

Interpolação

Prof. Afonso Paiva

Departamento de Matemática Aplicada e Estatística
Instituto de Ciências Matemáticas e de Computação
USP – São Carlos

Métodos Numéricos e Computacionais II – SME0306

A tabela abaixo mostra as temperaturas máximas (em graus Celsius) atingidas no mês de agosto a cada 5 dias:

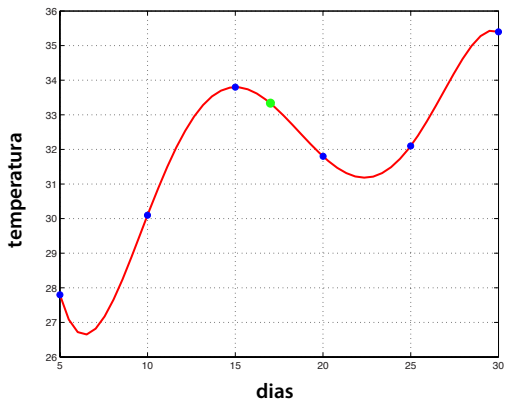
dia	5	10	15	20	25	30
temperatura	27.8	30.1	33.8	31.8	32.1	35.4

A tabela abaixo mostra as temperaturas máximas (em graus Celsius) atingidas no mês de agosto a cada 5 dias:

dia	5	10	15	20	25	30
temperatura	27.8	30.1	33.8	31.8	32.1	35.4

Como estimar a temperatura no dia 17?

Plote uma curva suave (de classe C^k) conectando esses pontos.



Suponha que temos uma função complicada:

$$f(x) = \frac{\log(x) e^{\sqrt{x}}}{\log(1 + e^x) + x^3}$$

Suponha que temos uma função complicada:

$$f(x) = \frac{\log(x) e^{\sqrt{x}}}{\log(1 + e^x) + x^3}$$

Como representar tal função de uma forma simples?

Em outras palavras, queremos avaliar, derivar, integrar $f(x)$ de uma maneira fácil e rápida.

Suponha que temos uma função complicada:

$$f(x) = \frac{\log(x) e^{\sqrt{x}}}{\log(1 + e^x) + x^3}$$

Como representar tal função de uma forma simples?

Em outras palavras, queremos avaliar, derivar, integrar $f(x)$ de uma maneira fácil e rápida.

- Uma estratégia:

Suponha que temos uma função complicada:

$$f(x) = \frac{\log(x) e^{\sqrt{x}}}{\log(1 + e^x) + x^3}$$

Como representar tal função de uma forma simples?

Em outras palavras, queremos avaliar, derivar, integrar $f(x)$ de uma maneira fácil e rápida.

■ Uma estratégia:

- 1 Avalie $y = f(x)$ em alguns pontos $\{x_0, x_1, \dots, x_n\}$;

Suponha que temos uma função complicada:

$$f(x) = \frac{\log(x) e^{\sqrt{x}}}{\log(1 + e^x) + x^3}$$

Como representar tal função de uma forma simples?

Em outras palavras, queremos avaliar, derivar, integrar $f(x)$ de uma maneira fácil e rápida.

■ Uma estratégia:

- 1 Avalie $y = f(x)$ em alguns pontos $\{x_0, x_1, \dots, x_n\}$;
- 2 Aproxime a função amostrada $y_i = f(x_i)$, $i = 0, \dots, n$, por uma função polinomial;

Suponha que temos uma função complicada:

$$f(x) = \frac{\log(x) e^{\sqrt{x}}}{\log(1 + e^x) + x^3}$$

Como representar tal função de uma forma simples?

Em outras palavras, queremos avaliar, derivar, integrar $f(x)$ de uma maneira fácil e rápida.

■ Uma estratégia:

- 1 Avalie $y = f(x)$ em alguns pontos $\{x_0, x_1, \dots, x_n\}$;
- 2 Aproxime a função amostrada $y_i = f(x_i)$, $i = 0, \dots, n$, por uma função polinomial;
- 3 Avalie, integre ou calcule as derivadas da função polinomial.

- Problema Básico de Interpolação:

■ Problema Básico de Interpolação:

Dados $(n + 1)$ pontos

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n) \quad \text{com} \quad x_0 < x_1 < \dots < x_n.$$

Determine uma função $F : \mathbb{R} \rightarrow \mathbb{R}$ tal que

$$y_i = F(x_i), \quad i = 0, \dots, n.$$

- Problema Básico de Interpolação:

Dados $(n + 1)$ pontos

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n) \quad \text{com} \quad x_0 < x_1 < \dots < x_n.$$

Determine uma função $F : \mathbb{R} \rightarrow \mathbb{R}$ tal que

$$y_i = F(x_i), \quad i = 0, \dots, n.$$

- A função $F(x)$ é a **função interpoladora**, ou **interpolante**, dos pontos dados;

- Problema Básico de Interpolação:

Dados $(n + 1)$ pontos

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n) \quad \text{com} \quad x_0 < x_1 < \dots < x_n.$$

Determine uma função $F : \mathbb{R} \rightarrow \mathbb{R}$ tal que

$$y_i = F(x_i), \quad i = 0, \dots, n.$$

- A função $F(x)$ é a **função interpoladora**, ou **interpolante**, dos pontos dados;
- Os pontos x_i são chamados de **nós da interpolação**.

Interpolação Polinomial

Dados $(n + 1)$ pontos

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n) \quad \text{com} \quad x_0 < x_1 < \dots < x_n.$$

Agora queremos encontrar um polinômio $P_n(x)$ de grau $\leq n$ que satisfaz as seguintes condições:

$$y_i = P_n(x_i), \quad i = 0, \dots, n.$$

O polinômio $P_n(x)$ é chamado de **polinômio de interpolação**.

Interpolação Polinomial

Dados $(n + 1)$ pontos

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n) \quad \text{com} \quad x_0 < x_1 < \dots < x_n.$$

Agora queremos encontrar um polinômio $P_n(x)$ de grau $\leq n$ que satisfaz as seguintes condições:

$$y_i = P_n(x_i), \quad i = 0, \dots, n.$$

O polinômio $P_n(x)$ é chamado de **polinômio de interpolação**.

Teorema

Dados $(n + 1)$ pontos $(x_0, y_0), \dots, (x_n, y_n)$ com $x_0 < \dots < x_n$, existe um único polinômio $P_n(x) \in \mathcal{P}_n$ que satisfaz as condições acima.

Interpolação Polinomial

Dados $(n + 1)$ pontos

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n) \quad \text{com} \quad x_0 < x_1 < \dots < x_n.$$

Agora queremos encontrar um polinômio $P_n(x)$ de grau $\leq n$ que satisfaz as seguintes condições:

$$y_i = P_n(x_i), \quad i = 0, \dots, n.$$

O polinômio $P_n(x)$ é chamado de **polinômio de interpolação**.

Teorema

Dados $(n + 1)$ pontos $(x_0, y_0), \dots, (x_n, y_n)$ com $x_0 < \dots < x_n$, existe um único polinômio $P_n(x) \in \mathcal{P}_n$ que satisfaz as condições acima.

Demonstração: Basta mostrar que o $\det(\mathbf{X}) \neq 0$, onde \mathbf{X} é a matriz de Vandermonnd.

Interpolação Polinomial

Dados $(n + 1)$ pontos

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n) \quad \text{com} \quad x_0 < x_1 < \dots < x_n.$$

Agora queremos encontrar um polinômio $P_n(x)$ de grau $\leq n$ que satisfaz as seguintes condições:

$$y_i = P_n(x_i), \quad i = 0, \dots, n.$$

O polinômio $P_n(x)$ é chamado de **polinômio de interpolação**.

Teorema

Dados $(n + 1)$ pontos $(x_0, y_0), \dots, (x_n, y_n)$ com $x_0 < \dots < x_n$, existe um único polinômio $P_n(x) \in \mathcal{P}_n$ que satisfaz as condições acima.

Demonstração: Basta mostrar que o $\det(\mathbf{X}) \neq 0$, onde \mathbf{X} é a matriz de Vandermond. **Mas como calcular $P_n(x)$?**

Forma de Lagrange

A **forma de Lagrange** para o polinômio de interpolação $P_n(x)$ nos pontos $(x_0, y_0), \dots, (x_n, y_n)$ é dado por:

$$P_n(x) = y_0 \ell_0(x) + y_1 \ell_1(x) + \dots + y_n \ell_n(x) = \sum_{k=0}^n y_k \ell_k(x),$$

onde $\ell_k(x) \in \mathcal{P}_n$ são polinômios que dependem apenas de x_0, \dots, x_n .

Forma de Lagrange

Dessa forma, $P_n(x_i) = y_0 \ell_0(x_i) + \cdots + y_i \ell_i(x_i) + \cdots + y_n \ell_n(x_i) = y_i$.

Logo,

$$\ell_k(x_i) = \delta_{ik} = \begin{cases} 1, & \text{se } i = k \\ 0, & \text{se } i \neq k \end{cases}$$

Forma de Lagrange

Dessa forma, $P_n(x_i) = y_0 \ell_0(x_i) + \cdots + y_i \ell_i(x_i) + \cdots + y_n \ell_n(x_i) = y_i$.
Logo,

$$\ell_k(x_i) = \delta_{ik} = \begin{cases} 1, & \text{se } i = k \\ 0, & \text{se } i \neq k \end{cases}$$

Para isso, basta definir cada **polinômio de Lagrange** $\ell_k(x)$ da forma:

$$\ell_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^n \frac{x - x_i}{x_k - x_i}, \quad k = 0, \dots, n.$$

Forma de Lagrange

Exemplo 1

Dada a função tabelada $y = f(x)$:

x	-2	0	3	5
y	3	-2	4	2

Calcule o polinômio de interpolação com a forma de Lagrange usando todos os pontos da tabela.

