

Introdução à Computação Gráfica

Curvas

Claudio Esperança
Paulo Roma Cavalcanti

Modelagem Geométrica

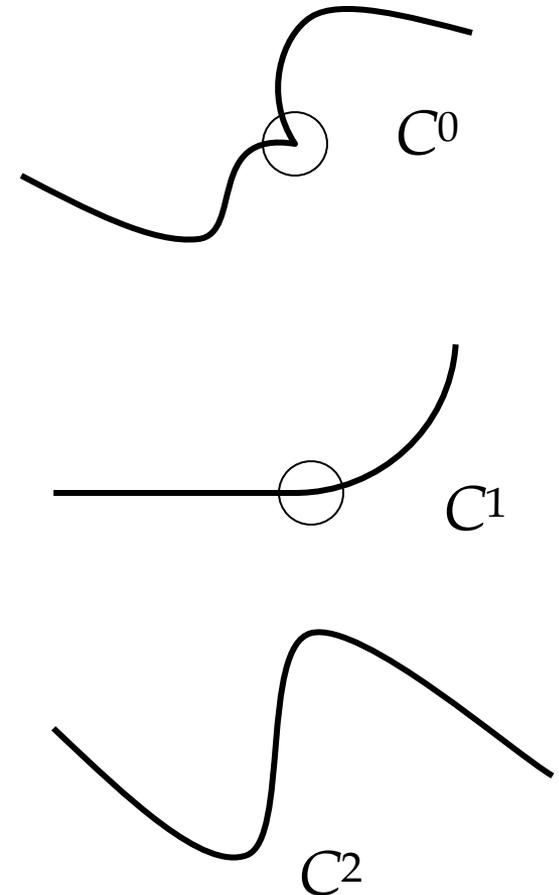
- Disciplina que visa obter representações algébricas para curvas e superfícies com determinado aspecto e/ou propriedades
- Até agora temos considerado quase que exclusivamente objetos geométricos compostos de segmentos de reta ou polígonos (curvas/superfícies lineares por parte)
 - ◆ Na maioria dos casos, são aproximações de curvas e superfícies algébricas
 - ◆ Mesmo quando só podemos desenhar segmentos de reta e polígonos, conhecer o objeto que estamos aproximando é fundamental

Curvas e Superfícies Paramétricas

- Normalmente, o resultado da modelagem é dado em forma paramétrica
 - ◆ Permite que a curva/superfície seja desenhada (aproximada) facilmente
 - ◆ Permite indicar que trechos da curva/superfície serão usados
 - ◆ Manipulação algébrica mais simples
- Curva em 3D é dada por
 - ◆ $C(t) = [C_x(t) \ C_y(t) \ C_z(t)]^T$
- Superfície em 3D é dada por
 - ◆ $S(u, v) = [S_x(u, v) \ S_y(u, v) \ S_z(u, v)]^T$

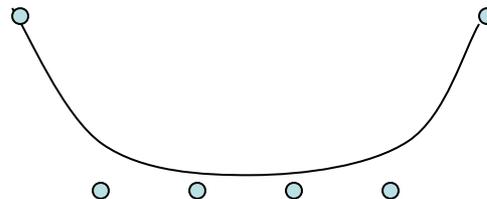
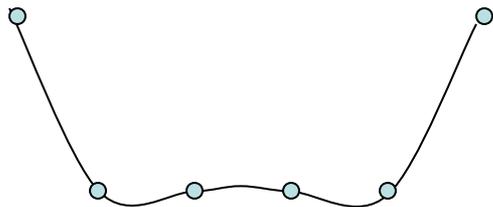
Continuidade

- Normalmente queremos curvas e superfícies “suaves”
- Critério de “suavidade” associado com critério de continuidade algébrica
 - ◆ Continuidade C^0 → funções paramétricas são contínuas, isto é, sem “pulos”
 - ◆ Continuidade C^1 → funções paramétricas têm primeiras derivadas contínuas, isto é, tangentes variam suavemente
 - ◆ Continuidade C^k → funções paramétricas têm k ’ésimas derivadas contínuas
- Alternativamente, G^k : continuidade geométrica
 - ◆ Independente de parametrização
 - ◆ Assumir curva parametrizada por comprimento de arco



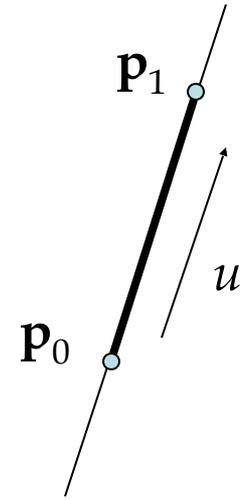
Interpolação x Aproximação

- É natural querermos modelar uma curva suave que passa por um conjunto de pontos dados
- Se a curva desejada é polinomial, chamamos tal curva de *interpolação polinomial lagrangeana*
- Entretanto, o resultado nem sempre é o esperado (oscilações)
- É mais comum querermos curvas que “passem perto” dos pontos dados, isto é, *aproximações*



Algoritmo de De Casteljau

- Suponha que queiramos aproximar uma curva polinomial entre dois pontos \mathbf{p}_0 e \mathbf{p}_1 dados
- A solução natural é um segmento de reta que passa por \mathbf{p}_0 e \mathbf{p}_1 cuja parametrização mais comum é
$$\mathbf{p}(u) = (1 - u) \mathbf{p}_0 + u \mathbf{p}_1$$
- Podemos pensar em $\mathbf{p}(u)$ como uma média ponderada entre \mathbf{p}_0 e \mathbf{p}_1
- Observe que os polinômios $(1 - u)$ e u somam 1 para qualquer valor de u
 - ◆ São chamadas de funções de mistura (*blending functions*)



Algoritmo de De Casteljau

- Para generalizar a idéia para três pontos \mathbf{p}_0 , \mathbf{p}_1 e \mathbf{p}_2 consideramos primeiramente os segmentos de reta $\mathbf{p}_0\mathbf{p}_1$ e $\mathbf{p}_1\mathbf{p}_2$

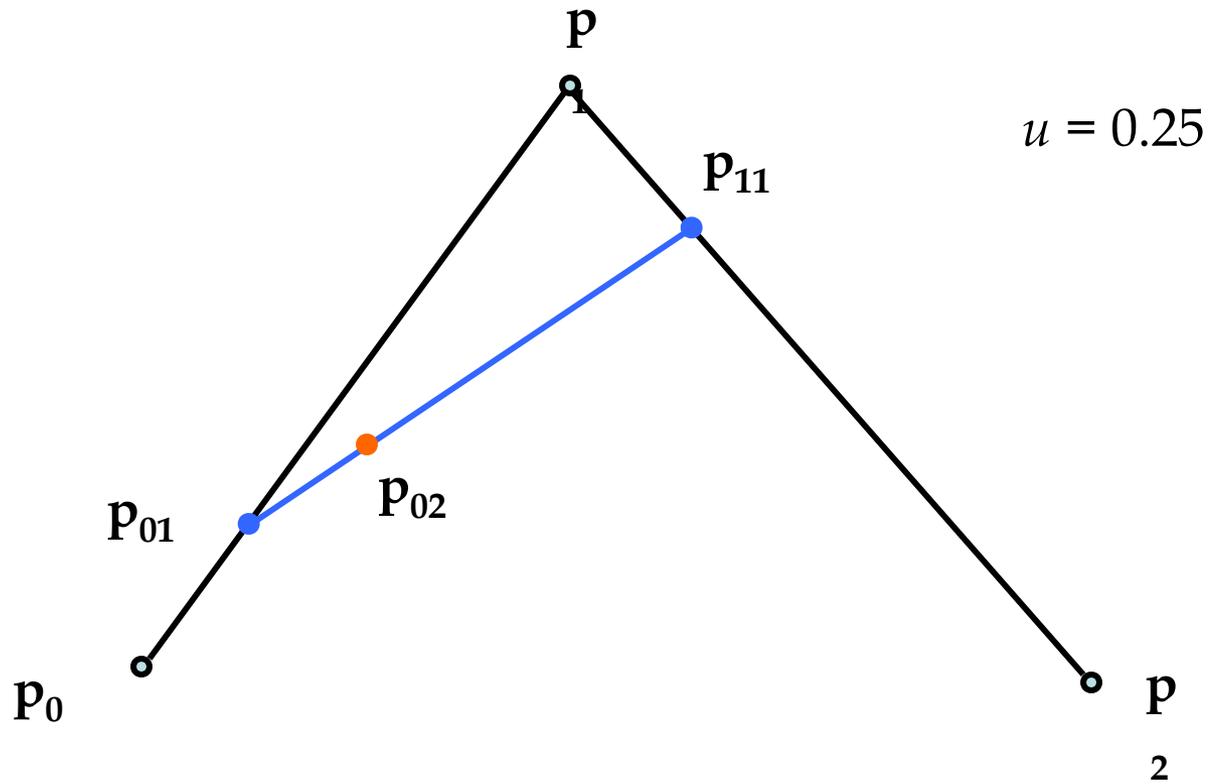
$$\mathbf{p}_{01}(u) = (1 - u) \mathbf{p}_0 + u \mathbf{p}_1$$

