Universidade do Minho
Escola de Ciências

Departamento de Matemática

Interpolação

- Interpolação Polinomial
- Interpolação segmentadas
- Funções Spline

Interpolação Polinomial

Seja $\mathcal{P}_{n-1}$ o conjunto de todos os polinómios (de coeficientes reais) de grau $\leq n-1$.

O Problema de Interpolação

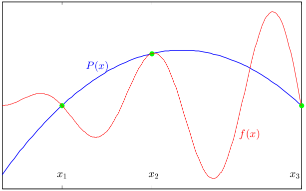
Dados $x_1, \ldots, x_n$ **distintos** e $y_1, \ldots, y_n$, construir um polinómio $P_{n-1} \in \mathcal{P}_{n-1}$ que satisfaça

$$P_{n-1}(x_i) = y_i, \quad i = 1, \ldots, n.$$

Frequentemente, $y_1, \ldots, y_n$ são valores de uma certa função f nos pontos $x_1, \ldots, x_n$, isto é,

$$y_i = f(x_i); \; i = 1, \ldots, n.$$

Nesse caso, dizemos que P_{n-1} interpola f nos pontos x_i. P_{n-1} é usado, geralmente, para estimar o valor de f num certo ponto x (em princípio, situado no intervalo $(\min\{x_i\}, \max\{x_i\})$).



A função $f(x)$, os pontos x_1, x_2, x_3 e o polinómio interpolador $P(x)$

Existência e Unicidade

Teorema

Sejam $x_1, \dots, x_n$, n pontos distintos (em $\mathbb{R}$) e sejam $y_1, \dots, y_n$ números reais dados. Então, **existe um e um só** *polinómio $P_{n-1} \in \mathcal{P}_{n-1}$ que satisfaz*

$$P_{n-1}(x_i) = y_i; i = 1, \dots, n.$$

Demonstração:

➠ Existência Se encontrarmos n polinómios $L_1(x), L_2(x), \dots, L_n(x)$ de grau $n-1$ tais que $L_i(x_j) = \delta_{ij}$, então o polinómio P_{n-1} dado por

$$P_{n-1}(x) = L_1(x)y_1 + L_2(x)y_2 + \dots + L_n(x)y_n$$

estará em $\mathcal{P}_{n-1}$ e satisfará as condições de interpolação. Como

- ➠ $L_i(x_j) = 0;\ j \neq i \Longrightarrow L_i(x) = A_i(x - x_1)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n)$
- ➠ $L_i(x_i) = 1 \Longrightarrow A_i = \dfrac{1}{(x_i - x_1)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)}$

obtém-se

$$L_i(x) = \prod_{\substack{j=1 \\ j\neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right); i = 1, \dots, n.$$

Está assim provada a existência do polinómio P_{n-1}. Os polinómios $L_i(x)$ são chamados polinómios de Lagrange relativos aos pontos x_i.

Demonstração: (cont.)

➠ **Unicidade** Sejam P_{n-1} e Q_{n-1} polinómios de grau não superior a $n-1$ tais que

$$P_{n-1}(x_i) = Q_{n-1}(x_i) = y_i; \; i = 1, \dots, n.$$

Seja $D_{n-1} := P_{n-1} - Q_{n-1}$. Então:

- ➠ D_{n-1} é um polinómio de grau $\leq n-1$
- ➠ $D_{n-1}(x_i) = y_i - y_i = 0; i = 1, \dots, n \Longrightarrow P_{n-1} \equiv Q_{n-1}$

Está assim provada a unicidade do polinómio P_{n-1}. □

➲ O polinómio

$$\boxed{P_{n-1}(x) = \sum_{i=1}^{n} \left(\prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \right) y_i}$$

é o único polinómio de grau não superior a $n-1$ que satisfaz as condições de interpolação

$P_{n-1}(x_i) = y_i; i = 1, \dots, n.$

➲ A expressão anterior do polinómio interpolador é dita forma de Lagrange do polinómio interpolador.

Forma de Lagrange do polinómio interpolador

Dados: x, y e z

Resultado: $P_{n-1}(z)$

```
for i=1:length(z)
  for k=1:length(x)   % Polinómios de Lagrange
     L(i,k)=prod((z(i)-drop(x,k))./(x(k)-drop(x,k)));
  end
end
valp=L*y % x e y devem ser vetores coluna
```

Escreva a função `drop` para formar o vetor $(x_1,\ldots,x_{k-1},x_{k+1},\ldots,x_n)$

Exemplo: Interpolação linear: pontos (x_1,y_1) e (x_2,y_2).

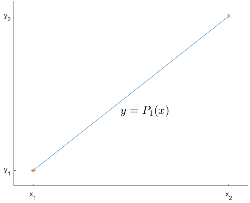
Polinómios de Lagrange:

$$L_1(x)=\frac{x-x_2}{x_1-x_2},\qquad L_2(x)=\frac{x-x_1}{x_2-x_1}$$

Polinómio interpolador:

$$P_1(x)=y_1+\left(\frac{y_2-y_1}{x_2-x_1}\right)(x-x_1)$$

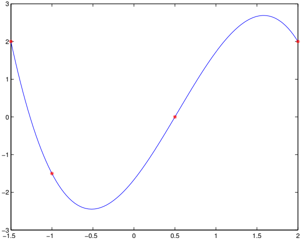
[Equação da reta que passa pelos pontos]



Exemplo: Representar graficamente o polinómio interpolador dos pontos da tabela.

x	-1.5	-1	0.5	2
y	2	-1.5	0	2

```
>> x=[-1.5;-1;0.5;2];
>> y=[2;-1.5;0;2];
>> z=-1.5:0.01:2;
>> valpz=polLagrange(x,y,z);
>> plot(z,valpz,'b',x,y,'r*')
```



Erro em Interpolação Polinomial

Teorema

Seja P_{n-1} o polinómio de grau $\leq n-1$ interpolador de uma função y em n pontos distintos $x_1, x_2, \ldots, x_n$. Se $y \in \mathscr{C}^n[a,b]$, onde $[a,b]$ é um intervalo que contém os pontos x_i, então, para todo o ponto $x \in [a,b]$, existe $\xi_x = \xi_x(x) \in (a,b)$ tal que

$$E_{n-1}(x) := y(x) - P_{n-1}(x) = \frac{y^{(n)}(\xi_x)}{n!} \prod_{i=1}^{n} (x - x_i)$$

Demonstração: Se $x = x_i$ para algum i, o resultado é trivialmente verdadeiro. Se x não é nenhum dos pontos de interpolação, definamos a seguinte função (na variável t, considerando o valor de x fixo):

$$Y(t) := y(t) - P_{n-1}(t) - \left\{ \frac{y(x) - P_{n-1}(x)}{\Pi_n(x)} \right\} \Pi_n(t),$$

onde

$$\Pi_n(t) := \prod_{i=1}^{n} (t - x_i).$$

Como $Y \in \mathscr{C}^n[a,b]$ e

- $Y(x_i) = y(x_i) - P_{n-1}(x_i) - E_{n-1}(x) \frac{\Pi_n(x_i)}{\Pi_n(x)} = 0; \; i = 1, \ldots, n,$
- $Y(x) = E_{n-1}(x) - E_{n-1}(x) \frac{\Pi_n(x)}{\Pi_n(x)} = 0,$

pelo Teorema de Rolle generalizado, conclui-se que $Y^{(n)}(\xi_x) = 0$, para algum $\xi_x \in (\min\{x_1, \ldots, x_n, x\}, \max\{x_1, \ldots, x_n, x\})$.