Forma de Lagrange

Solução:

$$P_3(x) = 3l_0(x) - 2l_1(x) + 4l_2(x) + 2l_3(x)$$

$$l_0(x) = \frac{x(x-3)(x-5)}{-70}$$

$$l_1(x) = \frac{(x+2)(x-3)(x-5)}{30}$$

$$l_2(x) = \frac{(x+2)x(x-5)}{-30}$$

$$l_3(x) = \frac{(x+2)x(x-3)}{70}$$

Forma de Lagrange

Solução:

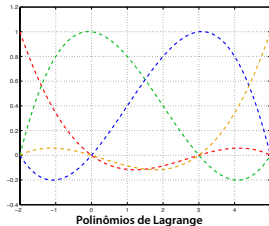
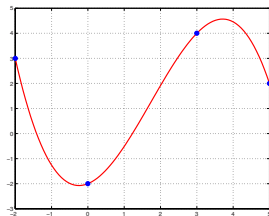
$$P_3(x) = 3l_0(x) - 2l_1(x) + 4l_2(x) + 2l_3(x)$$

$$l_0(x) = \frac{x(x-3)(x-5)}{-70}$$

$$l_1(x) = \frac{(x+2)(x-3)(x-5)}{30}$$

$$l_2(x) = \frac{(x+2)x(x-5)}{-30}$$

$$l_3(x) = \frac{(x+2)x(x-3)}{70}$$



Forma de Lagrange

Solução:

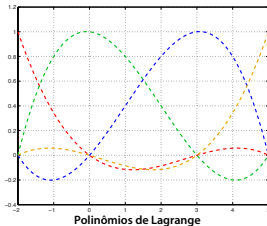
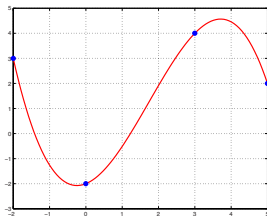
$$P_3(x) = 3l_0(x) - 2l_1(x) + 4l_2(x) + 2l_3(x)$$

$$l_0(x) = \frac{x(x-3)(x-5)}{-70}$$

$$l_1(x) = \frac{(x+2)(x-3)(x-5)}{30}$$

$$l_2(x) = \frac{(x+2)x(x-5)}{-30}$$

$$l_3(x) = \frac{(x+2)x(x-3)}{70}$$



Atenção: se adicionarmos mais um ponto $(x_{n+1}, y_{n+1}) \Rightarrow$ todos $l_k(x)$ precisam ser recalculados!

MATLAB – Forma de Lagrange

```
1 function y = lagrange_interp(xi,yi,x)
2 % xi, yi, x: vetor linha ou coluna
3 [m,n]= size (xi);
4 if (n == 1) xi = xi'; yi = yi'; x = x'; n = m; end
5
6 L = ones(n,length(x));
7
8 for i =1:n
9     for j =1:n
10        if ( i ~= j )
11            L(i,:) = L(i,:).*(x-xi(j))/(xi(i)-xi(j));
12        end
13    end
14 end
15
16 y = yi*L;
```

Forma de Newton

A **forma de Newton** para $P_n(x)$ é dada de maneira diferente:

$$P_n(x) = \alpha_0 + \alpha_1(x - x_0) + \alpha_2(x - x_0)(x - x_1) + \cdots + \\ + \alpha_n(x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

Forma de Newton

A **forma de Newton** para $P_n(x)$ é dada de maneira diferente:

$$P_n(x) = \alpha_0 + \alpha_1(x - x_0) + \alpha_2(x - x_0)(x - x_1) + \cdots + \\ + \alpha_n(x - x_0)(x - x_1) \cdots (x - x_{n-1})$$

Cada coeficiente α_k é determinado por uma **diferença dividida** de ordem k :

$$\alpha_k = f[x_0, x_1, \dots, x_k], \quad k = 0, 1, \dots, n.$$

Forma de Newton

Definição (diferenças divididas)

As diferenças divididas são definidas recursivamente:

$$f[x_i] := f(x_i), \quad i = 0, \dots, n.$$

$$f[x_i, x_{i+1}, \dots, x_{i+k}] := \frac{f[x_{i+1}, x_{i+2}, \dots, x_{i+k}] - f[x_i, x_{i+1}, \dots, x_{i+k-1}]}{x_{i+k} - x_i},$$

com $k = 1, \dots, n$ e $i = 0, \dots, n - k$.

Diferenças Divididas

- Diferença dividida de ordem 0:

$$f[x_i] = f(x_i)$$

Diferenças Divididas

- Diferença dividida de ordem 0:

$$f[x_i] = f(x_i)$$

- Diferença dividida de ordem 1:

$$f[x_i, x_j] = \frac{f[x_j] - f[x_i]}{x_j - x_i} = \frac{f(x_j) - f(x_i)}{x_j - x_i}$$

Diferenças Divididas

- Diferença dividida de ordem 0:

$$f[x_i] = f(x_i)$$

- Diferença dividida de ordem 1:

$$f[x_i, x_j] = \frac{f[x_j] - f[x_i]}{x_j - x_i} = \frac{f(x_j) - f(x_i)}{x_j - x_i}$$

- Diferença dividida de ordem k :

$$f[x_{i_0}, x_{i_1}, \dots, x_{i_k}] = \frac{f[x_{i_1}, x_{i_2}, \dots, x_{i_k}] - f[x_{i_0}, x_{i_1}, \dots, x_{i_{k-1}}]}{x_{i_k} - x_{i_0}}$$

Diferenças Divididas

Recursivamente temos a seguinte **tabela de diferenças divididas**.

Diferenças Divididas

Recursivamente temos a seguinte **tabela de diferenças divididas**.

x	ordem 0	ordem 1	ordem 2	ordem 3	...
x_0	$f[x_0] = \alpha_0$	$f[x_0, x_1] = \alpha_1$	$f[x_0, x_1, x_2] = \alpha_2$	$f[x_0, x_1, x_2, x_3] = \alpha_3$...
x_1	$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$	\vdots	
x_2	$f[x_2]$	$f[x_2, x_3]$	\vdots	\vdots	
x_3	$f[x_3]$	\vdots	\vdots	\vdots	
\vdots	\vdots	\vdots	\vdots	\vdots	

Forma de Newton

Exemplo 2

Calcule o polinômio de interpolação com a forma de Newton usando todos os pontos da tabela do Exemplo 1.

Forma de Newton

Exemplo 2

Calcule o polinômio de interpolação com a forma de Newton usando todos os pontos da tabela do Exemplo 1.

Solução: Primeiro vamos montar a tabela de diferenças divididas

x	ordem 0	ordem 1	ordem 2	ordem 3
-2	3	$-5/2$	$9/10$	$-3/14$
0	-2	2	$-3/5$	
3	4	-1		
5	2			

Forma de Newton

Exemplo 2

Calcule o polinômio de interpolação com a forma de Newton usando todos os pontos da tabela do Exemplo 1.

Solução: Primeiro vamos montar a tabela de diferenças divididas

x	ordem 0	ordem 1	ordem 2	ordem 3
-2	3	$-5/2$	$9/10$	$-3/14$
0	-2	2	$-3/5$	
3	4	-1		
5	2			

$$P_3(x) = 3 - \frac{5}{2}(x+2) + \frac{9}{10}(x+2)x - \frac{3}{14}(x+2)x(x-3)$$

MATLAB – Forma de Newton

```
1 function y = newton_interp(xi,yi,x)
2 % xi, yi, x: vetor linha ou coluna
3 [m,n]= size (x);
4 if (n == 1) xi = xi'; yi = yi'; n = m; end
5 n = length(xi); ni = length(x); N = ones(n,ni);
6 D=zeros(n); D(:,1) = yi';
7
8 for j=1:n % tabela de diferencas divididas
9     for i=1:n-j
10         D(i,j+1) = (D(i+1,j)-D(i,j))/(xi(i+j)-xi(i));
11     end
12 end
13 for i=2:n % forma de Newton
14     for k = 1:i-1
15         N(i,:) = N(i,:).*(x-xi(k));
16     end
17 end
18
19 y = D(1,:)*N;
```

Interpolando Derivadas

Agora vamos admitir também **derivadas** no nós de interpolação:

Interpolando Derivadas

Agora vamos admitir também **derivadas** no nós de interpolação:

$$f(0) = 2, \quad f'(0) = 1, \quad f(10) = 0.$$

Interpolando Derivadas

Agora vamos admitir também **derivadas** no nós de interpolação:

$$f(0) = 2, \quad f'(0) = 1, \quad f(10) = 0.$$

Dessa forma temos $\{(x_i, y_i)\} = \{(0, 2), (0, 1), (10, 0)\}$.

