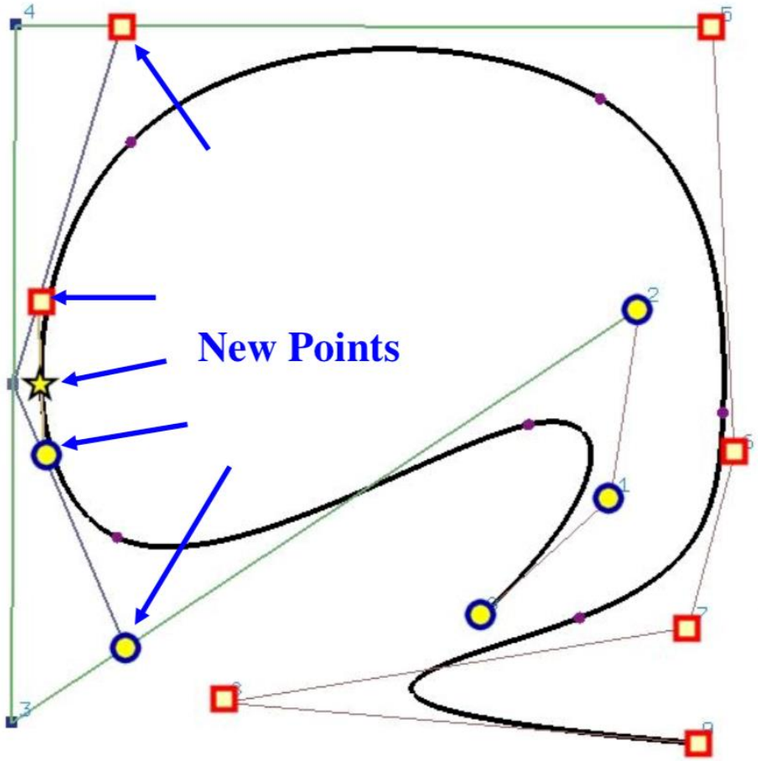
- B样条曲线计算：De Boor’s algorithm
  - 例如，在$p = 3$的B样条曲线上插入一个非节点的u
    - 三次计算可完成





B样条曲线细分

- 给定对于给定u，将degree p的B样条曲线细分为两条B样条曲线
- 使用de Boor's algorithm计算$\boldsymbol{C}(u)$，然后即可使用与贝塞尔曲线细分同样的方法得到细分后的曲线控制点