Como

$$Y^{(n)}(t) = y^{(n)}(t) - \frac{E_{n-1}(x)}{\Pi_n(x)} n!,$$

então

$$Y^{(n)}(\xi_x) = 0 \Leftrightarrow y^{(n)}(\xi_x) - \frac{E_{n-1}(x)}{\Pi_n(x)} n! = 0 \Leftrightarrow E_{n-1}(x) = \frac{y^{(n)}(\xi_x)}{n!} \Pi_n(x),$$

$$\Leftrightarrow E_{n-1}(x) = \frac{y^{(n)}(\xi_x)}{n!} \prod_{i=1}^{n} (x - x_i).$$

□

- Se $x \in (\min\{x_1, \ldots, x_{n-1}\}, \max\{x_1, \ldots, x_{n-1}\}) \longrightarrow$ interpolação
- Se $x \notin (\min\{x_1, \ldots, x_{n-1}\}, \max\{x_1, \ldots, x_{n-1}\}) \longrightarrow$ extrapolação

Em geral, não conhecemos $y^{(n)}$ nem $\xi_x = \xi(x)$. Se conhecermos M_n tal que $\max_{x \in [a,b]} |y^{(n)}(x)| \leq M_n$ teremos o seguinte limite superior para o erro (em valor absoluto)

$$\boxed{|E_{n-1}(x)| = |y(x) - P_{n-1}(x)| \leq \frac{M_n}{n!} \prod_{i=1}^{n} |x - x_i|.}$$

Nós igualmente espaçados

Se

$$x_i = a + (i-1)h; i = 1, 2, \ldots, n, \quad \text{onde} \quad h = \frac{b-a}{n-1},$$

então

$$\boxed{|E_{n-1}(x)| = |y(x) - P_{n-1}(x)| \leq \frac{h^n M_n}{4n}, \quad \forall x \in [a, b].}$$

Prove!

Exercício

Uma majoração mais cuidada de $\prod_{i=1}^{n} |x - x_i|$ permite melhorar os majorantes obtidos pela fórmula anterior. Obtenha os seguintes majorantes para o erro (casos $n = 2, 3, 4$):

$$|f(x) - P_1(x)| \leq \frac{h^2}{8} M_2$$

$$|f(x) - P_2(x)| \leq \frac{\sqrt{3}\, h^3}{27} M_3$$

$$|f(x) - P_3(x)| \leq \frac{h^4}{24} M_4.$$

Exemplo: Interpolação Linear $n = 2$

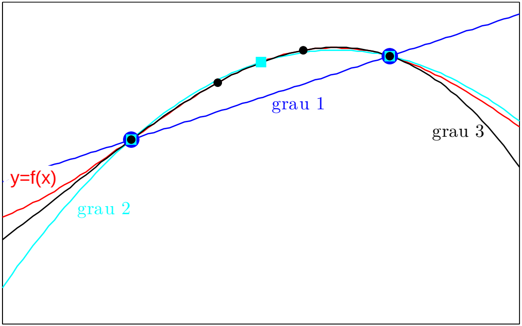
Vamos demonstrar o resultado anterior no caso $n = 2$. Seja $h := x_2 - x_1$. Temos, então

$$|y(x) - P_1(x)| \leq \frac{M_2}{2} |(x - x_1)(x - x_2)|.$$

Facilmente se verifica que $\max_{x \in [x_1, x_2]} |(x - x_1)(x - x_2)| = \frac{h}{2} \cdot \frac{h}{2} = \frac{h^2}{4}$.

Assim, tem-se $|E_1(x)| \leq \frac{h^2}{8} M_2, \quad \forall x \in [x_1, x_2].$ $\square$

Polinómios Interpoladores



Considerações sobre o erro de interpolação

$$|y(x) - P_{n-1}(x)| \leq \frac{M_n}{n!} \, |\underbrace{(x - x_1) \ldots (x - x_n)}_{\Pi_n(x)}|$$

- $|\Pi_n(x)|$ cresce muito rapidamente à medida que x se afasta de $[\min\{x_i\}, \max\{x_i\}]$.
- O erro é nulo quando x é um dos nós x_i, devendo, por continuidade, ser "pequeno", se x estiver próximo de um nó $\Rightarrow$ devemos interpolar em nós situados à volta do ponto onde pretendemos estimar $y(x)$.
- Nós igualmente espaçados Se existir uma constante M, independente de n, tal que, $|y^{(n)}(x)| \leq M, \ \forall x \in [a, b]$, i.e. se as derivadas forem uniformemente limitadas em $[a, b]$, então

$$\max_{x \in [a,b]} |y(x) - P_{n-1}(x)| \leq \frac{h^n M}{4\,n} \Longrightarrow \lim_{n \to \infty} \max_{x \in [a,b]} |y(x) - P_n(x)| = 0.$$

A sequência de polinómios interpoladores $(P_n)_{n \in \mathbb{N}}$ converge uniformemente para a função y em $[a, b]$.

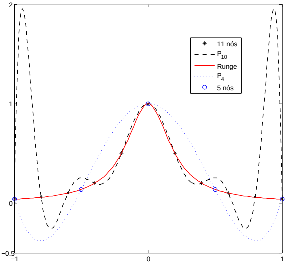
Fenómeno de Runge

Existem funções para as quais a interpolação num número crescente de pontos equidistantes não produz uma sequência de polinómios convergindo uniformemente para a função. Um exemplo clássico é a chamada função de Runge, definida, em $[-1, 1]$, por

$$y(x) = \frac{1}{1 + 25x^2}.$$

Esta função é infinitamente derivável.[1] No entanto, sendo P_n o polinómio de grau não superior a n interpolador de y em $n+1$ pontos igualmente espaçados no intervalo $[-1, 1]$, pode provar-se que a sequência de polinómios $(P_n)_{n\in\mathbb{N}}$ não converge uniformemente para y no intervalo $[-1, 1]$, tendo-se

$$\lim_{n\to\infty} \max_{x\in[-1,1]} |y(x) - P_n(x)| = +\infty.$$



Interpolação da função de Runge por P_4 e P_{10}

[1] Mas, as suas derivadas não são uniformemente limitadas.

