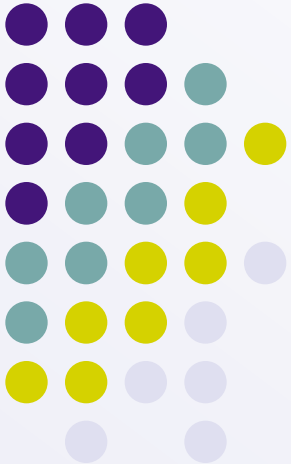
Universidade de São Paulo
Instituto de Matemática e Estatística

Introdução à Computação Gráfica

Marcel P. Jackowski
mjack@ime.usp.br

Aula #16

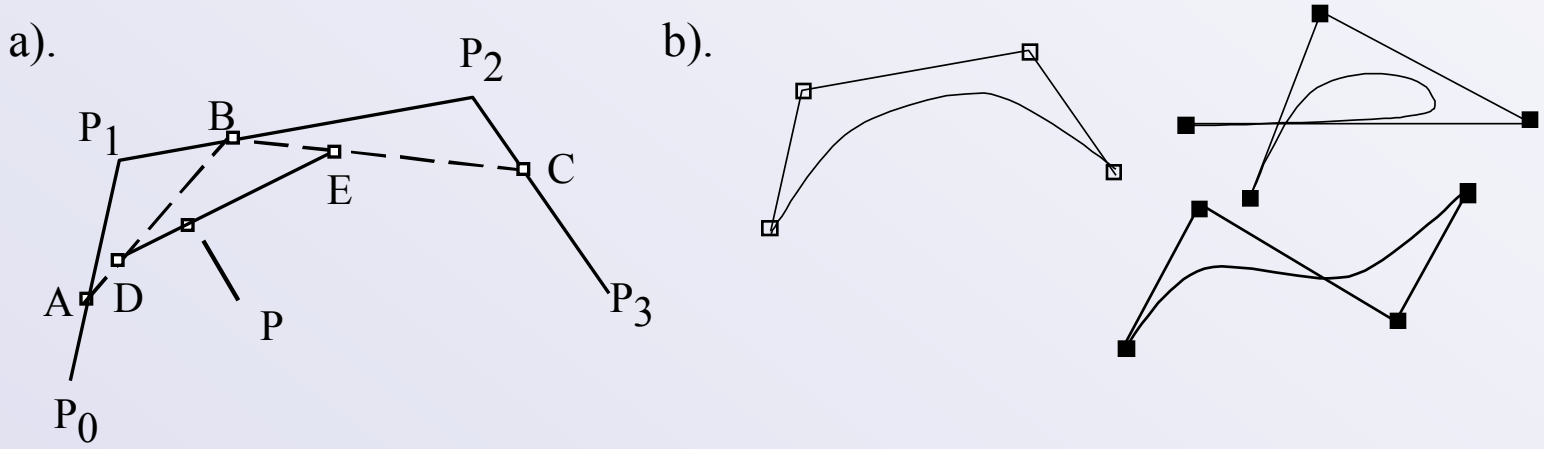


Objetivos

- Curvas Bézier
  - Polinômios de Bernstein
- Funções de ponderação genéricas
  - Funções splines
- Curvas B-splines

Curvas de Bézier

- Uma curva Bézier baseia-se em L+1 pontos de controle e é gerada através de uma combinação de polinômios de grau L.
- Polinômios de alto grau são mais dispensiosos computacionalmente e também são vulneráveis à erros de arredondamento.
- A forma mais comum das curvas Bezier utilizam 4 pontos de controle.



Generalização de curvas Bézier

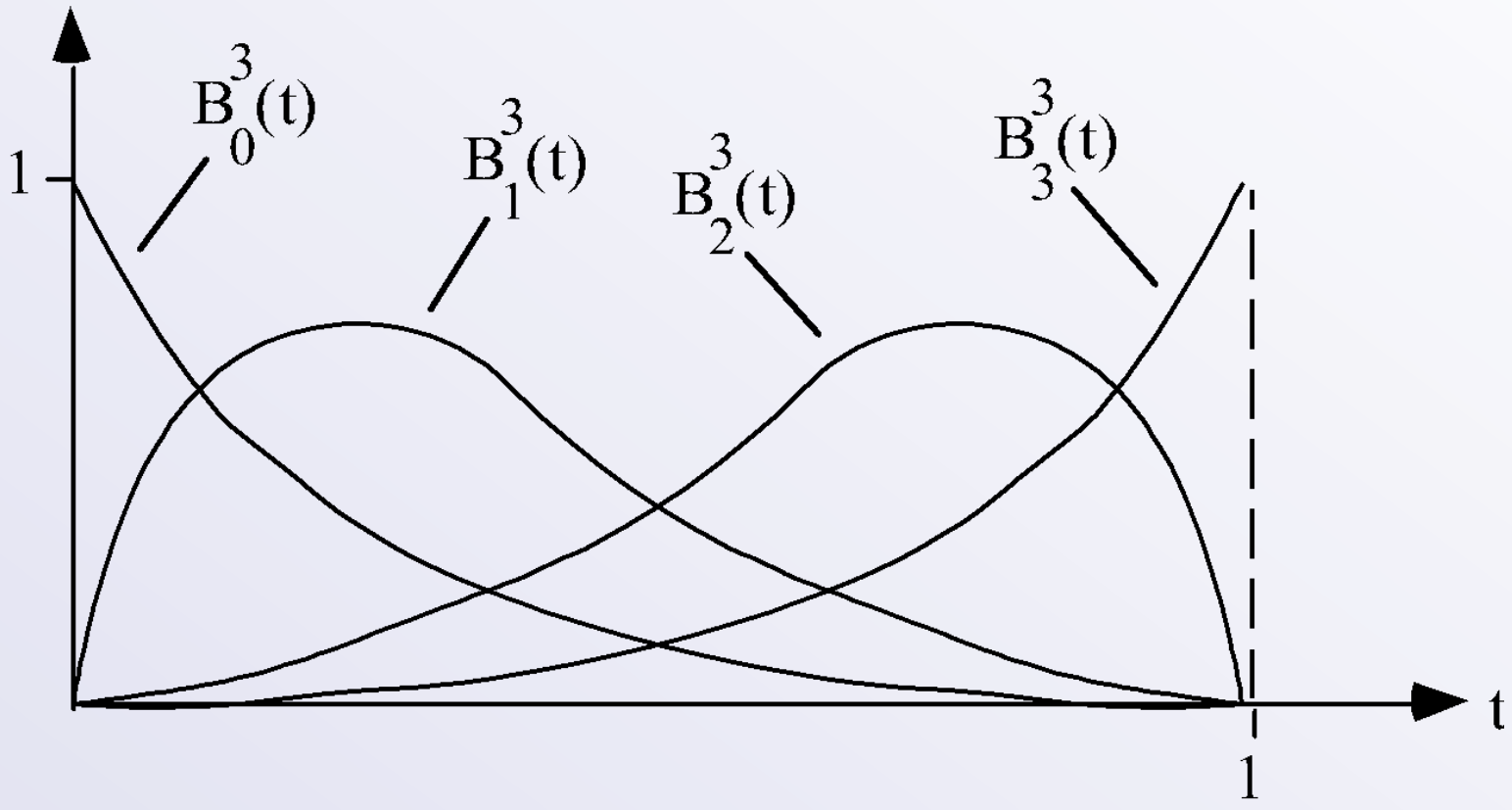
- A curva resultante será:

$$B_k^L(t) = \binom{L}{k}(1-t)^{L-k}\,t^k \qquad P(t) = \sum_{k=0}^{L} \mathrm{P}_k\, B_k^L(t)$$

- onde o coeficiente binomial é:

