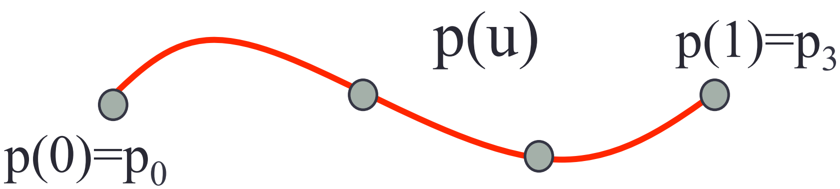
MAC420/5744: Introdução à Computação Gráfica

Marcel P. Jackowski
mjack@ime.usp.br

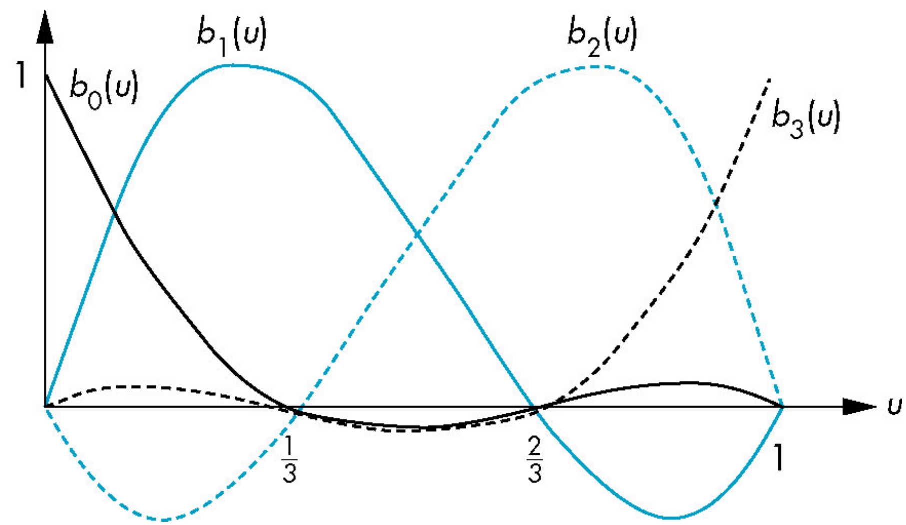
Aula #17: Curvas Splines

Curva polinomial cúbica

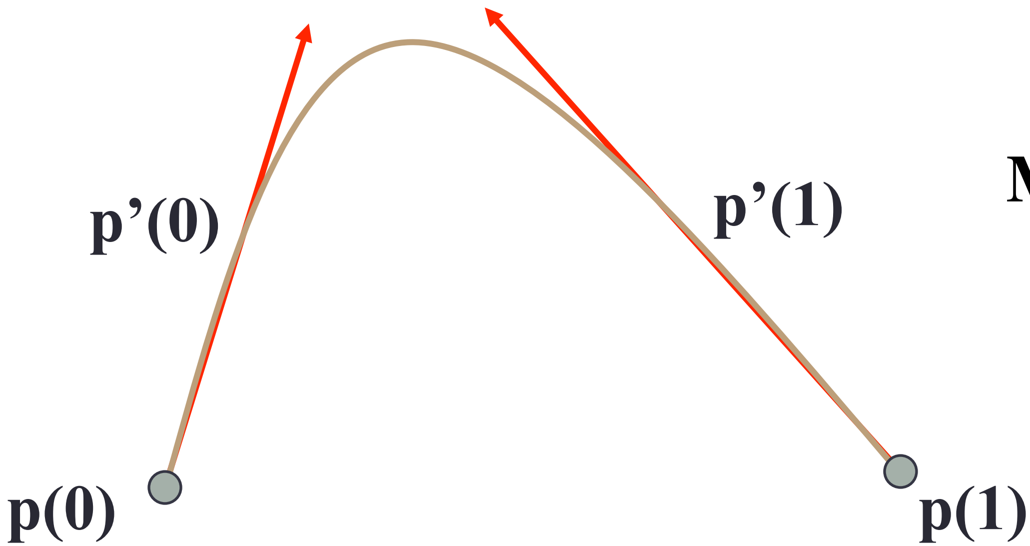


$$\mathbf{M}_I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -5.5 & 9 & -4.5 & 1 \\ 9 & -22.5 & 18 & -4.5 \\ -4.5 & 13.5 & -13.5 & 4.5 \end{bmatrix}$$

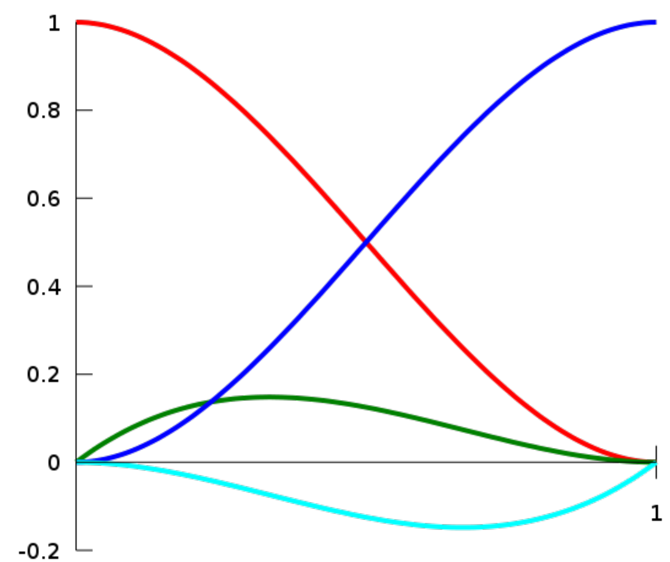
$$\mathrm{p(u)}=\mathbf{u}^{\mathrm{T}}\mathbf{c}=\mathbf{u}^{\mathrm{T}}\mathbf{M}_I\mathbf{p} = \mathbf{b(u)}^{\mathrm{T}}\mathbf{p}$$



Curvas de Hermite

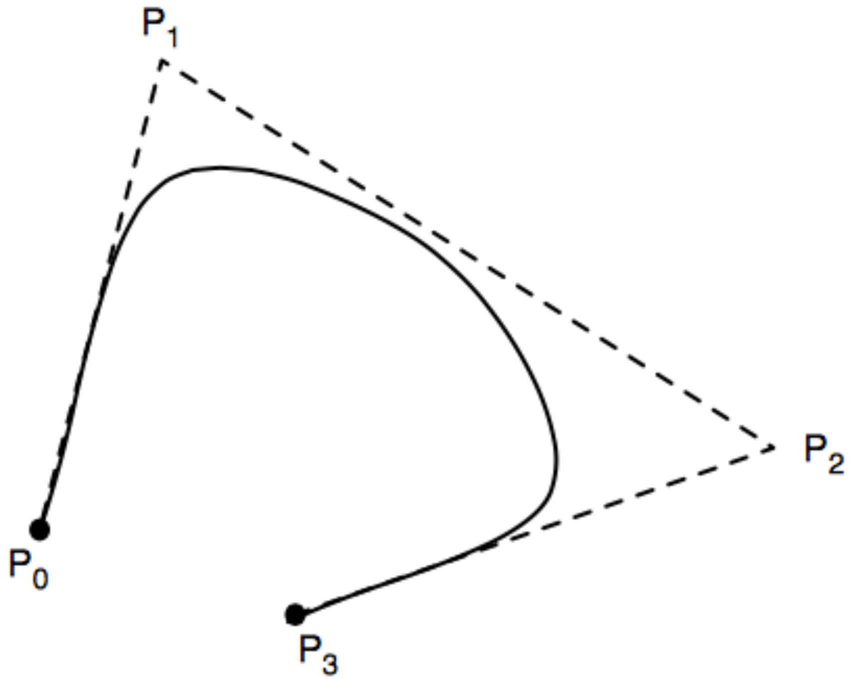


$$\mathbf{M}_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$



$$p(u)=\mathbf{u}^T\mathbf{c}=\mathbf{u}^T\mathbf{M}_H\mathbf{p} = \mathbf{b}_H(\mathbf{u})^T\mathbf{p}$$

Curvas Bézier



- Four **control points**: $\{P_0, P_1, P_2, P_3\}$
- Interpolates endpoints: P_0, P_3
- Tangent vectors at endpoints:
  $R_0 = 3(P_1 - P_0)$, $R_1 = 3(P_3 - P_2)$