$$\mathbf{p}_{11}(u) = (1 - u) \mathbf{p}_1 + u \mathbf{p}_2$$

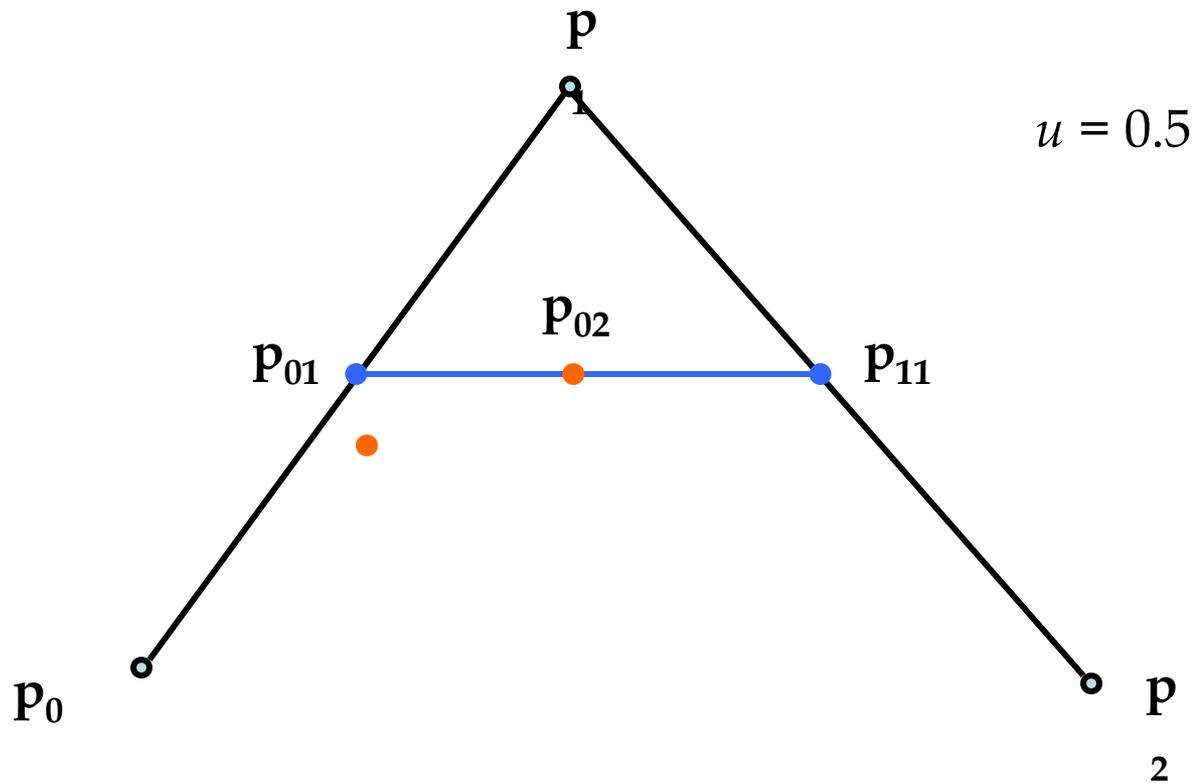
- Podemos agora realizar uma interpolação entre $\mathbf{p}_{01}(u)$ e $\mathbf{p}_{11}(u)$

$$\begin{aligned} \mathbf{p}_{02}(u) &= (1 - u) \mathbf{p}_{01}(u) + u \mathbf{p}_{11}(u) \\ &= (1 - u)^2 \mathbf{p}_0 + 2u(1 - u) \mathbf{p}_1 + u^2 \mathbf{p}_2 \end{aligned}$$

