

### Objetivos Específicos de Aprendizagem

Ao finalizar este Capítulo, você será capaz de:

- Efetuar interpolações de funções com uma ou várias variáveis independentes;
- Executar aproximações de funções por interpolação *spline*; e
- Executar aproximações de funções e não funções por curvas de Bézier.

## 5 Aproximação Polinomial por Interpolação, Spline e Bézier

Nos Capítulos 5, 6 e 7, abordaremos o tópico central do Cálculo Numérico: a aproximação de uma função  $y = f(x)$ ,  $x \in [a, b]$ , por meio de outra função  $z = g(x)$ .

Você pode se perguntar: por que utilizamos uma função aproximadora, ou representante, em vez de usar diretamente a função original? Há dois motivos básicos para essa opção:

Motivo 1: podemos **conhecer apenas um conjunto discreto de pontos** da função  $y = f(x)$ , do tipo

$x_i$	$x_1$	$x_2$	.....	$x_{n+1}$
$y_i = f(x_i)$	$y_1$	$y_2$	.....	$y_{n+1}$

Isso ocorre em situações como:

- Coleta de valores amostrais em experimentos para geração de bases de dados, por exemplo, a tabela, a seguir, contém amostras da relação entre o tempo de uso e o respectivo índice de desgaste de uma peça

$x$ (Tempo de uso em meses)	1	2	4	5	6
$y$ (Índice de desgaste)	0.05	0.09	0.18	0.23	0.31

- Em sistemas de computação gráfica, o usuário indica alguns pontos de referência, ou controle, e o computador deve preencher os caminhos intermediários desses pontos sem distorções e com alta velocidade de resposta. Para tanto, precisamos obter funções aproximadoras desses caminhos.

Motivo 2: a função  $y = f(x)$  pode possuir **expressão conhecida, porém ser ineficiente**, difícil ou até impossível de ser utilizada.

Isso ocorre em situações como:

- A construção de funções pré-definidas para bibliotecas de linguagens computacionais, que devem ter rotinas para obter valores confiáveis de funções como  $y = e^x$ ,  $y = \sin(x)$ ,  $y = \cos(x)$ ,  $y = \tan(x)$ , entre outras, utilizando apenas as operações aritméticas elementares.

- b) Nos sistemas dedicados, muitas vezes temos que obter, em tempo real, valores de funções extensas e compostas de outras funções específicas, por exemplo:

$$f(x) = \operatorname{sen} \left[ \exp \left( \sqrt[3]{a \coth(x)} \right) \right]$$

Nesses casos, o tempo de processamento acumulado de cada uma das subfunções pode ser inviável, sendo obrigatória a aproximação da  $f(x)$  por uma função  $g(x)$  única com tempo de resposta factível.

- c) Alguns modelos matemáticos são representados por funções, cuja expressão algébrica pode ser de uso inviável. Por exemplo, para efetuar analiticamente a integral definida  $I = \int_0^2 e^{-x^2} dx$ , temos que obter uma aproximadora  $z = g(x)$  da função integranda original  $f(x) = e^{-x^2}$  que tenha primitiva conhecida, pois  $f(x) = e^{-x^2}$  tem primitiva impossível de ser expressa como uma função elementar.

**Para fazer uso da função aproximadora, a questão fundamental que precisamos responder é: quem pode ser a aproximadora  $z = g(x)$ ?**

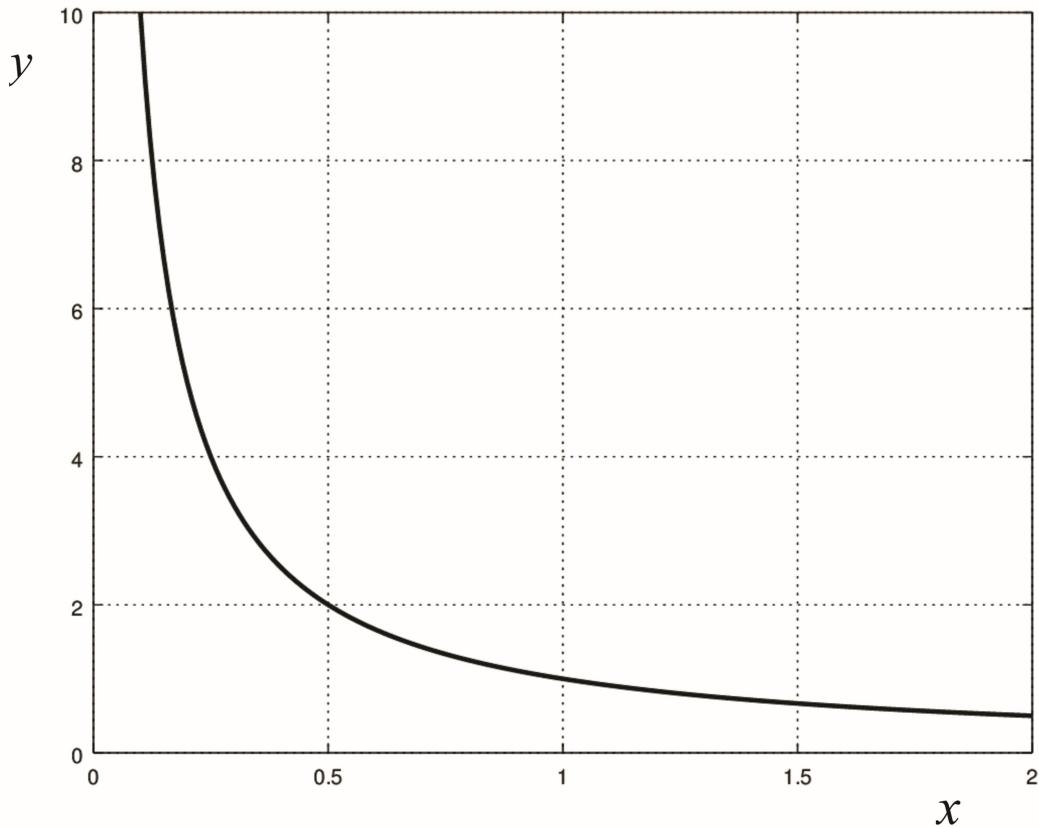
A premissa básica é que a  $z = g(x)$  deva ser simples e “bem comportada”, isto é, ser contínua, facilmente computável, derivável, integrável etc. Por consequência, o universo de busca das famílias de aproximadoras  $z = g(x)$  restringe-se às funções do tipo:

- a) **Polinomiais:**  $g(x) = \sum_{i=1}^{n+1} a_i x^{i-1} = P_n(x)$ , cujas vantagens são:
- i. envolvem somente operações elementares;
  - ii. facilmente deriváveis, integráveis etc;
  - iii. formam um anel, ou seja, todas as transformações algébricas nelas aplicadas resultam em outro polinômio.

- b) **Racionais:**  $g(x) = \frac{P_n(x)}{Q_m(x)}$ , em que  $P_n(x)$  e  $Q_m(x)$  são polinômios de

graus  $n$  e  $m$ , respectivamente, e permitem a aproximação de funções com gráfico assintótico, como o Gráfico 5.1.

Gráfico 5.1 – Função assintótica



Fonte: Elaboração própria

c) **Trigonométricas:**  $g(x) = \sum_{i=1}^m (a_i * \sin(i * x) + b_i * \cos(i * x))$ , que fornecem aproximações de funções representativas de fenômenos oscilantes.

Em virtude de suas características, as funções polinomiais são as mais utilizadas como aproximadoras. Contudo, tais vantagens teriam pouca valia se não existisse o suporte teórico de que podem sempre aproximar funções. Tal garantia encontra-se no **teorema de Weirstrass**:

“Se  $y=f(x)$  for contínua em  $[a,b]$ , então  $\forall \varepsilon > 0$  sempre existe  $P_n(x)$  e  $n$  depende da precisão  $\varepsilon$ , tal que  $|P_n(x) - f(x)| < \varepsilon$ ,  $\forall x \in [a,b]$ .”