Equações

As condições de interpolação permanecem as as mesmas nos dois extremos da curva

$$P(0) = p_0 = c_0$$
$$P(1) = p_3 = c_0+c_1+c_2+c_3$$

Diferenciando $\mathbf{P}$ encontramos $P'(u) = c_1+2uc_2+3u^2c_3$

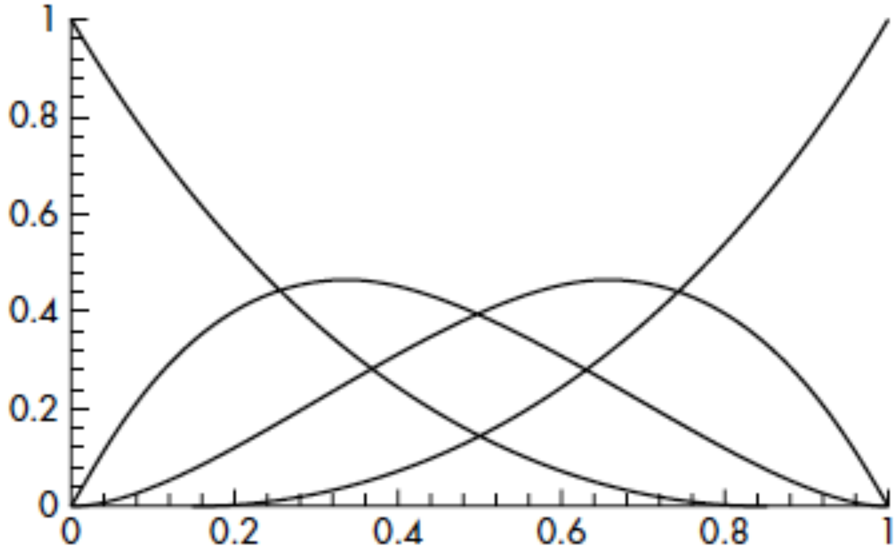
Avaliando nos pontos extremos:

$$P'(0) = 3(p_1-p_0) = c_1$$
$$P'(1) = 3(p_3-p_2) = c_1+2c_2+3c_3$$

Matriz de geometria

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

$$p(u)=\mathbf{u}^T\mathbf{c}=\mathbf{u}^T\mathbf{M}_B\mathbf{p} = \mathbf{b}_B(\mathbf{u})^T\mathbf{p}$$



Curvas Bézier

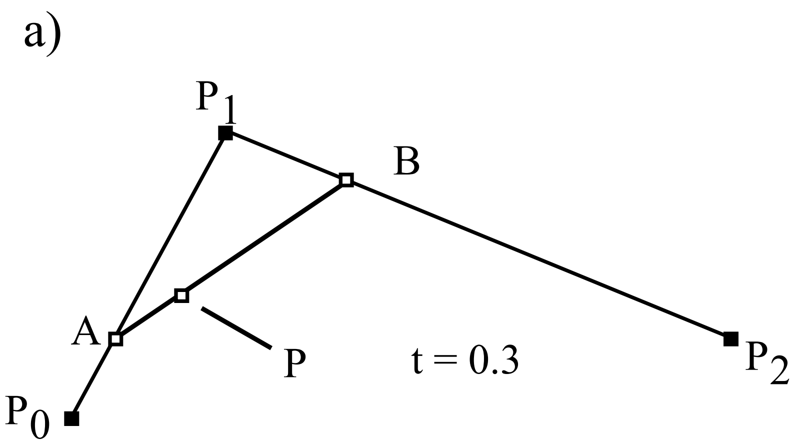
- As curvas de Bézier foram inventadas para auxiliar no design de automóveis
  - O algoritmo "de Casteljau"
- O algoritmo de De Casteljau baseia-se em uma seqüência de passos de transformação geométrica de fácil implementação
- Através desta transformação, é possível deduzir uma série de propriedades das que curvas que ela gera

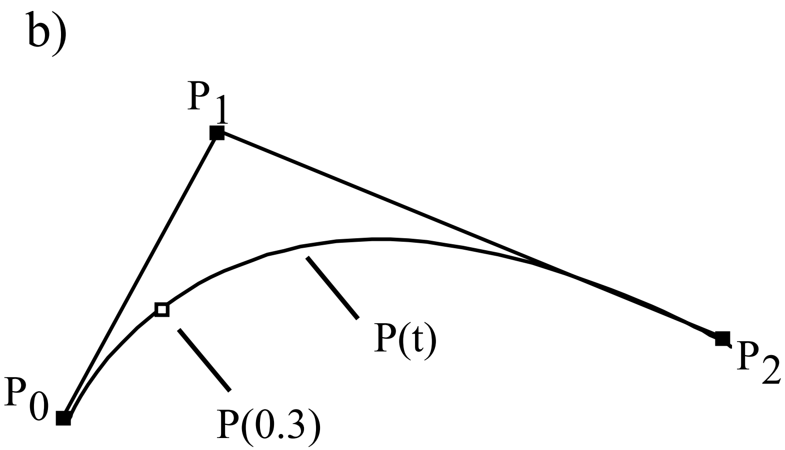
Curvas Bézier (ii)

- Transformação de 3 pontos para obtenção de uma parábola:
  - Escolha três pontos: P_0, P_1, and P_2
  - Escolha um valor de t entre 0 and 1, ex. $t = 0.3$
  - Localize o ponto A que está a uma fração t ao longo da linha de P_0 to P_1. Analogamente, localize B a mesma fração t entre os pontos P_1 e P_2.
  - Os novos pontos serão:
    - $A(t) = (1\text{-}t)P_0 + tP_1$
    - $B(t) = (1\text{-}t)P_1 + tP_2$

Curvas Bézier (iii)

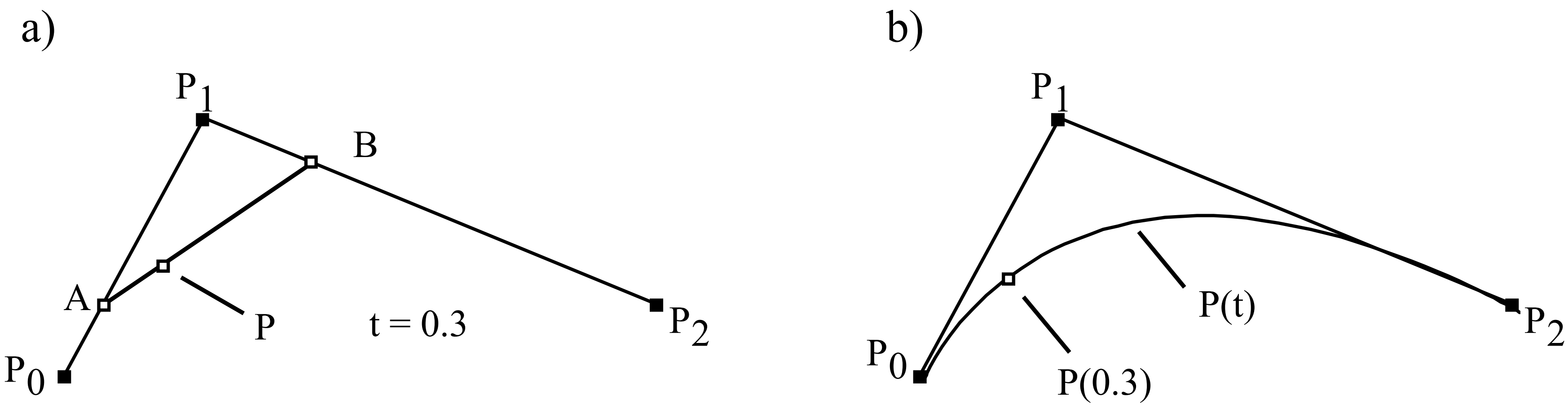
- Agora repita a interpolação linear nestes pontos (usando o mesmo valor de *t*)
- Ache o ponto, *P*(*t*), que está na fração *t* do caminho entre A e B: $P(t) = (1-t)A + tB$.