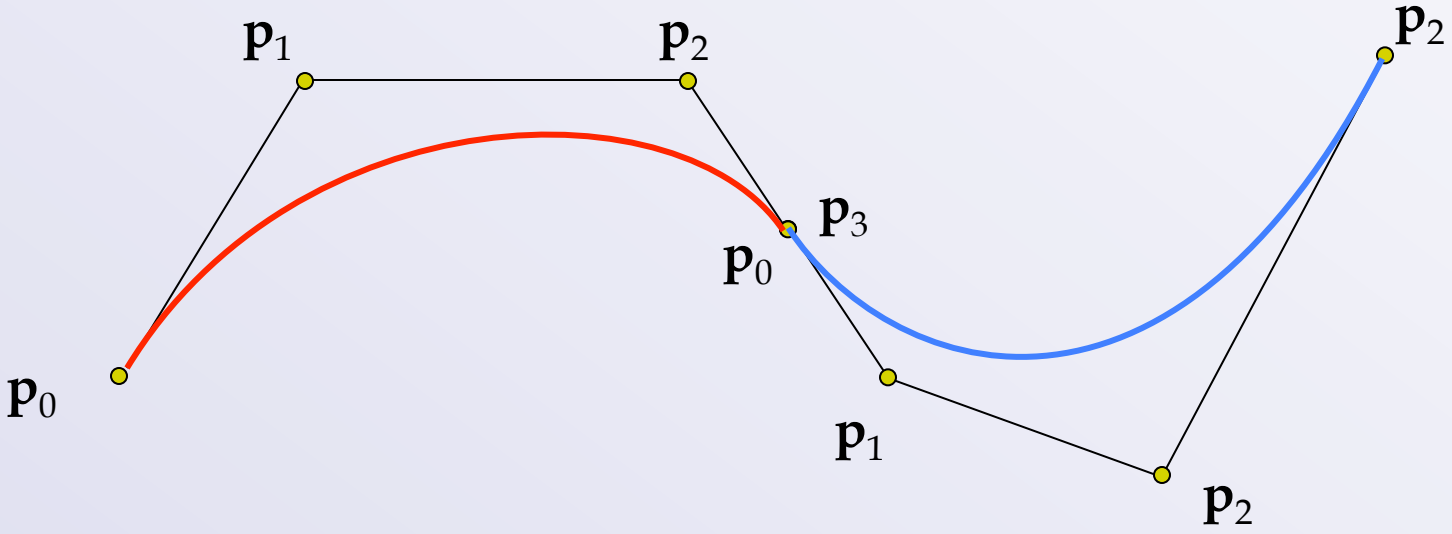
$$\binom{L}{k} = \frac{L!}{k!(L-k)!}$$

Polinômios de Bernstein



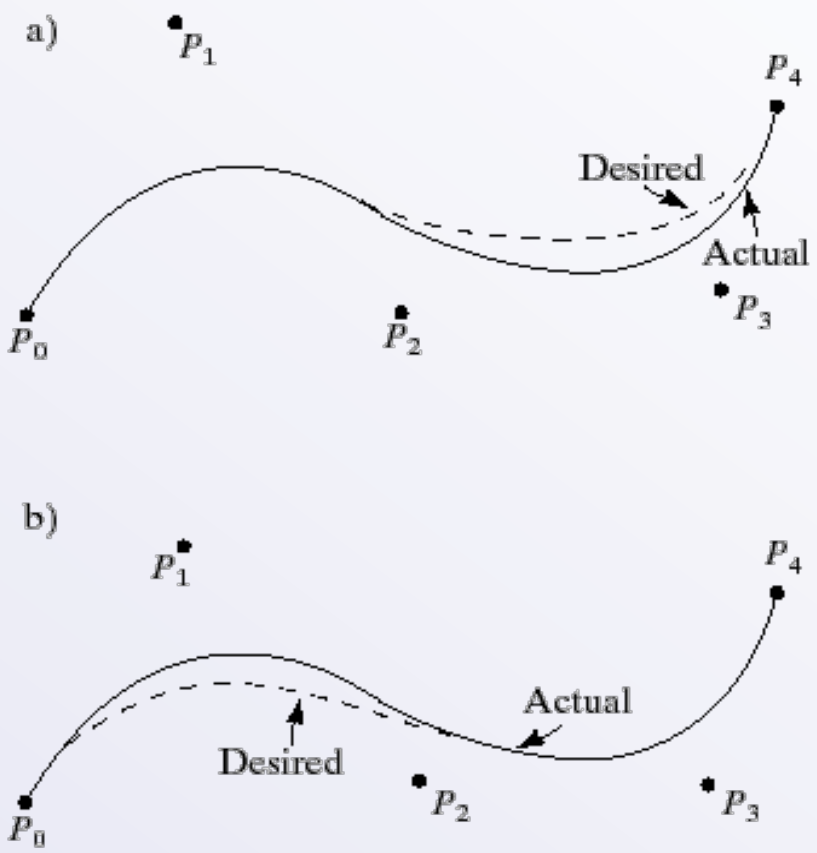
Emendando curvas Bézier

- Continuidade C^0/G^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

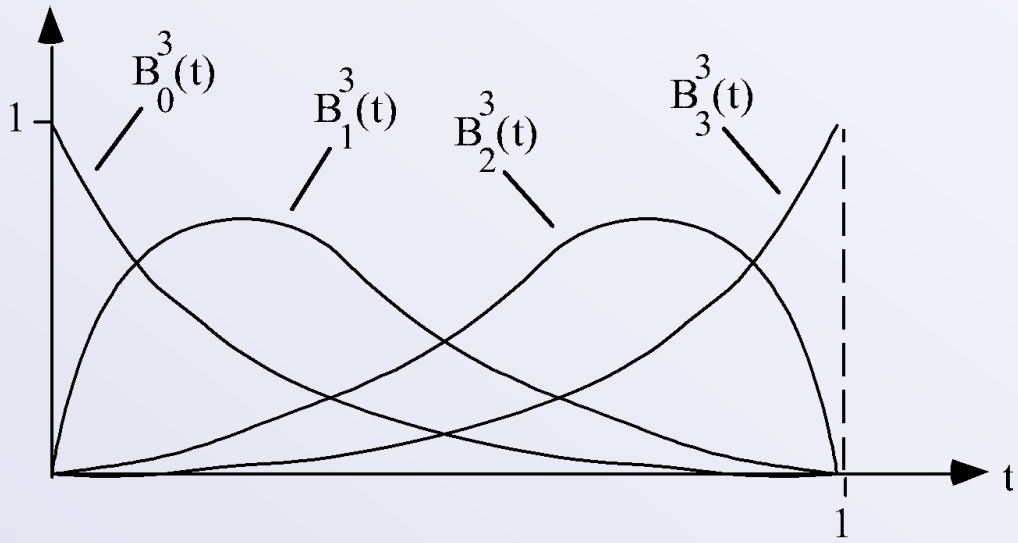


Grau de controle

- As curvas Bézier não oferecem **controle local** suficiente.
  - A figura ao lado mostra cinco pontos de controle e a curva Bézier correspondente.
  - Alterações na posição dos pontos de controle provocam mudanças na curva, mas não necessariamente permitem a obtenção da trajetória desejada.



Polinômios de Bernstein



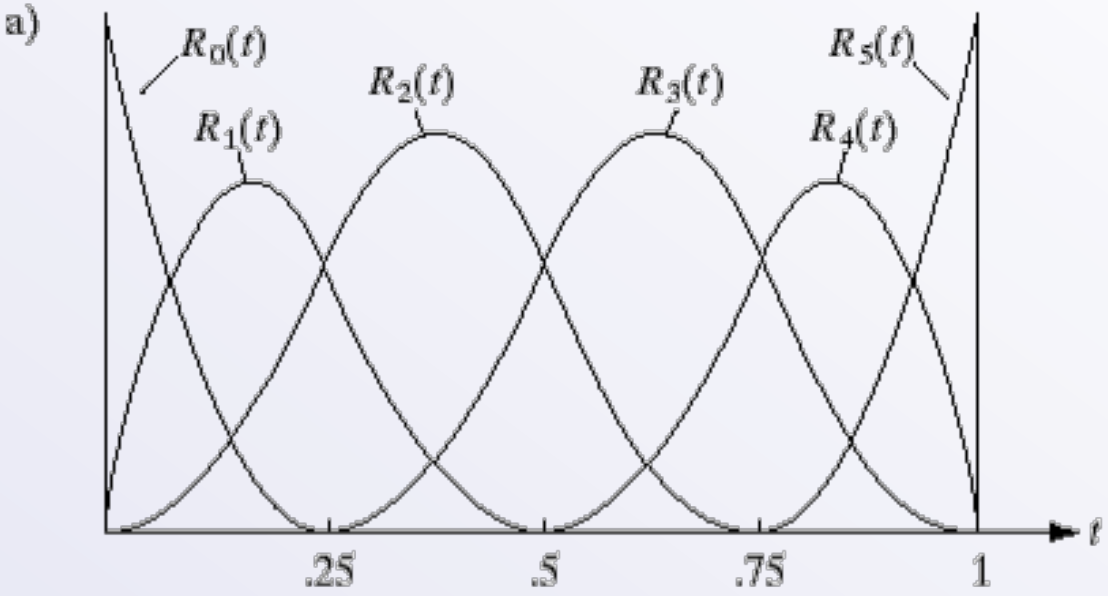
- O intervalo no qual uma função é não-zero é denominado suporte.
- Já que cada polinômio de Bernstein possui suporte em todo o intervalo [0, 1], e a curva é uma ponderação destas funções.
- Assim, o ajuste de qualquer ponto de controle afeta a curva globalmente, sem oferecer o controle local desejado.

Polinômios de Berstein são exemplos de funções de ponderação cujo suporte é [0,1]

Funções de ponderação

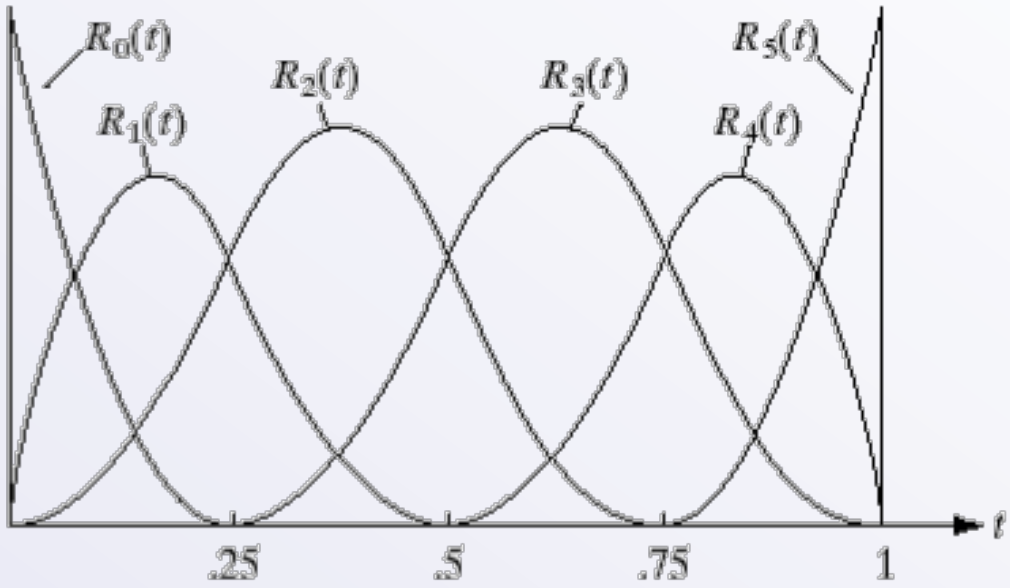
- Contraste o conjunto de funções de ponderação ao lado.