Ou seja, sempre existe uma aproximadora polinomial  $P_n(x)$  que esteja tão próxima quanto se queira de uma  $f(x)$  contínua no domínio  $[a,b]$ . Como esse teorema assegura a existência da aproximadora, mas não fornece um modo de determiná-la, temos mais uma questão a responder: como obter essa aproximadora?

A resposta dessa questão será abordada em todas as seções deste Capítulo.

## 5.1 Aproximação por Interpolação Polinomial

Para aproximar uma função  $y=f(x)$  contínua em  $[a,b]$ , podemos proceder como segue:

- a) Tomamos  $n+1$  pontos  $(x_i, y_i = f(x_i))$ , com  $i=1, 2, \dots, n+1$ , em  $x_i \in [a, b]$ , conforme:

$x_i$	$x_1$	$x_2$	.....	$x_{n+1}$
$y_i = f(x_i)$	$y_1$	$y_2$	.....	$y_{n+1}$

com  $x_1 = a$  e  $x_{n+1} = b$ , ou usamos uma função já representada por  $n+1$  pontos de uma tabela proveniente de um levantamento de dados.

- b) Tomamos um polinômio genérico de grau  $n$ :

$$P_n(x) = \sum_{i=1}^{n+1} a_i x^{i-1} = a_1 + a_2 x + \dots + a_n x^{n-1} + a_{n+1} x^n \quad (1)$$

E aplicamos a seguinte condição de aproximação:

$$P_n(x_i) = y_i, \quad i = 1, 2, \dots, n+1 \quad (2)$$

**DESTAQUE** Isto é, impomos a condição de que essa função aproximadora **polinomial deve passar por todos os  $n+1$  pontos** tabelados conforme item (a), ou seja, o **erro ou desvio**, sobre esses pontos escolhidos para definir a approximadora, é **nulo**. FIM DO DESTAQUE

Dessa condição, resulta a expressão:

$$\begin{cases} P_n(x_1) = a_1 + a_2 x_1 + \dots + a_n x_1^{n-1} + a_{n+1} x_1^n = y_1 \\ P_n(x_2) = a_1 + a_2 x_2 + \dots + a_n x_2^{n-1} + a_{n+1} x_2^n = y_2 \\ P_n(x_{n+1}) = a_1 + a_2 x_{n+1} + \dots + a_n x_{n+1}^{n-1} + a_{n+1} x_{n+1}^n = y_{n+1} \end{cases} \quad (3a)$$

Que é um sistema com  $n+1$  equações lineares e  $n+1$  incógnitas  $a_i$ .

Quando reescrevemos esse sistema na forma matricial, temos:

$$\begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} & x_1^n \\ 1 & x_2 & \cdots & x_2^{n-1} & x_2^n \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 1 & x_{n+1} & \cdots & x_{n+1}^{n-1} & x_{n+1}^n \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n+1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n+1} \end{bmatrix} \quad (3b)$$

Resolvendo o sistema (3b), obtemos os  $n+1$  coeficientes  $a_i$  do polinômio que passa por todos os  $n+1$  pontos  $(x_i, y_i = f(x_i))$ . Note que o número de pontos é igual ao número de coeficientes incógnitos  $a_i$ . FIM. Esse polinômio approximador é denominado de **interpolador** da função.

Os elementos  $u_{ij}$  da matriz dos coeficientes do sistema  $U * A = Y$  dado na eq. (3b) têm lei de formação genérica dada por:

$$u_{ij} = x_i^{j-1} \quad (4)$$

Resolvendo o sistema linear  $U * A = Y$  pelo método mais adequado entre aqueles que abordamos no Capítulo 2, obtemos os coeficientes do polinômio interpolador.

**Exemplo 5.1:** aproxime, por interpolação, a função  $y = \ln(x)$ ,  $x \in [1.0, 2.0]$ , dividindo esse intervalo em  $n = 2$  subintervalos iguais e estime  $\ln(1.14)$ . Estime ainda o erro do  $\ln(1.14)$  obtido pelo approximador de grau 2 comparando-o com seu valor exato.

**Solução:**

$$h = (b - a) / n = (2.0 - 1.0) / 2 = 0.5$$

$i$	1	2	$2+1=3$
$x_i$	1.0	1.5	2.0
$y_i = \ln(x_i)$	0.000000000000000	0.405465108108164	0.693147180559945

$$n+1=3 \text{ pontos} \Rightarrow n=2 \text{ subintervalos} \Rightarrow P_2(x) = a_1 + a_2 x + a_3 x^2$$

Aplicando o sistema dado na eq. (3b), temos:

$$\begin{bmatrix} (1.0) & (1.0)^1 & (1.0)^2 \\ (1.0) & (1.5)^1 & (1.5)^2 \\ (1.0) & (2.0)^1 & (2.0)^2 \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0.000000000000000 \\ 0.405465108108164 \\ 0.693147180559945 \end{bmatrix}$$

E resolvendo esse sistema por eliminação de Gauss, com variáveis *double*, temos:

$$a_1 = -1.164279323185479$$

$$a_2 = 1.399845394498246$$

$$a_3 = -0.235566071312767$$

Assim,

$$P_2(x) = -1.164279323185479 + 1.399845394498246x - 0.235566071312767x^2 \cong \ln(x)$$

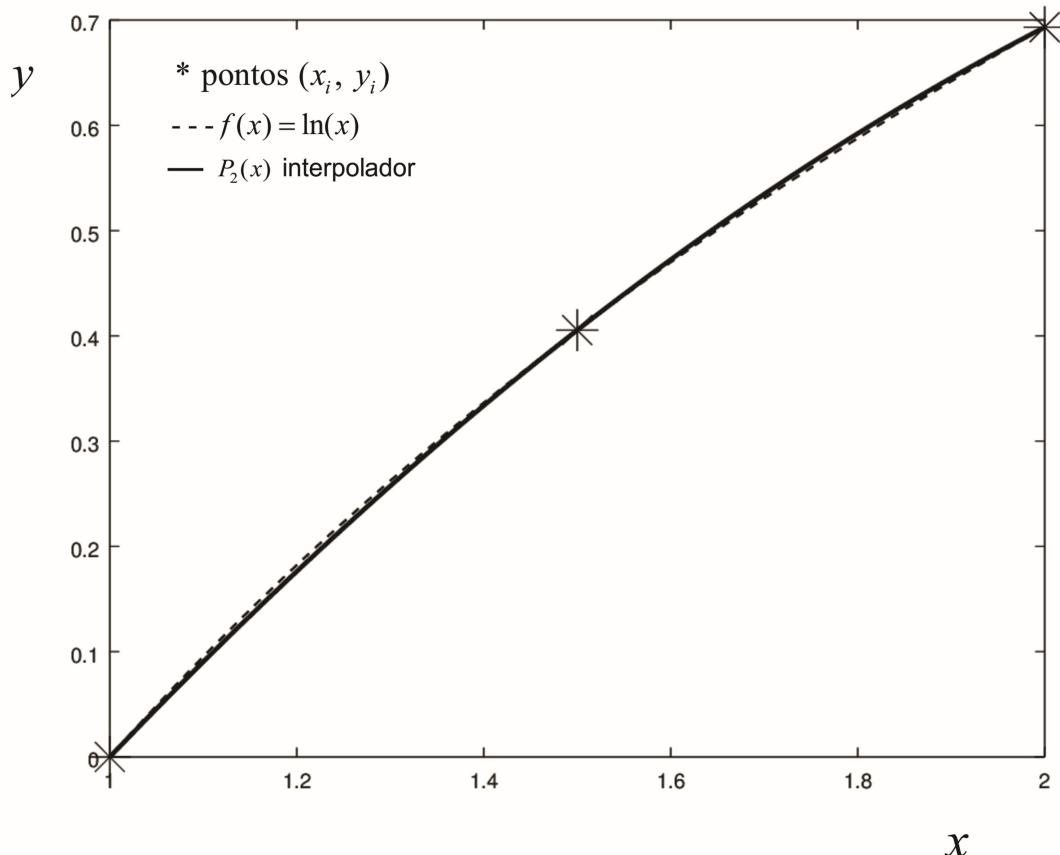
Valor Aproximado local:  $P_2(1.14) = 0.125402760264449 \Rightarrow$  avaliação por Horner.