Curvas Bézier (iv)

- Se este processo for repetido para *todo t* entre 0 e 1, a curva *P*(*t*) será gerada.
- A forma paramétrica resultante para tal curva será $P(t) = (1-t)^2P_0 + 2t(1-t)P_1 + t^2P_2$

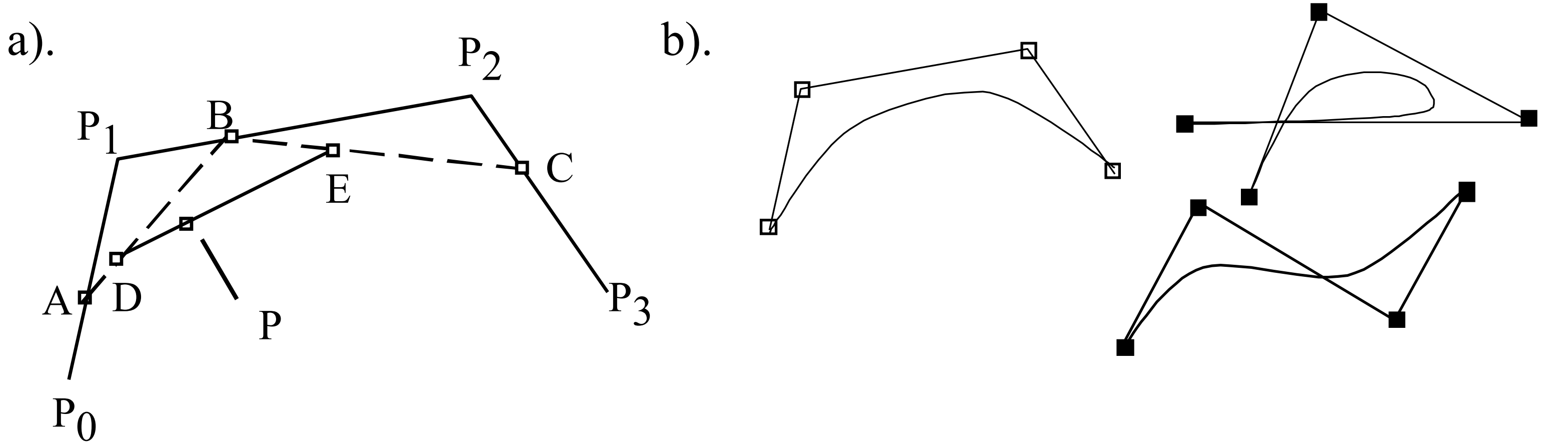


Curvas Bézier (v)

- A forma paramétrica para $P(t)$ é quadrática em t, então concluimos que tal curva é uma parábola
- Ela continuará sendo uma parábola mesmo quando t variar entre $-\infty$ to ∞.
- Ela passará em P_0 quando $t = 0$ e em P_2 quando $t = 1$
- Assim obtemos um processo bem-definido que gera uma curva parabólica suave baseada em três pontos de controle.

Curvas Bézier cúbicas

- Para um dado valor de *t*, o ponto *A* é posicionado a uma fração *t* entre P_0 e P_1, e similarmente para *B* e *C*.
- Então *D* é colocado a uma fração *t* do caminho entre *A* e *B*, e similarmente para o ponto *E*.
- Finalmente, o ponto desejado *P* está localizado a uma fração *t* do caminho entre *D* e *E*.

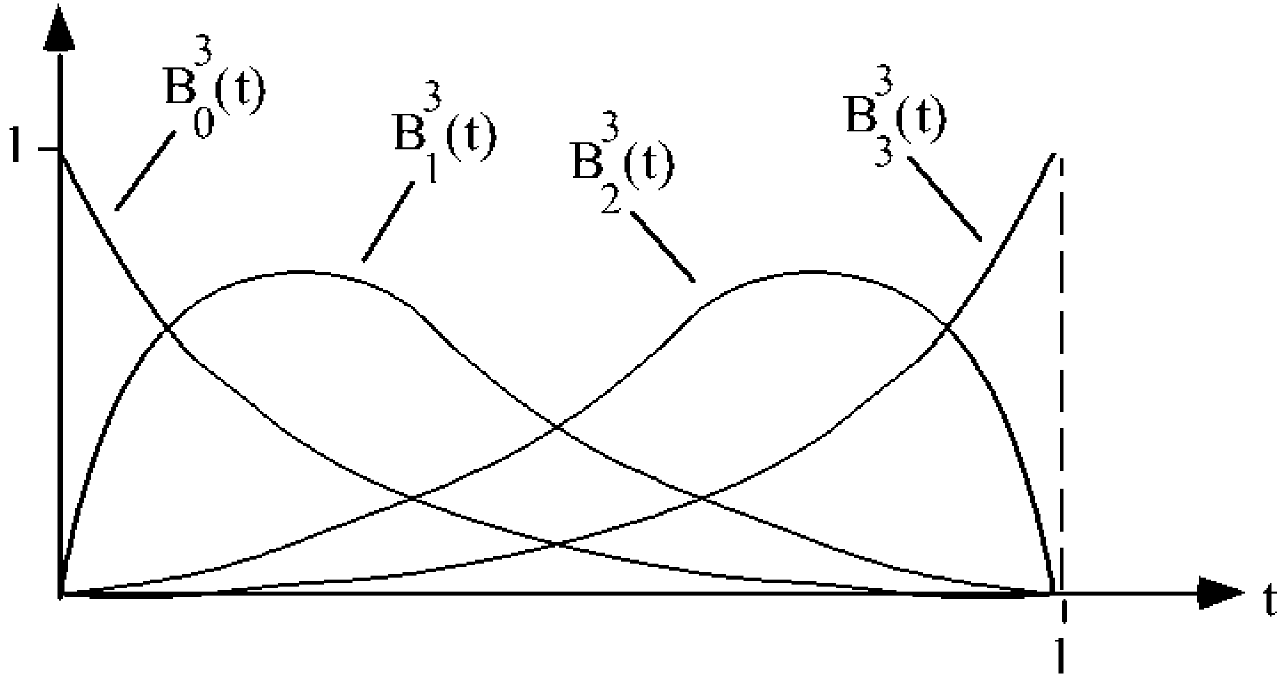


Curvas Bézier cúbica

- A curva Bézier baseada em quatro pontos de controle possui a forma paramétrica
  - $P(t) = P_0(1\text{-}t)^3 + P_1 3(1\text{-}t)^2 t + P_2 3(1\text{-}t)t^2 + P_3 t^3$.
- Cada ponto de controle P_i é pesado por um polinômio cúbico, e os termos são somados.
- Estes termos são chamados de *polinômios de Bernstein*:

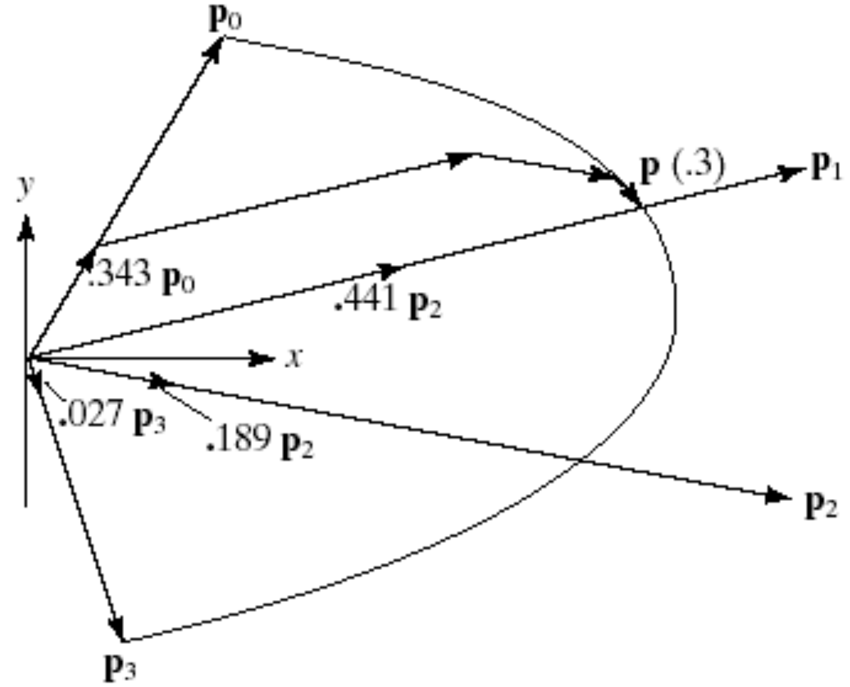
$$B_0^3 = (1-t)^3 \qquad B_1^3 = 3(1-t)^2 t \qquad B_2^3 = 3(1-t)t^2 \qquad B_3^3 = t^3$$

Polinômios de Bernstein



Ponderação com Bernstein

- Considere pontos como vetores na origem (ex., P_0 e t = 0.3)
- Então $\mathbf{p}(0.3) = 0.343\ \mathbf{p_0} + 0.441\ \mathbf{p_1} + 0.189\ \mathbf{p_2} + 0.027\ \mathbf{p_3}$
- Nesta figura os quatro vetores são modulados e os resultados são adicionados para formar o vetor **p**(0.3).