- Estas seis funções, $R_0(t)$, $R_1(t)$, ... , $R_5(t)$ possuem suporte em somente parte do intervalo [0, 1].



Funções de ponderação

- O suporte de $R_0(t)$ é [0, .25] e o suporte de $R_3(t)$ é [.25, 1.0].

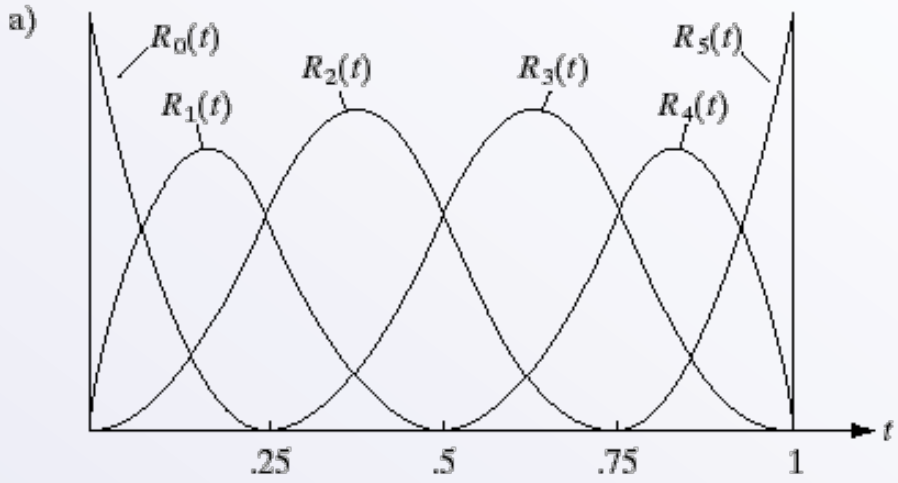


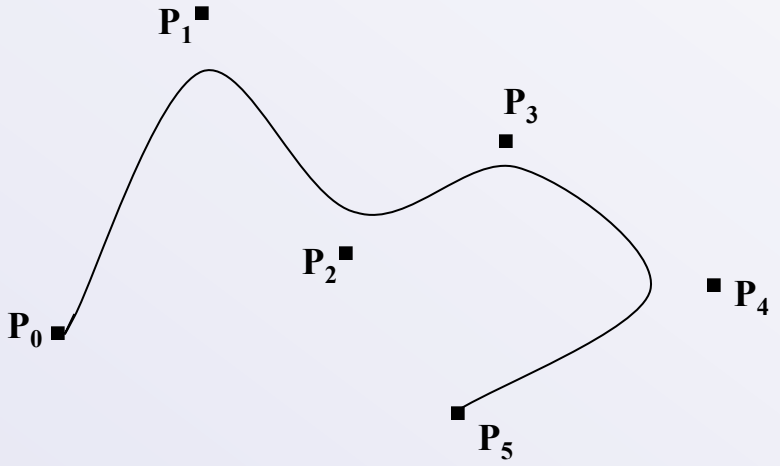
Para qualquer valor de *t*, não mais que três funções de ponderação estão ativas.

Funções de ponderação

- Podemos usar tais funções para construir $V(t)$, curva baseada em 6 pontos de controle, P_0, P_1, . . . , P_5.
- Utilizaremos a mesma forma paramétrica das curvas de Bézier:

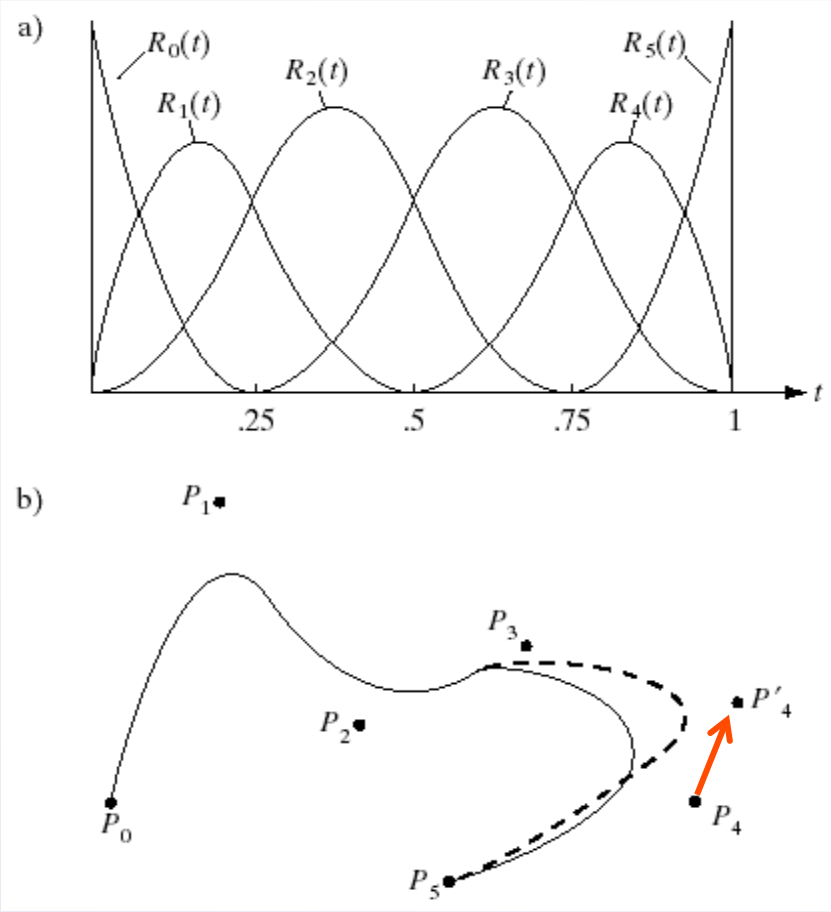
$$V(t) = \sum_{k=0}^{5} P_k R_k(t)$$





Funções de ponderação

- Em cada *t* a posição *V*(*t*) depende de não mais que três pontos de controle.
- Para todos *t* entre [0.75, 1.0] somente os pontos P_3, P_4, e P_5 controlam a forma da curva.
- Se P_4 for movido para P'_4, somente a porção indicada mudará.
- Estas funções de poderação dão um certo grau de controle local aos pontos de controle.



Requisito 1: estabilidade numérica

- Para minimizar o tempo de geração destas curvas, queremos que as funções de ponderação sejam computacionalmente simples
- Devem ser minimalmente susceptíveis à erros de arredondamento
- Consequentemente, devemos escolher polinômios para as funções de ponderação; e o grau dos polinômios não devem ser grandes
  - Funções trigonométricas devem ser evitadas, pois são dispendiosos computacionalmente.

Requisito 2: Unidade

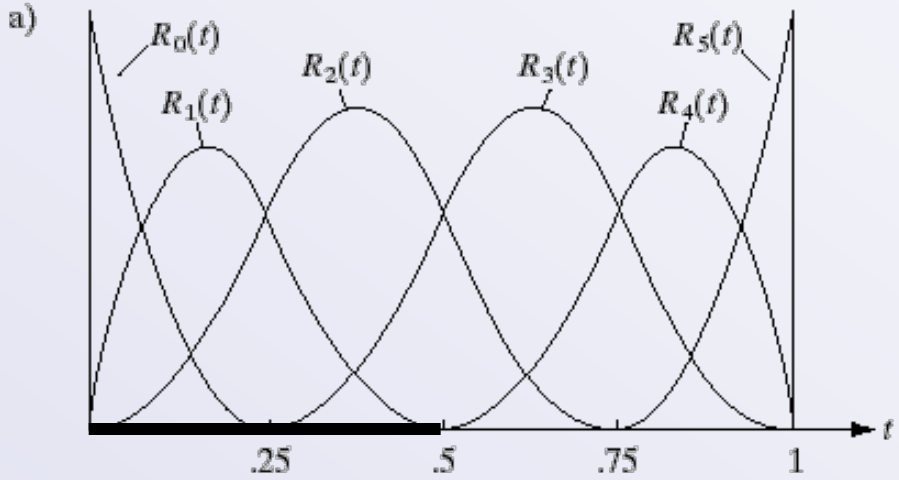
- $V(t)$ deve ser uma soma ponderada dos pontos em cada t dentro do intervalo [a, b]:

$$\sum_{k=0}^{L} R_k(t) = 1$$

- As funções de ponderação devem totalizar 1 para qualquer valor de t.
- Garantem que as curvas resultantes pertençam ao fecho convexo dos pontos de controle.

Requisito 3: Suporte limitado

- Para conguirmos controle local da curva, cada função de ponderação deverá ter suporte em uma faixa pequena do intervalo [*0*, *1*].