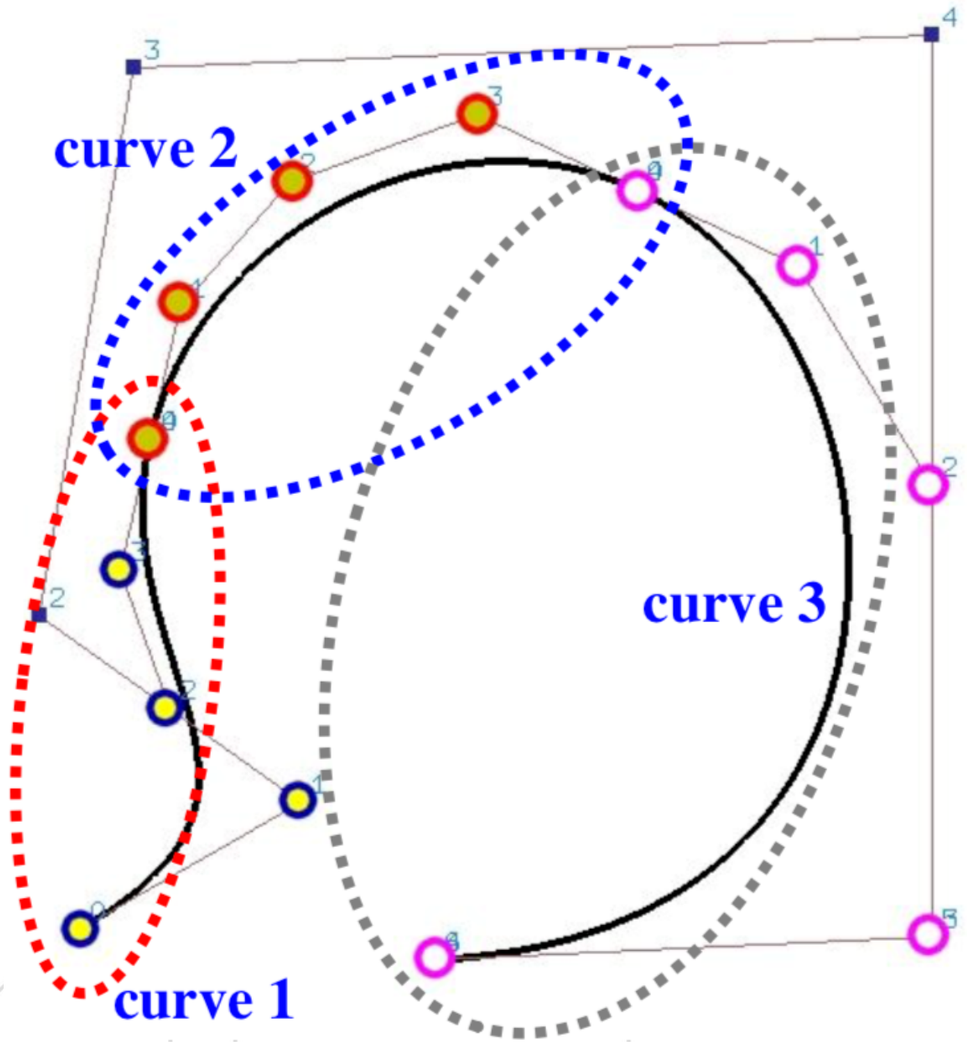


B样条曲线及曲面

B样条曲线细分

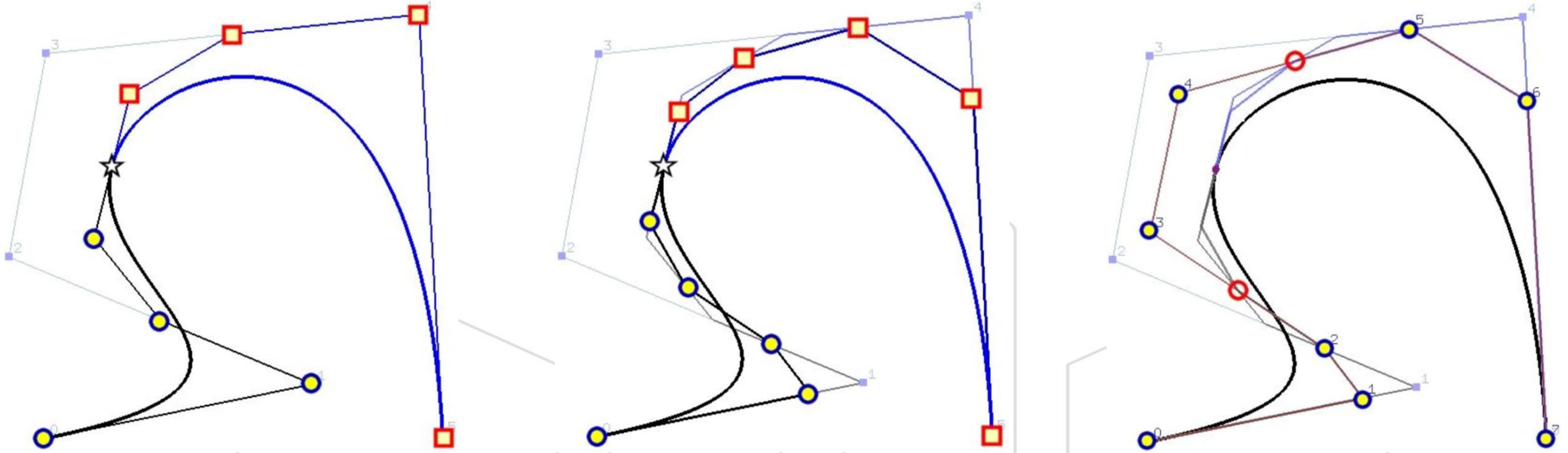
– 对于图中所示$p=4, n=6$的B样条曲线，其节点数组为

- $\{0,0,0,0,0,\frac{1}{3},\frac{2}{3},1,1,1,1,1\}$
  （$m=6+4+1=11$）
- 使用细分算法在$u=\frac{1}{3}$及$u=\frac{2}{3}$处将其细分为3段时，由于曲线没有$0,\frac{1}{3},\frac{2}{3},1$之外的节点(无内部节点)，细分后将得到3段贝塞尔曲线



B样条曲线的degree elevation

- 将degree p的B样条曲线的degree提高1
- 全过程较为复杂，在此我们只讨论大致思路
  - 1. 将B样条曲线在节点处拆分成多段贝塞尔曲线
  - 2. 对每段贝塞尔曲线进行degree elevation
  - 3. 去掉不必要的控制点与节点，从而得到1条degree $p+1$的B样条曲线



B样条曲线及曲面

B样条曲面

- B样条曲面$\boldsymbol{S}(u,v)$由以下参数定义
  - degree (p,q)
  - $(m+1)\times(n+1)$个控制点$P_{i,j}, i\in[0,m], j\in[0,n]$
  - $h+1$个节点组成的节点数组$U=\{u_0,u_1,\cdots,u_h\}$
  - k + 1个节点组成的节点数组$V=\{v_0,v_1,\cdots,v_k\}$
- 其表达式为