Generalização das curvas de 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:

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

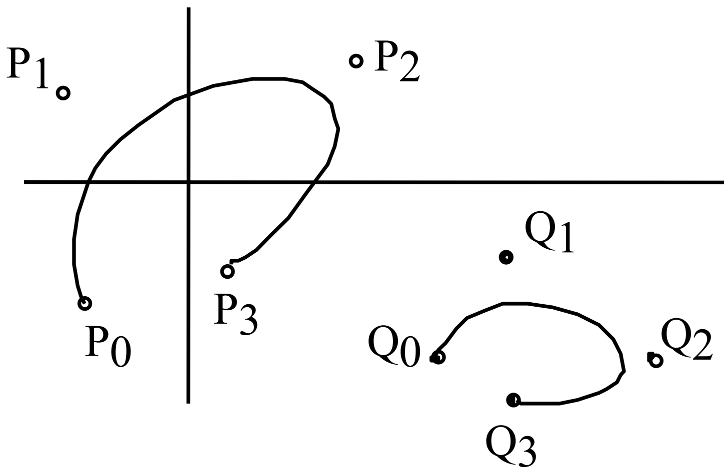
Propriedades das curvas Bézier

- As propriedades das curvas de Bézier fazem com que elas sejam perfeitas para o uso em CAD
  - <u>Interpolação dos pontos finais:</u> A curva Bézier $P(t)$ baseada nos pontos de controle $P_0, P_1, \ldots, P_L$ sempre interpola os pontos P_0 e P_L.
  - <u>Invariância afim:</u> para aplicar uma transformação afim T em todos os pontos $P(t)$ da curva, transformamos os pontos de controle uma vez, e usamos os novos pontos para recriar a curva transformada Q(t) em qualquer *t*.

$$Q(t) = \sum_{k=0}^{L} T(P_k) B_k^L(t) = T\left(\sum_{k=0}^{L} P_k B_k^L(t)\right)$$

Propriedades das curvas Bézier

- Exemplo: Uma curva Bezier é baseada em quatro pontos de controle $P_0,\ldots, P_3$. Os pontos são rotacionados, escalados e transladados para as novas posições Q_k.
- A curva Bezier resultante para Q_k é desenhada. Ela é idêntica ao resultado da transformação da curva Bezier original.



Propriedades das curvas Bézier

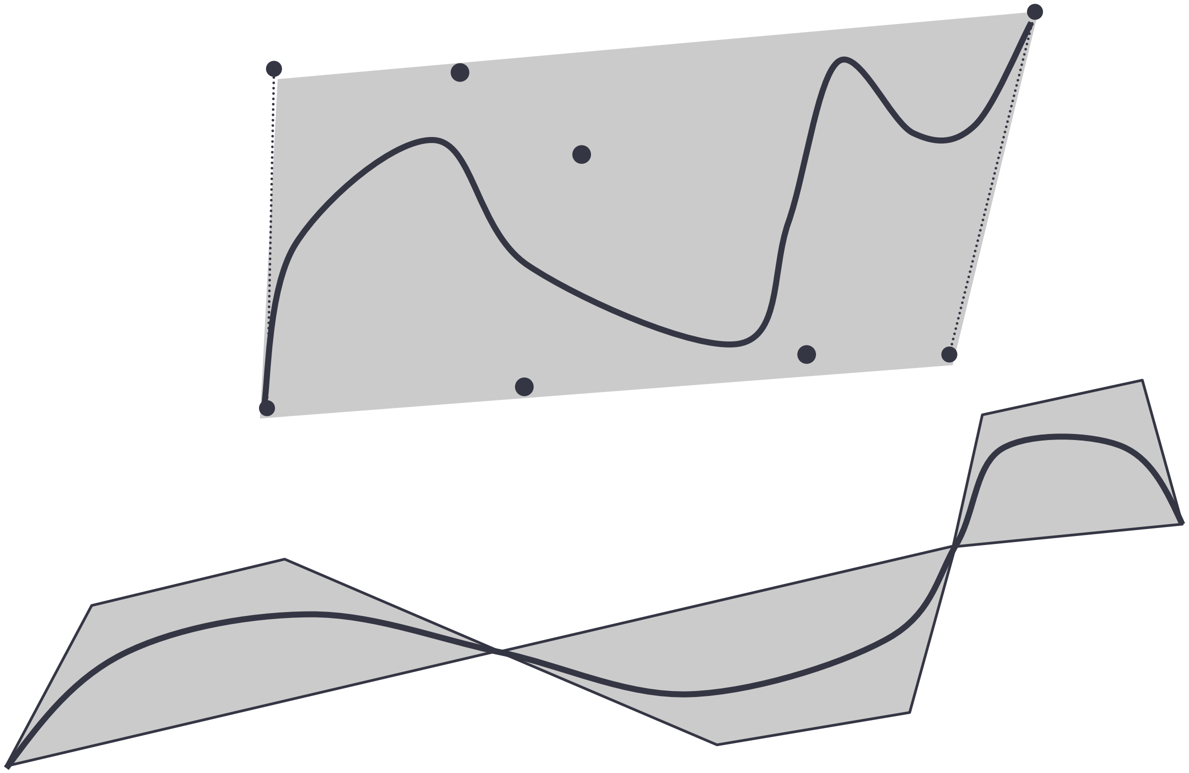
- Uma curva Bézier *P*(*t*), nunca deixa o seu fecho convexo
- O fecho convexo do conjunto de pontos V_0, V_1,... ., V_i é o conjunto de todas as suas *combinações convexas*; isto é, o conjunto de todos os pontos dados por

$$\boxed{P = \sum_{i=0}^{n} \alpha_i \vec{V}_i \quad com \quad \sum_{i=0}^{n} \alpha_i = 1}$$

onde cada α_i é positivo, e a soma é igual 1.

Fecho convexo

$$P = \sum_{i=0}^{n} \alpha_i \vec{V}_i \quad com \quad \sum_{i=0}^{n} \alpha_i = 1$$

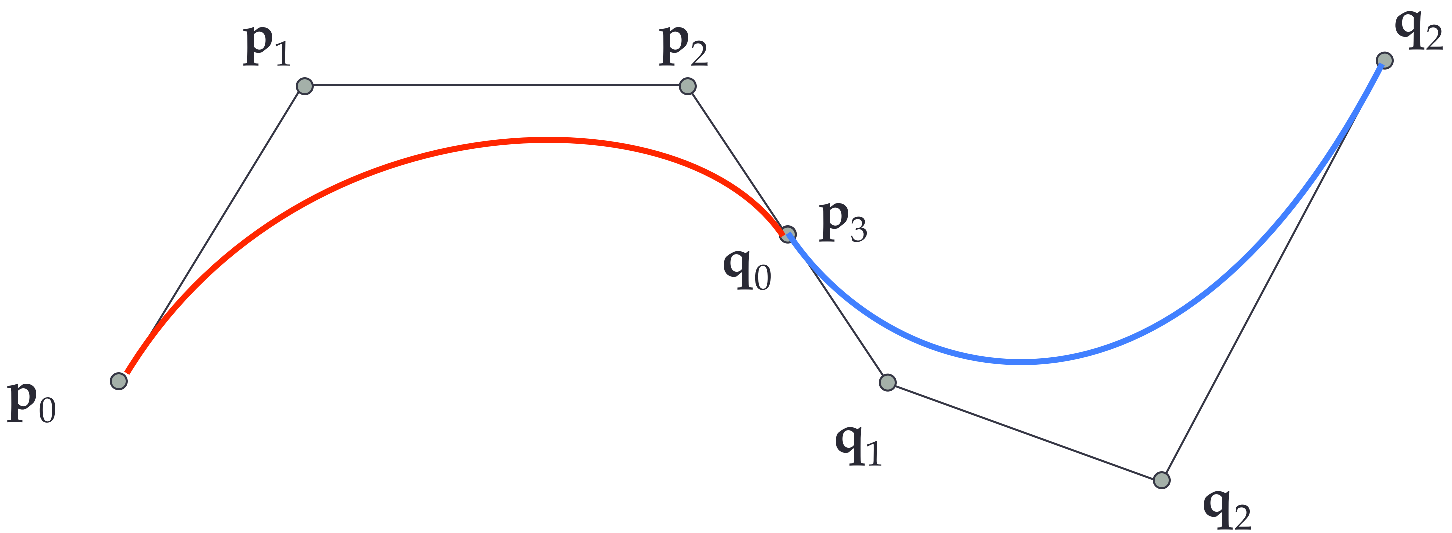


Desenhando curvas Bézier

- Curva normalmente é aproximada por uma linha poligonal
- Pontos podem ser obtidos avaliando a curva em t = $t_1, t_2 \ldots t_k$
  - 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"