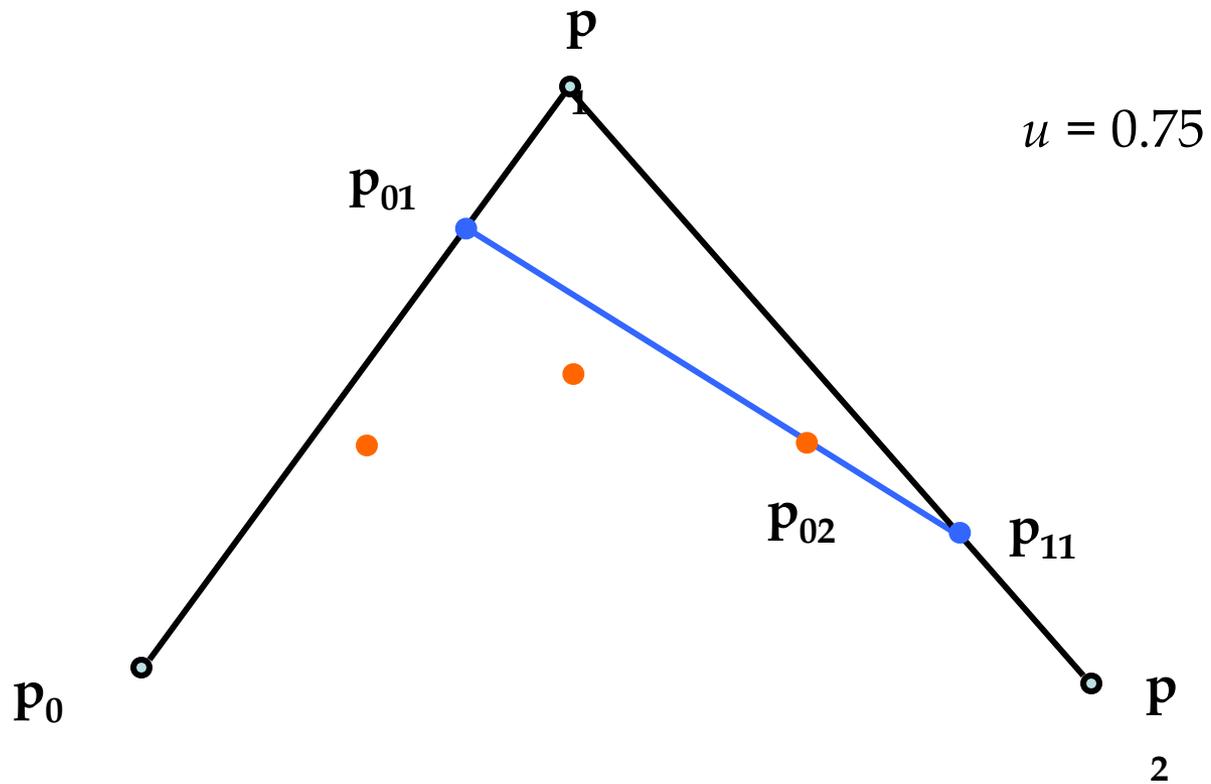
Algoritmo de De Casteljau



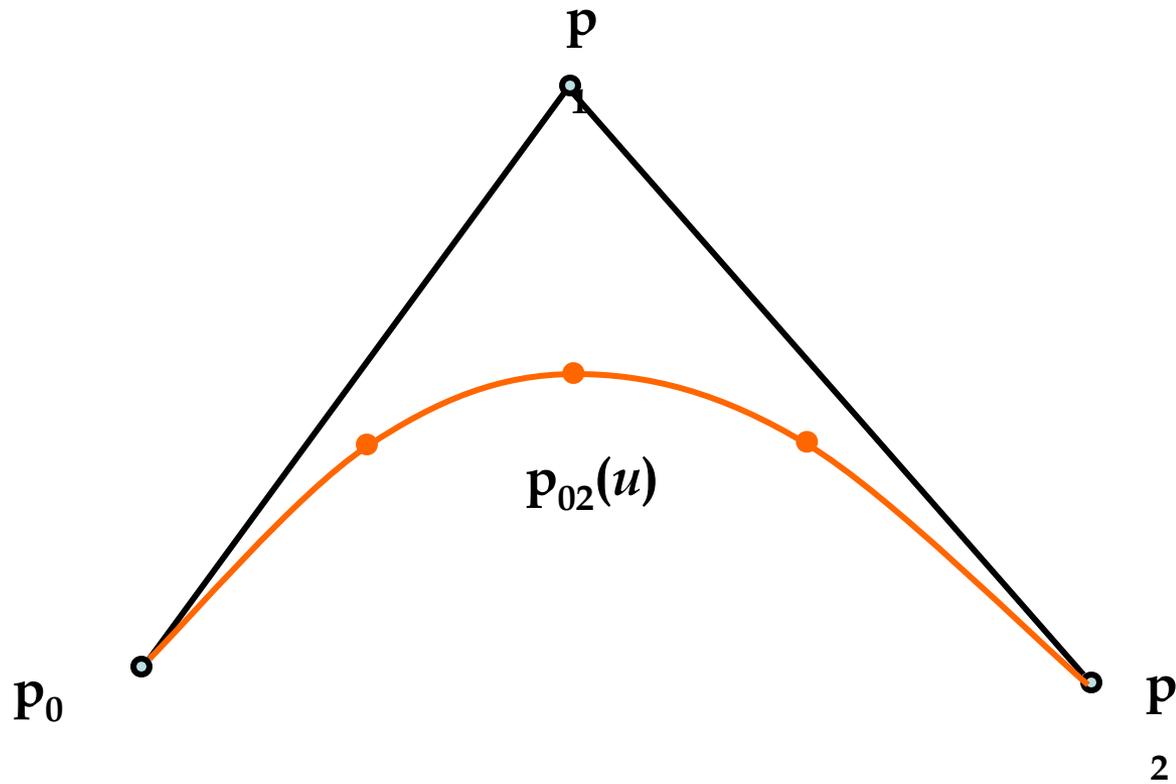
Algoritmo de De Casteljau



Algoritmo de De Casteljau



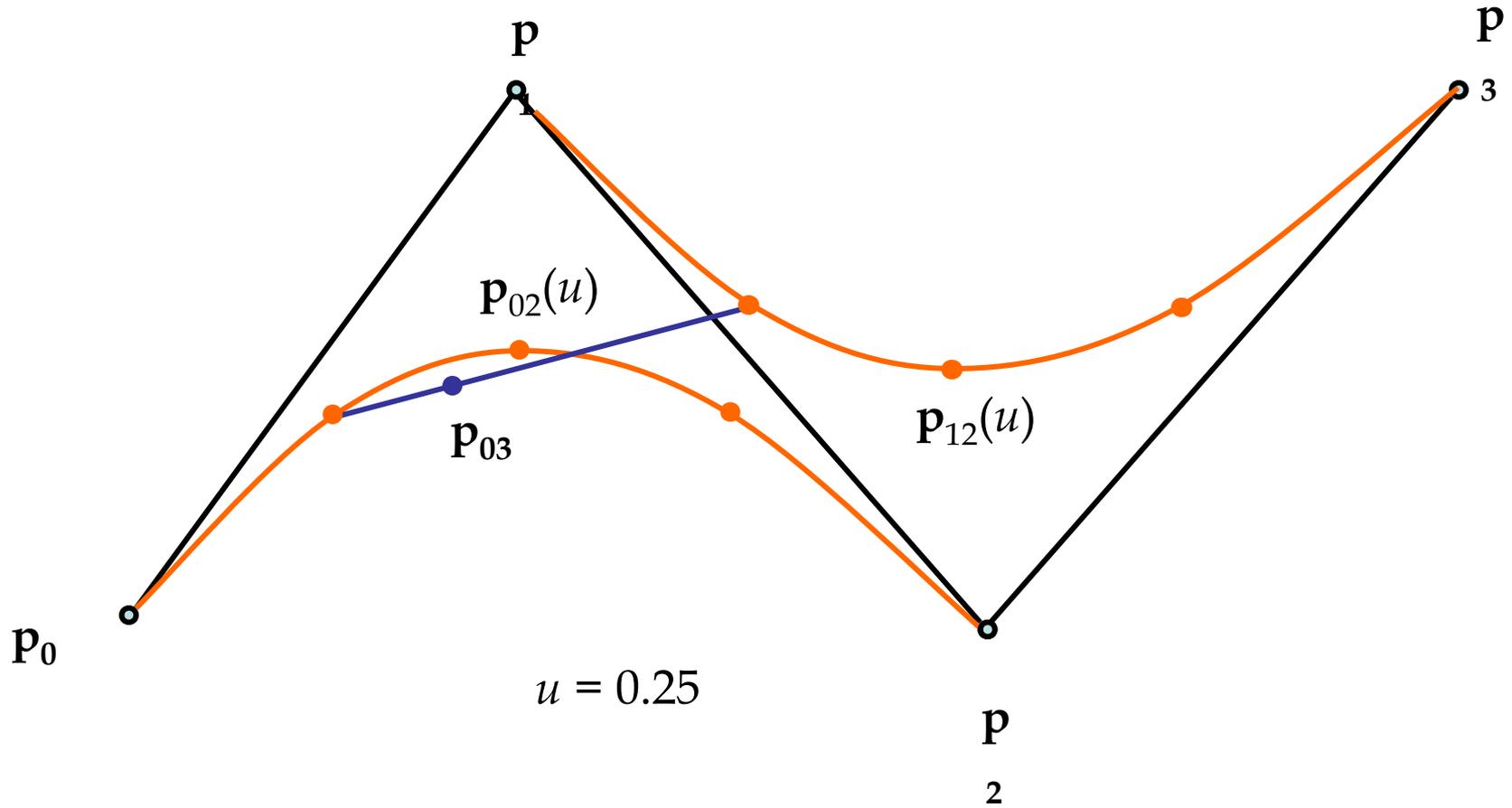
Algoritmo de De Casteljau



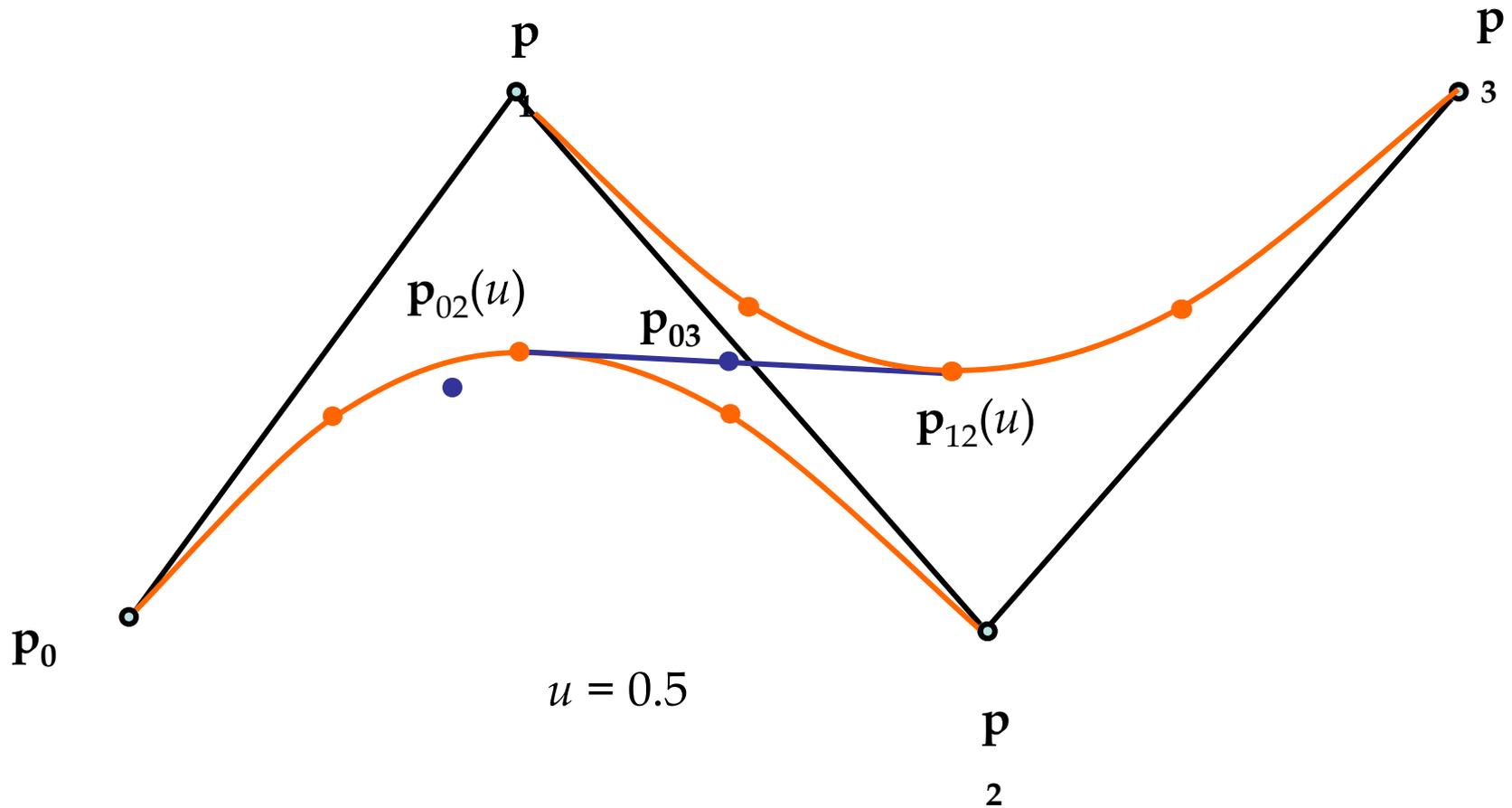
Algoritmo de De Casteljau

- A curva obtida pode ser entendida como a “mistura” dos pontos \mathbf{p}_0 , \mathbf{p}_1 e \mathbf{p}_2 por intermédio de três funções quadráticas:
 - ◆ $b_{02}(u) = (1 - u)^2$
 - ◆ $b_{12}(u) = 2u(1 - u)$
 - ◆ $b_{22}(u) = u^2$
- Aplicando mais uma vez a idéia podemos definir uma cúbica por 4 pontos
$$\mathbf{p}_{02}(u) = (1 - u)^2 \mathbf{p}_0 + 2u(1 - u) \mathbf{p}_1 + u^2 \mathbf{p}_2$$
$$\mathbf{p}_{12}(u) = (1 - u)^2 \mathbf{p}_1 + 2u(1 - u) \mathbf{p}_2 + u^2 \mathbf{p}_3$$
$$\mathbf{p}_{03}(u) = (1 - u) \mathbf{p}_{02}(u) + u \mathbf{p}_{12}(u)$$
$$= (1 - u)^3 \mathbf{p}_0 + 3u(1 - u)^2 \mathbf{p}_1 + 3u^2(1 - u) \mathbf{p}_2 + u^3 \mathbf{p}_3$$