Requisito 4: Interpolação

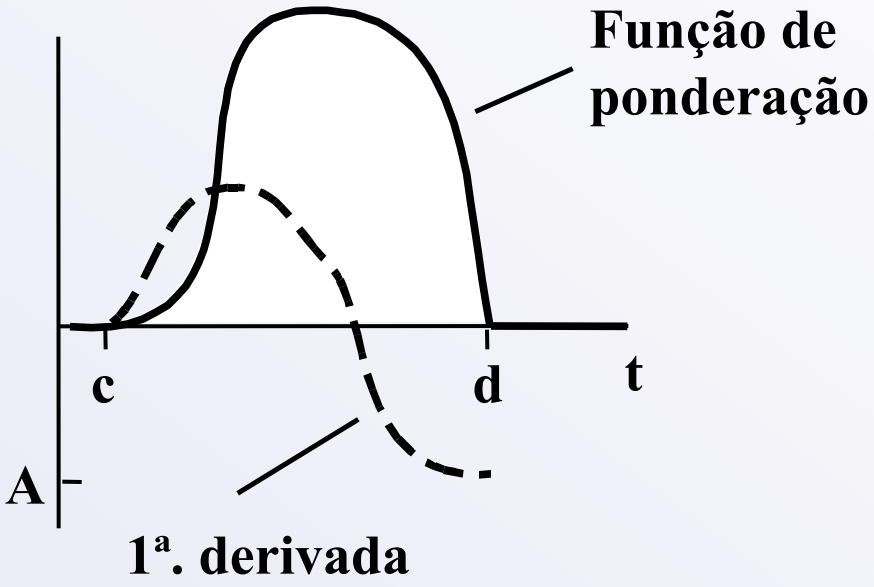
- O usuário pode desejar que $V(t)$ passe em alguns dos pontos de controle, ou ser somente atraída por outros pontos.
- Necessitaremos de um mecanismo para ajustar as funções de ponderação para que certos pontos de controle sejam interpolados.
- Como interpolar outros pontos de controle além do primeiro e último ?

Requisito 5: Suavidade

- Queremos que $V(t)$ seja suave para um conjunto de pontos de controle.
- Tipicamente $V(t)$ deve ser ao menos 1-suave, ou até 2-suave.
- A suavidade resultante de $V(t)$ depende da suavidade das funções de ponderação.
  - Se cada função de ponderação é 1-suave em [a, b], então $V(t)$ também será 1-suave em [a, b].
  - Curva resultante é uma combinação das funções de ponderação

Requisito 5: Suavidade (cont.)

- A derivada varia de forma contínua saido do zero em *t* = *c*, onde a função de ponderação inicia.
- Mas a derivada em *d* é discontínua, pulando de *A* para 0. Uma curva usando este tipo de função não será 1-suave em *t* = *d*.
- Assim, desejamos que tais funções sejam contínuas internamente, e que também começem e terminem com derivadas zero.



Função de ponderação candidata

$$R(t) = at^3 + bt^2 + ct + d$$

Precisamos fazer $R(0)=R(1)=R'(0)=R'(1)=0$.

- $R(0)=0 \rightarrow d = 0$
- $R(1)=0 \rightarrow a+b+c+d = 0$
- $R'(0)=0 \rightarrow c = 0$
- $R'(1)=0 \rightarrow 3a + 2b + c = 0$

Curvas polinomiais por partes

- Não é muito fácil achar uma função polinomial que atenda todos os requisitos e que seja ao mesmo tempo maleável
- Idéia:
  - Que tal emendar várias funções polinomiais de baixa ordem como funções de ponderação ?
  - As curvas são definidas por diferentes polinômios em diferentes intervalos de *t* e são chamadas de **polinomiais "piecewise" ou por partes**.

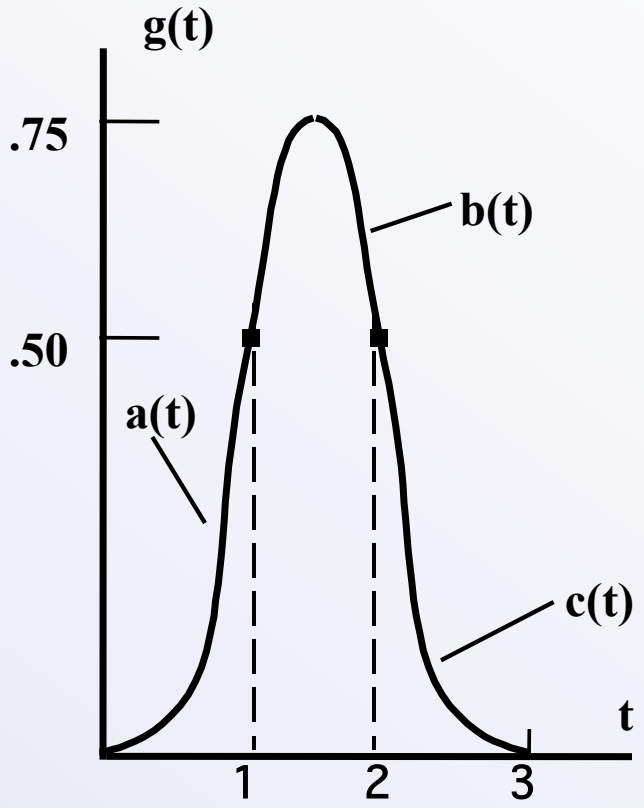
Funções polinomiais por partes

- $g(t)$ consiste de 3 **segmentos** polinomiais, definidos como:

$$a(t) = \frac{1}{2}t^2$$

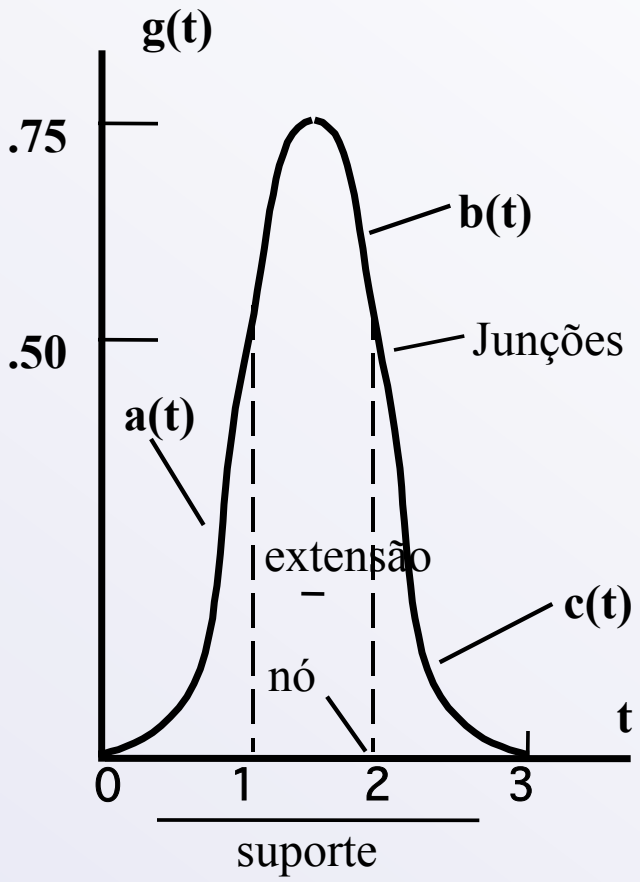
$$b(t) = \frac{3}{4} - (t - \frac{3}{2})^2$$

$$c(t) = \frac{1}{2}(3 - t)^2$$



Funções polinomiais por partes

- O suporte de *g*(*t*) é [0, 3].
  - *a*(*t*) é definido na **extensão** [0, 1], *b*(*t*) na extensão [1, 2], e *c*(*t*) na extensão [2, 3].
- Pontos onde um par de segmentos se encontram são denominados **junções**.
- Os valores de *t* nas junções são chamados de **nós**.
  - Existem quatro nós neste exemplo: 0, 1, 2, e 3.



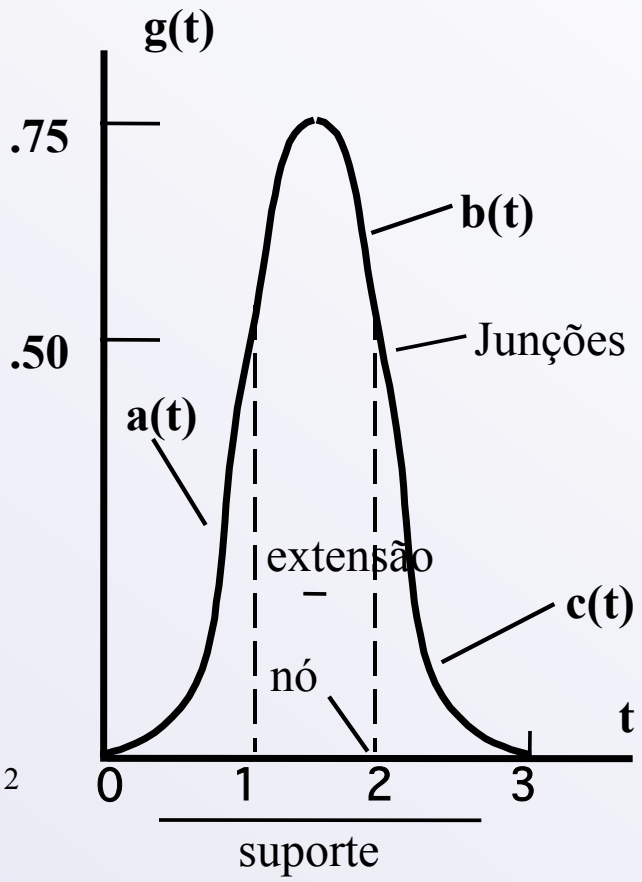
Funções polinomiais por partes

- $g(t)$ é contínua no seu suporte;
- Adicionalmente, $a(1) = b(1) = 1/2$, and $b(2) = c(2) = 1/2$.
- A derivada de $g(t)$ é contínua em todo o suporte: $g(t)$ é 1-suave em [0, 3].
  - A derivada é contínua dentro de cada segmento e $a'(1) = b'(1) = 1$, $b'(2) = c'(2) = -1$.