Interpolando Derivadas

Agora vamos admitir também **derivadas** no nós de interpolação:

$$f(0) = 2, \quad f'(0) = 1, \quad f(10) = 0.$$

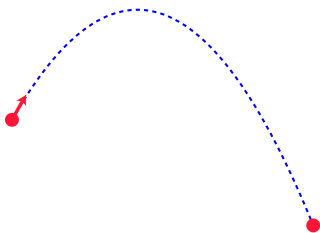
Dessa forma temos $\{(x_i, y_i)\} = \{(0, 2), (0, 1), (10, 0)\}$. Utilizando a base canônica de $\mathcal{P}_2 \Rightarrow P_2(x) = a + bx + cx^2$.

Interpolando Derivadas

Agora vamos admitir também **derivadas** no nós de interpolação:

$$f(0) = 2, \quad f'(0) = 1, \quad f(10) = 0.$$

Dessa forma temos $\{(x_i, y_i)\} = \{(0, 2), (0, 1), (10, 0)\}$. Utilizando a base canônica de $\mathcal{P}_2 \Rightarrow P_2(x) = a + bx + cx^2$. Logo,

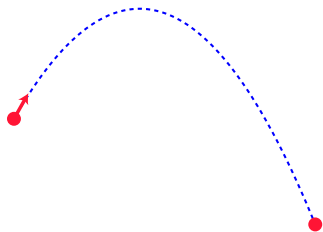


Interpolando Derivadas

Agora vamos admitir também **derivadas** no nós de interpolação:

$$f(0) = 2, \quad f'(0) = 1, \quad f(10) = 0.$$

Dessa forma temos $\{(x_i, y_i)\} = \{(0, 2), (0, 1), (10, 0)\}$. Utilizando a base canônica de $\mathcal{P}_2 \Rightarrow P_2(x) = a + bx + cx^2$. Logo,



$$2 = P_2(0) = a,$$

$$1 = P_2'(0) = b,$$

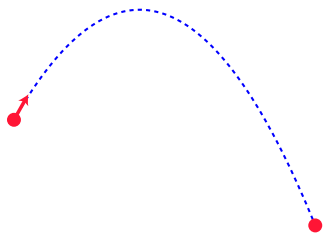
$$0 = P_2(10) = a + 10b + 100c \Rightarrow c = -\frac{12}{100}$$

Interpolando Derivadas

Agora vamos admitir também **derivadas** no nós de interpolação:

$$f(0) = 2, \quad f'(0) = 1, \quad f(10) = 0.$$

Dessa forma temos $\{(x_i, y_i)\} = \{(0, 2), (0, 1), (10, 0)\}$. Utilizando a base canônica de $\mathcal{P}_2 \Rightarrow P_2(x) = a + bx + cx^2$. Logo,



$$2 = P_2(0) = a,$$

$$1 = P_2'(0) = b,$$

$$0 = P_2(10) = a + 10b + 100c \Rightarrow c = -\frac{12}{100}$$

Portanto, $P_2(x) = 2 + x - \frac{12}{100}x^2$.

Interpolação de Hermite

- **Interpolação de Hermite** fornece um polinômio de interpolação de $f(x)$ e das derivadas $f'(x), f''(x), f^{(3)}(x), \text{ etc...}$

Interpolação de Hermite

- **Interpolação de Hermite** fornece um polinômio de interpolação de $f(x)$ e das derivadas $f'(x), f''(x), f^{(3)}(x)$, etc...
- As condições de interpolação em cada nó x_i são dadas por:

$$P_m^{(j)}(x_i) = f^{(j)}(x_i) = c_{ij}, \quad 0 \leq j \leq k_i - 1 \text{ e } 0 \leq i \leq n$$

$$m + 1 = \kappa_0 + \kappa_1 + \cdots + \kappa_n,$$

onde κ_i é o número de condições em um nó x_i .

Interpolação de Hermite

- **Interpolação de Hermite** fornece um polinômio de interpolação de $f(x)$ e das derivadas $f'(x), f''(x), f^{(3)}(x), \text{etc...}$
- As condições de interpolação em cada nó x_i são dadas por:

$$P_m^{(j)}(x_i) = f^{(j)}(x_i) = c_{ij}, \quad 0 \leq j \leq k_i - 1 \text{ e } 0 \leq i \leq n$$

$$m + 1 = \kappa_0 + \kappa_1 + \cdots + \kappa_n,$$

onde κ_i é o número de condições em um nó x_i .

Teorema

Existe um único polinômio $P_m(x) \in \mathcal{P}_m$ que satisfaz as condições (de interpolação de Hermite) acima.

Forma de Newton

Considerações:

- Para isso vamos usar diferenças divididas. Seja x_k um nó de interpolação **duplo**. Logo,

$$f[x_k, x_k] = \lim_{h \rightarrow 0} \frac{f(x_k + h) - f(x_k)}{(x_k + h) - x_k} = \lim_{h \rightarrow 0} \frac{f(x_k + h) - f(x_k)}{h} = f'(x_k).$$

Forma de Newton

Considerações:

- Para isso vamos usar diferenças divididas. Seja x_k um nó de interpolação **duplo**. Logo,

$$f[x_k, x_k] = \lim_{h \rightarrow 0} \frac{f(x_k + h) - f(x_k)}{(x_k + h) - x_k} = \lim_{h \rightarrow 0} \frac{f(x_k + h) - f(x_k)}{h} = f'(x_k).$$

- No caso geral, podemos escrever

$$f[\underbrace{x_k, \dots, x_k}_{m \text{ vezes}}] = \frac{1}{(m-1)!} f^{(m-1)}(x_k)$$

Forma de Newton

Considerações:

- Para isso vamos usar diferenças divididas. Seja x_k um nó de interpolação **duplo**. Logo,

$$f[x_k, x_k] = \lim_{h \rightarrow 0} \frac{f(x_k + h) - f(x_k)}{(x_k + h) - x_k} = \lim_{h \rightarrow 0} \frac{f(x_k + h) - f(x_k)}{h} = f'(x_k).$$

- No caso geral, podemos escrever

$$f[\underbrace{x_k, \dots, x_k}_{m \text{ vezes}}] = \frac{1}{(m-1)!} f^{(m-1)}(x_k)$$

- Perceba a relação com polinômios de Taylor!

$$P_n(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!} (x-a)^k \quad (\text{em torno de } a)$$

Forma de Newton

Dados $(n + 1)$ pontos $(x_0, y_0), \dots, (x_n, y_n)$ com $x_0 < \dots < x_n$.

Queremos obter $P_{2n+1}(x)$ que satisfaz as $(2n + 2)$ condições:

$$P_{2n+1}(x_i) = c_{i0}, \quad P'_{2n+1}(x_i) = c_{i1}, \quad i = 0, \dots, n.$$

Forma de Newton

Dados $(n + 1)$ pontos $(x_0, y_0), \dots, (x_n, y_n)$ com $x_0 < \dots < x_n$.

Queremos obter $P_{2n+1}(x)$ que satisfaz as $(2n + 2)$ condições:

$$P_{2n+1}(x_i) = c_{i0}, \quad P'_{2n+1}(x_i) = c_{i1}, \quad i = 0, \dots, n.$$

Para mostrar como é o procedimento, consideremos o caso $n = 1$

x	ordem 0	ordem 1	ordem 2	ordem 3
x_0	$f[x_0] = c_{00} = \alpha_0$	$f[x_0, x_0] = c_{01} = \alpha_1$	$f[x_0, x_0, x_1] = \alpha_2$	$f[x_0, x_0, x_1, x_1] = \alpha_3$
x_0	$f[x_0]$	$f[x_0, x_1]$	$f[x_0, x_1, x_1]$	
x_1	$f[x_1] = c_{10}$	$f[x_1, x_1] = c_{11}$		
x_1	$f[x_1]$			

O polinômio de Hermite possui termos quadráticos

$$P_3 = \alpha_0 + \alpha_1(x - x_0) + \alpha_2(x - x_0)^2 + \alpha_3(x - x_0)^2(x - x_1)$$

Forma de Newton

Exemplo 3

Calcule o polinômio $p(x)$ de Hermite usando a forma de Newton com as seguintes condições:

$$p(1) = 2 \quad p'(1) = 3 \quad p(2) = 6 \quad p'(2) = 7 \quad p''(2) = 8$$

Forma de Newton

Exemplo 3

Calcule o polinômio $p(x)$ de Hermite usando a forma de Newton com as seguintes condições:

$$p(1) = 2 \quad p'(1) = 3 \quad p(2) = 6 \quad p'(2) = 7 \quad p''(2) = 8$$

Solução: Primeiro vamos montar a tabela de diferenças divididas

x	ordem 0	ordem 1	ordem 2	ordem 3	ordem 4
1	2	3	1	2	-1
1	2	4	3	1	
2	6	7	4		
2	6	7			
2	6				

Forma de Newton

Exemplo 3

Calcule o polinômio $p(x)$ de Hermite usando a forma de Newton com as seguintes condições:

$$p(1) = 2 \quad p'(1) = 3 \quad p(2) = 6 \quad p'(2) = 7 \quad p''(2) = 8$$

Solução: Primeiro vamos montar a tabela de diferenças divididas

x	ordem 0	ordem 1	ordem 2	ordem 3	ordem 4
1	2	3	1	2	-1
1	2	4	3	1	
2	6	7	4		
2	6	7			
2	6				

$$p(x) = 2 + 3(x - 1) + (x - 1)^2 + 2(x - 1)^2(x - 2) - (x - 1)^2(x - 2)^2$$

Erro na Interpolação

Teorema (erro para interpolação de Lagrange e Newton)

Sejam $f \in \mathcal{C}^{n+1}([a, b])$, $a = x_0 < x_1 < \dots < x_n = b$ e $P_n(x)$ o polinômio de interpolação de $f(x)$ então

$$E_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^n (x - x_i),$$

onde $\xi = \xi(x) \in (a, b)$.