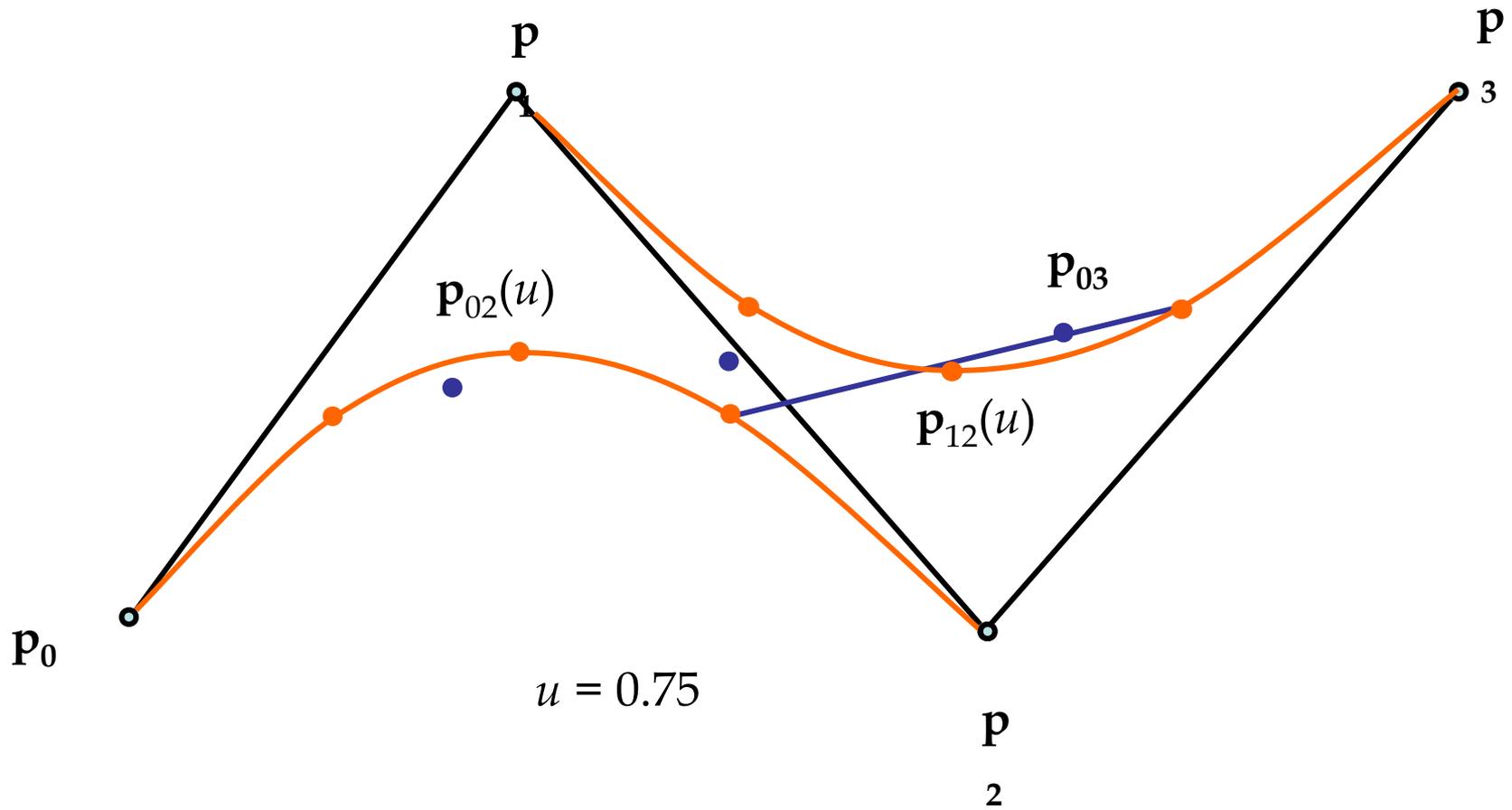
Algoritmo de De Casteljau



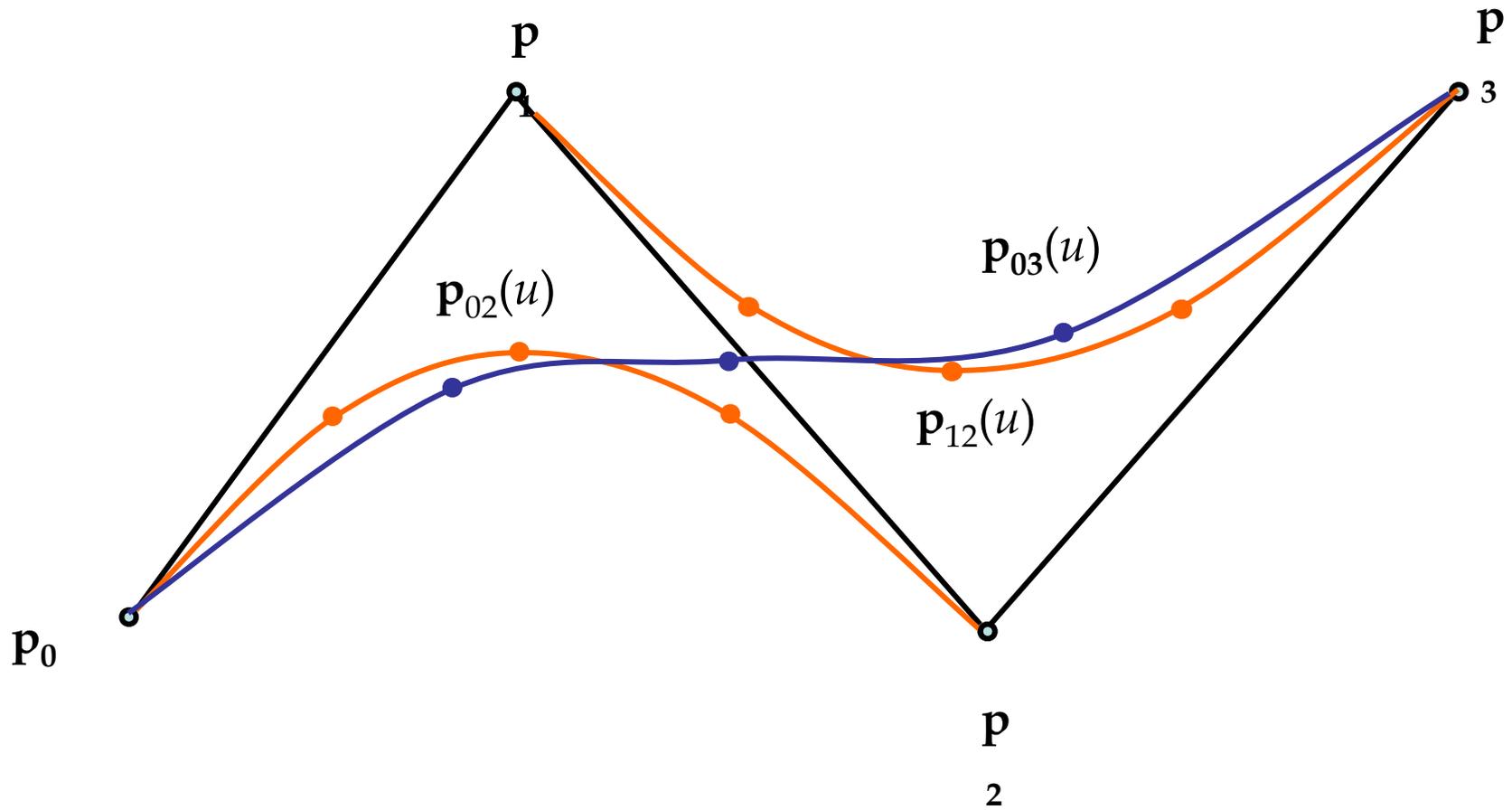
Algoritmo de De Casteljau



Algoritmo de De Casteljau



Algoritmo de De Casteljau



Algoritmo de De Casteljau

- Novamente temos uma curva dada pela soma de 4 funções de mistura (agora cúbicas), cada uma multiplicada por um dos 4 pontos
 - ◆ $b_{03}(u) = (1 - u)^3$
 - ◆ $b_{13}(u) = 3 u (1 - u)^2$
 - ◆ $b_{23}(u) = 3 u^2 (1 - u)$
 - ◆ $b_{33}(u) = u^3$
- Em geral, uma curva de grau n pode ser construída desta forma e será expressa por

$$\mathbf{p}_{0n}(u) = \sum_{j=0}^n b_{jn}(u) \mathbf{p}_j$$

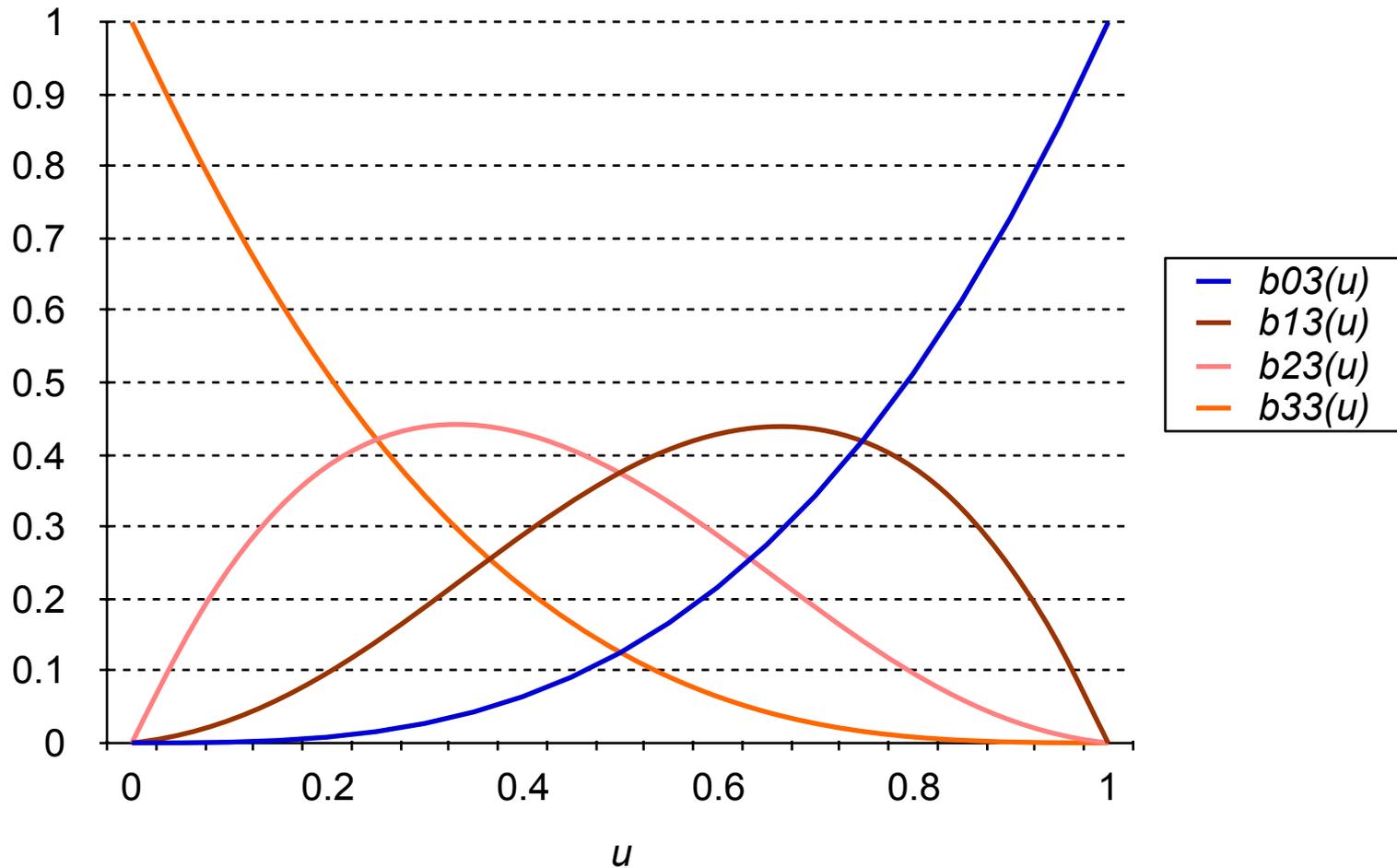
Curvas de Bézier e Polinômios de Bernstein