$$a(t) = \frac{1}{2}t^2$$

$$b(t) = \frac{3}{4} - (t - \frac{3}{2})^2$$

$$c(t) = \frac{1}{2}(3 - t)^2$$

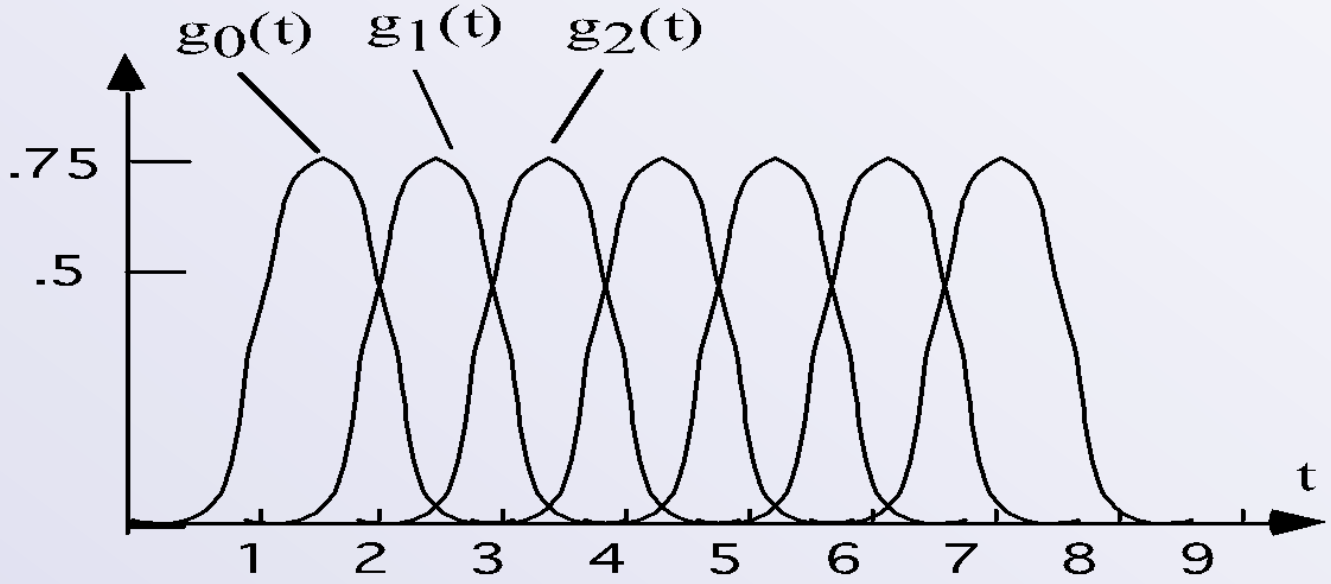


Splines

- A forma $g(t)$ é um exemplo de uma **função Spline**, uma função polinomial por partes que possui um grau de suavidade suficiente.
- **Definição:** Uma função spline de grau M é uma função polinomial por partes de grau M que é (M-1)-suave em cada nó.
  - O exemplo $g(t)$ é uma spline quadrática, pois possui grau 2 e sua primeira derivada é contínua em todo o intervalo.

Funções spline

- Usaremos versões transladas de $g(t)$, onde cada função de ponderação $g_k(t)$ é formada por uma translação a partir da origem. A figura mostra 7 funções $g_0(t),..,g_6(t)$ obtidas depois de transladar g(.) por valores inteiros: $g_k(t) = g(t-k)$ for k = 0, 1, 2, …



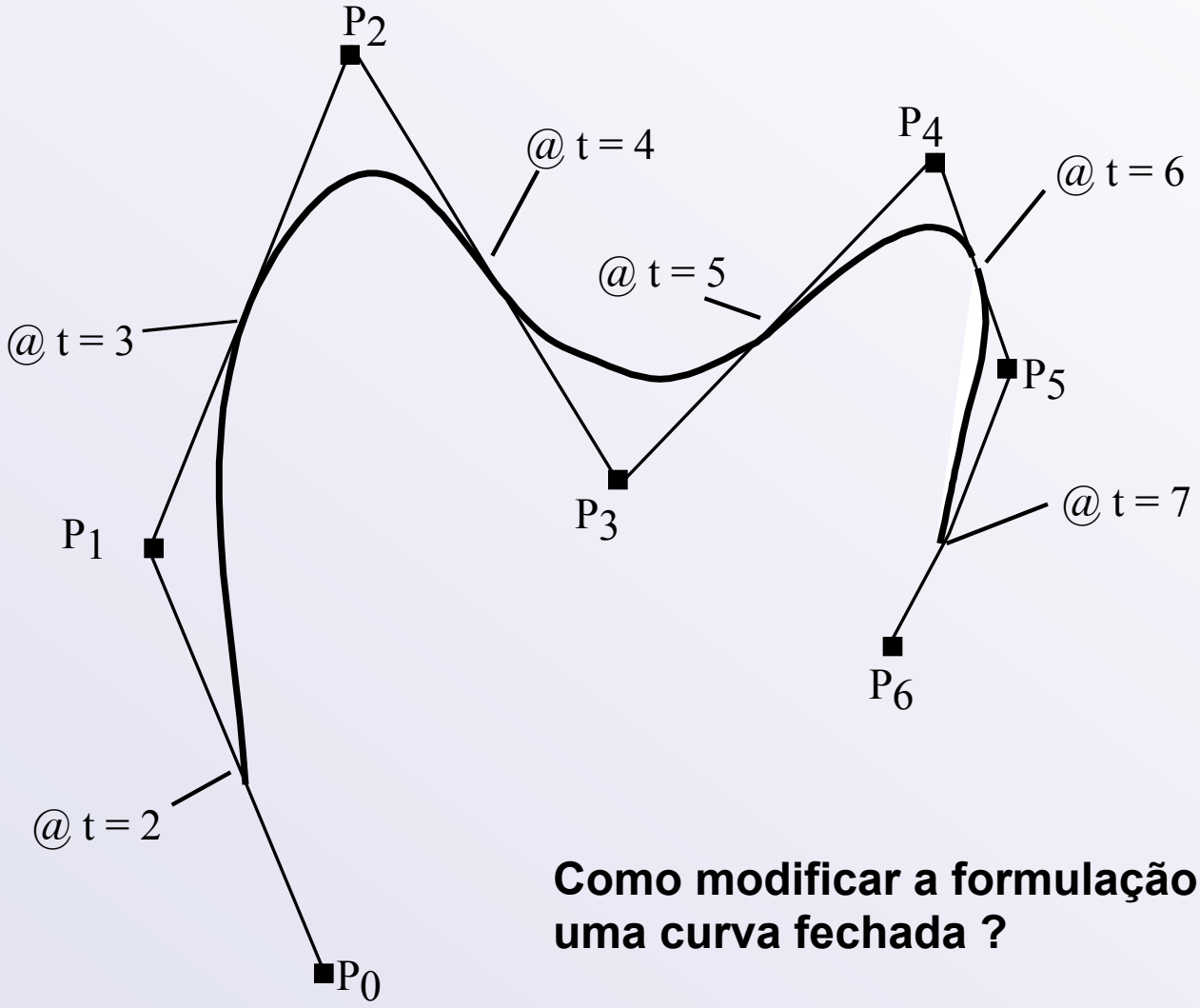
Funções spline

- O usuário escolhe 7 pontos de controle e gera a curva a partir da expressão

$$V(t) = \sum_{k=0}^{6} P_k \, g(t-k)$$

- Somente valores de *t* entre 2 e 7 poderão ser usados. Dentro deste intervalo, exatamente 3 funções de ponderação estarão ativas para cada valor de *t*, então obteremos um bom controle local.
- Quando *t* = 2, 3, .., e 7 somente duas destas funções estarão ativas e ambas terão valor 0.5. Desta forma, nestes valores de t, *V*(*t*) estará no ponto médio da linha conectando dois dos pontos de controle.

Curva spline resultante



Como modificar a formulação para criar uma curva fechada ?

Funções spline

- Já que os nós das várias versões de g(t) ocorrem em valores inteiros de t, os nós de cada curva se alinharão com os nós da próxima curva.
- Estas versões transladas formarão um conjunto legítimo de funções de ponderação somente se totalizarem 1 em cada valor de *t*. Porém isso somente será válido quando *t* estiver entre 2 e 7.

Propriedades da curva spline

- Possui **controle local**, já que o intervalo de suporte para cada função de ponderação tem tamanho 3.
- A curva **interpola pontos médios das arestas** entre pontos de controle; indicando uma propriedade geométrica.
- Já que cada função de ponderação é 1-suave, toda a **curva é 1-suave**.
- Nenhum ponto de controle é interpolado.
- Todos os polinômios possuem grau 2, facilitando a sua estabilidade e simplicidade de cálculo. **O grau dos polinômios não dependem dos pontos de controle**.
- Funciona para qualquer número de pontos de controle.