Forma de Newton do polinómio interpolador

Seja P_{n-2} o polinómio interpolador dos valores $y_1, y_2, \ldots, y_{n-1}$ nos pontos $x_1, x_2, \ldots, x_{n-1}$ e P_{n-1} o polinómio interpolador dos valores $y_1, y_2, \ldots, y_{n-1}$ e y_n nos pontos $x_1, x_2, \ldots, x_{n-1}$ e x_n.

Objetivo

Obter uma representação do polinómio interpolador que permita obter P_{n-1}, a partir de P_{n-2}, por junção de mais um termo.

$$P_{n-1}(x) = P_{n-2}(x) + Q(x); \; n = 1, 2, \ldots; P_0(x) = y_1.$$

Segue-se, de imediato, que Q deverá ser um polinómio de grau $\leq n-1$ e que $Q(x_i) = 0$; $i = 1, \ldots, n-1$. Logo

$$Q(x) \equiv Q_{n-1}(x) = d_{n-1}(x - x_1)\ldots(x - x_{n-1}).$$

e

$$P_{n-1}(x) = P_{n-2}(x) + d_{n-1}\prod_{k=1}^{n-1}(x - x_k)$$

Como

$$P_{n-1}(x) = \sum_{k=1}^{n} \left(\prod_{\substack{j=1 \\ j\neq k}}^{n} \left(\frac{x - x_j}{x_k - x_j} \right) \right) y_k,$$

igualando o coeficiente de x^{n-1} nas expressões anteriores, vem

$$d_{n-1} = \sum_{k=1}^{n} \left(\frac{y_k}{\prod_{\substack{j=1 \\ j\neq k}}^{n} (x_k - x_j)} \right)$$

➠ d_{n-1} diz-se a diferença dividida (de ordem $n-1$) dos valores $y_1, \ldots, y_n$ (relativamente aos pontos $x_1, \ldots, x_n$) e denota-se por $[y_1, \ldots, y_n]$, ou $y[x_1, \ldots, x_n]$

$$\boxed{[y_1, \ldots, y_n] := \sum_{k=1}^{n} \left(\frac{y_k}{\prod_{\substack{j=1 \\ j\neq k}}^{n} (x_k - x_j)} \right)}$$

➠ O valor da diferença dividida é independente da ordem dos seus argumentos.

Forma de Newton com diferenças divididas

$$P_{n-1}(x) = P_{n-2}(x) + [y_1, \dots, y_n] \prod_{k=1}^{n-1} (x - x_k)$$

$$= P_{n-3}(x) + [y_1, \dots, y_{n-1}] \prod_{k=1}^{n-2} (x - x_k) + [y_1, \dots, y_n] \prod_{k=1}^{n-1} (x - x_k)$$

$$\vdots$$

$$P_{n-1}(x) = y_1 + [y_1, y_2](x - x_1) + [y_1, y_2, y_3](x - x_1)(x - x_2)$$

$$+ \dots + [y_1, y_2, \dots, y_n](x - x_1)(x - x_2) \dots (x - x_{n-1})$$

$$\boxed{P_{n-1}(x) = \sum_{k=1}^{n} [y_1, \dots, y_k](x - x_1) \dots (x - x_{k-1})}$$

Forma de Newton do polinómio interpolador com diferenças divididas

Teorema (Fórmula recursiva para o cálculo das diferenças divididas)

$$[y_1, y_2, \ldots, y_n] = \frac{[y_2, \ldots, y_n] - [y_1, \ldots, y_{n-1}]}{x_n - x_1}; n = 2, 3, \ldots,$$

com $[y_i] := y_i$.

Demonstração: Seja $P(x)$ o polinómio interpolador dos pontos $(x_1, y_1), \ldots, (x_n, y_n)$.

- ordem dos nós de interpolação: $x_2, \ldots, x_{n-1}, x_1, x_n$

$$P(x) = y_2 + [y_2, y_3](x - x_2) + \cdots + [y_2, \ldots, y_{n-1}, y_1](x - x_2) \ldots (x - x_{n-1}) + [y_2, \ldots, y_{n-1}, y_1, y_n](x - x_2) \ldots (x - x_{n-1})(x - x_1)$$

- ordem dos nós de interpolação: $x_2, \ldots, x_{n-1}, x_n, x_1$

$$P(x) = y_2 + [y_2, y_3](x - x_2) + \cdots + [y_2, \ldots, y_{n-1}, y_n](x - x_2) \ldots (x - x_{n-1}) + [y_2, \ldots, y_{n-1}, y_n, y_1](x - x_2) \ldots (x - x_{n-1})(x - x_n)$$

Igualando estas duas formas do mesmo polinómio e cancelando os termos iguais, vem

$$[y_2, \ldots, y_{n-1}, y_1](x - x_2) \ldots (x - x_{n-1}) + [y_2, \ldots, y_{n-1}, y_1, y_n](x - x_2) \ldots (x - x_{n-1})(x - x_1)$$

$$= [y_2, \ldots, y_{n-1}, y_n](x - x_2) \ldots (x - x_{n-1}) + [y_2, \ldots, y_{n-1}, y_n, y_1](x - x_2) \ldots (x - x_{n-1})(x - x_n)$$

Como as diferenças divididas são independentes da ordem dos argumentos, obtém-se

$$[y_1, y_2, \ldots, y_{n-1}](x - x_2)\ldots(x - x_{n-1})$$
$$+ [y_1, y_2, \ldots, y_n](x - x_1)(x - x_2)\ldots(x - x_{n-1})$$

$$= [y_2, \ldots, y_{n-1}, y_n](x - x_2)\ldots(x - x_{n-1})$$
$$+ [y_1, y_2, \ldots, y_n](x - x_2)\ldots(x - x_{n-1})(x - x_n)$$

ou seja

$$[y_1, y_2, \ldots, y_{n-1}] + [y_1, y_2, \ldots, y_n](x - x_1) = [y_2, \ldots, y_{n-1}, y_n] + [y_1, y_2, \ldots, y_n](x - x_n).$$