- As curvas construídas pelo algoritmo de De Casteljaou são conhecidas como *curvas de Bézier* e as funções de mistura são chamadas de *base Bézier* ou *polinômios de Bernstein*
- Observamos que os polinômios de Bernstein de grau n têm como forma geral $b_{i,n}(u) = c_i u^i (1 - u)^{n-i}$
- Se escrevermos as constantes c_i para os diversos polinômios, teremos
 - ◆ 1º grau: 1 1
 - ◆ 2º grau: 1 2 1
 - ◆ 3º grau: 1 3 3 1
 - ◆ 4º grau: 1 4 6 4 1
- Vemos que o padrão de formação corresponde ao *Triângulo de Pascal* e portanto, podemos escrever

$$b_{i,n}(u) = \binom{n}{i} u^i (1 - u)^{n-i}$$

Polinômios de Bernstein

Polinômios de Bernstein de grau 3



Forma Matricial da Base Bézier

- Podemos escrever a equação para uma curva de Bézier cúbica na forma

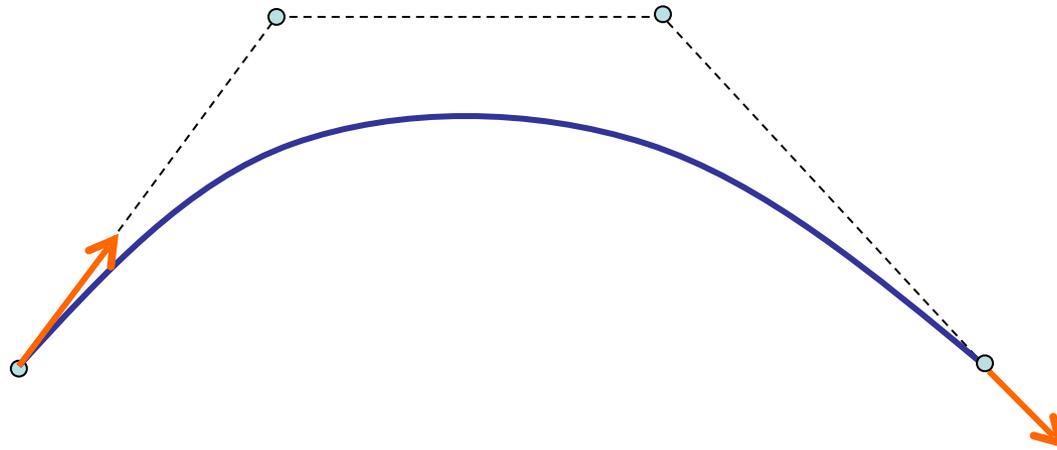
$$\mathbf{p}(u) = \mathbf{p}_{03}(u) = [1 \quad u \quad u^2 \quad u^3] \mathbf{M}_B \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

onde \mathbf{M}_B é a matriz de coeficientes da base Bézier

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

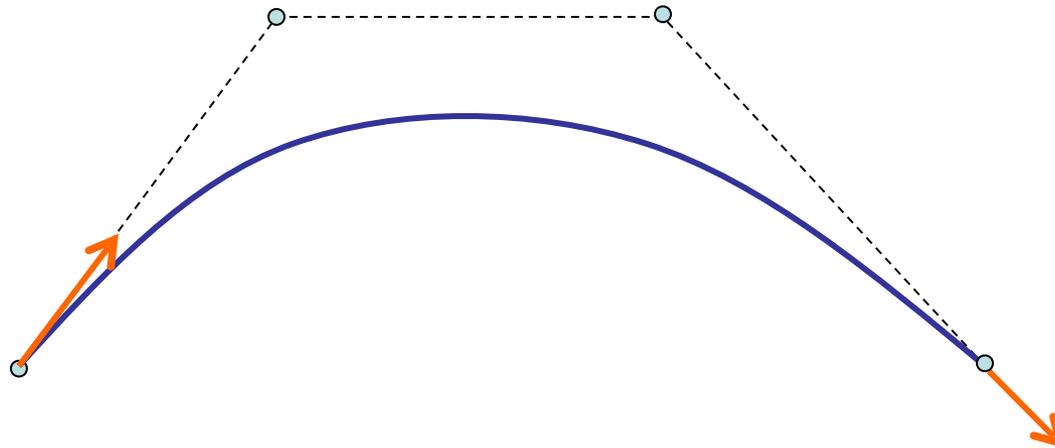
Propriedades de Curva de Bézier

- Continuidade infinita (todas as derivadas são contínuas)
- O grau da curva (do polinômio) é dado pelo número de pontos do polígono de controle menos 1
- A curva de Bézier está contida no fecho convexo do polígono de controle
 - ◆ Os polinômios de Bernstein somam 1 para qualquer u
- A curva interpola o primeiro e último ponto do polígono de controle



Propriedades de Curva de Bézier

- As tangentes à curva em \mathbf{p}_0 e \mathbf{p}_n têm a direção dos segmentos de reta $\mathbf{p}_0\mathbf{p}_1$ e $\mathbf{p}_{n-1}\mathbf{p}_n$, respectivamente
 - ◆ Para cúbicas, as derivadas são $3(\mathbf{p}_1 - \mathbf{p}_0)$ e $3(\mathbf{p}_2 - \mathbf{p}_3)$
- Qualquer linha reta intercepta a curva tantas ou menos vezes quanto intercepta o polígono de controle
 - ◆ Não pode oscilar demasiadamente
- Transformar os pontos de controle (transf. afim) e desenhar a curva é equivalente a desenhar a curva transformada

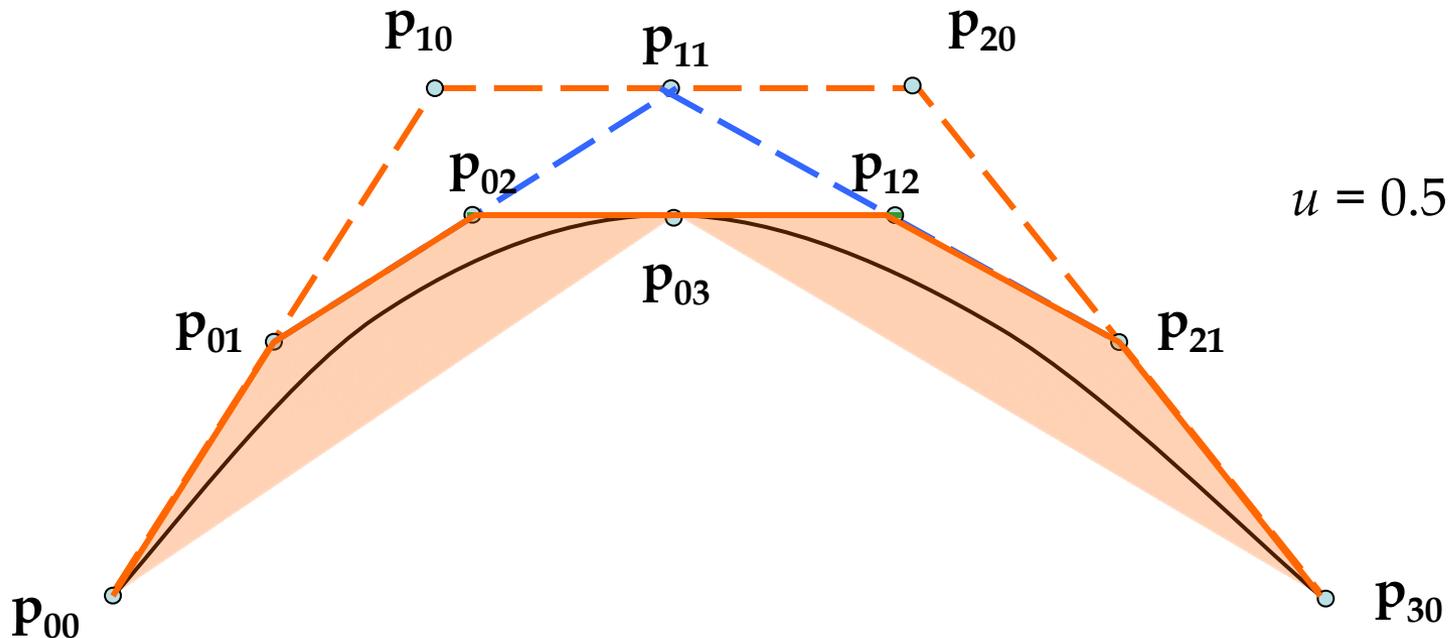


Desenhando Curvas Bézier

- Curva normalmente é aproximada por uma linha poligonal
- Pontos podem ser obtidos avaliando a curva em $u = u_1, u_2 \dots u_k$
 - ◆ Avaliar os polinômios de Bernstein
 - ◆ Usar o algoritmo recursivo de De Casteljau
- Quantos pontos?
 - ◆ Mais pontos em regiões de alta curvatura
- Idéia: subdividir recursivamente a curva em trechos até que cada trecho seja aproximadamente “reto”

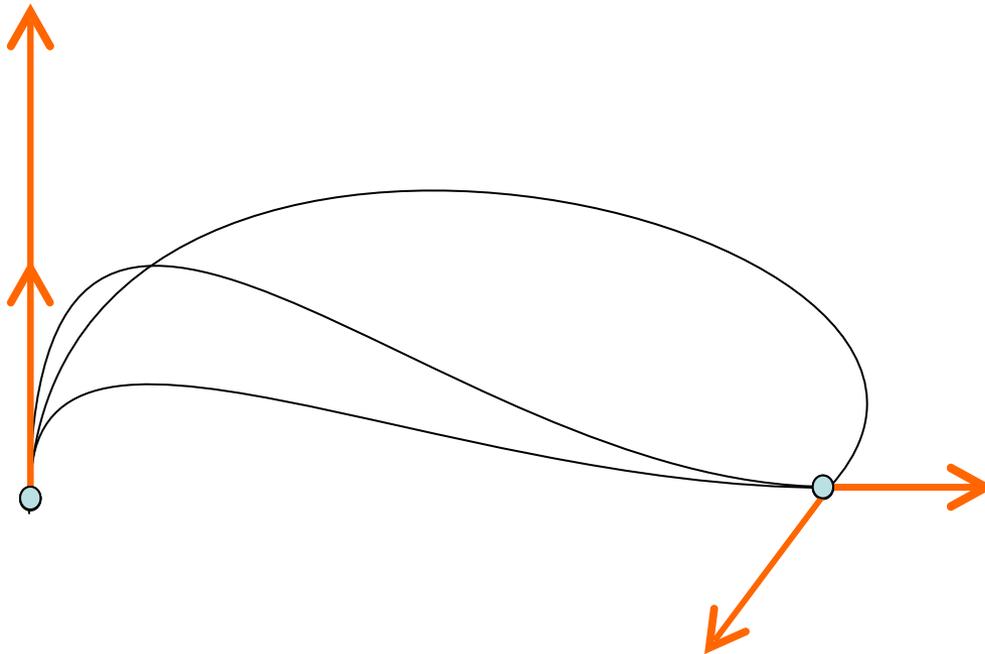
Subdivisão de Curvas Bézier

- Como saber se trecho da curva é “reto”?
 - ◆ Encontrar o polígono de controle do trecho
 - ◆ Parar se vértices do polígono forem aproximadamente colineares



Curvas de Hermite

- Ao invés de modelar a curva a partir de um polígono de controle (Bézier), especifica-se pontos de controle e vetores tangentes nesses pontos
- Vantagem: é fácil emendar várias curvas bastando especificar tangentes iguais nos pontos de emenda
- Exemplos (cúbicas):