Demo Spline

- bspline0.cpp

Funções de ponderação genéricas

- Poderíamos desejar maior controle:
  - A curva poderia entortar mais e ter maior suavidade (>1). Isso sugere a mudança para polinomiais cúbicas.
- Desejamos que o usuário possa especificar quais pontos de controle são interpolados.
- Também gostaríamos que um único algoritmo que englobasse todas as técnicas de design descritas até o momento — incluindo as curvas de Bézier:
  - Precisamos desenvolver famílias mais genéricas de funções de ponderação que reinforçem todas as propriedades discutidas até o momento.

Funções spline genéricas

- Continuaremos a usar a mesma forma paramétrica

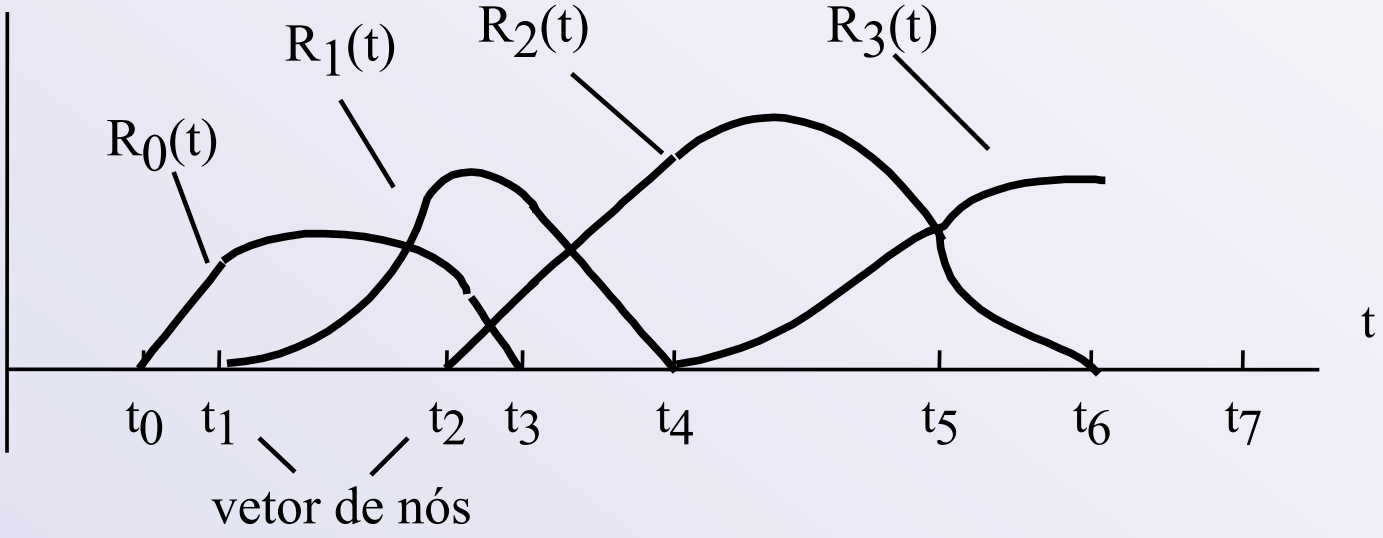
$$P(t) = \sum_{k=0}^{L} \mathrm{P}_k R_k(t)$$

  baseada em L+1 pontos de controle e L+1 funções de ponderação $R_0(t)$, ..., $R_L(t)$.

- Usaremos polinômios por partes para as funções de ponderação, mas eles agora serão definidos por uma seqüência de nós mais genéricos, chamado de **vetor de nós,** $T = [t_0, t_1, t_2, \ldots]$, com $t_i \leq t_{i+1}$.
  - Alguns dos nós podem ter o mesmo valor, mas serão dados nomes distintos.

Funções spline genéricas

- Cada função de ponderação $R_k(t)$ é um polinomial por partes que é zero até o tempo t_k, e é não zero em uma certa extensão do vetor de nós, e então retorna a zero novamente.
- Cada $R_k(t)$ deve ser uma função spline, assegurando um nível de suavidade em todo t dentro de seu suporte.



Funções de base spline

- Dado um vetor de nós, existirá uma família de funções de ponderação que permitam gerar quaisquer curvas splines possíveis definidas por aquele vetor de nós ?
- Tal família é chamada de **base** para as splines, o que significa que qualquer curva spline pode ser alcançada através da soma

$$P(t) = \sum_{k=0}^{L} P_k R_k(t)$$

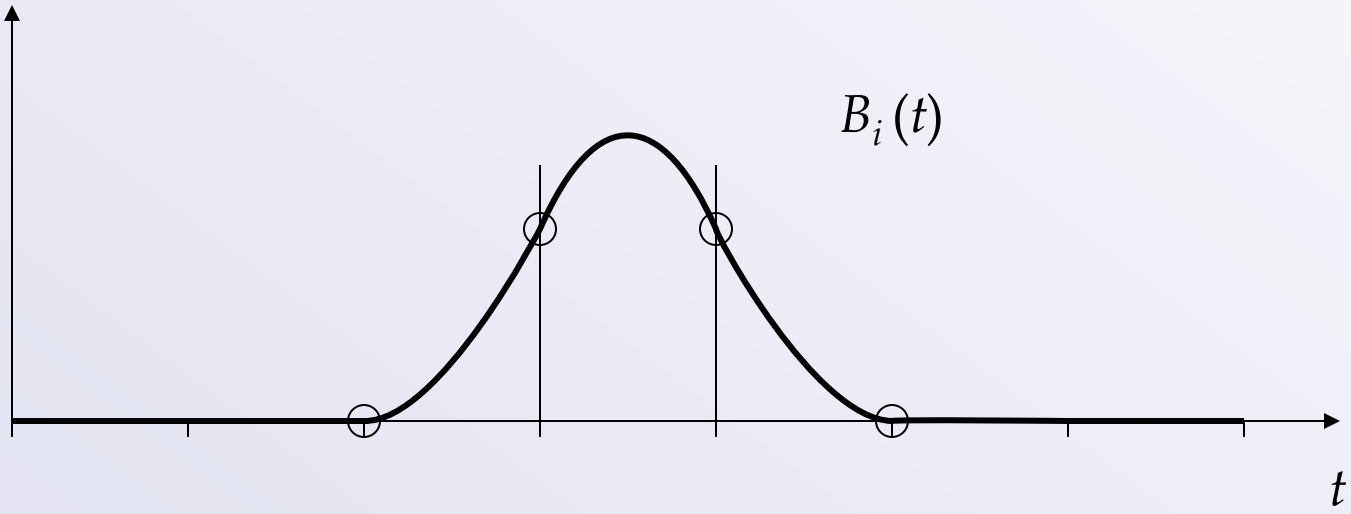
escolhendo-se os pontos de controle apropriados.