Erro na Interpolação

Teorema (erro para interpolação de Lagrange e Newton)

Sejam $f \in C^{n+1}([a, b])$, $a = x_0 < x_1 < \dots < x_n = b$ e $P_n(x)$ o polinômio de interpolação de $f(x)$ então

$$E_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^n (x - x_i),$$

onde $\xi = \xi(x) \in (a, b)$.

Demonstração: Burden & Faires, Seção 3.1.

Erro na Interpolação

Teorema (erro para interpolação de Hermite)

Sejam $f \in \mathcal{C}^{2n+2}([a, b])$, $a = x_0 < x_1 < \dots < x_n = b$ e $P_{2n+1}(x)$ o polinômio de interpolação de Hermite que satisfaz

$$P_{2n+1}(x_i) = f(x_i), \quad P'_{2n+1}(x_i) = f'(x_i), \quad i = 0, \dots, n$$

então para ponto $x \in [a, b]$ há um $\xi = \xi(x) \in (a, b)$, tal que

$$R_n(x) = f(x) - P_n(x) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \prod_{i=0}^n (x - x_i)^2.$$

Erro na Interpolação

Teorema (erro para interpolação de Hermite)

Sejam $f \in \mathcal{C}^{2n+2}([a, b])$, $a = x_0 < x_1 < \dots < x_n = b$ e $P_{2n+1}(x)$ o polinômio de interpolação de Hermite que satisfaz

$$P_{2n+1}(x_i) = f(x_i), \quad P'_{2n+1}(x_i) = f'(x_i), \quad i = 0, \dots, n$$

então para ponto $x \in [a, b]$ há um $\xi = \xi(x) \in (a, b)$, tal que

$$R_n(x) = f(x) - P_n(x) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \prod_{i=0}^n (x - x_i)^2.$$

Demonstração: Burden & Faires, Seção 3.3.

Estimativas de Erro na Interpolação

Corolário

- $|E_n(x)| \leq \frac{1}{(n+1)!} \|f^{(n+1)}\|_\infty \prod_{i=0}^n |x - x_i|$
- $\|E_n\|_\infty \leq \frac{1}{(n+1)!} \|f^{(n+1)}\|_\infty (b - a)^{n+1}$
- $|R_n(x)| \leq \frac{1}{(2n+2)!} \|f^{(2n+2)}\|_\infty \prod_{i=0}^n (x - x_i)^2$
- $\|R_n\|_\infty \leq \frac{1}{(2n+2)!} \|f^{(2n+2)}\|_\infty (b - a)^{2n+2}$

Corolário (distribuição uniforme de nós)

Dado x_0 , se $x_{i+1} = x_i + h$ para $i = 0, \dots, n - 1$ com $h = (b - a)/n$ então

$$\|E_n\|_\infty \leq \frac{h^{n+1}}{4(n+1)} \|f^{(n+1)}\|_\infty.$$

Estimativas de Erro na Interpolação

Exemplo 4

Se a função $f(x) = \cos(x)$ é aproximada por polinômio de grau 9 que interpola f em 10 pontos no intervalo $[0, 1]$, quão grande é o erro de interpolação neste intervalo?

Estimativas de Erro na Interpolação

Exemplo 4

Se a função $f(x) = \cos(x)$ é aproximada por polinômio de grau 9 que interpola f em 10 pontos no intervalo $[0, 1]$, quão grande é o erro de interpolação neste intervalo?

Solução: Vamos calcular um limitante superior para $\|E_9\|_\infty$. Logo,

$$\|f^{(10)}\|_\infty = \max_{t \in [0,1]} |f^{(10)}(t)| = 1 \quad \implies \quad \|E_9\|_\infty \leq \frac{1}{10!} < 2.8 \times 10^{-7}.$$

Estimativas de Erro na Interpolação

Exemplo 4

Se a função $f(x) = \cos(x)$ é aproximada por polinômio de grau 9 que interpola f em 10 pontos no intervalo $[0, 1]$, quão grande é o erro de interpolação neste intervalo?

Solução: Vamos calcular um limitante superior para $\|E_9\|_\infty$. Logo,

$$\|f^{(10)}\|_\infty = \max_{t \in [0,1]} |f^{(10)}(t)| = 1 \quad \implies \quad \|E_9\|_\infty \leq \frac{1}{10!} < 2.8 \times 10^{-7}.$$

Será que P_n **converge** para f quando $n \rightarrow \infty$?

Fenômeno de Runge

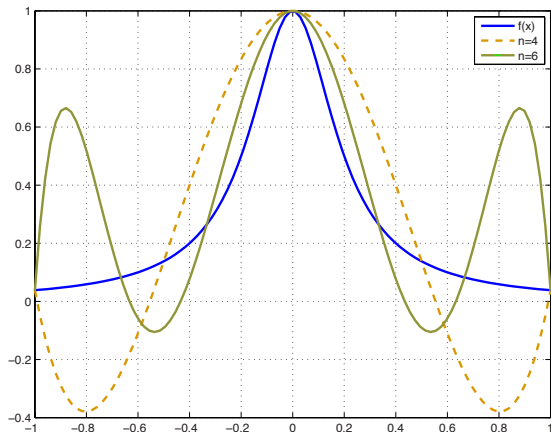
Exemplo 5

Considere a função

$$f(x) = \frac{1}{1 + 25x^2}, \quad x \in [-1, 1].$$

Calcule e plote no MATLAB o polinômio de interpolação de f usando a forma de Lagrange com 5, 7 e 16 nós de interpolação igualmente espaçados no intervalo $[-1, 1]$.

Fenômeno de Runge



Fenômeno de Runge

Conclusões:

- **Não** há garantias que P_n converge para f quando $n \rightarrow \infty$;
- Interpolação polinomial de alta ordem é **instável** em uma distribuição uniforme de nós.

Fenômeno de Runge

Conclusões:

- **Não** há garantias que P_n converge para f quando $n \rightarrow \infty$;
- Interpolação polinomial de alta ordem é **instável** em uma distribuição uniforme de nós.

Soluções:

- Usar uma distribuição **não uniforme** de nós que minimize o erro;
- Interpolação polinomial **por partes**.

Nós de Chebyshev

Como escolher nós que minimizem o limitante do erro?

$$\|E_n(x)\|_\infty \leq \frac{1}{(n+1)!} \|f^{(n+1)}\|_\infty \|\omega_n\|_\infty \quad \omega_n(x) = \prod_{i=0}^n (x - x_i)$$

Nós de Chebyshev

Como escolher nós que minimizem o limitante do erro?

$$\|E_n(x)\|_\infty \leq \frac{1}{(n+1)!} \|f^{(n+1)}\|_\infty \|\omega_n\|_\infty \quad \omega_n(x) = \prod_{i=0}^n (x - x_i)$$

Basta minimizar $\|\omega_n\|_\infty$, isso nos leva aos nós de Chebyshev:

Nós de Chebyshev

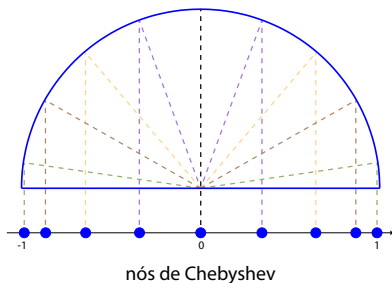
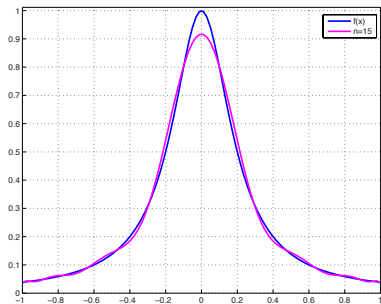
Como escolher nós que minimizem o limitante do erro?

$$\|E_n(x)\|_\infty \leq \frac{1}{(n+1)!} \|f^{(n+1)}\|_\infty \|\omega_n\|_\infty \quad \omega_n(x) = \prod_{i=0}^n (x - x_i)$$

Basta minimizar $\|\omega_n\|_\infty$, isso nos leva aos nós de Chebyshev:

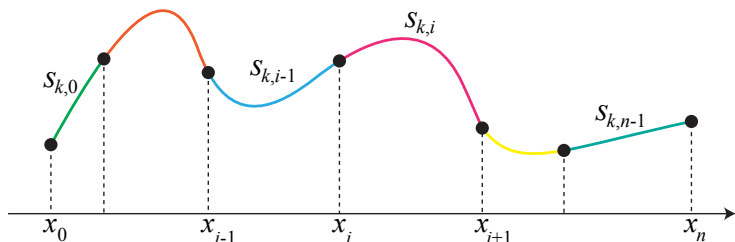
$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{2i+1}{2(n+1)}\pi\right), \quad i = 0, \dots, n$$

Nós de Chebyshev



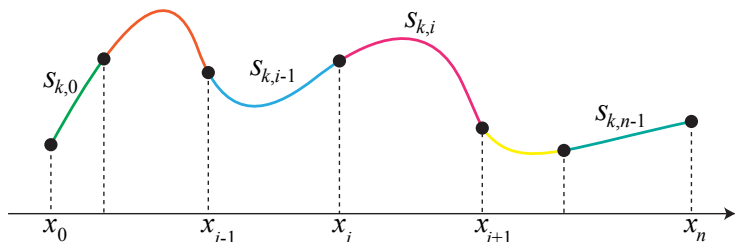
$$f(x) = \frac{1}{1 + 25x^2}, \quad x \in [-1, 1].$$

Splines



Dados $(n + 1)$ pontos $(x_0, y_0), \dots, (x_n, y_n)$ com $a = x_0 < \dots < x_n = b$.