Logo

$$[y_1, y_2, \ldots, y_n](x - x_1 - x + x_n) = [y_2, \ldots, y_{n-1}, y_n] - [y_1, y_2, \ldots, y_{n-1}],$$

e o resultado obtém-se de imediato. □

Tabela de Diferenças Divididas

x	y ([ ])	[,]	[, ,]	[, , ,]	[, , , ,]
x_1	$d_{11} = y_1$				
		$d_{12} = \frac{y_2 - y_1}{x_2 - x_1}$			
x_2	$d_{21} = y_2$		$d_{13} = \frac{d_{22} - d_{12}}{x_3 - x_1}$		
		$d_{22} = \frac{y_3 - y_2}{x_3 - x_2}$		$d_{14} = \frac{d_{23} - d_{13}}{x_4 - x_1}$	
x_3	$d_{31} = y_3$		$d_{23} = \frac{d_{32} - d_{22}}{x_4 - x_2}$		$d_{15} = \frac{d_{24} - d_{14}}{x_5 - x_1}$
		$d_{32} = \frac{y_4 - y_3}{x_4 - x_3}$		$d_{24} = \frac{d_{33} - d_{23}}{x_5 - x_2}$	
x_4	$d_{41} = y_4$		$d_{33} = \frac{d_{42} - d_{32}}{x_5 - x_3}$		
		$d_{42} = \frac{y_5 - y_4}{x_5 - x_4}$			
x_5	$d_{51} = y_5$				

Exemplo:

x	y	[,]	[, ,]	[, , ,]	[, , , ,]
0	1.0000				
		−0.0995			
0.2	0.9801		−0.4887		
		−0.2950		0.0442	
0.4	0.9211		−0.4667		0.0375
		−0.4350		0.0667	
0.5	0.8776		−0.4400		
		−0.5230			
0.6	0.8253				

Diferenças Divididas - algoritmo

Dados: $x_1, x_2 \cdots x_n$ e $y_1, y_2 \cdots y_n$

Resultado: matriz d de ordem n com as diferenças divididas

```
n=length(x);
d=zeros(n);        % Inicialização da matriz d
d(:,1)=y;          % a primeira coluna de d é o vetor y
for k=2:n
   for i=1:n-k+1   % Cálculo recursivo das dif. divididas e armazenamento
      d(i,k)=(d(i+1,k-1)-d(i,k-1))/(x(k+i-1)-x(i));   %  nas colunas de d
   end
end
```

Exemplo:

```
>> x=[1 3 4 7 10]; y=log(x);

>> tabDifDiv(x,y)

ans =
         0    0.5493   -0.0872    0.0103   -0.0009
    1.0986    0.2877   -0.0253    0.0020         0
    1.3863    0.1865   -0.0113         0         0
    1.9459    0.1189         0         0         0
    2.3026         0         0         0         0
```

Polinómio interpolador com Diferenças Divididas - algoritmo

Dados: $x, y\ z$

Resultado: $P_{n-1}(z)$

```
DD = tabDifDiv(x,y); % tabela das diferenças divididas
dfdv = DD(1,:); % Primeira linha de DD
n = length(dfdv);
valpol = ones(size(z)); % Inicialização do vetor valpol
valpol(:) = dfdv(n);
for i=n-1:-1:1
   valpol = dfdv(i)+valpol.*(z-x(i)); % Uso da forma encaixada
end
```

Exemplo:

```
>> x=[1 3 4 7 10]; y=log(x);
>> z=[3.5 4.5];
>> p=polDifDiv(x,y,z)
   p =
     1.2558    1.4979
>> erro=abs(p-log(z))
   erro =
        0.0030    0.0062
```

Fórmula do erro com diferenças divididas

Seja P_{n-1} o polinómio interpolador de f em $x_1, \ldots, x_n$ e P_n o polinómio interpolador de f em $x_1, \ldots, x_n, x, \quad x \neq x_i$. Então

$$P_n(t) = P_{n-1}(t) + y[x_1, \ldots, x_n, x](t - x_1) \ldots (t - x_n),$$

pelo que

$$P_n(x) = P_{n-1}(x) + y[x_1, \ldots, x_n, x](x - x_1) \ldots (x - x_n).$$

Mas, $P_n(x) = f(x)$, donde

$$f(x) = P_{n-1}(x) + y[x_1, \ldots, x_n, x](x - x_1) \ldots (x - x_n)$$

ou seja

$$E_{n-1}(x) = f(x) - P_{n-1}(x) = y[x_1, \ldots, x_n, x](x - x_1) \ldots (x - x_n)$$

Para “calcular”$y[x_1, \ldots, x_n, x]$ precisamos do valor $f(x)$; se as diferenças divididas de ordem n não variarem muito, podemos tomar $y[x_1, \ldots, x_n, x_{n+1}]$ como aproximação para $y[x_1, \ldots, x_n, x]$, obtendo, assim, a seguinte estimativa para o erro:

$$\boxed{E_{n-1}(x) \approx y[x_1, \ldots, x_n, x_{n+1}](x - x_1) \ldots (x - x_n)}$$

Erro é aproximadamente dado pelo valor do “próximo termo”!

Diferenças divididas e derivadas

Teorema

Se $f \in \mathscr{C}^{n-1}[a,b]$ e $x_1, \ldots, x_n \in [a,b]$ (distintos), então,

$$y[x_1, \ldots, x_n] = \frac{f^{(n-1)}(\xi)}{(n-1)!}$$

para um certo $\xi \in (a,b)$.

Demonstração:

Seja P_{n-2} o polinómio interpolador de f em $x_1, \ldots, x_{n-1}$. Então, existe $\xi \in (\min\{x_0, \ldots, x_{n-1}, x_n\}, \max\{x_0, \ldots, x_{n-1}, x_n\})$ tal que

$$f(x_n) - P_{n-2}(x_n) = \frac{f^{(n-1)}(\xi)}{(n-1)!}(x_n - x_1)\ldots(x_n - x_{n-1}).$$

Por outro lado, temos também

$$f(x_n) - P_{n-2}(x_n) = y[x_1, \ldots, x_n](x_n - x_1)\ldots(x_n - x_{n-1})$$

e o resultado segue de imediato.