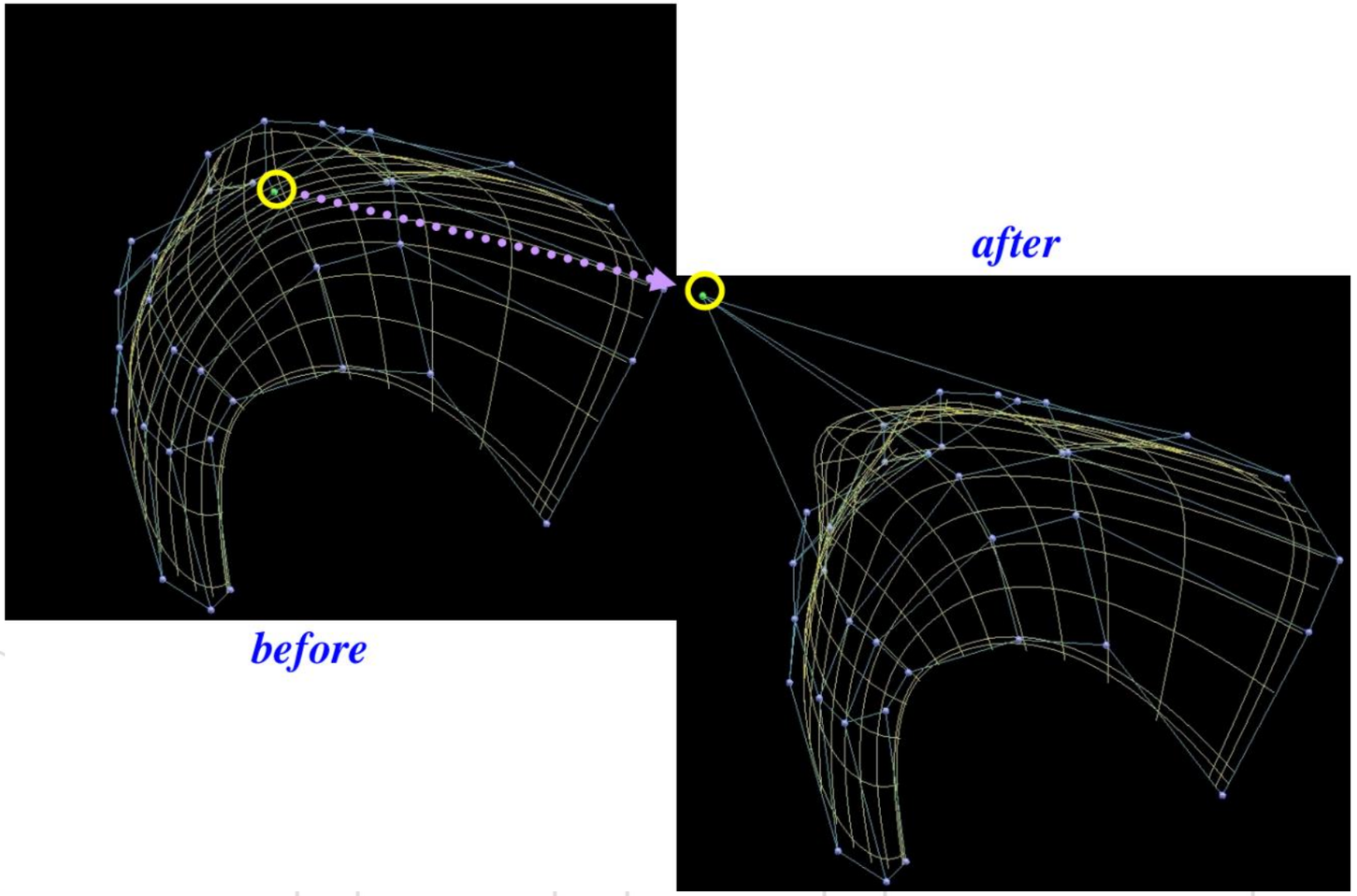
$$\boldsymbol{S}(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)N_{j,q}(v)\boldsymbol{P}_{i,j}$$