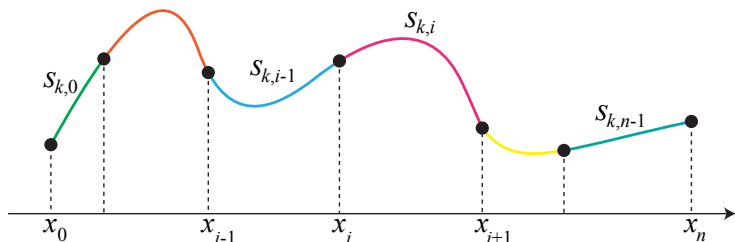
Uma função $S_k(x)$ e chamada de **spline** de grau k se satisfaz as seguintes condições:



Dados $(n + 1)$ pontos $(x_0, y_0), \dots, (x_n, y_n)$ com $a = x_0 < \dots < x_n = b$.

Uma função $S_k(x)$ e chamada de **spline** de grau k se satisfaz as seguintes condições:

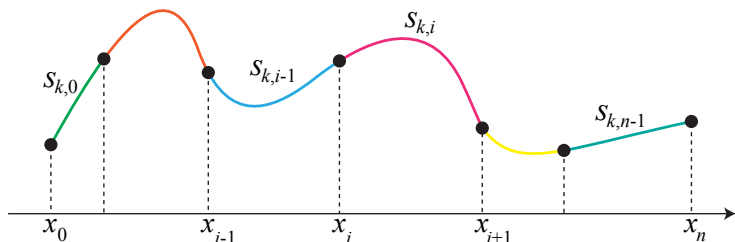
- $S_{k,i} = S_k|_{[x_i, x_{i+1}]}$, com $i = 0, \dots, n - 1$, é um polinômio de grau k ;



Dados $(n + 1)$ pontos $(x_0, y_0), \dots, (x_n, y_n)$ com $a = x_0 < \dots < x_n = b$.

Uma função $S_k(x)$ é chamada de **spline** de grau k se satisfaz as seguintes condições:

- $S_{k,i} = S_k|_{[x_i, x_{i+1}]}$, com $i = 0, \dots, n - 1$, é um polinômio de grau k ;
- $S_k \in \mathcal{C}^{k-1}([a, b])$;

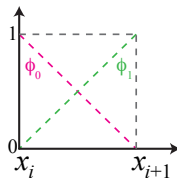
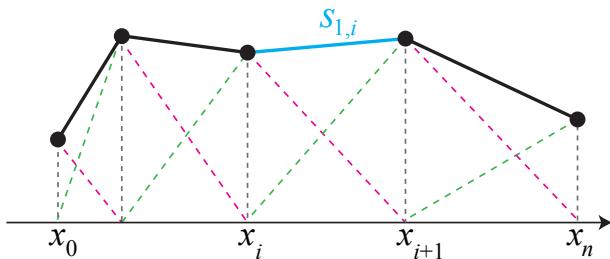


Dados $(n + 1)$ pontos $(x_0, y_0), \dots, (x_n, y_n)$ com $a = x_0 < \dots < x_n = b$.

Uma função $S_k(x)$ é chamada de **spline** de grau k se satisfaz as seguintes condições:

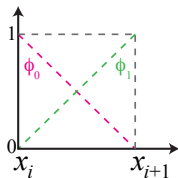
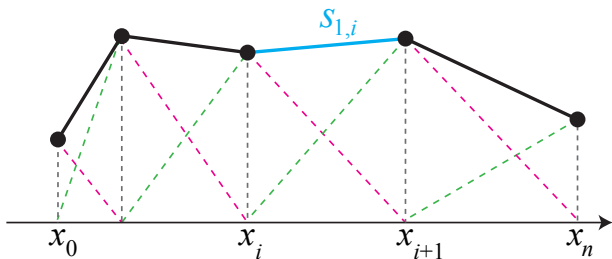
- $S_{k,i} = S_k|_{[x_i, x_{i+1}]}$, com $i = 0, \dots, n - 1$, é um polinômio de grau k ;
- $S_k \in \mathcal{C}^{k-1}([a, b])$;
- $S_k(x_i) = y_i$, com $i = 0, \dots, n$.

Spline Linear



$$S_{1,i}(x) = y_i \underbrace{\frac{x_{i+1} - x}{x_{i+1} - x_i}}_{\phi_0(x)} + y_{i+1} \underbrace{\frac{x - x_i}{x_{i+1} - x_i}}_{\phi_1(x)}, \quad x \in [x_i, x_{i+1}], \quad i = 0, \dots, n-1.$$

Spline Linear



$$S_{1,i}(x) = y_i \underbrace{\frac{x_{i+1} - x}{x_{i+1} - x_i}}_{\phi_0(x)} + y_{i+1} \underbrace{\frac{x - x_i}{x_{i+1} - x_i}}_{\phi_1(x)}, \quad x \in [x_i, x_{i+1}], \quad i = 0, \dots, n-1.$$

Note que, fazendo $t = \phi_1(x) \Rightarrow (1 - t) = \phi_0(x)$, logo

$$S_{1,i}(t) = (1 - t) y_i + t y_{i+1}.$$

Spline Linear

$$S_{1,i}(x) = y_i \frac{x_{i+1} - x}{x_{i+1} - x_i} + y_{i+1} \frac{x - x_i}{x_{i+1} - x_i}, \quad x \in [x_i, x_{i+1}], \quad i = 0, \dots, n-1.$$

Considerações:

Spline Linear

$$S_{1,i}(x) = y_i \frac{x_{i+1} - x}{x_{i+1} - x_i} + y_{i+1} \frac{x - x_i}{x_{i+1} - x_i}, \quad x \in [x_i, x_{i+1}], \quad i = 0, \dots, n-1.$$

Considerações:

- $S_1(x)$ é um polinômio de grau 1 em cada sub-intervalo $[x_i, x_{i+1}]$;

Spline Linear

$$S_{1,i}(x) = y_i \frac{x_{i+1} - x}{x_{i+1} - x_i} + y_{i+1} \frac{x - x_i}{x_{i+1} - x_i}, \quad x \in [x_i, x_{i+1}], \quad i = 0, \dots, n-1.$$

Considerações:

- $S_1(x)$ é um polinômio de grau 1 em cada sub-intervalo $[x_i, x_{i+1}]$;
- $S_{1,i-1}(x_{i-1}) = y_{i-1}$ e $S_{1,i-1}(x_i) = y_i$, $i = 1, \dots, n$;

Spline Linear

$$S_{1,i}(x) = y_i \frac{x_{i+1} - x}{x_{i+1} - x_i} + y_{i+1} \frac{x - x_i}{x_{i+1} - x_i}, \quad x \in [x_i, x_{i+1}], \quad i = 0, \dots, n-1.$$

Considerações:

- $S_1(x)$ é um polinômio de grau 1 em cada sub-intervalo $[x_i, x_{i+1}]$;
- $S_{1,i-1}(x_{i-1}) = y_{i-1}$ e $S_{1,i-1}(x_i) = y_i$, $i = 1, \dots, n$;
- $S_1 \in C^0([a, b])$, por construção $S_{1,i}(x_{i+1}) = y_{i+1} = S_{1,i+1}(x_{i+1})$;

Spline Linear

$$S_{1,i}(x) = y_i \frac{x_{i+1} - x}{x_{i+1} - x_i} + y_{i+1} \frac{x - x_i}{x_{i+1} - x_i}, \quad x \in [x_i, x_{i+1}], \quad i = 0, \dots, n-1.$$

Considerações:

- $S_1(x)$ é um polinômio de grau 1 em cada sub-intervalo $[x_i, x_{i+1}]$;
- $S_{1,i-1}(x_{i-1}) = y_{i-1}$ e $S_{1,i-1}(x_i) = y_i$, $i = 1, \dots, n$;
- $S_1 \in C^0([a, b])$, por construção $S_{1,i}(x_{i+1}) = y_{i+1} = S_{1,i+1}(x_{i+1})$;
- O erro é dado por

$$\|f - S_1\|_\infty \leq \frac{h^2}{8} \|f''\|_\infty \quad \text{com} \quad h = \max_i \{|x_i - x_{i+1}|\}.$$

Spline Linear

Exemplo 6

Dada a função tabelada $y = f(x)$:

x	1	2	4
y	1	3	5

Calcule a spline linear que interpola a função acima.

Spline Linear

Exemplo 6

Dada a função tabelada $y = f(x)$:

x	1	2	4
y	1	3	5

Calcule a spline linear que interpola a função acima.