Interpolação em pontos equidistantes - Diferenças finitas

Como simplificar a fórmula do polinómio interpolador, quando os pontos de interpolação são igualmente espaçados, i.e.

$$x_i = x_1 + (i-1)h; i = 1, 2, \dots.$$

➲ Chama-se diferença **descendente** de ordem k (e passo h) de y_i e denota-se por $\triangle_h^k y_i$ (ou $\triangle^k y_i$ ou $\triangle_i^k$) à quantidade definida por:

- $\triangle^1 y_i := y_{i+1} - y_i$
- $\Delta^k y_i := \triangle(\triangle^{k-1} y_i), \ k = 2, 3, \dots$ $\qquad \triangle y_i \equiv \triangle^1 y_i.$

➲ Chama-se diferença **ascendente** de ordem k (e passo h) de y_i e denota-se por $\nabla_h^k y_i$ (ou $\nabla^k y_i$ ou ∇_i^k) à quantidade definida por:

- $\nabla^1 y_i := y_i - y_{i-1}$
- $\nabla^k y_i := \nabla(\nabla^{k-1} y_i), \ k = 2, 3, \dots$ $\qquad \nabla y_i \equiv \nabla^1 y_i.$

Tabela de diferenças descendentes

x	y	$\triangle$	$\triangle^2$	$\triangle^3$	$\triangle^4$
x_1	y_1				
		$\triangle_1 = y_2 - y_1$			
x_2	y_2		$\triangle_1^2 = \triangle_2 - \triangle_1$		
		$\triangle_2 = y_3 - y_2$		$\triangle_1^3 = \triangle_2^2 - \triangle_1^2$	
x_3	y_3		$\triangle_2^2 = \triangle_3 - \triangle_2$		$\triangle_1^4 = \triangle_2^3 - \triangle_1^3$
		$\triangle_3 = y_4 - y_3$		$\triangle_2^3 = \triangle_3^2 - \triangle_2^2$	
x_4	y_4		$\triangle_3^2 = \triangle_4 - \triangle_3$		
		$\triangle_4 = y_5 - y_4$			
x_5	y_5				

$$\triangle^k y_i = \nabla^k y_{i+k}$$

$$\triangle^k y_i = \sum_{j=0}^{k} (-1)^{k+j} \binom{k}{j} y_{i+j}$$

$$[y_1, y_2, \ldots, y_k] = \frac{\triangle^{k-1} y_1}{h^{k-1}(k-1)!}$$

Forma de Newton com diferenças descendentes

- $P_{n-1}(x) = y_1 + \sum_{k=1}^{n-1} \frac{\triangle^k y_1}{h^k k!}(x - x_1)(x - x_2)\dots(x - x_k)$
- **Mudança de variável**: $s = s(x) = \dfrac{x - x_1}{h} \iff x = x(s) = x_1 + sh$

$$x - x_i = x - (x_1 + (i-1)h) = (s - i + 1)h$$

Forma encaixada diferenças descendentes

$$P_{n-1}(x_1 + sh) = y_1 + s\left\{\triangle y_1 + \frac{(s-1)}{2}\left\{\triangle^2 y_1 + \frac{(s-2)}{3}\left\{\triangle^3 y_1 + \right.\right.\right.$$
$$\left.\left.\left. + \dots + \left\{\triangle^{n-2} y_1 + \frac{(s-n+2)}{n-1}\triangle^{n-1} y_1\right\}\dots\right\}\right\}\right\}$$

Deduza a forma de Newton com diferenças ascendentes!

Diferenças Finitas - algoritmo

Dados: $y_1, y_2 \cdots y_n$

Resultado: matriz df de ordem n com as diferenças finitas

```
n=length(y);
df=zeros(n);        % Inicialização da matriz df
df(:,1)=y;          % a primeira coluna de df é o vetor y
for k=1:n-1   % Cálculo das diferenças finitas (usando a função DIFF)
   df(1:n-k,k+1)=diff(y,k);   %   e armazenamento nas colunas de df
end
```

Exemplo:

```
>> x=1:2:10; y=log(x);
>> tabDifFin(y)

0       1.0986   -0.5878    0.4134   -0.3242
1.0986    0.5108   -0.1744    0.0892         0
1.6094    0.3365   -0.0852         0         0
1.9459    0.2513         0         0         0
2.1972         0         0         0         0
```

Polinómio interpolador com Diferenças Descendentes - algoritmo

Dados: a, h, y e z

Resultado: $P_{n-1}(z)$

```
DF = tabDifFin(y); % tabela das diferenças finitas
dffn = DF(1,:); % Primeira linha de DF
n = length(dffn);
s = (z-a)./h;   % Mudança de variável
valpol = dffn(n)*ones(size(z));  % Inicialização do vetor valpol
for i=n-1:-1:1
valpol = (valpol.*(s-k+1)./k)+dffn(k); % Forma encaixada
end
```

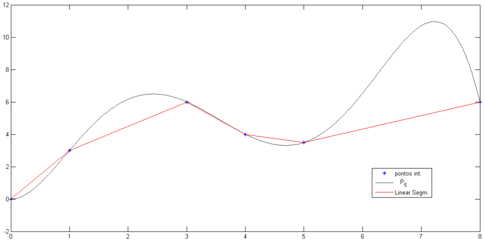
Exemplo:

```
>> x=1:2:10; y=log(x);
>> z=[3.5 5.5];
>> p=polDifDes(1,2,y,z)
p =
1.2597    1.7009
>> erro=abs(p-log(z))
erro =
0.0070    0.0039
```

Interpolação segmentada

- ↓ Vimos que a interpolação num número crescente de nós não conduz necessariamente a melhores resultados (em todo o intervalo de interpolação).
- ↓ Polinómios de grau muito elevado *oscilam* muito.
- ↑ "Partir"o intervalo de interpolação em subintervalos e usar polinómios diferentes em diferentes subintervalos (*polinómios segmentados*).

Por exemplo, podemos tomar, em cada subintervalo $[x_{i-1}, x_i]$, o polinómio linear interpolador dos pontos (x_{i-1}, y_{i-1}) e (x_i, y_i) $\longrightarrow$ interpolação linear segmentada .



Interpolação polinomial de grau 5 e interpolação linear segmentada de 6 pontos

Algumas notações

- Uma sequência de pontos $\Omega_n = \{x_i\}_{i=1}^{n}$, onde

$$a = x_1 < x_2 < \cdots < x_{n-1} < x_n = b,$$

é chamada uma sequência de nós no intervalo $[a, b]$.

- Os nós $x_2, \ldots, x_{n-1}$ são chamados nós interiores.
- Os nós $x_1 = a$ e $x_n = b$ são chamados nós fronteiros.

Teorema

Seja $f \in \mathscr{C}^2[x_1, x_n]$ e seja q_1 um polinómio segmentado de grau um, interpolador de uma função f nos nós $x_1, \ldots, x_n$. Seja $h = \max_{1 \leq i \leq n-1} |x_{i+1} - x_i|$. Se $\max_{x \in [x_1, x_n]} |f^{(2)}(x)| \leq M_2$, então

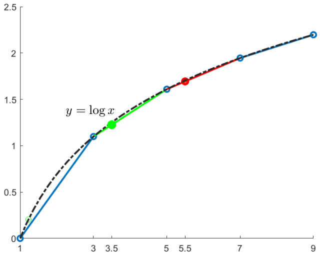
$$E_1(x) := |f(x) - q_1(x)| \leq \frac{M_2}{8} h^2$$

Demonstração: Veja, por exemplo, M. Raquel Valença, Métodos Numéricos.