Valor Exato local:  $f(1.14) = \ln(1.14) = 0.131028262406404$

$$\Rightarrow \text{Erro local} = |P_2(1.14) - f(1.14)| = 0.00562550214195459$$

Podemos avaliar a aderência da função interpoladora  $P_2(x)$  com a função exata  $f(x) = \ln(x)$  no Gráfico 5.2.

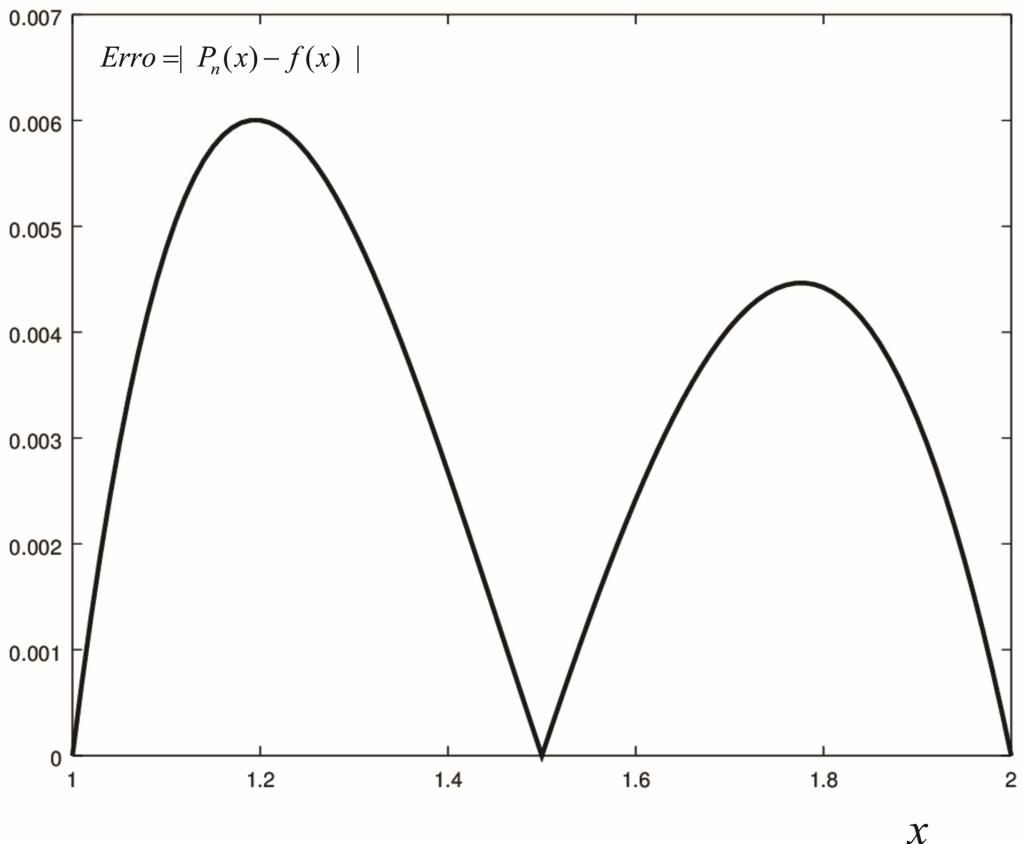
Gráfico 5.2 – Função interpoladora  $P_2(x)$  e função exata  $f(x) = \ln(x)$



Fonte: Elaboração própria

Podemos também representar o erro local exato entre  $P_2(x)$  e  $f(x)$  no intervalo  $x \in [1.0, 2.0]$ , conforme o Gráfico 5.3.

Gráfico 5.3 – Erro local exato entre  $P_2(x)$  e  $f(x)$



Fonte: Elaboração própria

Observe que os erros locais, entre a função aproximadora  $P_2(x)$  e a exata  $f(x)$ , são nulos sobre os pontos usados para definir o interpolador  $(x_i, y_i = f(x_i))$ , e o erro máximo é  $ErroMax \approx 0.006$ . Observe também que os erros têm pico máximo na região intermediária de cada subintervalo entre os pontos  $(x_i, y_i = f(x_i))$  interpolados e serão maiores nos subintervalos das duas extremidades, conforme poderemos ver no exercício 5.3.

No **Caderno de Exercícios e Respostas**, disponível no link <http://sergiopeters.prof.ufsc.br/exercicios-e-respostas/>, você encontrará os exercícios atualizados deste livro.

Já no **Caderno de Algoritmos**, disponível no link <http://sergiopeters.prof.ufsc.br/algoritmos-livro/>, apresentamos um **algoritmo que implementa a interpolação polinomial geral** LINK Veremos, nas seções 5.1.2.1 e

5.1.2.2, que existem algoritmos alternativos para determinar o polinômio interpolador. FIM DO LINK no arquivo [Cap5exem5.1coefinterPn1D.m](#), contendo o cálculo dos coeficientes  $a_i$  do polinômio interpolador e seu uso.

Fazendo uma análise da técnica de aproximação por interpolação polinomial, devemos considerar pelo menos três questões fundamentais:

- Será que o sistema  $U * A = Y$ , dado pela eq. (3b), sempre tem solução?  
Caso exista solução, esta será única?
- Será possível melhorar a eficiência computacional (menor tempo de resposta e demanda de memória) para a obtenção e o uso do  $P_n(x)$ ?
- Qual é o erro de truncamento máximo associado ao se tomar  $P_n(x)$  como aproximador de  $f(x)$ ,  $\forall x \in [a,b]$  com  $x \neq x_i$ ?

Vamos tratar de responder a cada uma dessas questões nas três seções a seguir.

### 5.1.1 Unicidade do Interpolador

Conforme detalhamos anteriormente, para obter o polinômio interpolador de uma função definida por uma tabela de pontos:

$x_i$	$x_1$	$x_2$	.....	$x_{n+1}$
$y_i = f(x_i)$	$y_1$	$y_2$	.....	$y_{n+1}$

geramos um sistema de equações lineares dado pela eq. (3b), cuja matriz  $U$  dos coeficientes é especial, devido à lei de formação dos seus elementos. Esse tipo de matriz é denominado de **Vandermonde**, cuja forma simples e eficiente de obter o seu determinante é:

$$Det(U) = Det \begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} & x_1^n \\ 1 & x_2 & \cdots & x_2^{n-1} & x_2^n \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{n+1} & \cdots & x_{n+1}^{n-1} & x_{n+1}^n \end{bmatrix} = \prod_{\substack{i=n+1,2 \\ j=i-1,1 \\ (i \neq j)}} (x_i - x_j) \quad (5a)$$

ou

$$\begin{aligned}
\text{Det}(U) = & (x_{n+1} - x_n)(x_{n+1} - x_{n-1}) \dots (x_{n+1} - x_1) * \\
& * (x_n - x_{n-1})(x_n - x_{n-2}) \dots (x_n - x_1) * \\
& \vdots \\
& * (x_3 - x_2)(x_3 - x_1) * \\
& * (x_2 - x_1)
\end{aligned} \tag{5b}$$

Como no conjunto de pontos  $(x_i, y_i = f(x_i))$  geradores da matriz  $U$  não existem valores de  $x$  repetidos por ser uma função, isto é  $x_i \neq x_j$  para  $i \neq j$ , logo  $\text{Det}(U) \neq 0$ , conforme a eq. (5a) ou (5b), e o sistema  $U^* A = Y$  dado na eq. (3b) terá solução única. Assim, o polinômio gerado será único, independentemente da maneira de expressá-lo. Essa demonstração generaliza o fato conhecido de que por dois pontos quaisquer passa uma única reta.