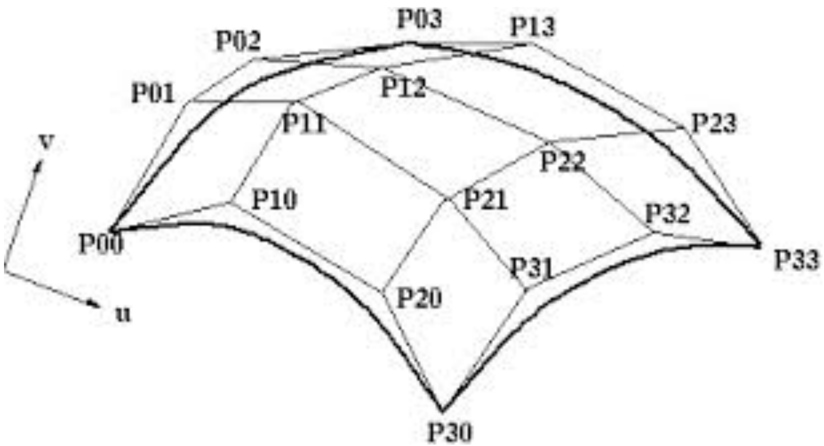
Emendando curvas Bézier

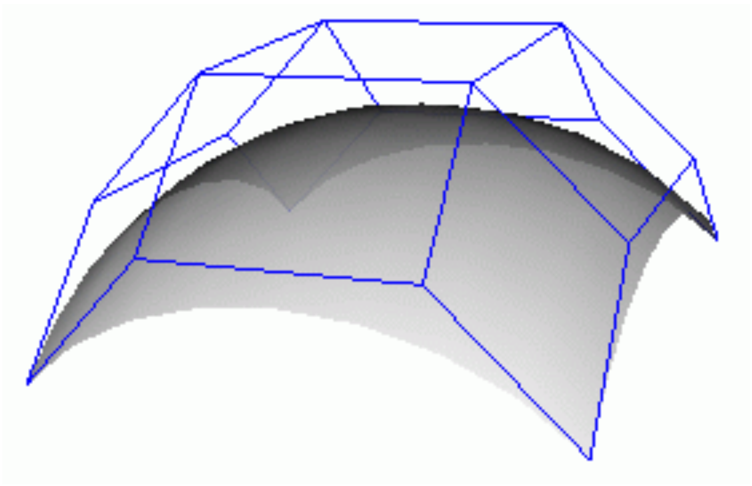
- Continuidade C^0: $\mathbf{p}_3=\mathbf{q}_0$
- Continuidade C^1: C^0 e segmento $\mathbf{p}_2\mathbf{p}_3$ com mesma direção e comprimento que o segmento $\mathbf{q}_0\mathbf{q}_1$
- Continuidade C^2: C^1 mais restrições sobre pontos $\mathbf{p}_1$ e $\mathbf{q}_2$



Superfície Bézier

$$p(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_i^n(u) B_j^m(v) p_{i,j}$$





Polinômios de Bernstein

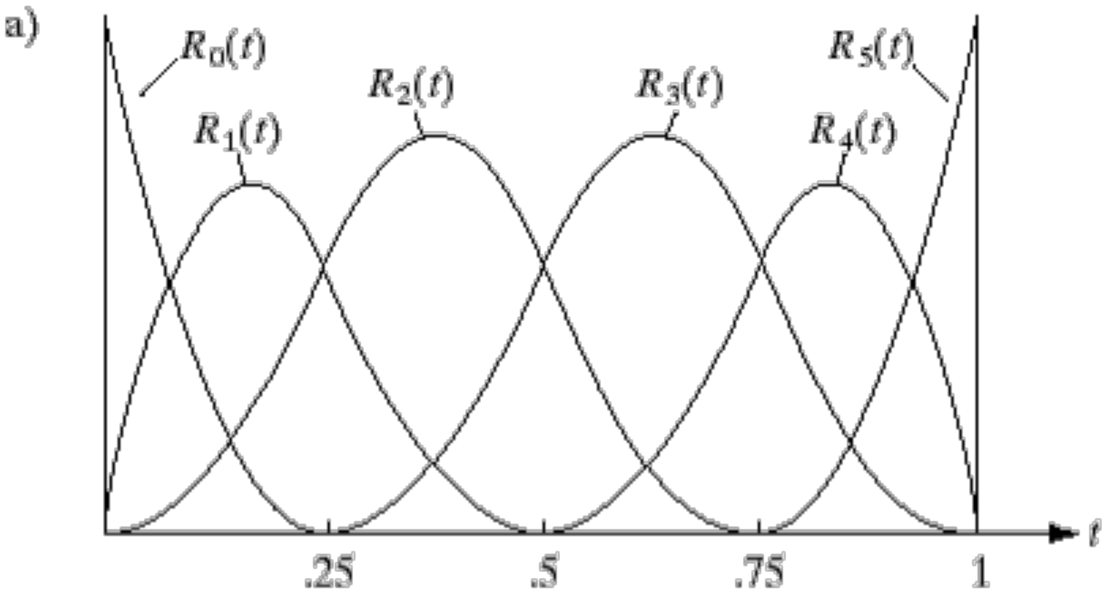
- A mudança de qualquer ponto de controle altera a curva inteira
- Cada polinômio de Bernstein é "ativo" (não-zero) dentro de todo intervalo [0, 1]
- O intervalo no qual uma função é não-zero é denominado **suporte**
  - Já que cada polinômio de Bernstein possui suporte em todo o intervalo [0, 1], e a curva é uma ponderação destas funções, cada ponto de controle tem um efeito na curva em todos os valores de t entre 0 e 1.
- Assim, o ajuste de qualquer ponto de controle afeta a curva globalmente, sem oferecer o controle local desejado.

Curvas longas

- Curvas Bézier com L+1 pontos são de grau L
  - Curvas de grau alto são difíceis de desenhar
  - 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

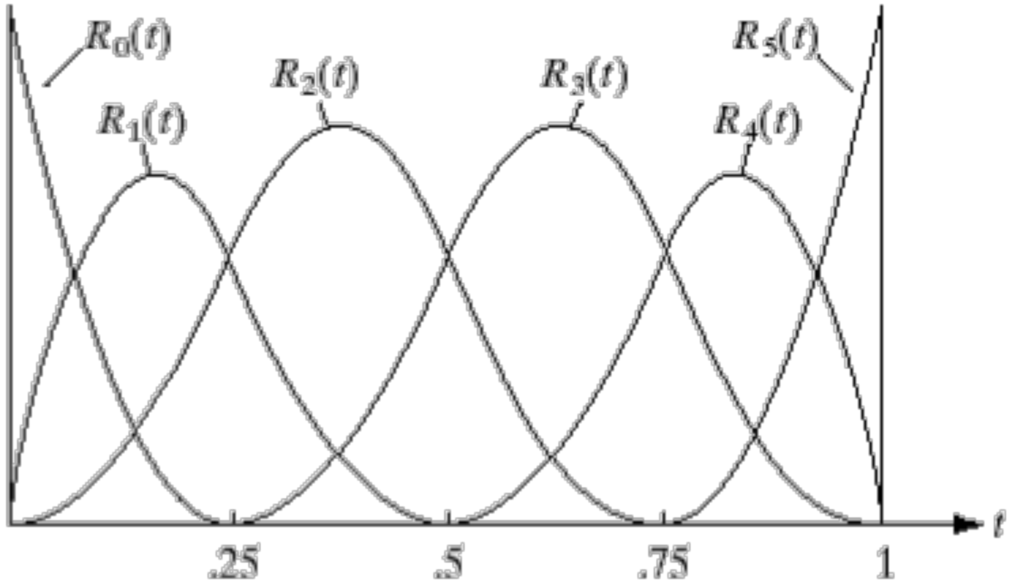
Funções de ponderação

- Agora, contraste o conjunto de funções de ponderação ao lado.
- Estas seis funções, $R_0(t), R_1(t), \ldots, R_5(t)$ 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].