Exemplo:

Considerando novamente os dados do exemplo anterior e usando interpolação linear segmentada:

```
>> x=1:2:10; y=log(x);
>> z=[3.5 5.5];
>> v1=polDifDes(x(2),2,y(2:3),z(1))
   v1 =
      1.2263
>> v2=polDifDes(x(3),2,y(3:4),z(2))
   v2 =
      1.6936
```



Funções *spline*

Definição

Seja dada uma sequência de nós $\Omega_n : a = x_1 < x_2 < \ldots < x_n = b$ *num intervalo* $[a, b]$ *e seja* $k \in \mathbb{N}$. *Uma função* $s : [a, b] \longrightarrow \mathbb{R}$ *diz-se uma função spline de grau* k *com nós* Ω_n, *se:*

- *Em cada intervalo* $[x_i, x_{i+1}]$, s *é um polinómio de grau não superior a* k;
- $s \in C^{(k-1)}[a, b]$.

Nota: Como só definimos *splines* de grau $k \geq 1$, teremos sempre funções contínuas. Por esse motivo, na definição anterior podemos usar os intervalos $[x_i, x_{i+1}]$ fechados em ambos os extremos, não havendo qualquer incompatibilidade na definição de s nos nós interiores.

De entre as funções *spline*, são especialmente importantes as funções *spline* cúbicas e será destas que iremos tratar de seguida com mais pormenor.

Splines cúbicas interpoladoras

Seja dada uma sequência de nós $\Omega_n : a = x_1 < x_2 < \ldots < x_n = b$ de um intervalo $[a, b]$ e n valores y_i (em geral, valores de uma dada função y nos nós x_i).

Problema

Determinar uma função *spline* cúbica s, com nós Ω_n, e interpoladora dos valores y_i nos nós x_i.
Por outras palavras, pretende-se encontrar uma função s que satisfaça:

- Em cada um dos subintervalos $[x_i, x_{i+1}]; i = 1, \ldots, n-1$, s é um polinómio de grau não superior a três;
- $s \in C^2[a, b]$;
- $s(x_i) = y_i;\ i = 1, 2, \ldots, n.$

Graus de liberdade: $4 \times (n-1) = 4n - 4$

Condições a impor: Continuidade de s, s' e s'' nos nós interiores + condições de interpolação $\rightarrow$ $3 \times (n-2) + n = 4n - 6 \Longrightarrow$ o problema não está totalmente definido, sendo necessário especificar duas condições adicionais.

Caso de nós igualmente espaçados

Por uma questão de simplicidade, consideramos apenas o caso em que os nós são igualmente espaçados, com espaçamento h, i.e., em que temos

$$x_i = a + (i-1)h \;\; i = 1, 2, \ldots n, \;\; h = \frac{b-a}{n-1}.$$

Vamos introduzir a seguinte notação para os valores que a segunda derivada de s assume em cada um dos nós

$$\boxed{M_i := s''(x_i)}$$

Se os valores M_i forem conhecidos, facilmente se obtém a expressão do polinómio cúbico que "forma"a função *spline* s num dado intervalo $[x_i, x_{i+1}]$. Comecemos por observar que, em $[x_i, x_{i+1}]$, a segunda derivada de s será um polinómio linear, cuja expressão será dada por

$$s''(x) = \frac{(x_{i+1} - x)M_i + (x - x_i)M_{i+1}}{h}.$$

Integrando duas vezes a expressão anterior, vem

$$s(x) = \frac{(x_{i+1} - x)^3 M_i + (x - x_i)^3 M_{i+1}}{6h} + Ax + B,$$

com A, B constantes.

Por uma questão de conveniência reescrevamos a expressão anterior como

$$s(x) = \frac{(x_{i+1}-x)^3 M_i + (x-x_i)^3 M_{i+1}}{6h} + \tilde{A}(x_{i+1}-x) + \tilde{B}(x-x_i). \tag{1}$$

Da condição de interpolação $s(x_i) = y_i$, vem

$$\frac{h^3}{6h} M_i + \tilde{A}h = y_i$$

de onde se obtém a seguinte expressão para $\tilde{A}$:

$$\tilde{A} = \frac{y_i}{h} - \frac{h}{6} M_i. \tag{2}$$

De modo análogo, usando a condição $s(x_{i+1}) = y_{i+1}$, obtém-se a seguinte expressão para $\tilde{B}$:

$$\tilde{B} = \frac{y_{i+1}}{h} - \frac{h}{6} M_{i+1}. \tag{3}$$

Substituindo as expressões (2) e (3) em (1), obtemos a seguinte expressão para o polinómio cúbico que define s no intervalo $[x_i, x_{i+1}]$:

$$\begin{aligned} s(x) = \frac{(x_{i+1}-x)^3 M_i + (x-x_i)^3 M_{i+1}}{6h} + \left(\frac{y_i}{h} - \frac{h}{6} M_i\right)(x_{i+1}-x) + \\ + \left(\frac{y_{i+1}}{h} - \frac{h}{6} M_{i+1}\right)(x-x_i) \end{aligned} \tag{4}$$

Como determinar os valores M_i a usar na fórmula anterior?

A função s cuja expressão, em cada um dos intervalos $[x_i, x_{i+1}]$, é dada por (4) é tal que $s(x_i^-) = s(x_i^+) = y_i$ e $s''(x_i^-) = s''(x_i^+) = M_i$ em cada nó interior. Para que s seja uma função *spline* cúbica, deveremos também ter

$$s'(x_i^+) = s'(x_i^-); i = 2, \ldots, n-1.$$

Mas:

- Em $[x_i, x_{i+1}]; i = 1, 2, \ldots, n-1$, tem-se

$$s'(x) = \frac{-(x_{i+1}-x)^2 M_i + (x-x_i)^2 M_{i+1}}{2h} + \frac{y_{i+1}-y_i}{h} - \frac{h}{6}(M_{i+1}-M_i).$$

- Em $[x_{i-1}, x_i]; i = 2, 3, \ldots, n-1, n$, tem-se

$$s'(x) = \frac{-(x_i-x)^2 M_{i-1} + (x-x_{i-1})^2 M_i}{2h} + \frac{y_i-y_{i-1}}{h} - \frac{h}{6}(M_i-M_{i-1}).$$

Assim, tem-se

$$s'(x_i^+) = -\frac{h}{2}M_i + \frac{y_{i+1}-y_i}{h} - \frac{h}{6}(M_{i+1}-M_i); i = 1, 2, \ldots, n-1$$

e

$$s'(x_i^-) = \frac{h}{2}M_i + \frac{y_i-y_{i-1}}{h} - \frac{h}{6}(M_i-M_{i-1}), i = 2, 3, \ldots, n-1, n$$

Relações de consistência para os M_i

Igualando as expressões anteriores (para $i = 2, \ldots, n-2$) e efetuando alguma manipulação simples, vem que teremos de ter

$$M_{i-1} + 4M_i + M_{i+1} = \frac{6}{h^2}\left(y_{i-1} - 2y_i + y_{i+1}\right); \; i = 2, \ldots, n-1. \tag{5}$$

As $n-2$ equações anteriores são conhecidas como relações de consistência para os valores $M_i = s''(x_i)$. Estas equações não são suficientes para determinar as n incógnitas $M_i; i = 1, \ldots, n$ necessárias para a construção de s.