Na sequência, vamos mostrar formas mais eficientes de determinar o polinômio interpolador de grau  $n$ .

### 5.1.2 Determinação Eficiente do Interpolador

A consequência da unicidade do interpolador é que podemos tentar determiná-lo sem a geração do sistema linear (3b), que demanda a ordem de  $O(n^2)$  multiplicações para gerar a matriz  $U$  e cuja solução por eliminação de Gauss ou Crout exige número de operações aritméticas na ordem de  $O(2n^3 / 3)$ .

Para a função discretizada

$i$	1	2	.....	$n + 1$
$x_i$	$x_1$	$x_2$	.....	$x_{n+1}$
$y_i = f(x_i)$	$y_1$	$y_2$	.....	$y_{n+1}$

propomos representações de polinômios interpoladores em **bases** alternativas, pois a eq. (1) é escrita como uma combinação linear direta da **base canônica** dos polinômios:  $\{x^0, x^1, \dots, x^{n-1}, x^n\}$ , com **coeficientes**  $a_i$ , ou seja,

$$P_n(x) = \sum_{i=1}^{n+1} a_i x^{i-1} = a_1 x^0 + a_2 x^1 + \dots + a_n x^{n-1} + a_{n+1} x^n$$

#### 5.1.2.1 Expressão do Interpolador Polinomial $P_n(x)$ na Base dos Polinômios de Lagrange

Nesse caso, o interpolador  $P_n(x)$  é expresso na forma de uma combinação linear da **base** definida por **polinômios de Lagrange** de grau  $n$ ,  $L_i(x)$ ,  $i = 1, \dots, n+1$ ,

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n+1} \frac{(x - x_j)}{(x_i - x_j)} \quad (6a)$$

com as propriedades:

$$L_i(x_i) = 1 \text{ para qualquer } i \text{ e } L_i(x_j) = 0 \text{ para } j \neq i \quad (6b)$$

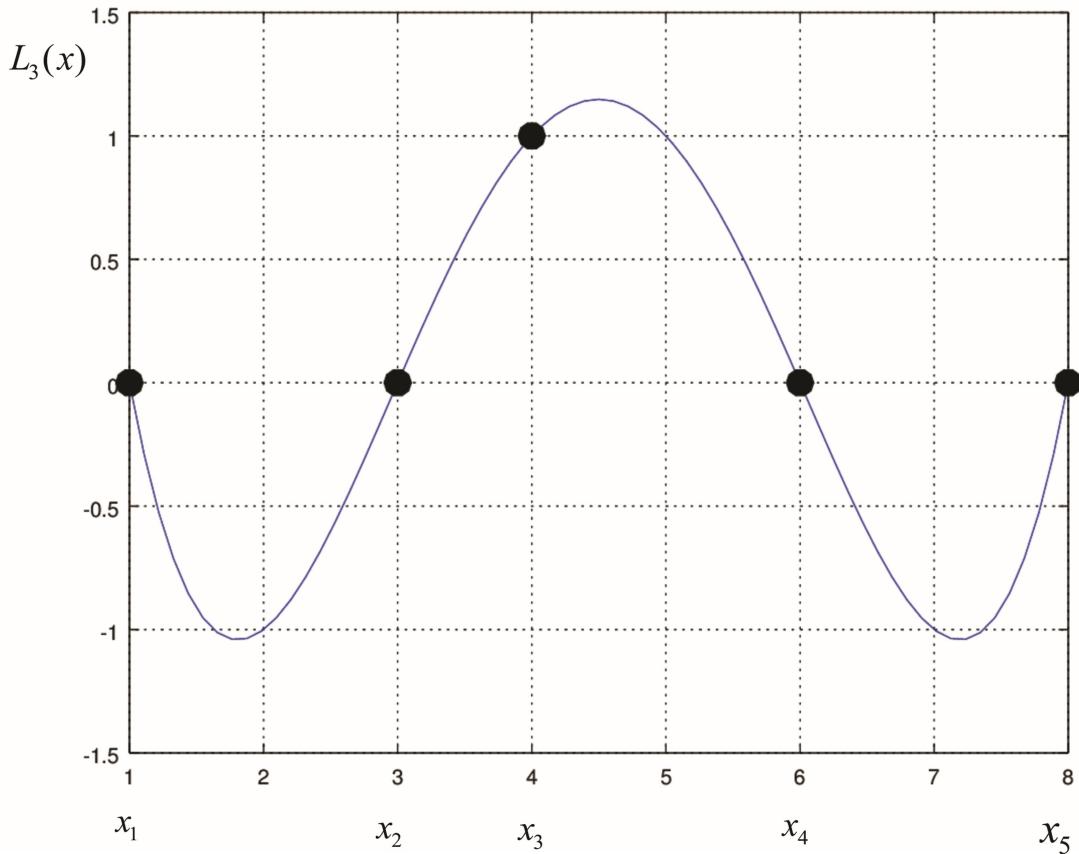
**DESTAQUE** Cada polinômio  $L_i(x)$  se anula em todos os pontos conhecidos  $x_j$ , com exceção de um deles,  $x_i$ . Os polinômios de Lagrange podem ser considerados uma função “peso”, com valor 1 sobre  $x_i$  e 0 sobre  $x_j$ . Cada  $L_i(x)$  é definido em torno de um  $x_i$  e é uma função linearmente independente de outra  $L_j(x)$ .

Por exemplo, para o conjunto de 5 pontos:

$i$	1	2	3	4	5
$x_i$	1	3	4	6	8

O polinômio de Lagrange de grau 4 definido em torno de  $x_3 = 4$ ,  $L_3(x)$  resulta no Gráfico 5.4.

Gráfico 5.4 – Polinômio de Lagrange  $L_3(x)$  de grau 4 em  $x_3 = 4$



Fonte: Elaboração própria

Observe que o polinômio  $L_3(x)$  do Gráfico 5.4 tem valor nulo sobre  $x_1 = 1$ ,  $x_2 = 3$ ,  $x_4 = 6$ ,  $x_5 = 8$  e valor unitário em  $x_3 = 4$ . FIM DO DESTAQUE

---

Assim, um interpolador genérico  $P_n(x)$  de grau  $n$  pode ser expresso como combinação linear da **base de polinômios de Lagrange**  $L_i(x)$ , ponderados diretamente por **coeficientes** com valores iguais a  $y_i$ , em função das propriedades estabelecidas nas eqs. (6b), conforme segue:

$$P_n(x) = \sum_{i=1}^{n+1} y_i L_i(x) \quad (7)$$

$$P_n(x) = \sum_{i=1}^{n+1} y_i \prod_{\substack{j=1 \\ j \neq i}}^{n+1} \frac{(x - x_j)}{(x_i - x_j)} \quad (8a)$$

ou

$$\begin{aligned}
P_n(x) = & y_1 \frac{(x-x_2)(x-x_3) \dots (x-x_{n+1})}{(x_1-x_2)(x_1-x_3) \dots (x_1-x_{n+1})} + \\
& + y_2 \frac{(x-x_1)(x-x_3) \dots (x-x_{n+1})}{(x_2-x_1)(x_2-x_3) \dots (x_2-x_{n+1})} + \\
& + \dots \\
& + y_{n+1} \frac{(x-x_1)(x-x_2) \dots (x-x_n)}{(x_{n+1}-x_1)(x_{n+1}-x_2) \dots (x_{n+1}-x_n)}
\end{aligned} \tag{8b}$$

Como  $L_i(x_i) = 1$  e  $L_i(x_j) = 0$ , então

$$\left\{ \begin{array}{l} P_n(x_1) = y_1 \\ P_n(x_2) = y_2 \\ \vdots \\ P_n(x_{n+1}) = y_{n+1} \end{array} \right.$$

Ou seja,  $P_n(x)$  passa por todos os pontos  $(x_i, y_i)$ .