Funções B-spline

- 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}

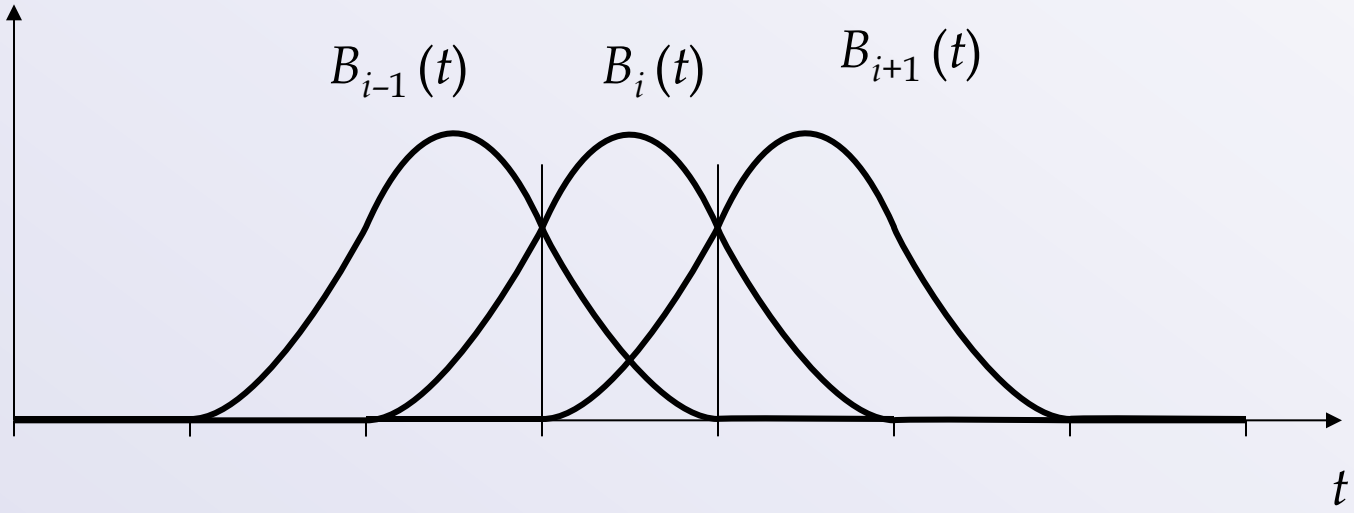
Funções B-spline

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



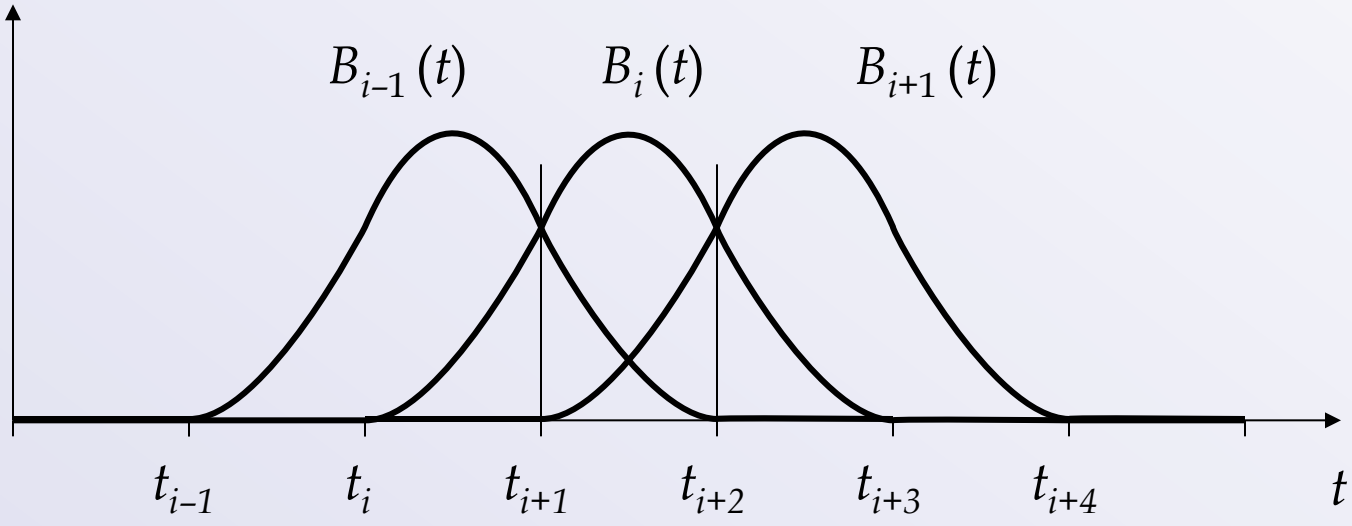
Funções B-spline

- 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



Funções B-spline

- Os valores t_i do espaço de parâmetro que delimitam os intervalos são chamados de *nós.*
- Podemos pensar em intervalos regulares por enquanto (B-splines uniformes).



Funções de Base Spline

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

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

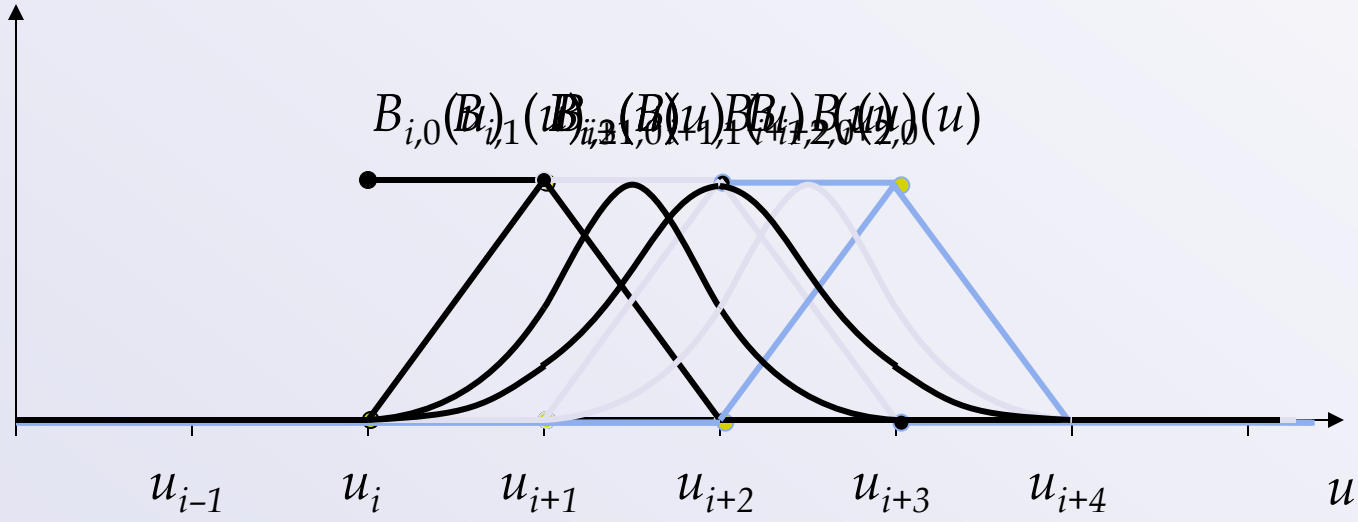
  onde *m* é o número de pontos do polígono de controle e *d* é o grau da B-spline que se quer usar

- Para derivar as funções da base B-spline pode-se resolver um sistema de equações
  - Para B-splines cúbicas, requere-se continuidade C^2 nos nós, a propriedade do fecho convexo, etc
- Uma maneira mais natural é utilizar a recorrência 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

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

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



Carl De Boor and M.G. Cox, 1972.

Recorrência Cox-De Boor

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

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

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

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

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

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

$\mathbf{p}_i$

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

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

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

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

$d = 0$

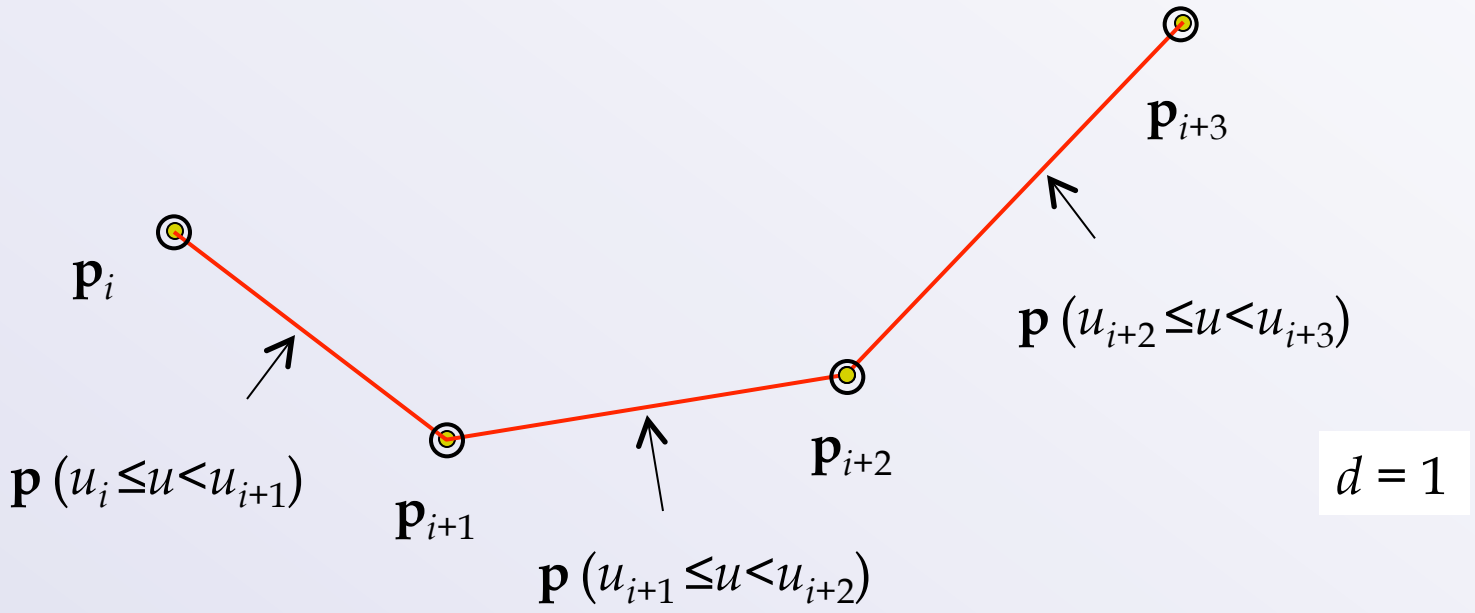
(assumir que para $u = u_i$
Spline de grau 0 passa por $\mathbf{p}_i$)