- 与贝塞尔曲面类似，B样条曲面可视为两次B样条曲线计算

B样条曲线及曲面

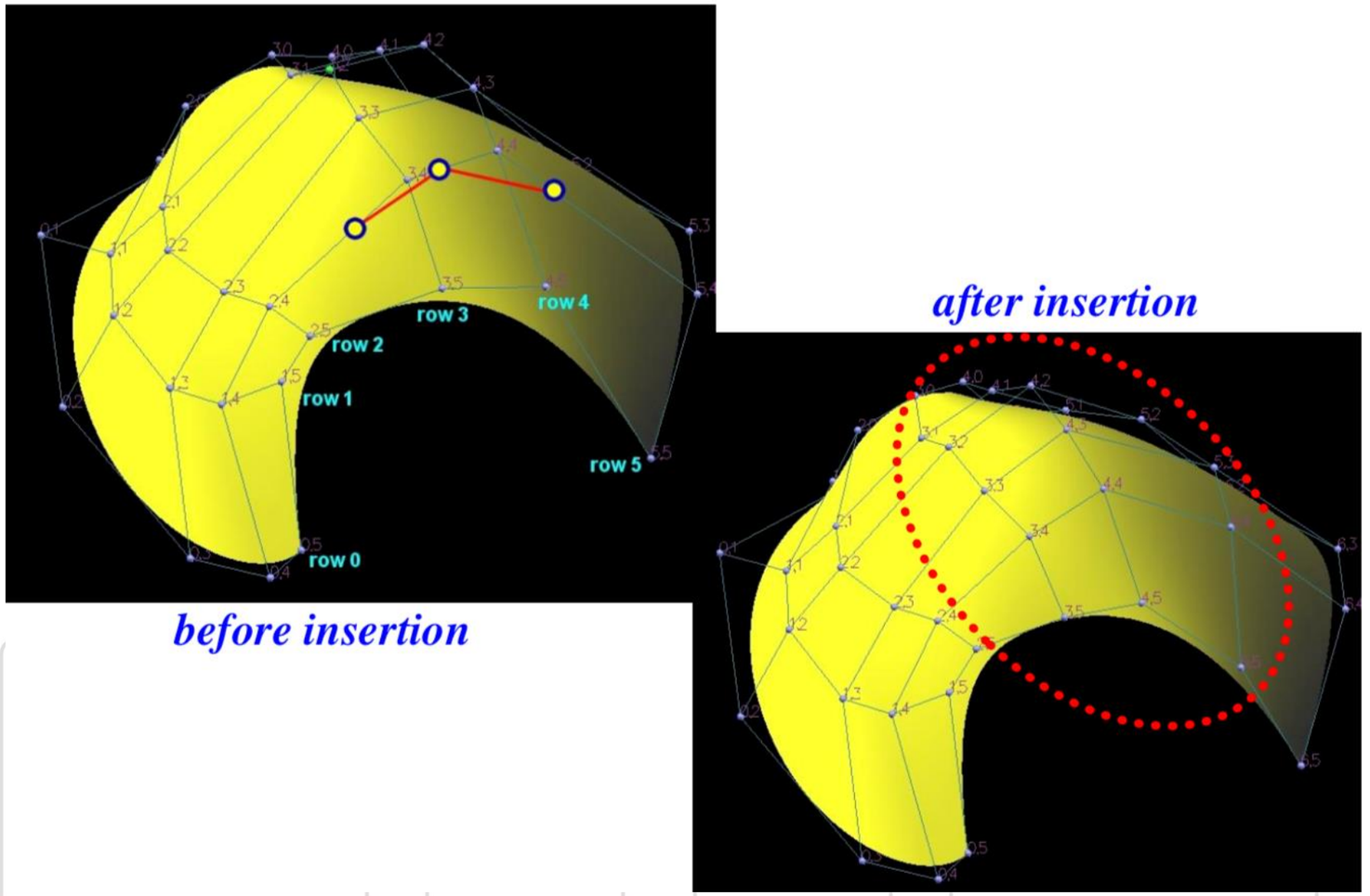
B样条曲面同样具有local modification property

– 更改控制点只对B样条曲线局部产生影响，因此，曲面也只有局部受影响

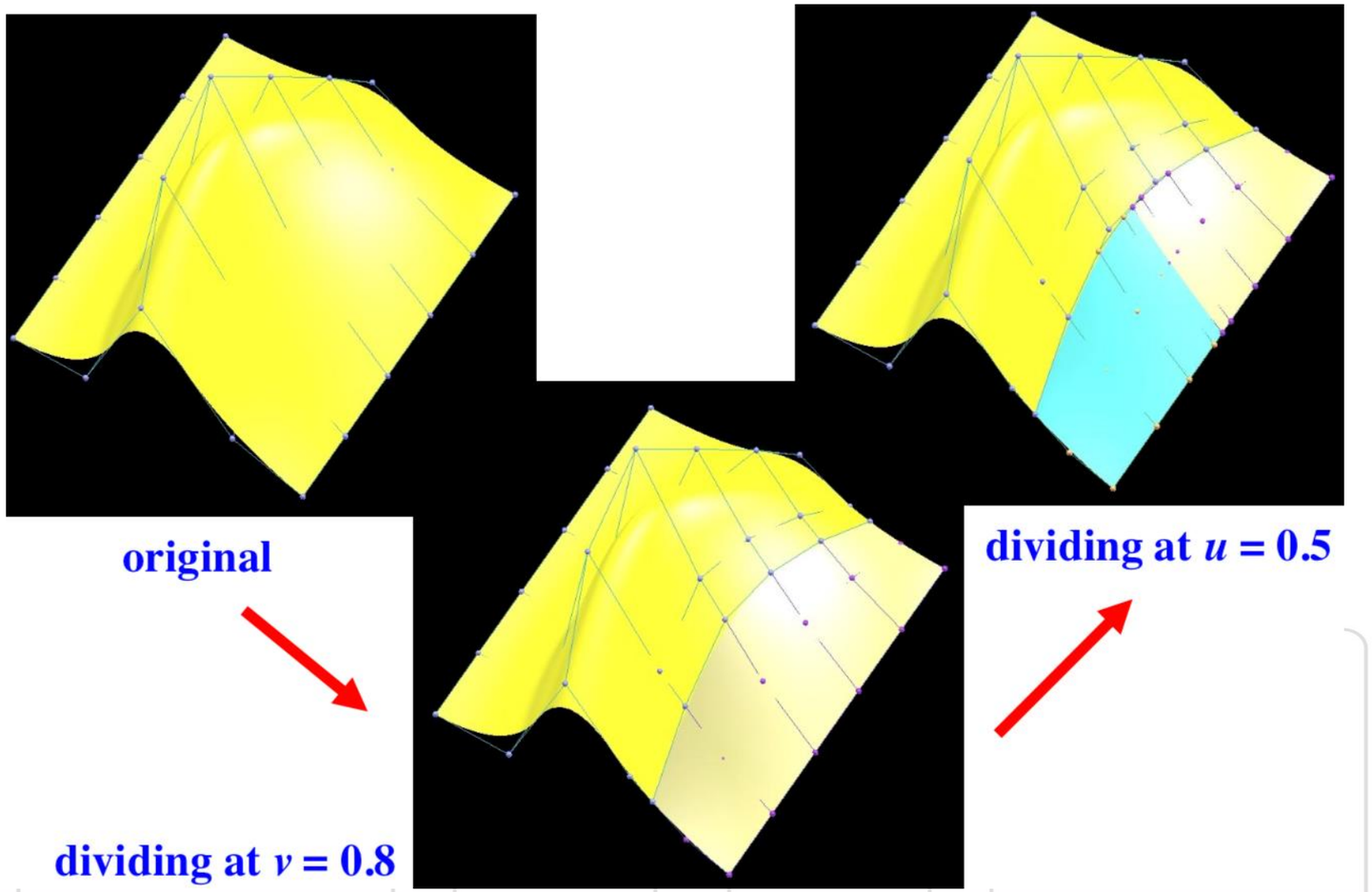


B样条曲线及曲面

- B样条曲面插入节点
  - 可对行、列所对应的节点数组及相应控制点分别进行操作

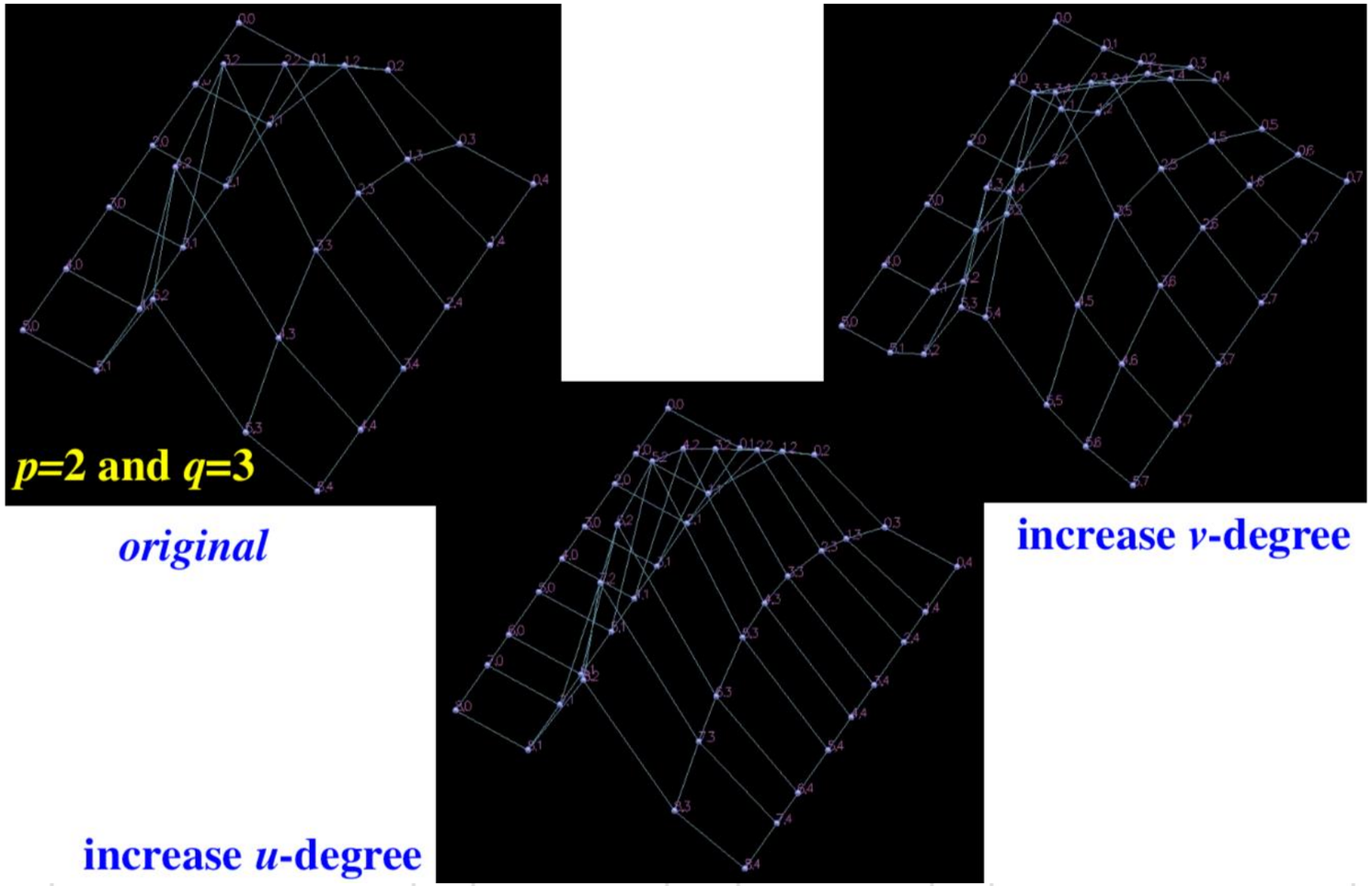


- B样条曲面细分
  - 可对行、列所对应的节点数组及相应控制点分别进行操作



B样条曲线及曲面

- B样条曲面的degree elevation
  - 可对行、列所对应的节点数组及相应控制点分别进行操作



B样条曲线及曲面

在OpenGL中使用evaluator绘制B样条曲线

```
GLUnurbsObj *theNurbs;                  /* declare a NURBS Obj*/
glEnable(GL_MAP1_VERTEX_3);             /* activate mapping */
gluBeginCurve(theNurbs);                /* start NURBS curve */
gluNurbsCurve(theNurbs,                 /* draw a NURBS curve */
              m,                        /* # of knots */
              &knot_v[0],               /* knot vector */
              3,                        /* u stride: 3 nos gap*/
              &points_v[0][0],          /* control points */
              p + 1,                    /* order = degree + 1 */
              GL_MAP1_VERTEX_3);        /* do it as here */
gluEndCurve(theNurbs);                  /* end of NURBS curve */
```