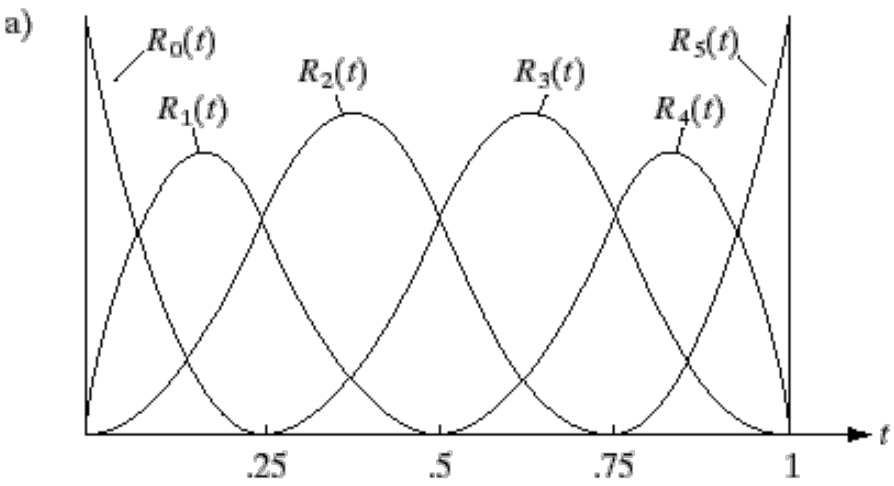


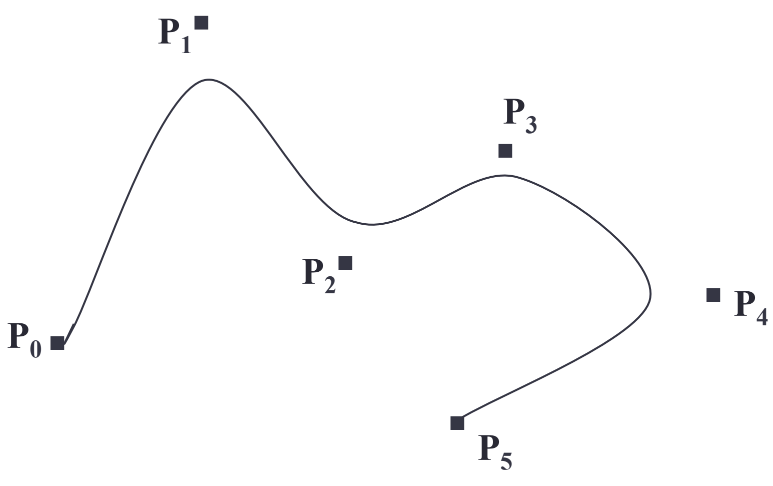
Para qualquer valor de *t*, não mais que três funções de ponderação estão ativas.

Funções de ponderação

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

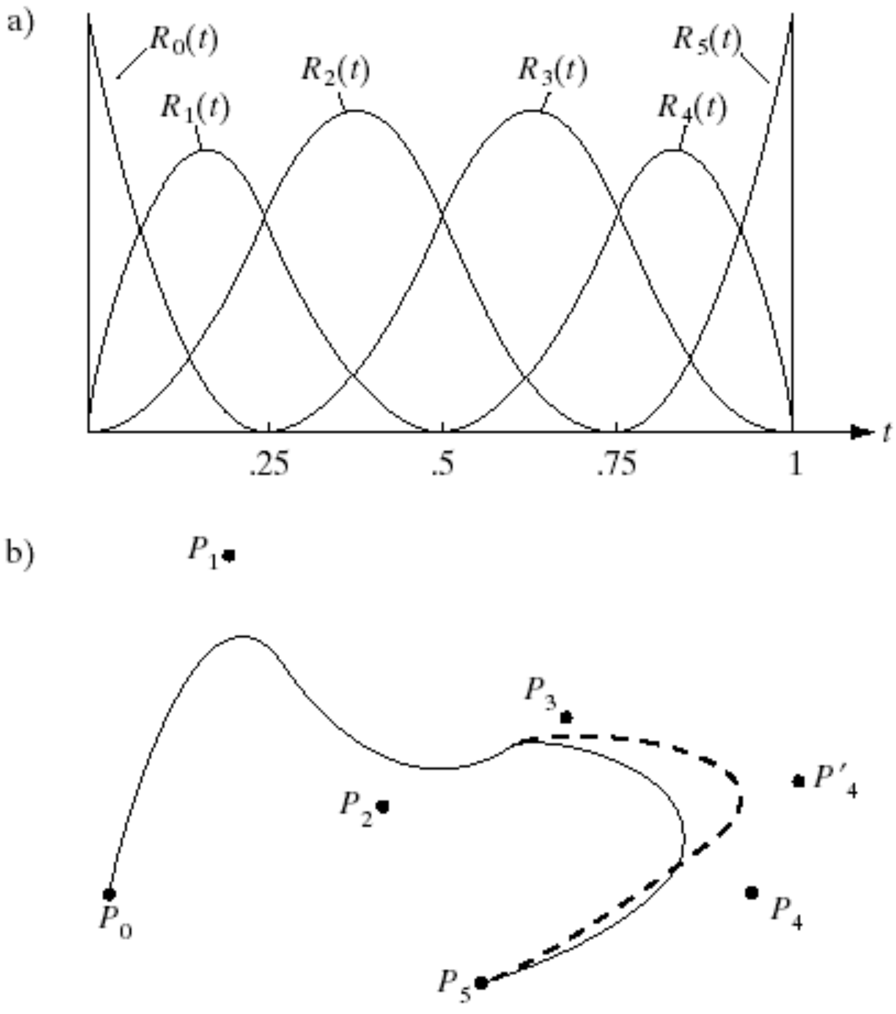
$$V(t) = \sum_{k=0}^{5} P_k R_k(t)$$





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.



Requisitos: 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 devem ser pequenos
  - Funções como *sen()* e *cos()* devem ser evitadas, pois são dispendiosos computacionalmente.

Requisitos: unidade

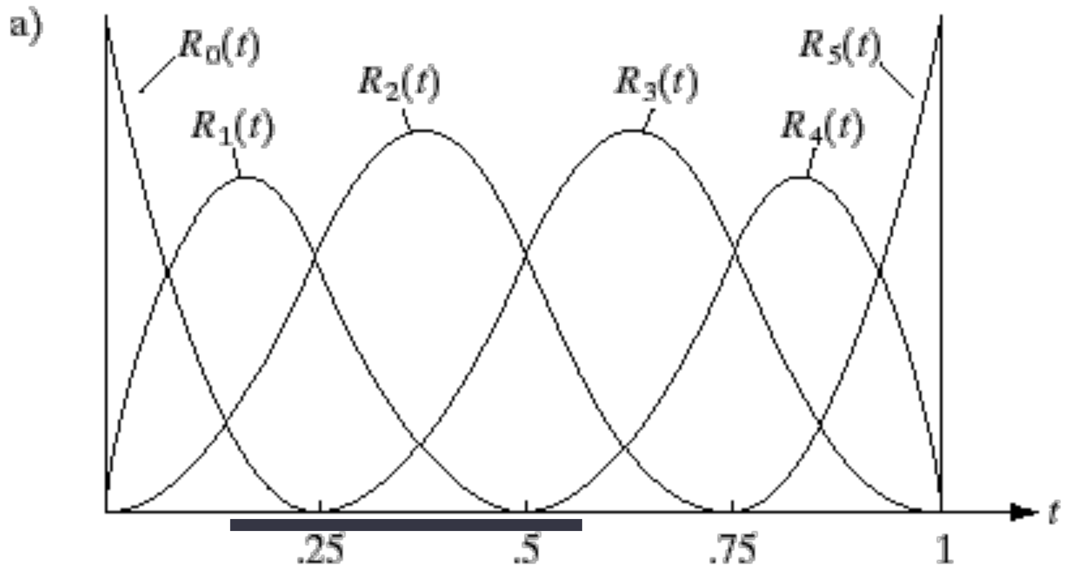
- $V(t)$ deve ser uma soma ponderada dos pontos em cada t dentro do intervalo $[a, b]$:

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

- As funções de ponderação devem totalizar 1 para qualquer valor de t.