Recorrência Cox-De Boor

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

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

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



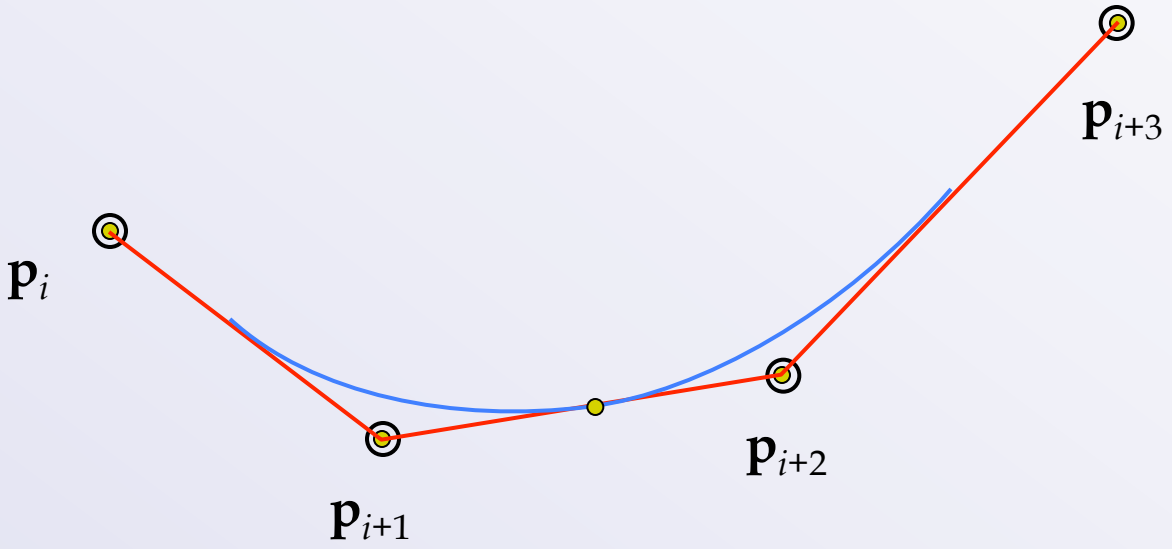
$d = 1$

Recorrência Cox-De Boor

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

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

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



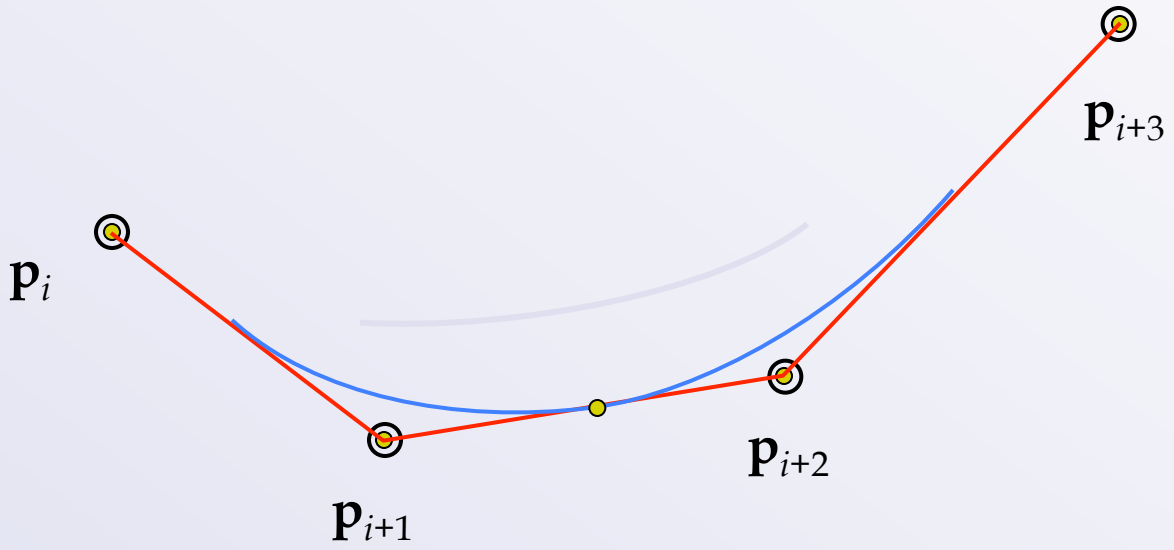
$d = 2$

Recorrência Cox-De Boor

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

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

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



$d = 3$

Propriedades das B-Splines

- Dados n+1 pontos ($\mathbf{p}_0$ ... $\mathbf{p}_n$) , 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 , ... , 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

Demo: Efeito dos nós

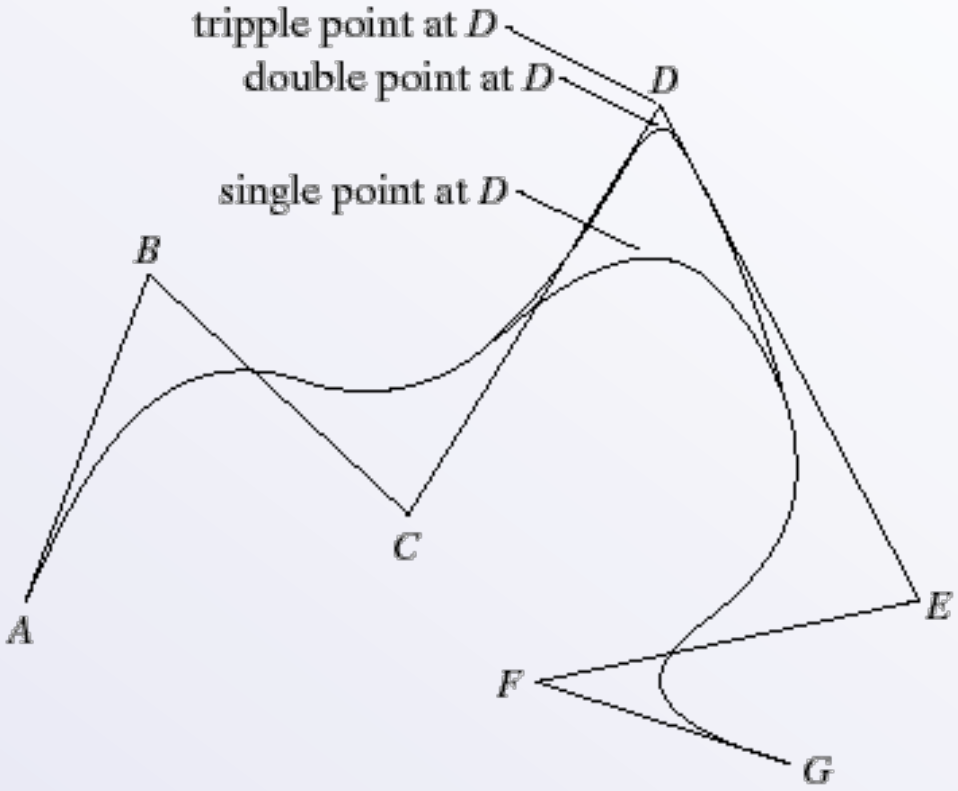
http://www.ibiblio.org/e-notes/Splines/Basis.htm

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 contendo 4 pontos de controle e interpolar o primeiro e último pontos, podemos usar o vetor de nós: 0, 0, 0, 0, 1, 1, 1, 1

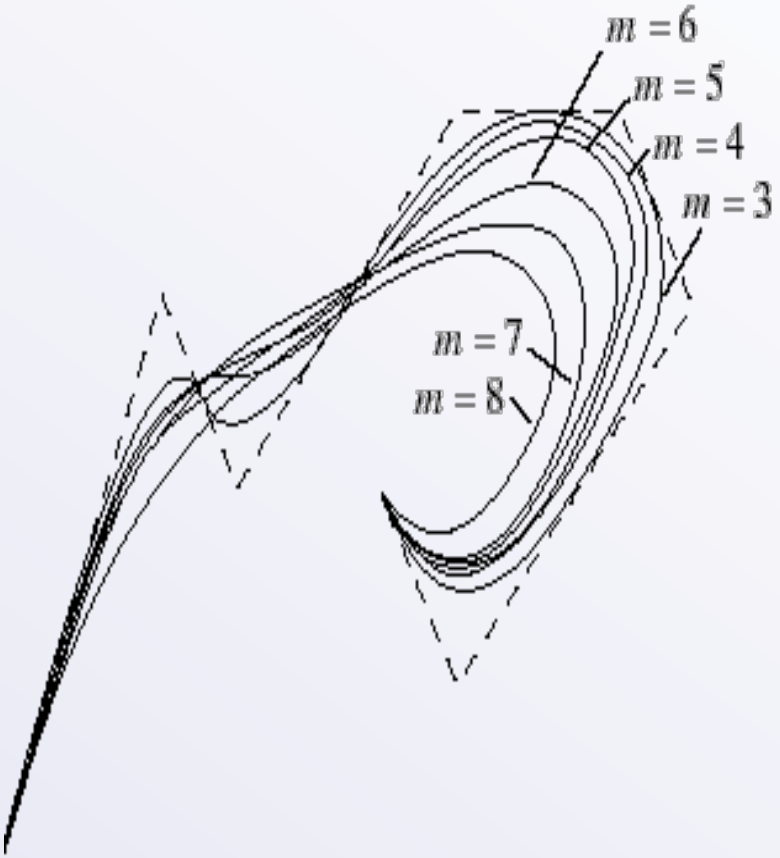
Efeitos dos nós

- Com B-spline cúbicas, a curva baseada nos pontros de controle *A*, *B*, *C*, *D*, *E*, *F*, e *G* é obtida.
- Quando um ponto duplo é usado em *D*, (o polígono de controle se torna *A*, *B*, *C*, *D*, *D*, *E*, *F*, *G)*, a curva é puxada para *D*.
- Quando um ponto triplo é colocado em *D*, (polígono *A*, *B*, *C*, *D*, *D*, *D*, *E*, *F*, *G*) a curva deve interpolar D.



B-splines e curvas Bézier

- As curvas B-spline dão controle local da forma desejada.
- Quando a ordem dos polinomiais B-spline aumenta em 1, o suporte aumenta, reduzindo o grau de controle local.
- Quando *m* alcança o limite *L*+1, a curva Bézier é obtida, e o controle local se torna mínimo.



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!

Demo Bézier em OpenGL

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
  - Podem representar perfeitamente seções cônicas tais como círculos, elipses, etc

Exercício

- Como transformar uma curva Bézier em uma Bspline (e vice-versa) ?