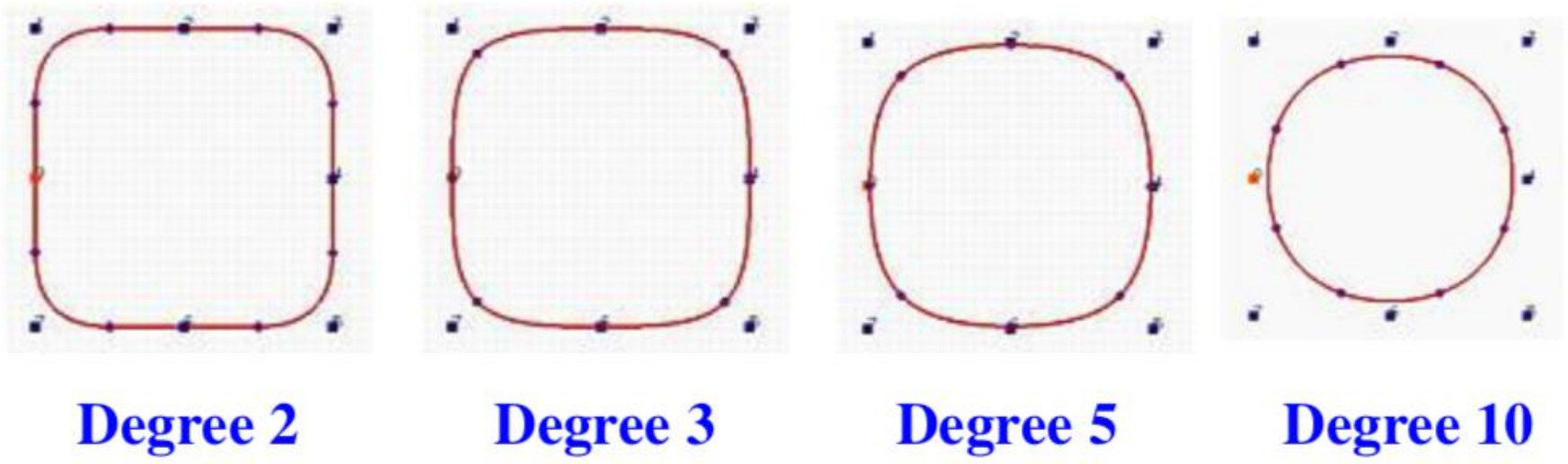
B样条曲线及曲面

在OpenGL中使用evaluator绘制B样条曲面

```
GLUnurbsObj *theNurbs;                  /* declare a NURBS Obj */
glEnable(GL_MAP2_VERTEX_3);             /* activate mapping */
gluBeginSurface(theNurbs);              /* start NURBS surface */
gluNurbsSurface(theNurbs,               /* draw a NURBS surface*/
                u_knots,                /* # of u knots */
                &knot_u[0],             /* U knot vector */
                v_knots,                /* # of v knots */
                &knot_v[0],             /* V knot vector */
                u_stride,               /* multiple of 3 floats*/
                v_stride,               /* 3 for (x,y,z) */
                &points_v[0][0][0],     /* ctrl points */
                u_order,                /* order = degree + 1 */
                v_order,                /* order = degree + 1 */
                GL_MAP2_VERTEX_3);      /* do it as here */
gluEndSurface(theNurbs);                /* end of NURBS surface*/
```

NURBS（Non-Uniform Rational B-Spline）

- 然而，B样条曲线无法表示圆，椭圆，抛物线，等
- 如下图所示，使用degree 2，3，5，10及8个控制点定义的B样条曲线，都无法准确地表示圆



B样条曲线及曲面

- NURBS（Non-Uniform Rational B-Spline）
  - 在B样条曲线原有参数之外，增加了控制点的权重数组
    - Degree： p
    - $m+1$个节点（knots）组成的节点数组：$\{u_0, u_1, \cdots, u_m\}$
    - $n+1$个控制点： $\boldsymbol{P}_0, \boldsymbol{P}_1, \cdots, \boldsymbol{P}_n$
    - 每个控制点$\boldsymbol{P}_i$同时带有一个权重w_i
  - NURBS曲线的表达式为

$$\boldsymbol{C}(u) = \frac{1}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \sum_{i=0}^{n} N_{i,p}(u) w_i \boldsymbol{P}_i$$

  - 其中， $N_{i,p}(u)$为B样条曲线的基函数
    - 显然，当所有w_i都为1时，NURBS为B样条曲线

NURBS（Non-Uniform Rational B-Spline）

– 从B样条曲线到NURBS

- 将B样条曲线上的控制点$\boldsymbol{P}_i = [x_i, y_i, z_i]^T$表示为齐次坐标

$$\boldsymbol{P}_i = [x_i, y_i, z_i, 1]^T$$

- 注意三维空间中的每个点对应的都是齐次坐标下的一条直线

$$\boldsymbol{P}_i^w = [w_i x_i, w_i y_i, w_i z_i, w_i]^T$$

- 因此，NURBS曲线表达式可写作

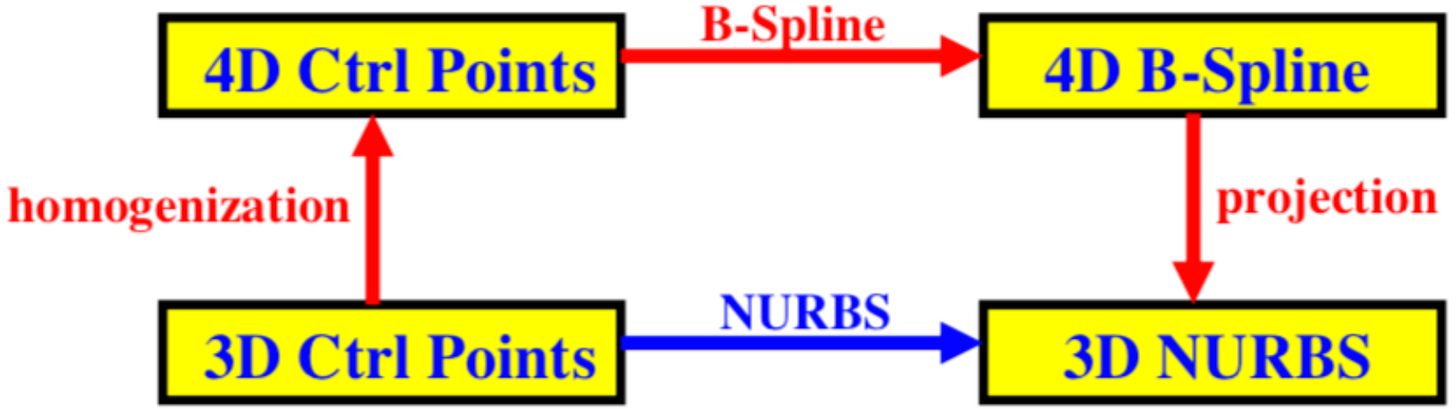
$$\boldsymbol{C}(u) = \frac{1}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \sum_{i=0}^{n} N_{i,p}(u) w_i \boldsymbol{P}_i$$