Curvas de Hermite

- No caso de cúbicas, temos o ponto inicial e final além dos vetores tangentes

$$p(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} M_H \begin{bmatrix} p_0 \\ p_1 \\ p'_0 \\ p'_1 \end{bmatrix}$$

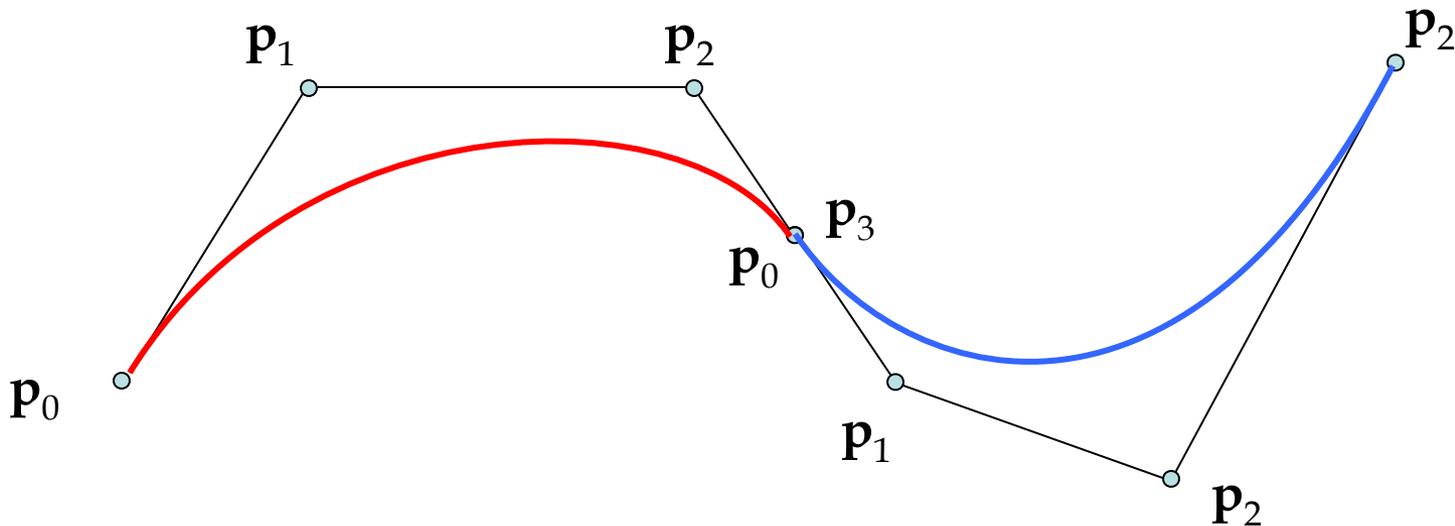
$$\text{onde } M_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

Curvas Longas

- Curvas Bézier com k pontos de controle são de grau $k-1$
- Curvas de grau alto são difíceis de desenhar
 - ◆ Complexas
 - ◆ Sujeitas a erros de precisão
- Normalmente, queremos que pontos de controle tenham efeito *local*
 - ◆ Em curvas Bézier, todos os pontos de controle têm efeito *global*
- Solução:
 - ◆ Emendar curvas polinomiais de grau baixo
 - ◆ Relaxar condições de continuidade

Emendando Curvas Bézier

- Continuidade C^0 : Último ponto da primeira = primeiro ponto da segunda
- Continuidade C^1 : C^0 e segmento $\mathbf{p}_2\mathbf{p}_3$ da primeira com mesma direção e comprimento que o segmento $\mathbf{p}_0\mathbf{p}_1$ da segunda
- Continuidade C^2 : C^1 e + restrições sobre pontos \mathbf{p}_1 da primeira e \mathbf{p}_2 da segunda

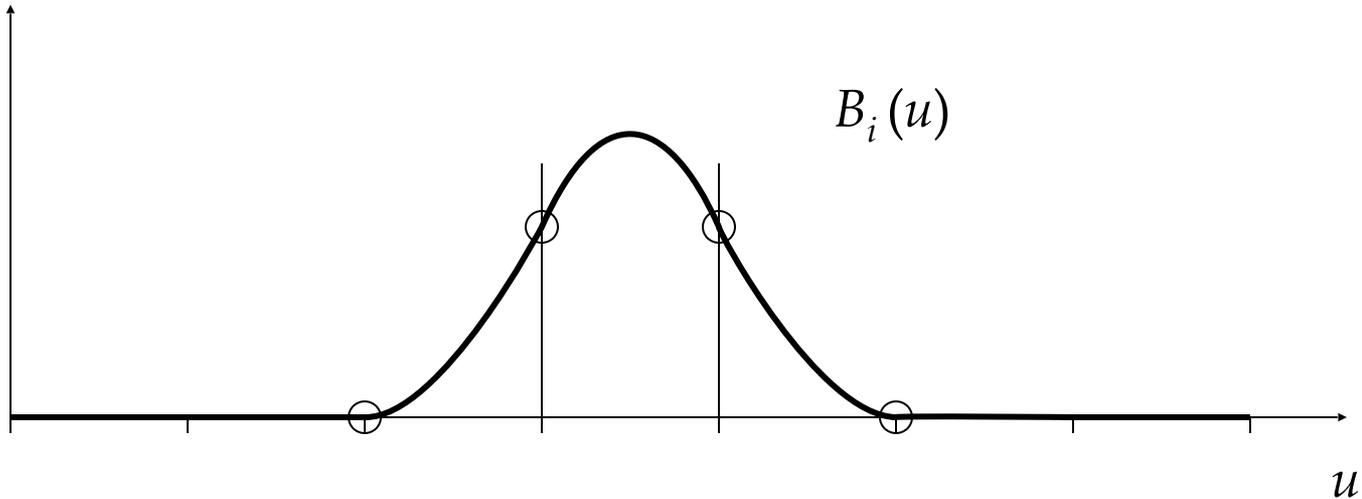


Splines

- A base de Bézier não é própria para a modelagem de curvas longas
 - ◆ Bézier única: suporte não local
 - ◆ Trechos emendados: restrições não são naturais
- Base alternativa: B-Splines
 - ◆ Nome vem de um instrumento usado por desenhistas
 - ◆ Modelagem por polígonos de controle sem restrições adicionais
 - ◆ Suporte local
 - Alteração de um vértice afeta curva apenas na vizinhança
 - ◆ Existem muitos tipos de Splines, mas vamos nos concentrar em B-splines uniformes
 - Uma B-spline uniforme de grau d tem continuidade C^{d-1}

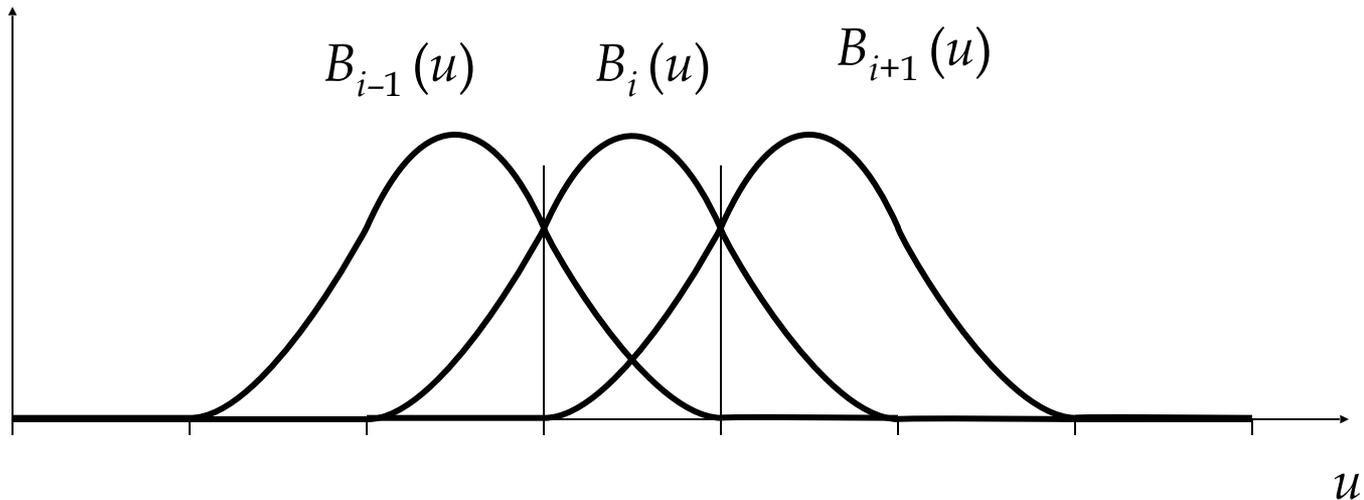
Curvas B-Spline

- Funções de base são não nulas apenas em um intervalo no espaço do parâmetro
 - ◆ Como é impossível obter isso com apenas 1 polinomial, cada função de base é composta da emenda de funções polinomiais
 - ◆ Por exemplo, uma função de base de uma B-spline quadrática tem 3 trechos (não nulos) emendados com continuidade C^1