Solução:

$$S_1(x) = \begin{cases} S_{1,0}(x) = (2-x) + 3(x-1) = 2x - 1 & , \text{se } x \in [1,2] \\ S_{1,1}(x) = 3\frac{4-x}{2} + 5\frac{x-2}{2} = x + 1 & , \text{se } x \in [2,4] \end{cases}$$

Spline Cúbica

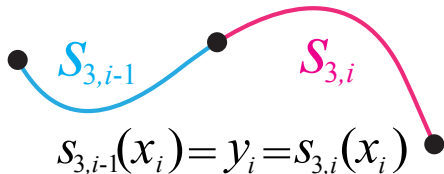
Uma função $S_3(x)$ e uma **spline cúbica** se satisfaz as seguintes condições:

- $S_{3,i} = S_3|_{[x_i, x_{i+1}]}$, com $i = 0, \dots, n - 1$, é um polinômio de grau 3;
- $S_3(x_i) = y_i$, com $i = 0, \dots, n$.
- $S_3 \in \mathcal{C}^2([a, b])$;

Spline Cúbica

Uma função $S_3(x)$ e uma **spline cúbica** se satisfaz as seguintes condições:

- $S_{3,i} = S_3|_{[x_i, x_{i+1}]}$, com $i = 0, \dots, n-1$, é um polinômio de grau 3;
- $S_3(x_i) = y_i$, com $i = 0, \dots, n$.
- $S_3 \in \mathcal{C}^2([a, b])$;



$$S_{3,i-1}(x_i) = y_i = S_{3,i}(x_i)$$

$$S'_{3,i-1}(x_i) = S'_{3,i}(x_i)$$

$$S''_{3,i-1}(x_i) = S''_{3,i}(x_i)$$

Spline Cúbica

Existem n polinômios cúbicos $S_{3,i}(x)$ que satisfazem as condições:

Spline Cúbica

Existem n polinômios cúbicos $S_{3,i}(x)$ que satisfazem as condições:

$$\mathbf{1} \quad S_{3,i-1}(x_{i-1}) = y_{i-1} \text{ e } S_{3,i-1}(x_i) = y_i, \quad i = 1, \dots, n;$$

Spline Cúbica

Existem n polinômios cúbicos $S_{3,i}(x)$ que satisfazem as condições:

- 1 $S_{3,i-1}(x_{i-1}) = y_{i-1}$ e $S_{3,i-1}(x_i) = y_i, \quad i = 1, \dots, n;$
- 2 $S_{3,i-1}(x_i) = S_{3,i}(x_i), \quad i = 1, \dots, n - 1;$

Spline Cúbica

Existem n polinômios cúbicos $S_{3,i}(x)$ que satisfazem as condições:

- 1 $S_{3,i-1}(x_{i-1}) = y_{i-1}$ e $S_{3,i-1}(x_i) = y_i, \quad i = 1, \dots, n;$
- 2 $S_{3,i-1}(x_i) = S_{3,i}(x_i), \quad i = 1, \dots, n - 1;$
- 3 $S'_{3,i-1}(x_i) = S'_{3,i}(x_i), \quad i = 1, \dots, n - 1;$

Spline Cúbica

Existem n polinômios cúbicos $S_{3,i}(x)$ que satisfazem as condições:

- 1 $S_{3,i-1}(x_{i-1}) = y_{i-1}$ e $S_{3,i-1}(x_i) = y_i, \quad i = 1, \dots, n;$
- 2 $S_{3,i-1}(x_i) = S_{3,i}(x_i), \quad i = 1, \dots, n - 1;$
- 3 $S'_{3,i-1}(x_i) = S'_{3,i}(x_i), \quad i = 1, \dots, n - 1;$
- 4 $S''_{3,i-1}(x_i) = S''_{3,i}(x_i), \quad i = 1, \dots, n - 1;$

Spline Cúbica

Existem n polinômios cúbicos $S_{3,i}(x)$ que satisfazem as condições:

- 1 $S_{3,i-1}(x_{i-1}) = y_{i-1}$ e $S_{3,i-1}(x_i) = y_i$, $i = 1, \dots, n$;
- 2 $S_{3,i-1}(x_i) = S_{3,i}(x_i)$, $i = 1, \dots, n - 1$;
- 3 $S'_{3,i-1}(x_i) = S'_{3,i}(x_i)$, $i = 1, \dots, n - 1$;
- 4 $S''_{3,i-1}(x_i) = S''_{3,i}(x_i)$, $i = 1, \dots, n - 1$;
- 5 Condições de contorno **naturais**: $S''_3(x_0) = S''_3(x_n) = 0$.

Spline Cúbica

Existem n polinômios cúbicos $S_{3,i}(x)$ que satisfazem as condições:

- 1 $S_{3,i-1}(x_{i-1}) = y_{i-1}$ e $S_{3,i-1}(x_i) = y_i$, $i = 1, \dots, n$;
- 2 $S_{3,i-1}(x_i) = S_{3,i}(x_i)$, $i = 1, \dots, n - 1$;
- 3 $S'_{3,i-1}(x_i) = S'_{3,i}(x_i)$, $i = 1, \dots, n - 1$;
- 4 $S''_{3,i-1}(x_i) = S''_{3,i}(x_i)$, $i = 1, \dots, n - 1$;
- 5 Condições de contorno **naturais**: $S''_3(x_0) = S''_3(x_n) = 0$.

Por simplicidade, vamos denotar $S_i(x)$ como

$$S_{3,i}(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad i = 0, \dots, n - 1.$$

Spline Cúbica

Existem n polinômios cúbicos $S_{3,i}(x)$ que satisfazem as condições:

- 1 $S_{3,i-1}(x_{i-1}) = y_{i-1}$ e $S_{3,i-1}(x_i) = y_i$, $i = 1, \dots, n$;
- 2 $S_{3,i-1}(x_i) = S_{3,i}(x_i)$, $i = 1, \dots, n - 1$;
- 3 $S'_{3,i-1}(x_i) = S'_{3,i}(x_i)$, $i = 1, \dots, n - 1$;
- 4 $S''_{3,i-1}(x_i) = S''_{3,i}(x_i)$, $i = 1, \dots, n - 1$;
- 5 Condições de contorno **naturais**: $S''_3(x_0) = S''_3(x_n) = 0$.

Por simplicidade, vamos denotar $S_i(x)$ como

$$S_{3,i}(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad i = 0, \dots, n - 1.$$

Para determinar $S_3(x)$ precisamos determinar cada $S_{3,i}(x)$, isto é:

$$\{a_0, b_0, c_0, d_0, \dots, a_{n-1}, b_{n-1}, c_{n-1}, d_{n-1}\} \Rightarrow 4n \text{ incógnitas.}$$

Spline Cúbica

Pelo fato de $S''_{3,i}(x)$ ser uma função linear e da condição (4) $\Rightarrow S''_{3,i}(x)$ é uma **spline linear**.

Spline Cúbica

Pelo fato de $S''_{3,i}(x)$ ser uma função linear e da condição (4) $\Rightarrow S''_{3,i}(x)$ é uma **spline linear**. Definindo $z_i = S''_{3,i}(x_i)$ e $h_i = x_{i+1} - x_i$, logo:

$$S''_{3,i}(x) = \frac{z_i}{h_i} (x_{i+1} - x) + \frac{z_{i+1}}{h_i} (x - x_i).$$

Spline Cúbica

Pelo fato de $S''_{3,i}(x)$ ser uma função linear e da condição (4) $\Rightarrow S''_{3,i}(x)$ é uma **spline linear**. Definindo $z_i = S''_{3,i}(x_i)$ e $h_i = x_{i+1} - x_i$, logo:

$$S''_{3,i}(x) = \frac{z_i}{h_i} (x_{i+1} - x) + \frac{z_{i+1}}{h_i} (x - x_i).$$

Integrando a equação acima 2 vezes, temos:

$$S_{3,i}(x) = \frac{z_i}{6h_i} (x_{i+1} - x)^3 + \frac{z_{i+1}}{6h_i} (x - x_i)^3 + C(x - x_i) + D,$$

onde C e D são constantes de integração.

Spline Cúbica

Determinamos C e D usando o fato de $\Rightarrow S_{3,i}(x_i) = y_i$ e $S_{3,i}(x_{i+1}) = y_{i+1}$:

$$\begin{aligned} S_{3,i}(x) &= \frac{z_i}{6h_i}(x_{i+1} - x)^3 + \frac{z_{i+1}}{6h_i}(x - x_i)^3 \\ &+ \left[\frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6}(z_{i+1} - z_i) \right] (x - x_i) \\ &+ y_i - \frac{z_i h_i^2}{6} \end{aligned}$$

Spline Cúbica

Determinamos C e D usando o fato de $\Rightarrow S_{3,i}(x_i) = y_i$ e $S_{3,i}(x_{i+1}) = y_{i+1}$:

$$\begin{aligned} S_{3,i}(x) &= \frac{z_i}{6h_i}(x_{i+1} - x)^3 + \frac{z_{i+1}}{6h_i}(x - x_i)^3 \\ &+ \left[\frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6}(z_{i+1} - z_i) \right] (x - x_i) \\ &+ y_i - \frac{z_i h_i^2}{6} \end{aligned}$$

Fazendo $x_{i+1} = x_i + h_i$, temos:

$$S_{3,i}(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad \text{com}$$

$$a_i = \frac{z_{i+1} - z_i}{6h_i}, \quad b_i = \frac{z_i}{2}, \quad c_i = -\frac{h_i z_{i+1}}{6} - \frac{h_i z_i}{3} + \frac{y_{i+1} - y_i}{h_i}, \quad d_i = y_i$$