Uma vez mais confirmamos que o problema da determinação de uma função *spline* cúbica satisfazendo as condições de interpolação $s(x_i) = y_i; \; i = 1, \ldots, n$, não está completamente definido, sendo necessário especificar duas condições adicionais. Essas condições são, geralmente, impostas nos nós fronteiros (ou em nós próximos destes) e designam-se condições finais.

Condições finais

As escolhas mais usuais para as condições finais são as seguintes, tomando as respetivas funções *spline* os nomes indicados:

- $s''(x_1) = s''(x_n) = 0 \longrightarrow$ *spline* natural.

- $s'(x_1) = d_1$ e $s'(x_n) = d_n$, com d_1 e d_n dados $\longrightarrow$ *spline* completa.

  Geralmente construímos este tipo de função spline quando estamos a interpolar uma função y nos nós x_i, isto é, quando $y_i = y(x_i)$ e, além disso, conhecemos os valores da primeira derivada de y nos nós fronteiros x_1 e x_n; nesse caso, devemos tomar $d_1 = y'(x_1)$ e $d_n = y'(x_n)$.

- $s^{(3)}(x_2^+) = s^{(3)}(x_2^-)$ e $s^{(3)}(x_{n-1}^+) = s^{(3)}(x_{n-1}^-) \longrightarrow$ *spline* sem-nó.

  A condição de continuidade imposta à terceira derivada de s no nó x_2 (juntamente com a continuidade de s, s' e s'' nesse ponto) implica que as cúbicas que definem a *spline* nos intervalos $[x_1, x_2]$ e $[x_2, x_3]$, sejam a mesma, isto é, que não haja um verdadeiro nó em x_2 (o mesmo se passando, naturalmente, com o nó x_{n-1}). Isto justifica a designação escolhida para esta função *spline*.

Spline cúbica natural interpoladora

Existência e unicidade

Naturalmente, teremos de verificar que, para quaisquer das condições finais indicadas, o problema da construção da respetiva função *spline* interpoladora tem uma e uma só solução. Analisemos, então, o caso da *spline* natural.

As relações de consistência fornecem-nos $n-2$ equações para a determinação dos valores M_i; como, neste caso, $M_1 = M_n = 0$, na realidade há apenas $n-2$ incógnitas a determinar.

O sistema para a determinação dessas incógnitas é dado por

$$\begin{cases} 4M_2 + M_3 = b_2 \\ M_{i-1} + 4M_i + M_{i+1} = b_i;\, i = 3, \ldots, n-2, \\ M_{n-2} + 4M_{n-1} = b_{n-1} \end{cases}$$

onde

$$b_i = \frac{6}{h^2}(y_{i-1} - 2y_i + y_{i+1}).$$

A matriz do sistema anterior é

$$\begin{pmatrix} 4 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & \cdots & & 1 & 4 \end{pmatrix}.$$

Trata-se de uma matriz de diagonal estritamente dominante, portanto invertível, pelo que o sistema correspondente tem solução única. Podemos, assim, concluir que o problema da determinação de uma função *spline* cúbica natural interpoladora de n valores y_i em n nós x_i tem uma e uma só solução.

Nota: Além disso, a matriz do sistema é tridiagonal, o que torna o sistema especialmente simples de resolver.

De modo análogo, embora um pouco mais trabalhoso, se podem analisar os casos das funções *spline* sem-nó ou *spline* completa. Haverá, nestes casos, que expressar as condições finais respetivas em termos dos M_i; por exemplo, as condições finais $s'(x_1) = d_1$ e $s'(x_n) = d_n$ da *spline* completa podem escrever-se como

$$2M_1 + M_2 = \frac{6}{h^2}(y_2 - y_1) - \frac{6}{h}d_1$$

e

$$M_{n-1} + 2M_n = \frac{6}{h}d_n - \frac{6}{h^2}(y_n - y_{n-1}),$$

respetivamente. (Verifique!) Adicionando estas duas equações às $n-2$ relações de consistência, obtém-se novamente um sistema de matriz de diagonal estritamente dominante e tridiagonal.

Construção de *splines* cúbicas interpoladoras :: Passos

1. Usar as relações de consistência (5) + condições finais (expressas em termos dos M_i) para obter um sistema para determinação dos M_i;
2. Resolver esse sistema;
3. Usar a fórmula (4) para obter a expressão de s em cada intervalo $[x_i, x_{i+1}]$.

Teorema (Erro em interpolação por *splines* completas)

Seja s a função spline cúbica completa interpoladora de uma dada função y nos nós $x_i = a + (i-1)h,\ i = 1\ldots, n;\ h = \frac{b-a}{n-1}$. Suponhamos que $y \in C^4[a, b]$ e seja M_4 tal que $\max_{x\in[a,b]} |y^{(4)}(x)| \leq M_4$. Então, tem-se o seguinte resultado.[2]

$$\max_{x\in[a,b]} |s(x) - y(x)| \leq \frac{5}{384} h^4 M_4. \tag{6}$$

Omitiremos a demonstração, por ser demasiado ténica; ver, e.g, Hall, C.A., Meyer, W., *Optimal Error Bounds for Spline Interpolation* , J. Approx. Theory,**16**, 105-112 (1976).

[2] Estamos aqui a admitir que os valores d_1 e d_n usados nas condições finais são os valores da primeira derivada de y nos extremos, isto é, que $d_1 = y'(x_1)$ e $d_n = y'(x_n)$.

O resultado anterior costuma escrever-se, usando o símbolo de Landau $\mathcal{O}$, da seguinte forma:

$$\|s-y\|_\infty := \max_{x\in[a,b]} |s(x)-y(x)| = \mathcal{O}(h^4), \quad \text{quando } h \to 0.$$

Podemos assim concluir que, se (s_n) for uma sequência de *splines* completas interpoladoras de uma determinada função $y \in C^4[a,b]$ em n nós igualmente espaçados nesse intervalo $[a,b]$, então (s_n) converge uniformemente para y quando $n \to \infty$ (ou seja, quando o espaçamento h entre os nós tende para zero).