$$= \left[\sum_{i=0}^{n} N_{i,p}(u)(w_i x_i), \sum_{i=0}^{n} N_{i,p}(u)(w_i y_i), \sum_{i=0}^{n} N_{i,p}(u)(w_i z_i), \sum_{i=0}^{n} N_{i,p}(u) w_i \right]$$

$$= \sum_{i=0}^{n} N_{i,p}(u) \boldsymbol{P}_i^w$$

B样条曲线及曲面

- NURBS（Non-Uniform Rational B-Spline）
  - 因此，三维空间中的NURBS曲线可以看做是四维齐次坐标系下的B样条曲线在三维中的投影
  - B样条曲线中，与度量无关的性质仍然使用与NURBS曲线，如strong convex hull property及local modification property



B样条曲线及曲面

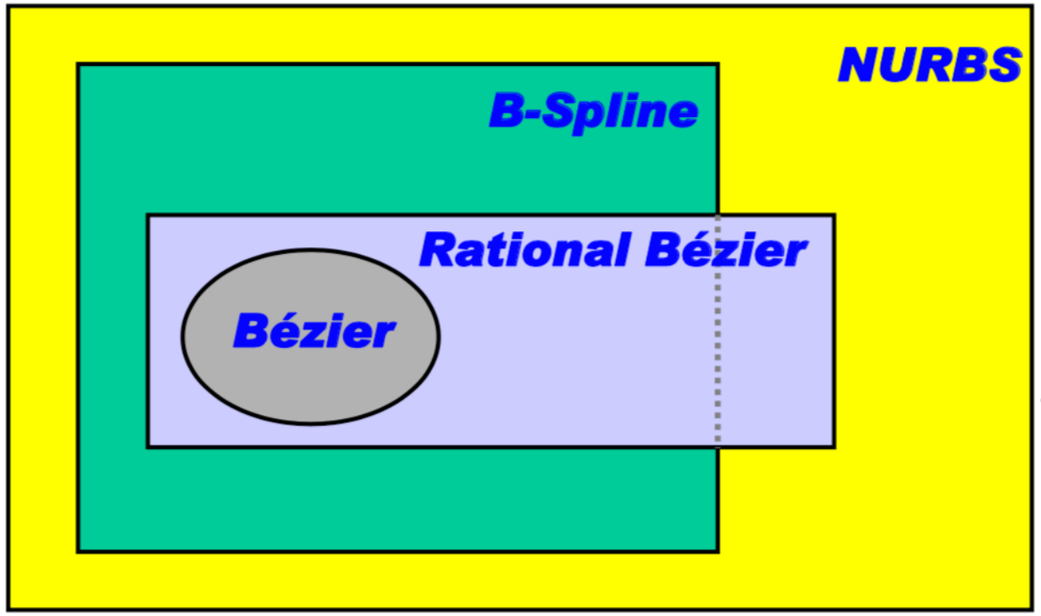
在OpenGL中使用evaluator绘制NURBS

```
GLUnurbsObj *theNurbs;                  /* declare a NURBS Obj*/
glEnable(GL_MAP1_VERTEX_4);             /* activate mapping */
gluBeginCurve(theNurbs);                /* start NURBS curve */
gluNurbsCurve(theNurbs,                 /* draw a NURBS curve */
              m,                        /* # of knots */
              &knot_v[0],               /* knot vector */
              4,                        /* u stride: 4 nos gap*/
              &points_v[0][0],          /* control points */
              p + 1,                    /* order = degree + 1 */
              GL_MAP1_VERTEX_4);        /* do it as here */
gluEndCurve(theNurbs);                  /* end of NURBS curve */
```

小结

贝塞尔，B样条，及NURBS曲线/曲面

- 都是基于控制点的插值
- 贝塞尔：对所有控制点进行插值
- B样条：依据节点数组及degree，对一定范围内控制点进行插值
- NURBS：在插值过程中赋予控制点额外的权重



Questions?