Como o interpolador  $P_n(x)$  dado pela eq. (8a) passa por todos os  $n+1$  pontos  $(x_i, y_i)$ , ele será o mesmo que o obtido anteriormente pelo sistema  $U^*A=Y$  dado pela eq. (3b), em consequência da unicidade do polinômio interpolador. Na forma de Lagrange não temos a necessidade de gerar e nem de resolver um sistema linear, basta substituir diretamente os valores dos pontos  $(x_i, y_i)$  e o  $x$  desejado na eq. (8a).

**Exemplo 5.2:** determine o interpolador de Lagrange para a função discretizada:

$x_i$	0	1	3	4
$y_i = f(x_i)$	2	4	0	1

Aproxime  $f(x=2)$  e  $f(x=5)$ .

**Solução:**

Temos  $n+1=4$  pontos de  $f(x)$  e  $n=3$  grau de  $P_n(x)$ .

$$\begin{aligned}
P_3(x) = & y_1 \frac{(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)} + \\
& + y_2 \frac{(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)} + \\
& + y_3 \frac{(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)} + \\
& + y_4 \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)}
\end{aligned}$$

$$P_3(x) = 2 \frac{(x-1)(x-3)(x-4)}{(0-1)(0-3)(0-4)} + 4 \frac{(x-0)(x-3)(x-4)}{(1-0)(1-3)(1-4)} + \\ + 0 \frac{(x-0)(x-1)(x-3)}{(3-0)(3-1)(3-4)} + 1 \frac{(x-0)(x-1)(x-3)}{(4-0)(4-1)(4-3)}$$

Resultando nas estimativas:

$$f(x=2) \cong P_3(x=2) = 2.167$$

$$f(x=5) \cong P_3(x=5) = 8.667$$

Também poderíamos expandir algebricamente a expressão de  $P_n(x)$  dada pela eq. (8b) e chegaríamos à mesma expressão dada pela eq. (1) com coeficientes dados pela eq. (3b), possivelmente com menos erros de arredondamento.

Já para efeitos de otimização do algoritmo, podemos reescrever a expressão dada pela eq. (8a) desta forma:

$$\begin{cases} Num = \prod_{j=1}^{n+1} (x - x_j) \\ P_n(x) = \sum_{i=1}^{n+1} y_i \frac{Num}{(x - x_i) \prod_{\substack{j=1 \\ j \neq i}}^{n+1} (x_i - x_j)}, \quad \forall x \neq x_i \end{cases} \quad (8c)$$

Antes de continuar a sua leitura, teste os algoritmos na forma da eq. (8a) e na forma otimizada da eq. (8c) disponíveis no **Caderno de Algoritmos** no arquivo **Cap5exem5.2PnLagrange1D.m**.

Observe que o número de operações envolvido nesse algoritmo otimizado, eq. (8c), fica reduzido a  $2n^2 + 9n$  operações aritméticas para cada avaliação de  $P_n(v) \cong f(v)$ , com a restrição de que  $v \neq x_i$ .

### 5.1.2.2 Interpolador de Gregory-Newton com Diferenças

Há duas definições importantes a considerar em relação ao polinômio de Gregory-Newton.

Definição 1: para a função discretizada  $(x_i, y_i)$

$i$	1	2	.....	$n + 1$
$x_i$	$x_1$	$x_2$	.....	$x_{n+1}$
$y_i = f(x_i)$	$y_1$	$y_2$	.....	$y_{n+1}$

definimos as **diferenças divididas**  $\Delta^k y_i$ , no sentido ascendente, por:

$$\Delta^1 y_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad \forall i = 1, \dots, n \quad \Rightarrow \quad \text{diferença de 1ª ordem.}$$

$$\Delta^2 y_i = \Delta(\Delta y_i) = \frac{\Delta y_{i+1} - \Delta y_i}{x_{i+2} - x_i}, \quad \forall i = 1, \dots, n-1 \quad \Rightarrow \quad \text{diferença de 2ª ordem.}$$

$$\Delta^k y_i = \frac{\Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i}{x_{i+k} - x_i}, \quad \forall i = 1, \dots, n+1-k \quad \Rightarrow \quad \text{diferença de } k\text{-ésima ordem.}$$

Note a semelhança entre as definições das diferenças divididas e das derivadas de várias ordens de uma função com expressão conhecida.

**Exemplo 5.3:** determine todas as diferenças divididas da função discretizada

$i$	1	2	3	4
$x_i$	1	3	4	7
$y_i = f(x_i)$	2	0	1	3

**Solução:**

Aplicando a definição 1 na tabela, a seguir, temos:

$i$	$x_i$	$y_i$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$
1	<b>1</b>	<b>2</b>	-1	2/3	-1/8
2	3	0	1	-1/12	-
3	4	1	2/3	-	-
4	7	3	-	-	-

$$\Delta^1 y_1 = \frac{[(0) - (2)]}{(3-1)} = -\frac{2}{2} = -1$$

$$\Delta^2 y_1 = \frac{[(1) - (-1)]}{(4-1)} = \frac{2}{3}$$

$$\Delta^3 y_1 = \frac{[(-1/12) - (2/3)]}{(7-1)} = -\frac{1}{8}$$

Observe que as diferenças divididas no 1º ponto, em  $i=1$  (destacados em negrito), calculadas ao lado da tabela de resultados, dependem de todos os pontos.

Definição 2: para a função discretizada

$i$	1	2	.....	$n + 1$
$x_i$	$x_1$	$x_2$	.....	$x_{n+1}$
$y_i = f(x_i)$	$y_1$	$y_2$	.....	$y_{n+1}$

definimos o polinômio interpolador  $P_n(x)$  de grau  $n$ , na forma de **Gregory-Newton**, como combinação linear da **base**  $\left\{1, (x-x_1), (x-x_1)(x-x_2), \dots, \prod_{j=1}^n (x-x_j)\right\}$ , ponderada diretamente pelos **coeficientes** dados pelas diferenças divididas, conforme segue:

$$P_n(x) = y_1 + \sum_{k=1}^n \Delta^k y_1 * \left[ \prod_{j=1}^k (x-x_j) \right] \quad (9a)$$

ou

$$P_n(x) = y_1 + \Delta y_1(x-x_1) + \Delta^2 y_1(x-x_1)(x-x_2) + \dots + \Delta^n y_1(x-x_1) \dots (x-x_n) \quad (9b)$$

resulta que  $P_n(x)$  passa por todos os pontos discretizados, pois

$$\begin{cases} P_n(x_1) = y_1 \\ P_n(x_2) = y_1 + \frac{(y_2 - y_1)}{(x_2 - x_1)}(x_2 - x_1) = y_2 \\ P_n(x_3) = y_3 \\ \vdots \\ P_n(x_{n+1}) = y_{n+1} \end{cases}$$

Todas essas igualdades podem ser comprovadas por indução finita. Logo, o polinômio  $P_n(x)$  dado pela eq. (9a) também é uma forma alternativa do único interpolador de grau  $n$  que passa sobre todos os pontos discretizados, assim como as formas dadas anteriormente pela eq. (1), com coeficientes dados por (3b), e pela eq. (8a).