Spline Cúbica

Para determinar z_i , vamos usar a condição (3):

$$S'_{3,i}(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i$$

Supondo $S'_{3,i-1}(x_i) = S'_{3,i}(x_i)$:

$$3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1} = c_i$$

Spline Cúbica

Para determinar z_i , vamos usar a condição (3):

$$S'_{3,i}(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i$$

Supondo $S'_{3,i-1}(x_i) = S'_{3,i}(x_i)$:

$$3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1} = c_i$$

Portanto, para $i = 1, \dots, n - 1$, temos:

$$h_{i-1}z_{i-1} + 2(h_i + h_{i-1})z_i + h_i z_{i+1} = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1})$$

e

$$z_0 = z_n = 0 \quad (\text{condições de contorno naturais}).$$

Spline Cúbica

Melhor, resolvendo o sistema linear de ordem $(n - 1)$:

$$\begin{bmatrix} u_1 & h_1 & & & & \\ h_1 & u_2 & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-3} & u_{n-2} & h_{n-2} & \\ & & & h_{n-2} & u_{n-1} & \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix}$$

onde

$$u_i = 2(h_i + h_{i-1}), \quad v_i = w_i - w_{i-1}, \quad w_i = \frac{6}{h_i}(y_{i+1} - y_i)$$

Spline Cúbica

Melhor, resolvendo o sistema linear de ordem $(n - 1)$:

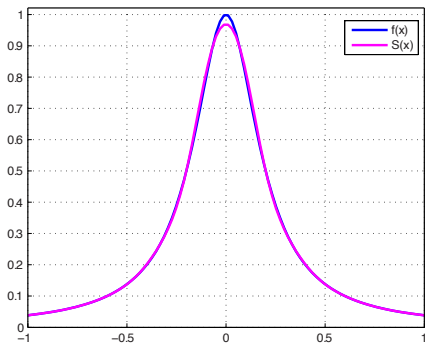
$$\begin{bmatrix} u_1 & h_1 & & & & \\ h_1 & u_2 & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-3} & u_{n-2} & h_{n-2} & \\ & & & h_{n-2} & u_{n-1} & \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix}$$

onde

$$u_i = 2(h_i + h_{i-1}), \quad v_i = w_i - w_{i-1}, \quad w_i = \frac{6}{h_i}(y_{i+1} - y_i)$$

Considerações: as splines cúbicas naturais são obtidas resolvendo um sistema linear $n - 1$ ao invés de um sistema de ordem $4n$.

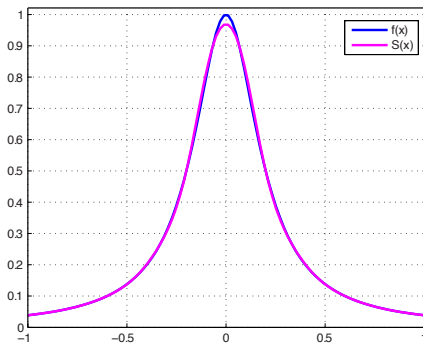
Spline Cúbica



16 nós igualmente espaçados

$$f(x) = \frac{1}{1+25x^2}, \quad x \in [-1, 1].$$

Spline Cúbica



16 nós igualmente espaçados

$$f(x) = \frac{1}{1+25x^2}, \quad x \in [-1, 1].$$

O erro é dado por

$$\|f - S_3\|_{\infty} \leq \frac{5h^4}{384} \|f^{(4)}\|_{\infty} \quad \text{com} \quad h = \max_i \{|x_i - x_{i+1}|\}.$$

Spline Cúbica

Exemplo 7

Dada a função tabelada $y = f(x)$:

x	0	1	2	3
y	3	1	3	2

Calcule uma aproximação de $f(0.5)$ usando uma spline cúbica natural que interpola a função acima.

Spline Cúbica

Solução: Sabemos que $h_0 = h_1 = h_2 = 1$ e $z_0 = z_3 = 0$. Para determinar z_1 e z_2 temos que resolver o sistema abaixo:

Spline Cúbica

Solução: Sabemos que $h_0 = h_1 = h_2 = 1$ e $z_0 = z_3 = 0$. Para determinar z_1 e z_2 temos que resolver o sistema abaixo:

$$\begin{bmatrix} u_1 & h_1 \\ h_2 & u_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Spline Cúbica

Solução: Sabemos que $h_0 = h_1 = h_2 = 1$ e $z_0 = z_3 = 0$. Para determinar z_1 e z_2 temos que resolver o sistema abaixo:

$$\begin{bmatrix} u_1 & h_1 \\ h_2 & u_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 24 \\ -18 \end{bmatrix}$$

Spline Cúbica

Solução: Sabemos que $h_0 = h_1 = h_2 = 1$ e $z_0 = z_3 = 0$. Para determinar z_1 e z_2 temos que resolver o sistema abaixo:

$$\begin{bmatrix} u_1 & h_1 \\ h_2 & u_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 24 \\ -18 \end{bmatrix}$$

Cuja a solução é $z_1 = 7.6$ e $z_2 = -6.4$. Logo,

Spline Cúbica

Solução: Sabemos que $h_0 = h_1 = h_2 = 1$ e $z_0 = z_3 = 0$. Para determinar z_1 e z_2 temos que resolver o sistema abaixo:

$$\begin{bmatrix} u_1 & h_1 \\ h_2 & u_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 24 \\ -18 \end{bmatrix}$$

Cuja a solução é $z_1 = 7.6$ e $z_2 = -6.4$. Logo,

$$S_{3,0}(x) = a_0(x - x_0)^3 + b_0(x - x_0)^2 + c_0(x - x_0) + d_0 \quad \text{com}$$

$$a_0 = (z_1 - z_0)/(6h_0) = 7.6/6 = 1.2667$$

$$b_0 = z_0/2 = 0$$

$$c_0 = -(h_0 z_1)/6 - (h_0 z_0)/3 + (y_1 - y_0)/h_0 = -1.2667 + 1 - 3 = -0.7333$$

$$d_0 = y_0 = 3$$

Spline Cúbica

Solução: Sabemos que $h_0 = h_1 = h_2 = 1$ e $z_0 = z_3 = 0$. Para determinar z_1 e z_2 temos que resolver o sistema abaixo:

$$\begin{bmatrix} u_1 & h_1 \\ h_2 & u_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 24 \\ -18 \end{bmatrix}$$

Cuja a solução é $z_1 = 7.6$ e $z_2 = -6.4$. Logo,

$$S_{3,0}(x) = a_0(x - x_0)^3 + b_0(x - x_0)^2 + c_0(x - x_0) + d_0 \quad \text{com}$$

$$a_0 = (z_1 - z_0)/(6h_0) = 7.6/6 = 1.2667$$

$$b_0 = z_0/2 = 0$$

$$c_0 = -(h_0 z_1)/6 - (h_0 z_0)/3 + (y_1 - y_0)/h_0 = -1.2667 + 1 - 3 = -0.7333$$

$$d_0 = y_0 = 3$$

Portanto, $f(0.5) \approx S_{3,0}(0.5) = 2.7917$

MATLAB: Spline Cúbica Natural

Primeiro vamos resolver o sistema linear $A\mathbf{z} = \mathbf{v}$:

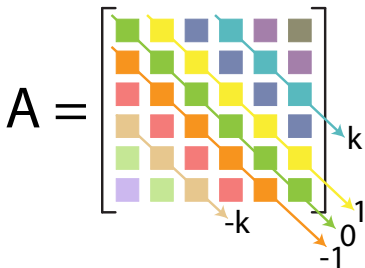
$$A = \begin{bmatrix} u_1 & h_1 & & & & \\ h_1 & u_2 & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & & h_{n-2} & u_{n-1} \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix}$$

onde

$$h_i = x_{i+1} - x_i, \quad u_i = 2(h_i + h_{i-1}), \quad v_i = w_i - w_{i-1}, \quad w_i = \frac{6}{h_i}(y_{i+1} - y_i)$$

Matrizes Diagonais no MATLAB

- Uma matriz $m \times n$ possui $(m + n - 1)$ diagonais.



- $D = \text{diag}(v, k)$: cria uma matriz diagonal dado um vetor v ;
- $v = \text{diag}(A, k)$: retorna a diagonal da matriz A ;
- % k : índice da diagonal entre $(-m + 1)$ e $(n - 1)$;

MATLAB: Spline Cúbica Natural

Vamos montar a matriz tridiagonal:

$$A = \begin{bmatrix} u_1 & h_1 & & & & \\ h_1 & u_2 & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & & h_{n-2} & u_{n-1} \end{bmatrix}$$

onde $u_i = 2(h_i + h_{i-1})$

MATLAB: Spline Cúbica Natural

Vamos montar a matriz tridiagonal:

$$A = \begin{bmatrix} u_1 & h_1 & & & & & \\ h_1 & u_2 & h_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & h_{n-3} & u_{n-2} & h_{n-2} & \\ & & & & h_{n-2} & u_{n-1} & \end{bmatrix}$$

onde $u_i = 2(h_i + h_{i-1})$



```
h = xi(2:end) - xi(1:end-1);
```

```
u = 2*(h(1:end-1)+h(2:end));
```

```
A = diag(h(2:end-1),-1) + diag(u) + diag(h(2:end-1),1);
```

MATLAB: Spline Cúbica Natural

Vamos montar a matriz tridiagonal:

$$A = \begin{bmatrix} u_1 & h_1 & & & & \\ h_1 & u_2 & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & & h_{n-2} & u_{n-1} \end{bmatrix}$$

onde $u_i = 2(h_i + h_{i-1})$



```
h = xi(2:end) - xi(1:end-1);
```

```
u = 2*(h(1:end-1)+h(2:end));
```

```
A = diag(h(2:end-1),-1) + diag(u) + diag(h(2:end-1),1);
```

A possui muitos zeros (matriz esparsa) \Rightarrow gasto de memória!