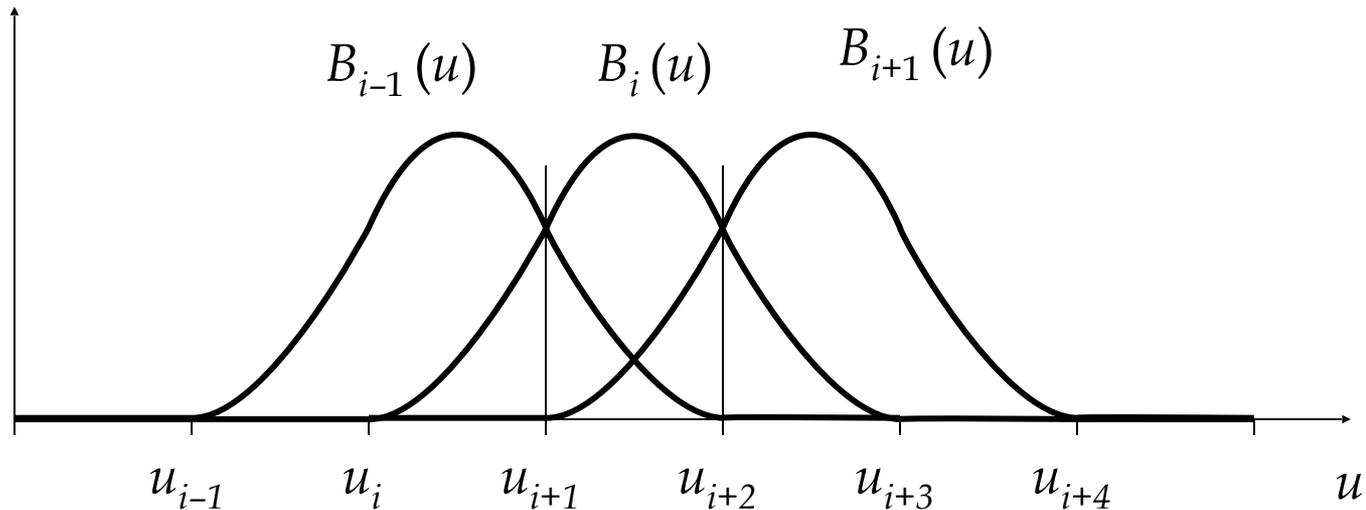
Curvas B-Spline

- Todas as funções de base têm a mesma forma, mas são deslocadas entre si em intervalos no espaço de parâmetros
- Num determinado intervalo, apenas um pequeno número de funções de base são não-nulas
 - ◆ Numa B-spline quadrática, cada intervalo é influenciado por 3 funções de base



Curvas B-Spline

- Os valores u_i do espaço de parâmetro que delimitam os intervalos são chamados de *nós*
- Podemos pensar em intervalos regulares por enquanto (B-Splines uniformes) isto é, $u_i = 1$



Funções da Base B-Spline

- Queremos exprimir curvas como pontos mesclados por intermédio de funções da base B-Spline

$$\mathbf{p}(u) = \sum_{i=0}^m B_{i,d}(u) \mathbf{p}_i$$

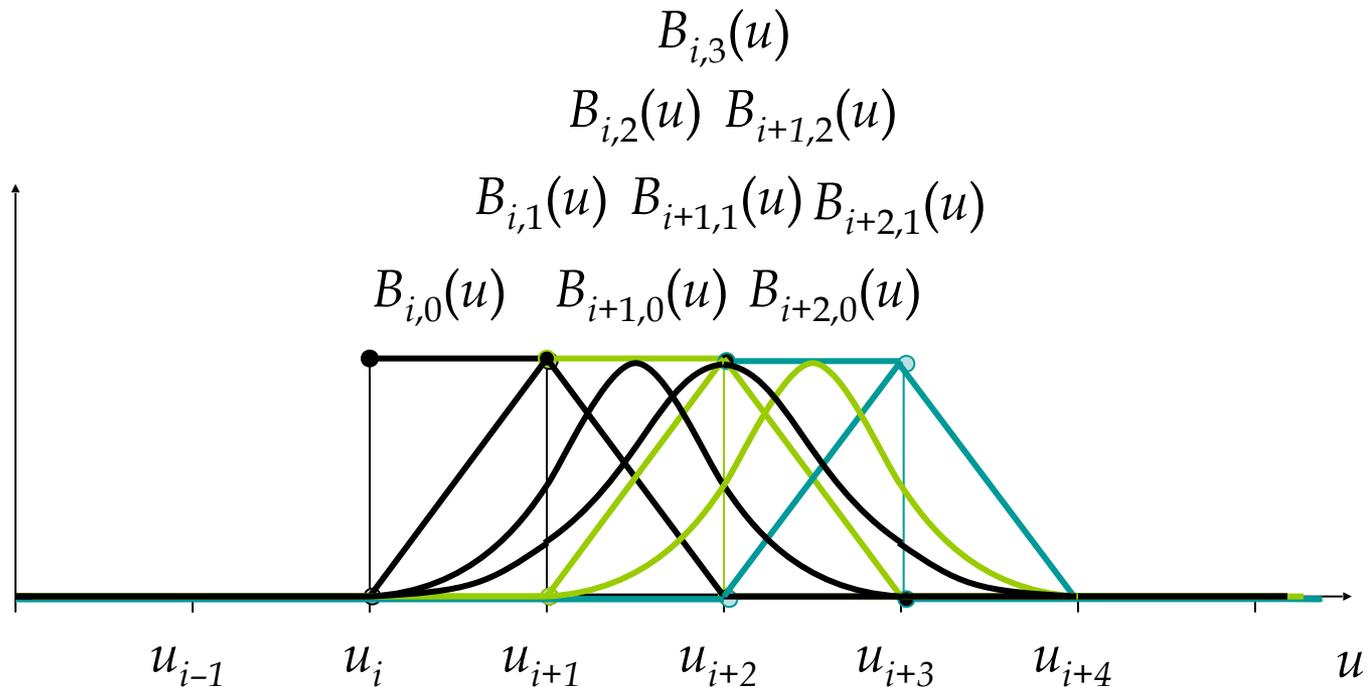
onde m é o número de pontos do polígono de controle e d é o grau da B-spline que se quer usar

- Para derivar as funções da base B-spline pode-se resolver um sistema de equações
 - ◆ Para B-splines cúbicas, requiere-se continuidade C^2 nos nós, a propriedade do fecho convexo, etc
- Uma maneira mais natural é utilizar a recorrência de Cox-de Boor que exprime as funções da base B-Spline de grau k como uma interpolação linear das funções de grau $k-1$

Recorrência Cox-de Boor

$$B_{k,0}(u) = \begin{cases} 1 & \text{para } u_k \leq u < u_{k+1}, \\ 0 & \text{caso contrário.} \end{cases}$$

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



Recorrência Cox-de Boor

$$B_{k,0}(u) = \begin{cases} 1 & \text{para } u_k \leq u < u_{k+1}, \\ 0 & \text{caso contrário.} \end{cases}$$

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

$$\mathbf{p}(u) = \sum_{i=0}^m B_{i,d}(u) \mathbf{p}_i$$

$$\mathbf{p}(u_i \leq u < u_{i+1})$$

$$\mathbf{p}_i \odot$$

$$\mathbf{p}(u_{i+1} \leq u < u_{i+2})$$

$$\odot$$

$$\mathbf{p}_{i+1}$$

$$\mathbf{p}(u_{i+2} \leq u < u_{i+3})$$

$$\odot$$

$$\mathbf{p}_{i+2}$$

$$\mathbf{p}(u_{i+2} \leq u < u_{i+4})$$

$$\odot$$

$$\mathbf{p}_{i+3}$$

$$d = 0$$

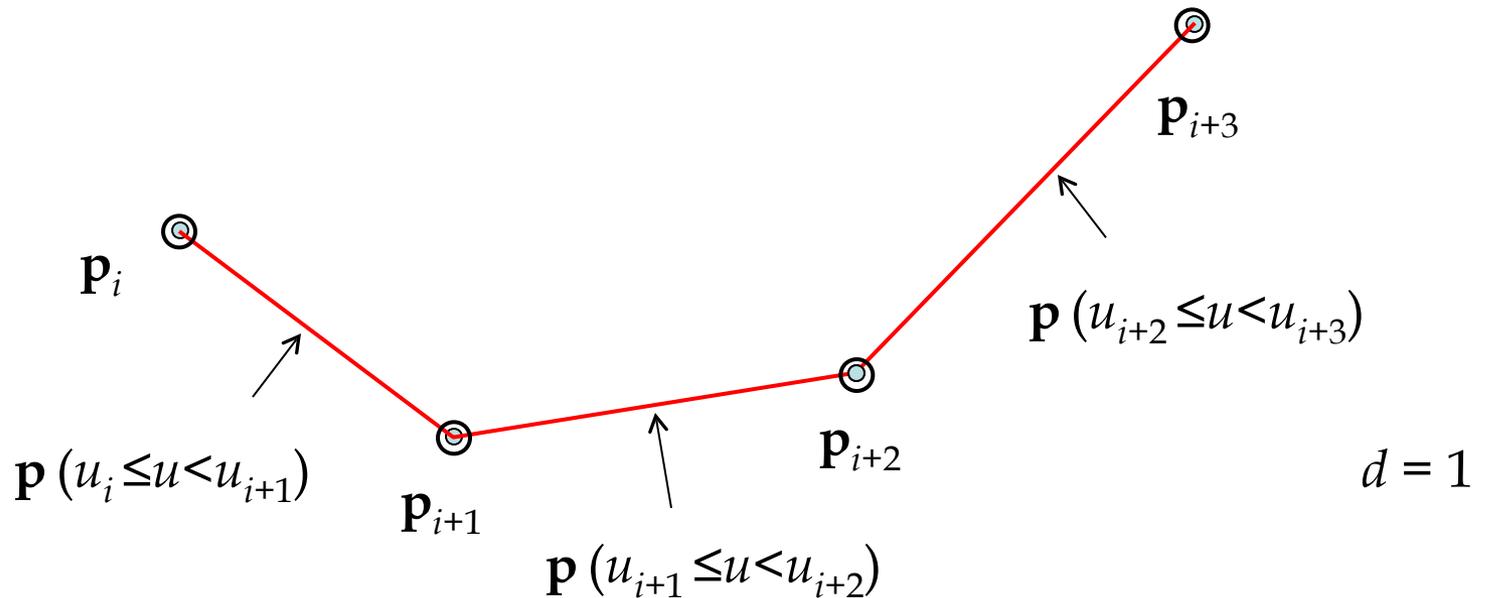
(assumir que para $u = u_i$
Spline de grau 0 passa por \mathbf{p}_i)