**Exemplo 5.4:** determine o interpolador de Gregory-Newton da função

$x_i$	1	3	4	7
$y_i = f(x_i)$	2	0	1	3

Estime  $f(x=5)$  e  $f(x=7.5)$ .

**Solução:**

Temos  $n+1 = 4 \Rightarrow n = 3$

$$P_3(x) = y_1 + \Delta y_1(x-x_1) + \Delta^2 y_1(x-x_1)(x-x_2) + \Delta^3 y_1(x-x_1)(x-x_2)(x-x_3)$$

Usando as diferenças divididas obtidas no **Exemplo 5.3**, temos:

$$P_3(x) = 2(1) + (-1)(x-1) + (2/3)(x-1)(x-3) + (-1/8)(x-1)(x-3)(x-4)$$

E efetuando as estimativas, resultam:

$$f(5.0) \cong P_3(5.0) = 2.233$$

$$f(7.5) \cong P_3(7.5) = 2.203$$

Na algoritmização desse interpolador, podemos fazer o armazenamento das diferenças divididas na forma de um vetor, haja vista que apenas as diferenças no 1º ponto, em  $i=1$ , são utilizadas na forma final, portanto não há necessidade de uma matriz para esse fim. No momento da determinação de todas as diferenças, bastará usar mais um vetor auxiliar.

Confira o algoritmo de Gregory-Newton no **Caderno de Algoritmos** no arquivo **Cap5exem5.4PnGregoryNewton1D.m**. A seguir, apresentamos algumas considerações sobre os interpoladores estudados até este momento.

O número de operações aritméticas executadas em cada tipo de interpolador é:

- a) Interpolador geral de base canônica:  $(n-1)n$  para gerar o sistema,  
 $O(2n^3 / 3)$  para solvê-lo e  $2n$  operações para obter  $f(v) \cong P_n(v)$ .
- b) Interpolador de Lagrange (otimizado):  $2n^2 + 9n$  operações para obter  $f(v) \cong P_n(v)$ .
- c) Interpolador de Gregory-Newton:  $3(n^2 + n) / 2$  operações para obter as diferenças divididas e  $4n$  operações para obter  $f(v) \cong P_n(v)$ .

Cada método será mais eficiente de acordo com a aplicação desejada e o número de valores a serem estimados. Por exemplo, devido às características das expressões algébricas, Gregory-Newton é mais eficiente quando temos que efetuar muitas estimativas em um mesmo conjunto de pontos, pois as diferenças divididas podem ser calculadas previamente e reutilizadas quantas vezes forem necessárias. Já o método de Lagrange é mais eficiente quando temos de efetuar estimativas com os mesmos valores independentes  $x$ , de modo que os produtórios que envolvem apenas  $x$ , e que acompanham os valores dependentes  $y_i$ , possam ser calculados previamente e reutilizados.

No método de Gregory-Newton é possível acrescentar um ponto qualquer no final de um conjunto de pontos discretos existente, mesmo que a sequência de pontos fique desordenada, e avaliar o novo interpolador correspondente incluindo mais uma parcela de diferenças divididas. Por exemplo, para acrescentar o 5º ponto  $(9, 5)$  à tabela de pontos do **Exemplo 5.3**, procedemos desta forma:

$i$	$x_i$	$y_i$	$\Delta^1 y_i$	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$
1	1	2	-1	2/3	-1/8	<b>3/160</b>
2	3	0	1	-1/12	<b>1/40</b>	-
3	4	1	2/3	<b>1/15</b>	-	-
4	7	3	<b>1</b>	-	-	-
5	<b>9</b>	<b>5</b>	-	-	-	-

Assim, acrescentamos apenas o termo  $\Delta^4 y_1(x-x_1)(x-x_2)(x-x_3)(x-x_4)$  ao interpolador  $P_3(x)$  existente, gerando o  $P_4(x)$  a seguir:

$$P_4(x) = 2 + (-1)(x-1) + (2/3)(x-1)(x-3) + (-1/8)(x-1)(x-3)(x-4) + \\ + (3/160)(x-1)(x-3)(x-4)(x-7)$$

No método de Lagrange, também podemos acrescentar um ponto qualquer a um conjunto de pontos existente utilizando o método de Neville, que faz uso de relações de recorrência para avaliar o novo interpolador (BURDEN; FAIRES, 2011).

Para dados  $(x_i, y_i)$  ( $i=1,2,\dots,n+1$ ) com  $x_{i+1} > x_i$  e espaçamento constante ( $x_{i+1} - x_i = h, \forall i$ ), os cálculos das diferenças divididas  $\Delta^k y_i$  podem ser simplificados. Nesse caso, definimos as chamadas diferenças finitas  $\bar{\Delta}^k y_i$ , no sentido ascendente, por:

$$\bar{\Delta} y_i = y_{i+1} - y_i \quad \Rightarrow \quad \text{diferença de 1ª ordem.}$$

$$\bar{\Delta}^k y_i = \bar{\Delta}^{k-1} y_{i+1} - \bar{\Delta}^{k-1} y_i \quad \Rightarrow \quad \text{diferença de } k\text{-ésima ordem.}$$

Logo, a relação das diferenças finitas com as diferenças divididas é a seguinte:

$$\Delta^k y_i = \frac{\bar{\Delta}^k y_i}{h^k k!}$$

Essa expressão substituída na eq. (9a) resulta no seguinte interpolador:

$$P_n(x) = y_1 + \sum_{k=1}^n \frac{\bar{\Delta}^k y_1}{h^k k!} \left[ \prod_{j=1}^k (x - x_j) \right] \quad (10)$$

Assim, nas estimativas de valores  $f(\beta) \equiv P_n(\beta)$ :

- a) Se  $\beta \in [a, b]$ , temos uma **interpolação**.
- b) Se  $\beta \notin [a, b]$ , temos uma **extrapolação**.

Caso queiramos obter o valor de  $x = \beta = ?$ , correspondente ao valor  $y = \gamma$ ,  $f(\beta) = \gamma$ , teremos uma **interpolação inversa**. Para efetuá-la, podemos:

- Se os pontos discretos forem de uma função injetora, isto é,  $\forall x_i \neq x_j \Rightarrow y_i \neq y_j$ , inverter as variáveis, tornando os  $y_i$  independentes e os  $x_i$  dependentes; efetuar a interpolação em  $(y_i, x_i)$ , obtendo o polinômio  $P_n(y)$ , e a estimativa nesse polinômio em  $y = \gamma$  gerará  $\beta = P_n(\gamma)$ .
- Se os pontos discretos não forem de uma função injetora, obter o interpolador  $P_n(x)$ ; igualar esse polinômio ao  $\gamma$ , resultando numa equação polinomial  $P_n(x) = \gamma$ ; resolver essa equação  $P_n(x) - \gamma = 0$  conforme vimos no Capítulo 3; e desconsiderar as  $n-1$  soluções espúrias.

**Exemplo 5.5:** na função, a seguir, representamos o consumo de energia elétrica, em anos, de um determinado local. Determine quando ( $t = ?$ ) o consumo atingirá o limite instalado de 7.5 MW.