Matrizes Esparsas no MATLAB



`S = sparse(A)`: converte uma matriz cheia para esparsa;

`A = full(S)`: converte uma matriz esparsa para cheia;

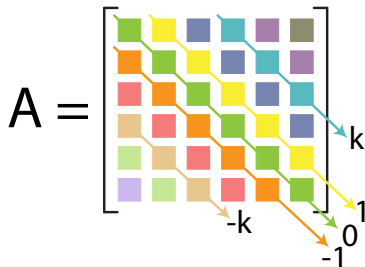


`S = sparse(m,n)`: cria uma esparsa $m \times n$;

`S = sparse(i,j,val)`: cria uma esparsa com $S(i(k),j(k)) = val(k)$;

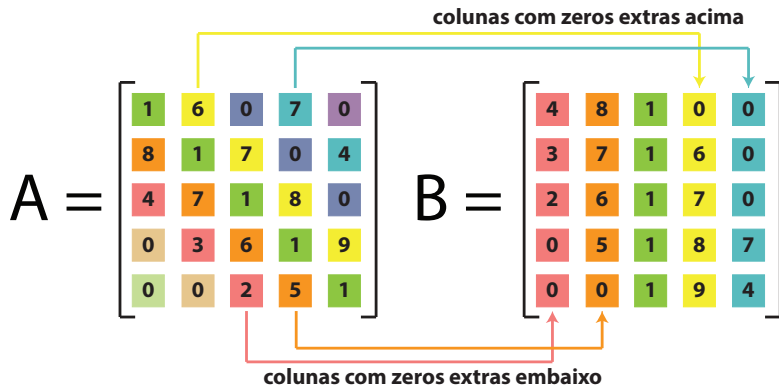
`[i,j,val] = find(S)`: encontra índices e coeficientes não-nulos;

Matrizes Diagonais Esparsas no MATLAB

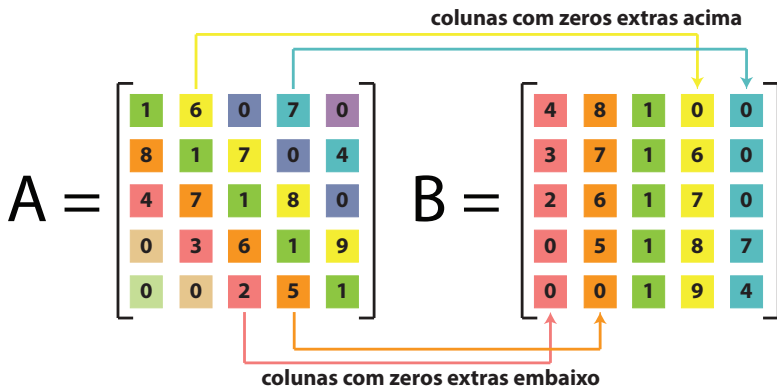


$A = \text{spdiags}(B, d, m, n)$: cria uma matriz diagonal esparsa $m \times n$;
 % B: matriz cujas colunas são as diagonais de A;
 % d: índices das diagonais entre $(-m + 1)$ e $(n - 1)$;

Matrizes Diagonais Esparsas no MATLAB



Matrizes Diagonais Esparsas no MATLAB



```
n = length(xi);
B = [ [h(2:end-1); 0] u [0; h(2:end-1)] ];
A = spdiags(B,-1:1,n-2,n-2);
```

MATLAB: Spline Cúbica Natural

Determinado o vetor solução $z=A \setminus v$, finalmente vem a montagem das splines cúbicas:

$$S_{3,i}(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad \text{com}$$

$$a_i = \frac{z_{i+1} - z_i}{6h_i}, \quad b_i = \frac{z_i}{2}, \quad c_i = -\frac{h_i z_{i+1}}{6} - \frac{h_i z_i}{3} + \frac{y_{i+1} - y_i}{h_i}, \quad d_i = y_i$$

MATLAB: Spline Cúbica Natural

Determinado o vetor solução $z=A \setminus v$, finalmente vem a montagem das splines cúbicas:

$$S_{3,i}(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad \text{com}$$

$$a_i = \frac{z_{i+1} - z_i}{6h_i}, \quad b_i = \frac{z_i}{2}, \quad c_i = -\frac{h_i z_{i+1}}{6} - \frac{h_i z_i}{3} + \frac{y_{i+1} - y_i}{h_i}, \quad d_i = y_i$$



```
a = (z(2:end)-z(1:end-1))./(6*h);
```

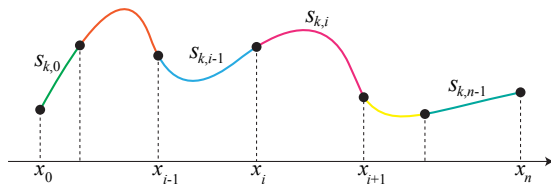
```
b = z(1:end-1)/2;
```

```
c = -(h/6).*(z(2:end)+2*z(1:end-1)) + (yi(2:end)-yi(1:end-1))./h;
```

```
d = yi(1:end-1);
```

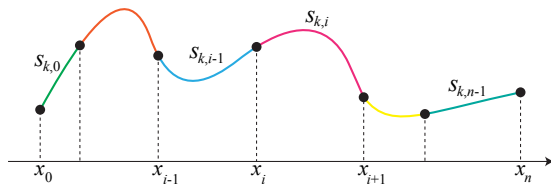
Polinômio por Partes no MATLAB

Relembrando que um polinômio $P_n(x) = a_n x^n + \dots + a_2 x^2 + a_1 x + a_0$ é representado no MATLAB por um vetor da forma $p = [a_n, \dots, a_2, a_1, a_0]$



Polinômio por Partes no MATLAB

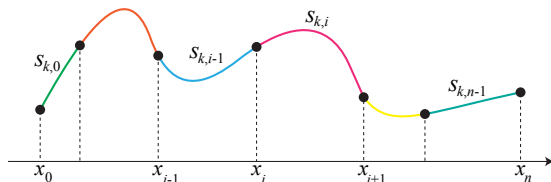
Relembrando que um polinômio $P_n(x) = a_n x^n + \dots + a_2 x^2 + a_1 x + a_0$ é representado no MATLAB por um vetor da forma $p = [a_n, \dots, a_2, a_1, a_0]$




```
pp = mkpp(xi, coefs): cria polinômio por partes;
% xi: nós do polinômio;
% coefs: matriz dos coeficientes do polinômio;
```

Polinômio por Partes no MATLAB


Relembrando que um polinômio $P_n(x) = a_n x^n + \dots + a_2 x^2 + a_1 x + a_0$ é representado no MATLAB por um vetor da forma $p = [a_n, \dots, a_2, a_1, a_0]$



 `pp = mkpp(xi, coefs)`: cria polinômio por partes;

`% xi`: nós do polinômio;

`% coefs`: matriz dos coeficientes do polinômio;

 `s = ppval(pp, x)`: avalia um polinômio por partes;

`% pp`: polinômio por partes;

`% x`: pontos a serem avaliados;

MATLAB: Spline Cúbica Natural

```
1 function s = cubic_spline(xi,yi,x)
2
3 [n,m]= size (xi);
4 if (n == 1) xi = xi'; yi = yi'; n = m; end
5
6 h = xi(2:end) - xi(1:end-1);
7 u = 2*(h(1:end-1)+h(2:end));
8 A = spdiags([ h(2:end-1);0] u [0;h(2:end-1)]),-1:1,n-2,n-2);
9 w = 6*(yi(2:end)-yi(1:end-1))./h;
10 v = w(2:end)-w(1:end-1);
11 z = A\v; z = [0;z;0];
12
13 a = (z(2:end)-z(1:end-1))./(6*h);
14 b = z(1:end-1)/2;
15 c = -(h/6).* (z(2:end)+2*z(1:end-1))+(yi(2:end)-yi(1:end-1))./h;
16 d = yi(1:end-1);
17
18 pp = mkpp (xi,[a b c d]);
19 s = ppval(pp ,x);
```

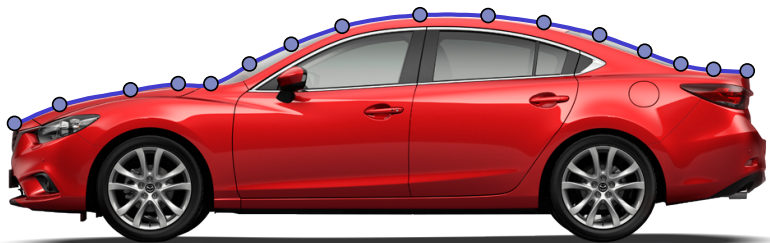
Splines no MATLAB

O MATLAB implementa a sua própria spline cúbica através da função:



```
s = spline(xi,yi,x): avalia uma spline cúbica;  
% xi,yi: pontos de interpolação;  
% x: pontos a serem avaliados;
```

CAD em Engenharia



x_i	18.5	73.5	160	218	258	305	356	418	513	596	664	732	787	831	871	912
y_i	157.5	108.5	198.5	206	206	230	254	276	290	289	280	265	245.5	230	223.5	221.5