Requisitos: suporte limitado

- Para conguirmos controle local da curva, devemos ter cada função de ponderação com suporte em uma faixa pequena do intervalo [*a*, *b*].



Requisitos: interpolação

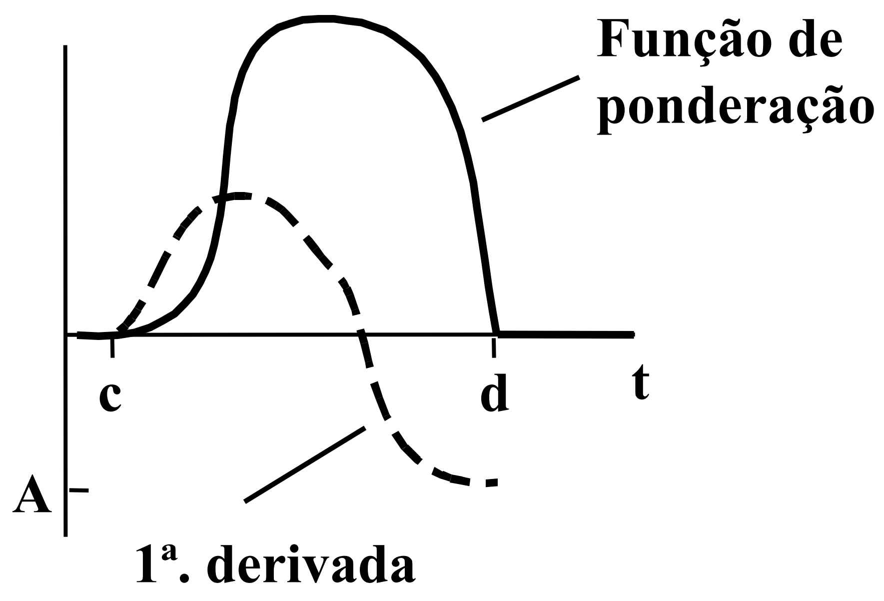
- O usuário pode desejar que $V(t)$ passe em alguns dos pontos de controle, ou ser somente atraída por outros pontos.
- Necessitaremos de um mecanismo para ajustar as funções de ponderação para que certos pontos de controle sejam interpolados.
- Como interpolar outros pontos de controle além do primeiro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

Requisitos: 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

Suavidade

- 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.



Curvas polinomiais por partes

- Não é fácil achar uma função polinomial que atenda todos os requisitos e que seja ao mesmo tempo maleável.
- Idéia: 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 por partes ou "piecewise"*.

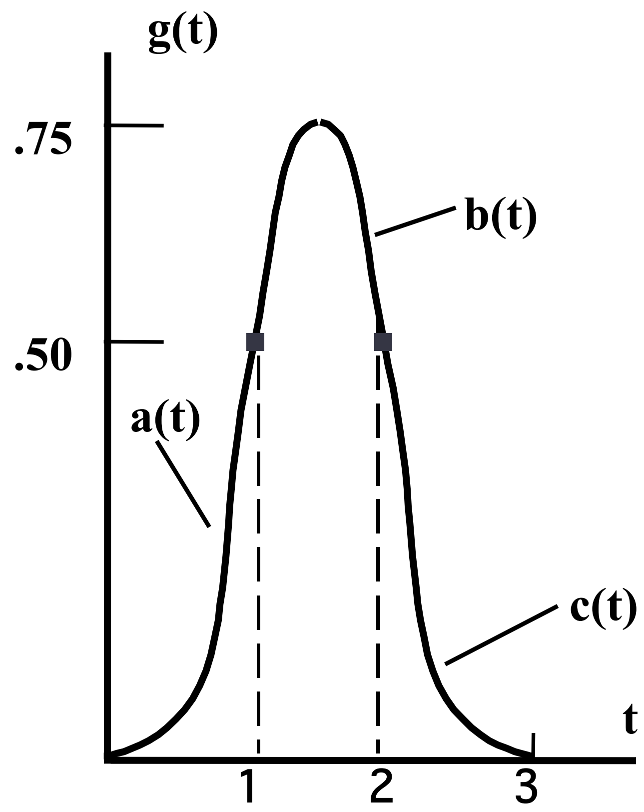
Funções polinomiais por partes

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

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

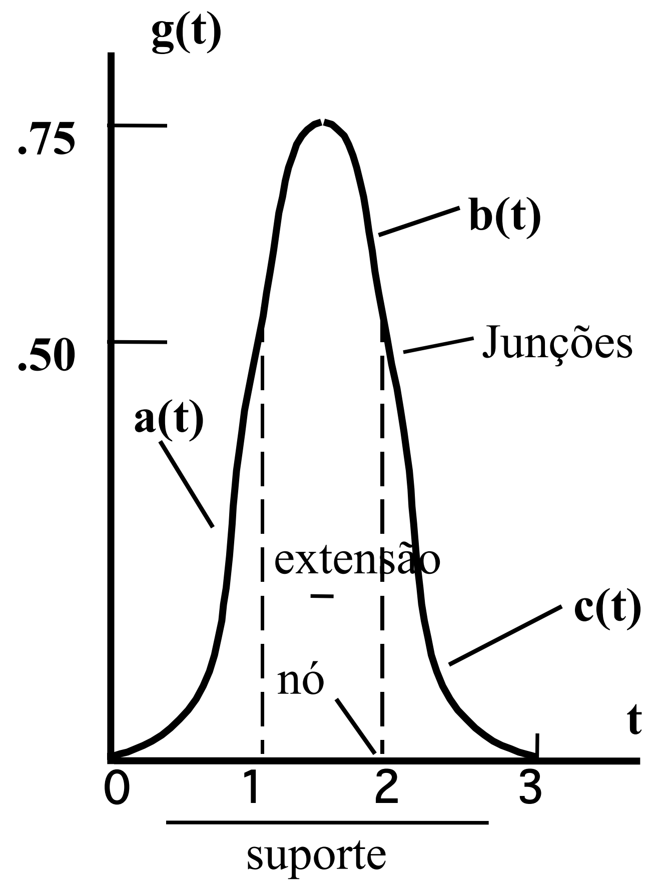
$$b(t) = \frac{3}{4} - (t - \frac{3}{2})^2$$

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



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;
- Já que ela é feita com polinômios, ela é certamente contínua dentro de cada extensão, e $a(1) = b(1) = 1/2$, 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 $a'(1) = b'(1) = 1$, $b'(2) = c'(2) = -1$.

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

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

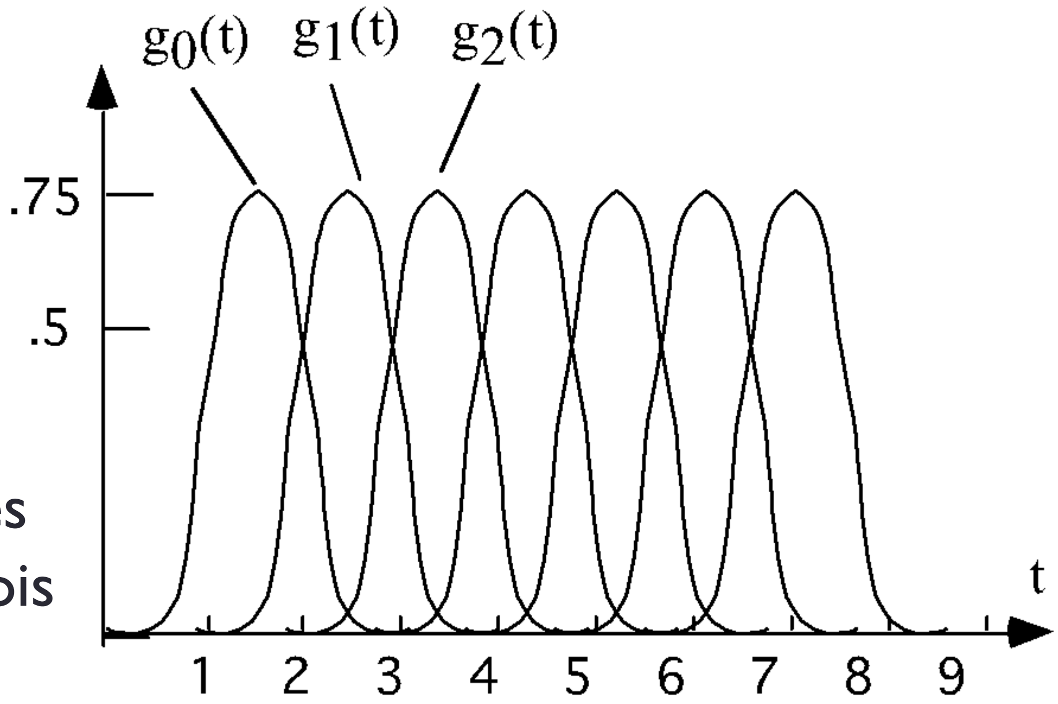
$$c(t) = \frac{1}{2}(3 - t)^2$$

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 de ponderação

- 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 $g_0(t),\ldots,g_6(t)$ obtidas depois de transladar g(.) por valores inteiros.