Recorrência Cox-de Boor

$$B_{k,0}(u) = \begin{cases} 1 & \text{para } u_k \leq u < u_{k+1}, \\ 0 & \text{caso contrário.} \end{cases}$$

$$\mathbf{p}(u) = \sum_{i=0}^m B_{i,d}(u) \mathbf{p}_i$$

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

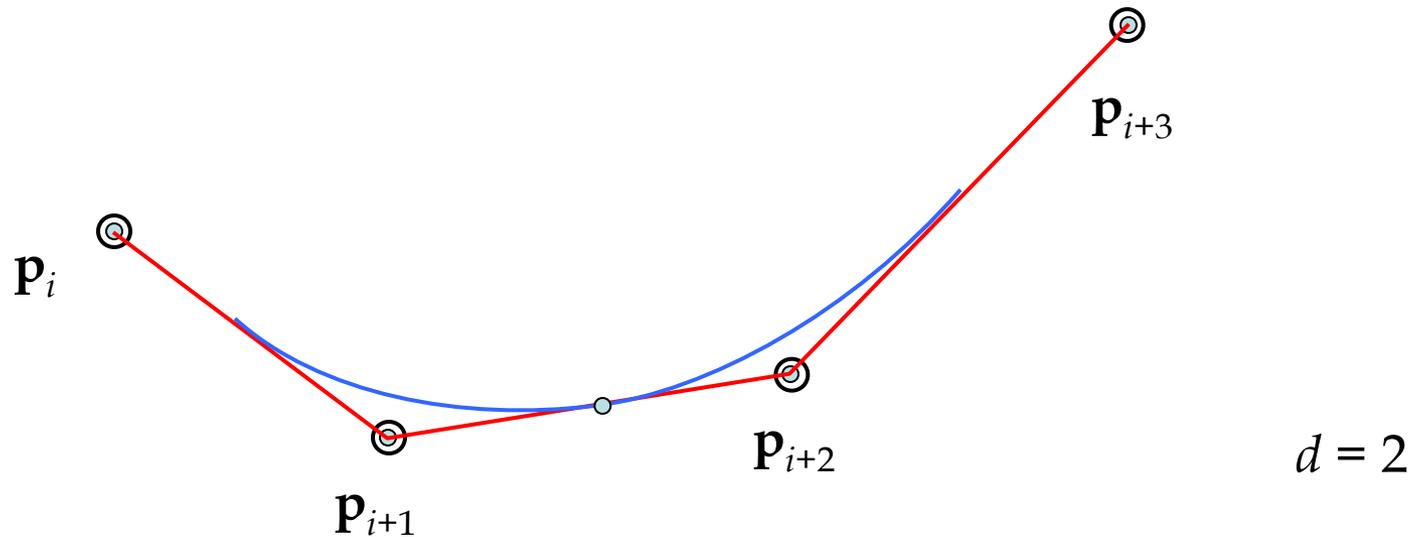


Recorrência Cox-de Boor

$$B_{k,0}(u) = \begin{cases} 1 & \text{para } u_k \leq u < u_{k+1}, \\ 0 & \text{caso contrário.} \end{cases}$$

$$\mathbf{p}(u) = \sum_{i=0}^m B_{i,d}(u) \mathbf{p}_i$$

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

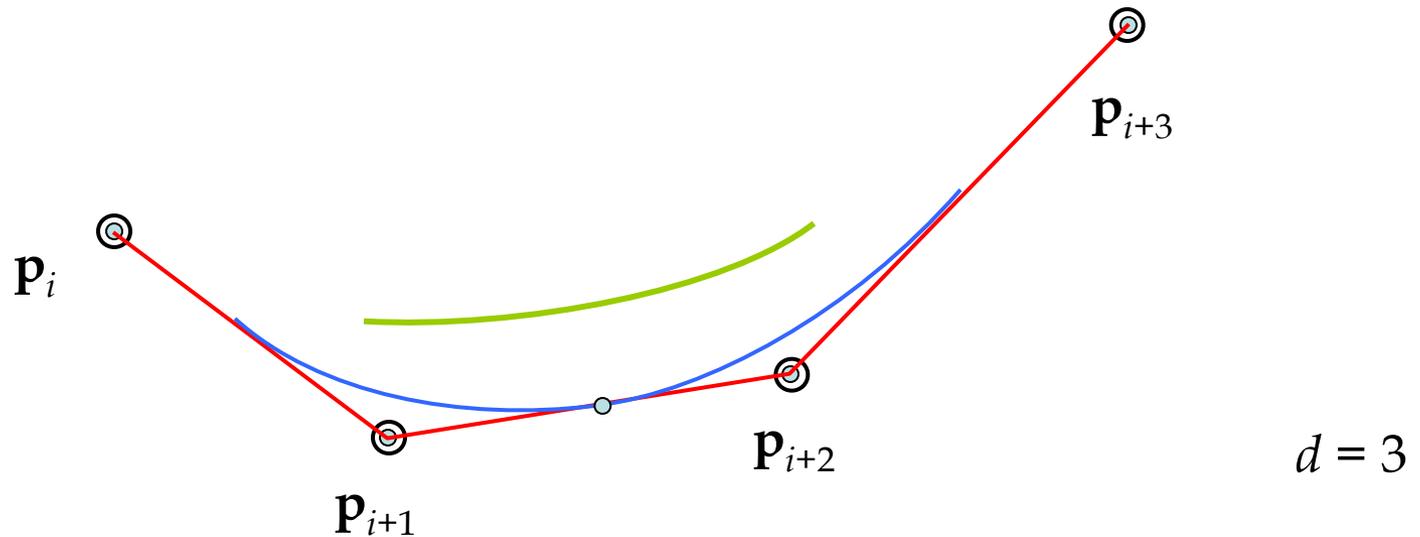


Recorrência Cox-de Boor

$$B_{k,0}(u) = \begin{cases} 1 & \text{para } u_k \leq u < u_{k+1}, \\ 0 & \text{caso contrário.} \end{cases}$$

$$\mathbf{p}(u) = \sum_{i=0}^m B_{i,d}(u) \mathbf{p}_i$$

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

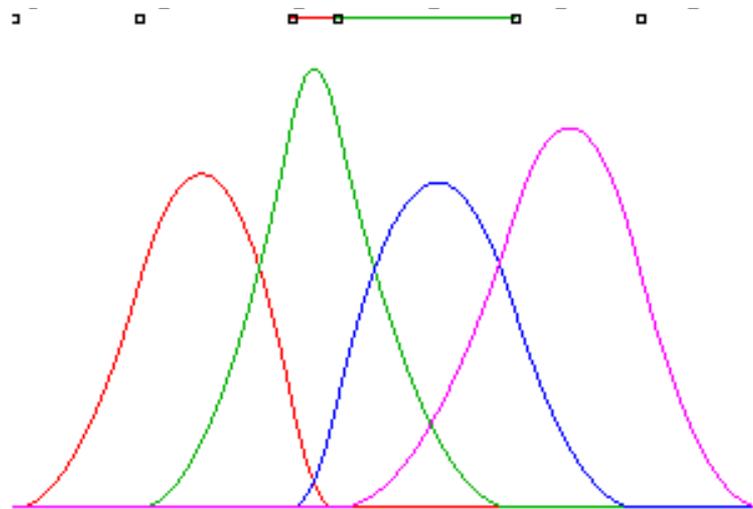
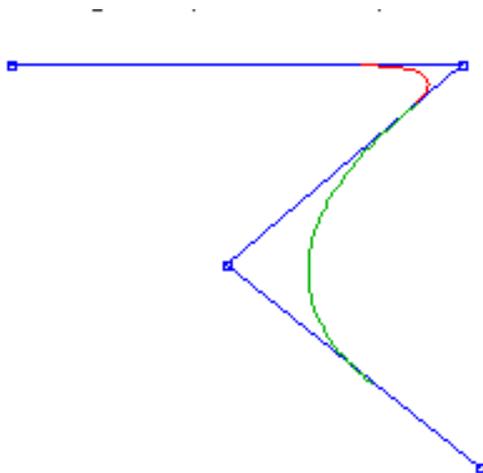


Propriedades das B-Splines

- Dados $n+1$ pontos $(\mathbf{p}_0 \dots \mathbf{p}_n)$, é composta de $(n-d+1)$ curvas Bézier de grau d emendadas com continuidade $d-1$ nos $n+d+1$ nós $u_0, u_1, \dots, u_{n+d+1}$
- Cada ponto da curva é afetado por $d+1$ pontos de controle
- Cada ponto de controle afeta $d+1$ segmentos
- Curva restrita ao fecho convexo do polígono de controle
- Invariância sob transformações afim

Efeito dos Nós

- Os intervalos entre nós influenciam a importância dos pontos de controle
 - ◆ Exemplo: B-spline Quádrica



Inserindo Nós

- Podemos ver que as B-splines uniformes em geral não passam pelos pontos de controle
- Entretanto, se repetirmos nós podemos fazer a curva se aproximar dos pontos de controle
 - ◆ Para fazer a interpolação do primeiro ponto usando uma B-Spline cúbica, fazemos $u_0 = u_1 = u_2 = u_3$
 - ◆ Para fazer uma B-spline cúbica passando por 4 pontos podemos usar o vetor de nós: 0, 0, 0, 0, 1, 1, 1, 1
 - ◆ De fato, com este vetor de nós, teremos uma Bézier cúbica

Curvas Racionais

- Funções são razões

- ◆ Avaliados em coordenadas homogêneas:

$$[x(t), y(t), z(t), w(t)] \rightarrow \left[\frac{x(t)}{w(t)}, \frac{y(t)}{w(t)}, \frac{z(t)}{w(t)} \right]$$

- ◆ NURBS (Non-Uniform Rational B-Splines): $x(t)$, $y(t)$, $z(t)$ e $w(t)$ são B-splines não uniformes

- Vantagens:

- ◆ Invariantes sob transformações perspectivas e portanto podem ser avaliadas no espaço da imagem
- ◆ Podem representar perfeitamente seções cônicas tais como círculos, elipses, etc

Parametrização de um Círculo

- Por exemplo, uma parametrização conhecida do círculo é dada por

$$x(u) = \frac{1 - u^2}{1 + u^2}$$

$$y(u) = \frac{2u}{1 + u^2}$$

- Podemos expressar essa parametrização em coordenadas homogêneas por:

$$x(u) = 1 - u^2$$

$$y(u) = 2u$$

$$w(u) = 1 + u^2$$

OpenGL e Curvas Paramétricas

- OpenGL define o que são chamados de *avaliadores* que podem avaliar uma curva Bézier para um valor do parâmetro
 - ◆ Para definir os pontos de controle:
 - `glMap1f(...)`
 - ◆ Para avaliar um ponto:
 - `glEvalCoord(param)`
 - ◆ Para avaliar uma seqüência de pontos:
 - `glMapGrid1f(n, t1, t2)`
 - `glEvalMesh1f(mode, p1, p2)`
- Essas rotinas avaliam a curva em intervalos regulares no espaço de parâmetros
 - ◆ Não necessariamente a melhor maneira!