$t$ (ano)	85	89	93	95	96	...	?
Consumo (MW)	5	5.7	6.2	6.7	7.0	...	7.5

**Solução:**

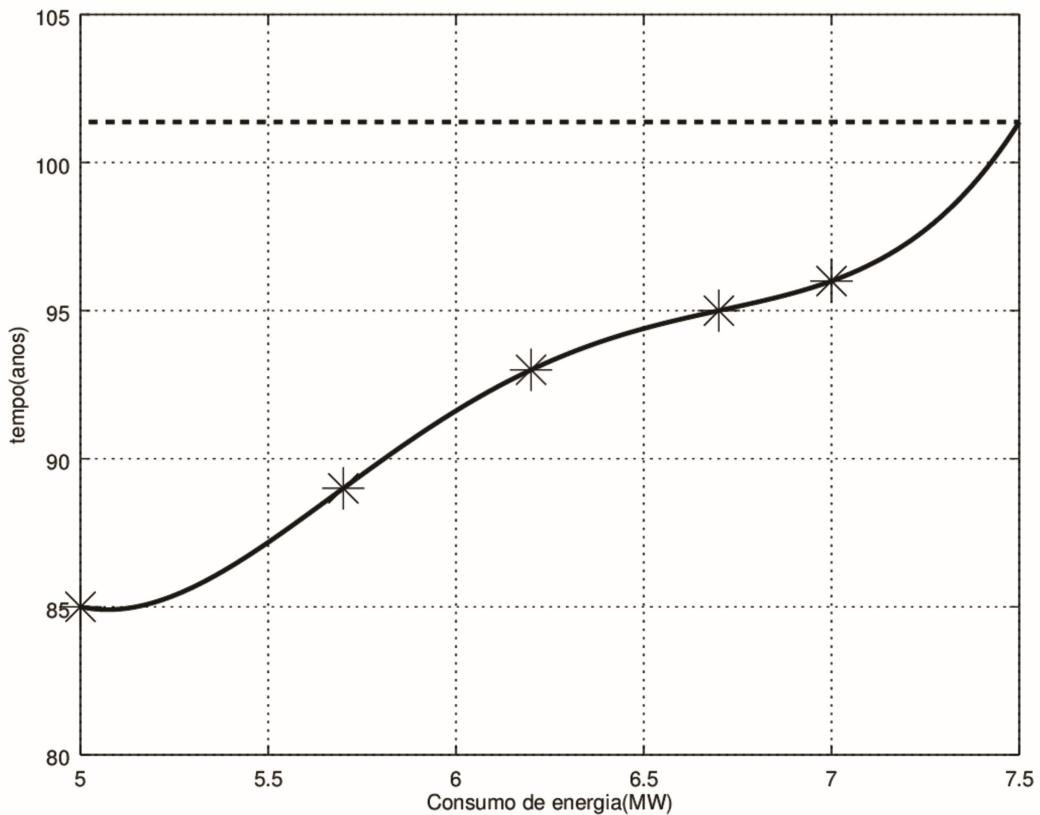
Neste exemplo, temos um problema de extrapolação (para fora dos limites de dados disponíveis).

Como os dados são de uma função injetora, se invertermos as variáveis, ela resultará na função:

Consumo (MW)	5	5.7	6.2	6.7	7.0	...	7.5
$t$ (ano)	85	89	93	95	96	...	?

No Gráfico 5.5a, temos a expressão de tempo em decorrência do consumo.

Gráfico 5.5a – Comportamento do tempo (ano) em função do consumo (MW)



Fonte: Elaboração própria

Obtendo todas as diferenças divididas do  $\Delta^k y_1$  temos:

$$\Delta^1 y_1 = 5.7143$$

$$\Delta^2 y_1 = 1.9048$$

$$\Delta^3 y_1 = -3.4734$$

$$\Delta^4 y_1 = 2.9546$$

Logo, se Consumo = 7.50MW, então tempo = 101.37anos (calculado pelo algoritmo de Gregory-Newton). Então, depois do quarto mês de 2001, o consumo deverá atingir 7.5MW.

Alternativamente, podemos também obter uma expressão para o consumo em decorrência do tempo:

$$P_4(t) = 7.5, \quad t = ?$$

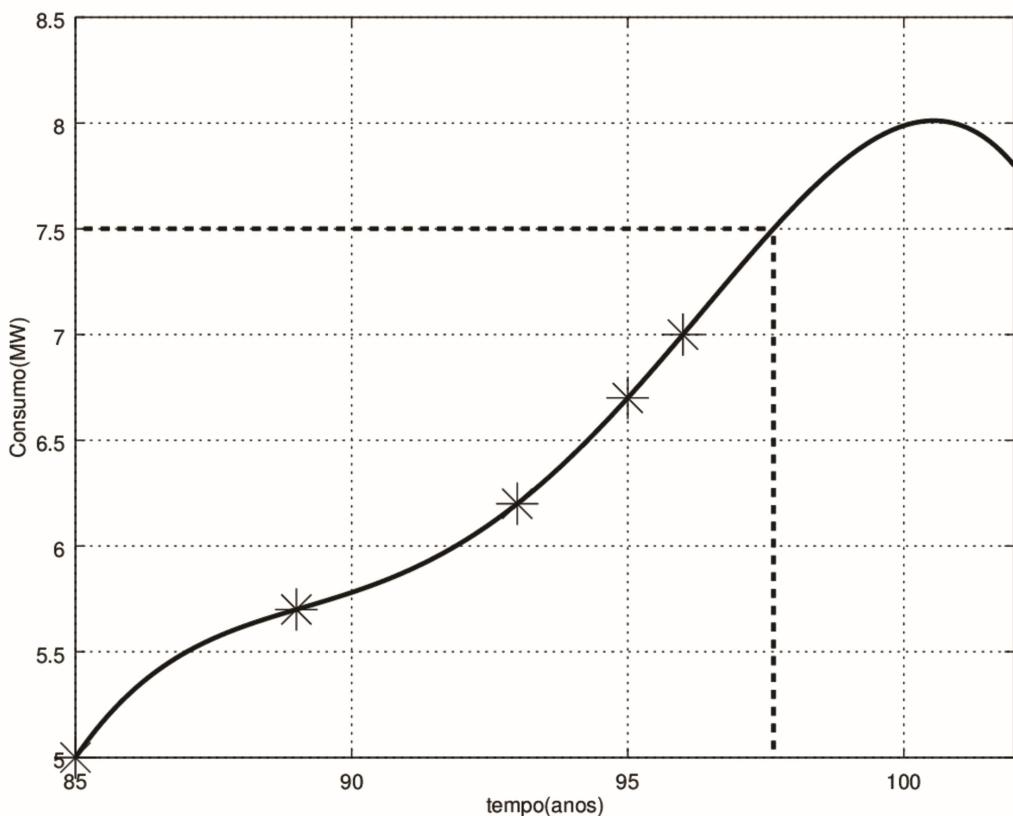
$t$ (ano)	85	89	93	95	96	...	?
Consumo(MW)	5	5.7	6.2	6.7	7.0	...	7.5

Com  $n+1=5$  pontos, obtemos o interpolador de grau  $n=4$  expresso na forma geral:

$$P_4(t) = -3.00324675446144e-04 \cdot t^4 + 1.11425865844944e-01 \cdot t^3 + \\ -1.54789204605392e+01 \cdot t^2 + 9.54391171898778e+02 \cdot t - 2.20353233848153e+04$$

$P_4(t) = 7.5 \Rightarrow t = 97.64$  (depois do 7º mês de 1997, o consumo deverá atingir 7.5MW).

Gráfico 5.5b – Comportamento do consumo de energia (MW) em função do tempo (ano)



Fonte: Elaboração própria

Observe que, em casos de extração (avaliações fora do intervalo  $[a,b]$ ), podemos ter comportamentos diferentes daquele observado no intervalo de medições  $[a,b]$ . No Gráfico 5.5a, observamos um crescimento acentuado do tempo a partir do consumo 7.0MW e, no Gráfico 5.5b, observamos uma inversão inesperada de consumo a partir do ano 2000 ( $t=100$ ). Logo, o resultado mais confiável deve ser o mais conservador, com o tempo de 97.64 anos para atingir o consumo 7.5MW. Esse é um exemplo típico de aplicação em um **problema de previsão por extração**. LINK No Capítulo 7, abordaremos uma metodologia mais adequada para esse tipo de problema. FIM DO LINK

Por fim, a aproximação por interpolação polinomial também pode ser estendida a funções com várias variáveis independentes, como veremos na seção 5.2.