Funções splines de ponderação

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

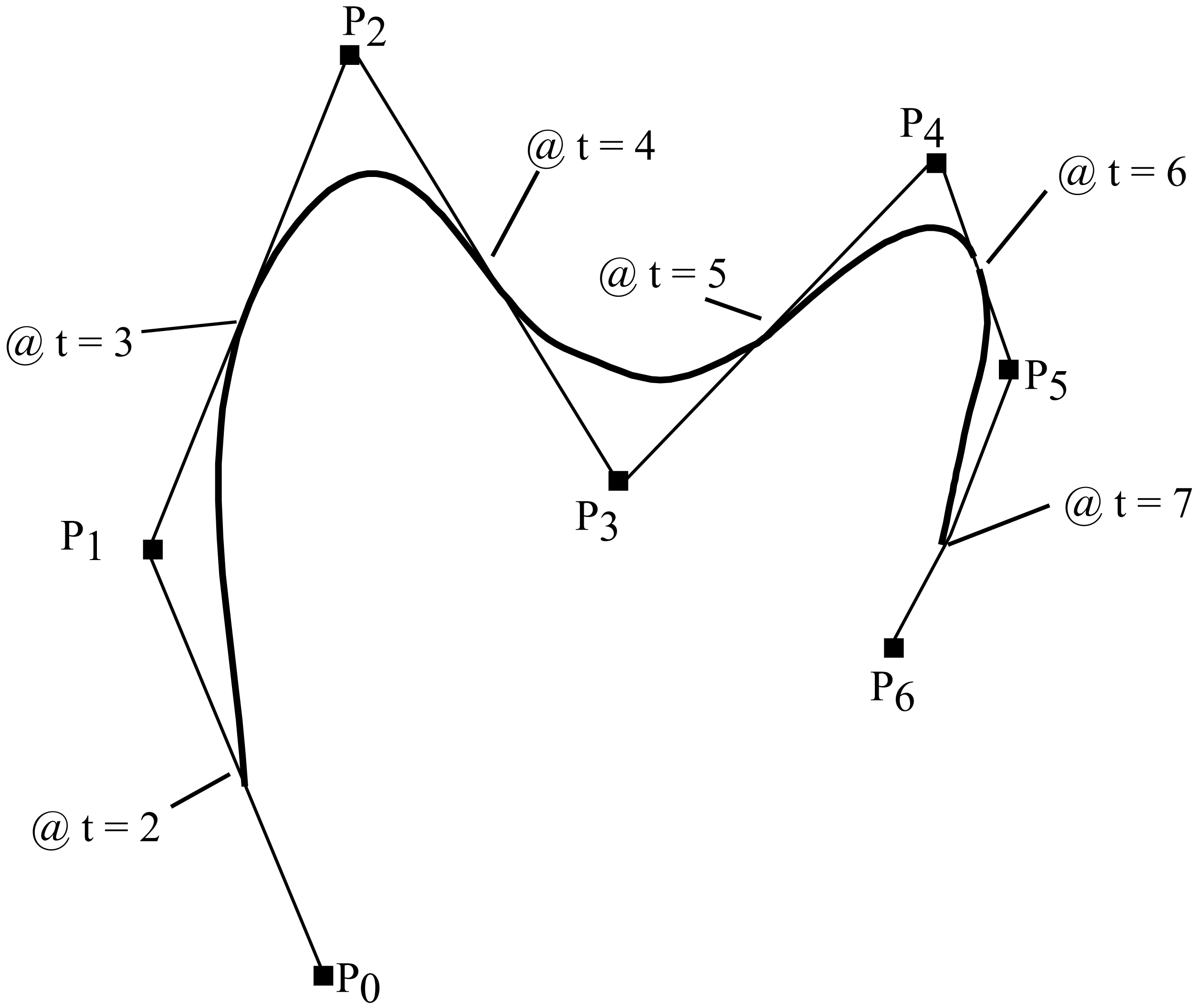
$$V(t) = \sum_{k=0}^{6} P_k g(t-k)$$

- 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.

Funções splines de ponderação

- 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.

Exemplo de curva spline



Propriedades da curva spline

- Possui **controle local**, já que o intervalo de suporte para cada função de ponderação tem tamanho 3.
- A curva deve **interpolar 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.

Funções genéricas de ponderação

- Precisamos maior controle: a curva deveria entortar mais e ter maior suavidade (>1).
  - Isso sugere a mudança para polinomiais cúbicas
- Queremos que o usuário possa especificar quais pontos de controle são interpolados
- Também gostaríamos que um único algoritmo 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

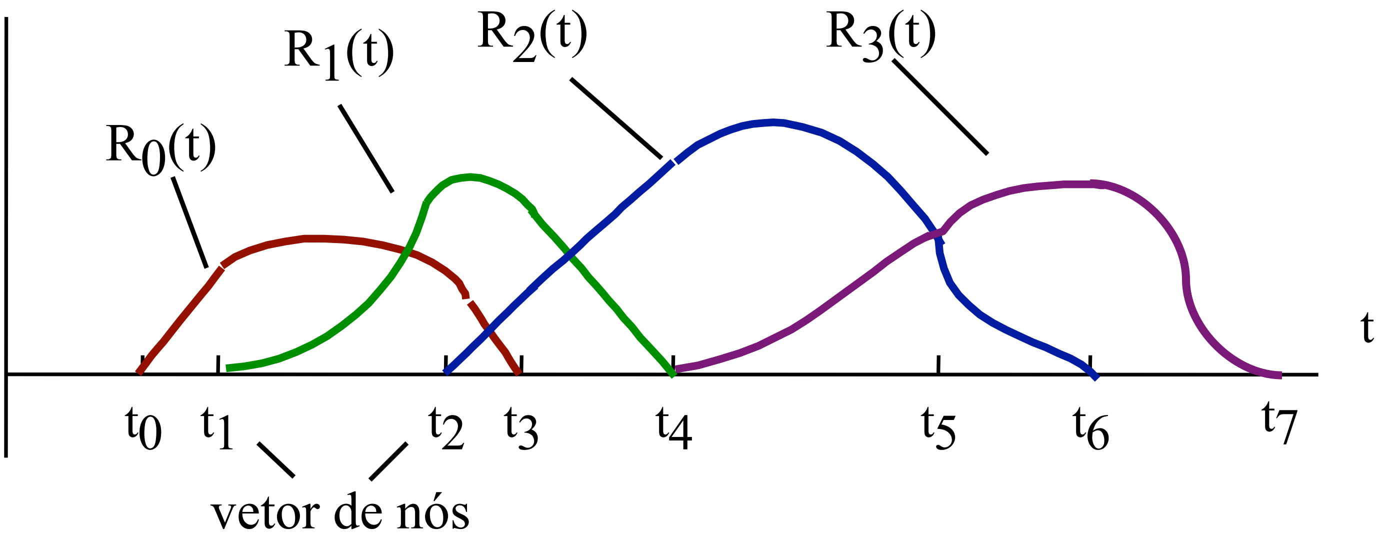
- Continuaremos a usar a mesma forma paramétrica

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

  baseada em $L+1$ pontos de controle e $L+1$ funções de ponderação $R_0(t), \ldots, R_L(t)$.
- $P(t)$ deve ser um somatório dos pontos.
- Usaremos polinomiais *piecewise* para as funções de ponderação, mas elas agora serão definidas através de uma seqüência de nós mais genéricas, 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 identificados de forma distinta.

Funções genéricas de ponderação

- Cada função de ponderação $R_k(t)$ é um polinômio "piecewise" que é zero até o tempo t_k, 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 e nós apropriados.

Tarefa de casa

- Leitura livro-texto
  - Shirley and Marschner. Fundamentals of Computer Graphics, CRC Press, 3rd Ed. 2010
  - Capítulo 15