Pode provar-se também que a função spline sem-nó satisfaz

$$\|s-y\|_\infty = \mathcal{O}(h^4),$$

ou seja que a ordem de convergência deste tipo de *splines* é idêntica à da *spline* completa.

Quanto à função *spline* natural, tem-se o seguinte resultado

$$\|s-y\|_\infty = \mathcal{O}(h^2).$$

Isto significa que, contrariamente ao que o nome natural possa sugerir, a interpolação por este tipo de *splines* é, do ponto de vista da aproximação obtida, pouco recomendável.

Nota: No entanto, pode mostrar-se que a influência negativa das condições finais da função *spline* natural diminui à medida que se consideram valores de x mais “interiores”no intervalo de interpolação.

Na figura seguinte apresenta-se novamente a função de Runge e a função *spline* cúbica sem-nó interpoladora dessa função em 11 nós igualmente espaçados no intervalo $[-1, 1]$.

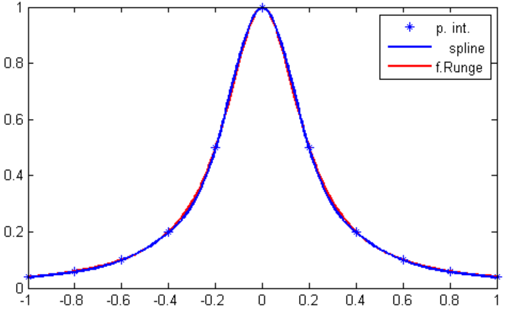


Figura: Função de Runge e função spline cúbica sem-nó

Um *spline* é um instrumento mecânico usado para traçar curvas suaves passando por determinados pontos (por exemplo, pontos de um mapa, estabelecendo a rota de um navio). Entende-se, assim, a escolha deste nome para designar as funções que temos vindo a estudar.

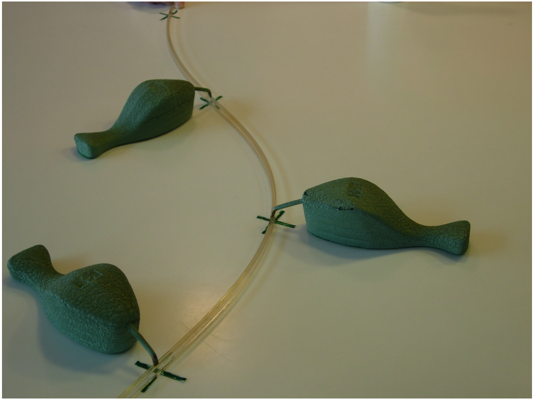


Figura: Um *spline* mecânico

Funções Matlab

Funções pré-definidas

`diff`	`interp1`	`makpp`	`poly`	`polyfit`
`polyval`	`ppval`	`roots`	`spline`	`unmkpp`

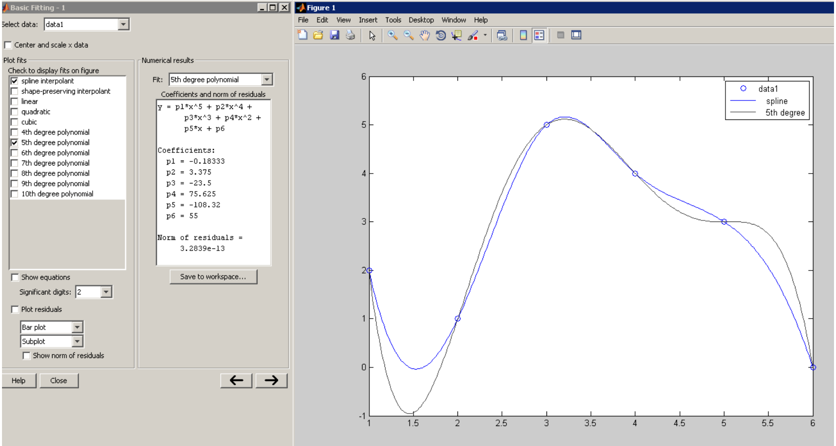
.

Funções toolbox AN

`polDifDes`	`polDifDiv`	`tabDifDiv`	`tabDifFin`

.

A ferramenta Basic Fitting do Matlab

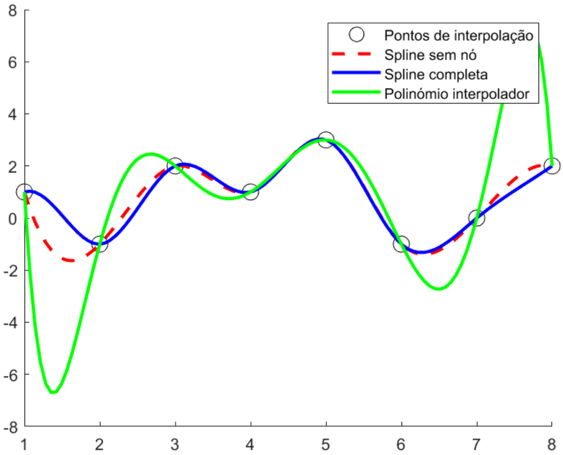


Exemplo: Considere a seguinte tabela de valores de uma função

x	1	2	3	4	5	6	7	8
y	1	-1	2	1	3	-1	0	2

A figura seguinte mostra a spline completa (com $S'(1) = 1$ e $S'(8) = 2$), a spline sem nó e o polinómio interpolador de grau ≤ 7 que interpolam os valores apresentados. Foram usadas as funções do Matlab `spline` e `polyfit` e o código:

```
>> x=[1, 2, 3, 4, 5, 6, 7, 8];
>> y=[1, -1, 2, 1, 3, -1, 0, 2];
>> hold on
>> plot(x,y,'ko','MarkerSize',10)
>> points=linspace(1,8);
>> splineSN=spline(x,y,points);
>> splineC=spline(x,[1 y 2],points);
>> plot(points,splineSN,'r--','LineWidth',2)
>> plot(points,splineC,'b-','LineWidth',2)
>> plot(points,polyval(polyfit(x,y,7),points),'g-','LineWidth',2)
>> legend('Pontos de interpolação','Spline sem nó',...
'Spline completa','Polinómio interpolador')
>> hold off
```


8
6
4
2
0
-2
-4
-6
-8
1
2
3
4
5
6
7
8
Pontos de interpolação
Spline sem nó
Spline completa
Polinómio interpolador