### 5.1.3 Avaliação do Erro de Truncamento na Interpolação

Quando aproximamos uma função  $y=f(x)$  com expressão conhecida por um interpolador  $P_n(x)$  e  $x \in [x_1, x_{n+1}]$ , cometemos um erro de truncamento em cada  $x \neq x_i$  a ser estimado definido por:

$$Erro\ P_n(x) = |P_n(x) - f(x)|, \quad \forall x \in [x_1, x_{n+1}] \text{ e } x \neq x_i \quad (11)$$

A relação entre a aproximanda  $f(x)$  e sua interpoladora  $P_n(x)$  pode ser observada no exemplo do Gráfico 5.2, enquanto o respectivo erro pode ser visto no Gráfico 5.3.

A delimitação do erro de truncamento  $Erro\ P_n(x = \bar{x})$ , para qualquer  $\bar{x} \in [x_1, x_{n+1}]$ , indicará o grau de confiança dos resultados fornecidos pela aproximação  $P_n(\bar{x}) \cong f(\bar{x})$ .

Podemos obter o erro de truncamento do interpolador polinomial  $P_n(x)$  antes de determiná-lo via:

**Teorema 1:** se  $P_n(x)$  é o interpolador de

$i$	1	2	.....	$n + 1$
$x_i$	$x_1$	$x_2$	.....	$x_{n+1}$
$y_i = f(x_i)$	$y_1$	$y_2$	.....	$y_{n+1}$

com  $f(x)$  continuamente diferenciável em  $[x_1, x_{n+1}]$ , então  $\forall \bar{x} \in [x_1, x_{n+1}]$ ,

$$\exists \xi \in [x_1, x_{n+1}] / Erro\ P_n(\bar{x}) = |P_n(\bar{x}) - f(\bar{x})| = \left| \frac{f^{(n+1)}(\xi) \prod_{i=1}^{n+1} (\bar{x} - x_i)}{(n+1)!} \right|.$$

LINK  $f^{(n+1)}(x)$  refere-se à derivada de ordem  $(n+1)$  de  $f(x)$ . Essa demonstração da expressão do erro contida no **Teorema 1** e das demais fórmulas simplificadas na sequência podem ser encontradas na obra “Cálculo Numérico: aspectos teóricos e computacionais”, de Ruggiero e Lopes, 2<sup>a</sup> edição. FIM DO LINK

A determinação do erro de truncamento usando a expressão do **Teorema 1** é muito difícil, pois:

- a) pode ser complicado, ou impossível, obter  $f^{(n+1)}(x)$  (derivada  $n+1$ -ésima de  $f(x)$ );

- b) não sabemos qual é o valor de  $\xi$ , apenas conhecemos a sua região de localização; e
- c) para cada valor  $\bar{x}$  a ser estimado, temos que reavaliar o  $Erro P_n(\bar{x})$ .

Embora o **Teorema 1** seja de difícil aplicação, tem grande valia devido aos dois próximos corolários.

**Corolário 1:** sob as hipóteses do **Teorema 1**, se  $M = \max_{x \in [x_1, x_{n+1}]} |f^{(n+1)}(x)|$ , então  
 $Erro P_n(\bar{x}) \leq \frac{M}{(n+1)!} \left| \prod_{i=1}^{n+1} (\bar{x} - x_i) \right|$  em um ponto específico  $x = \bar{x}$ .

**DESTAQUE** Trata-se do majorante do **erro local**, em um ponto específico  $x = \bar{x}$  (limite superior do erro em  $x = \bar{x}$ ). FIM DO DESTAQUE

**Exemplo 5.6:** delimitre o erro de truncamento cometido em  $\bar{x} = 0.65$  ao aproximar  $f(x) = \ln(x)$  por um interpolador polinomial, considerando  $x \in [0.4, 0.7]$  e  $n = 3$  subdivisões.

**Solução:**

Para  $n = 3$ , então  $h = (0.7 - 0.4) / 3 = 0.1$

$i$	1	2	3	4
$x_i$	0.4	0.5	0.6	0.7
$y_i = f(x_i)$	-0.916290732	-0.693147181	-0.510825624	-0.356674944

Aplicando o **Corolário 1**, temos:

$$f(x) = \ln(x) \Rightarrow f'(x) = 1/x \Rightarrow f''(x) = -1/x^2 \Rightarrow f'''(x) = 2/x^3 \Rightarrow f^{(iv)}(x) = -6/x^4$$

$f^{(iv)}(x)$  é uma função de módulo decrescente em  $[0.4, 0.7]$ , e o seu máximo local é  $M = |f^{(iv)}(0.4)| = 234.375$ . Então,

$$\begin{aligned} Erro P_n(0.65) &\leq \frac{234.375}{(3+1)!} \left| \prod_{i=0}^3 (0.65 - x_i) \right| = \frac{234.375}{(3+1)!} |(0.65 - x_1)(0.65 - x_2)(0.65 - x_3)(0.65 - x_4)| \\ Erro P_n(0.65) &\leq \frac{234.375}{(3+1)!} |(0.65 - 0.4)(0.65 - 0.5)(0.65 - 0.6)(0.65 - 0.7)| \leq 9.15 \cdot 10^{-4} \end{aligned}$$

Comparando com o valor exato de  $\ln(x)$ , temos:

$$f(\bar{x} = 0.65) = \ln(0.65) = -0.430782916$$

$$P_3(\bar{x} = 0.65) = -0.431019619 \text{ por Lagrange.}$$

$$\text{Erro exato } P_n(\bar{x}) = |P_n(\bar{x}) - f(\bar{x})| = 2.36703 \cdot 10^{-4}.$$

Em  $\bar{x}=0.65$ , vemos que o erro exato  $2.36703 \cdot 10^{-4}$  é menor do que o erro de truncamento máximo dado pelo **Corolário 1**,  $\text{Erro } P_n(0.65) \leq 9.15 \cdot 10^{-4}$ , como era esperado. Observe que  $\bar{x}=0.65$  é um ponto médio de um dos intervalos e deve ser um dos pontos de maior erro de truncamento de todo intervalo  $[a,b]$ .

Seria interessante ter o majorante do erro global em todo o intervalo  $[a,b]$  e não apenas em um ponto específico  $x=\bar{x}$ . Assim, temos o limite superior global do erro no intervalo  $[a,b]$  dado pelo **Corolário 2**.

**Corolário 2:** se na função discretizada ocorrer  $x_{i+1} > x_i$  (pontos ordenados) e  $x_{i+1} - x_i = h$  (igualmente espaçados)  $\forall i$ , então  $\text{Erro } P_n(x) \leq \frac{M h^{n+1}}{4(n+1)}$  com

$$M = \max_{x \in [x_1, x_{n+1}]} |f^{(n+1)}(x)| \text{ em todo } x \text{ do intervalo } x \in [x_1, x_{n+1}].$$

**DESTAQUE** Trata-se do majorante do **erro global**, em todo  $x$  do intervalo. FIM DO DESTAQUE

**Exemplo 5.7:** delimitar o erro de truncamento máximo no intervalo  $x \in [0.4, 0.7]$  ao aproximar  $f(x) = \ln(x)$  por um interpolador polinomial e com  $n=3$  subdivisões. Considere os  $x_i$  ordenados e igualmente espaçados. Avalie o erro exato em  $x = 0.65$  e compare se ele fica abaixo do erro máximo do intervalo  $[a,b]$ .

**Solução:**

$$n=3 \Rightarrow h = \frac{0.7 - 0.4}{3} = 0.1$$

$$f(x) = \ln(x) \Rightarrow f'(x) = 1/x \Rightarrow f''(x) = -1/x^2 \Rightarrow f'''(x) = 2/x^3 \Rightarrow f^{(iv)}(x) = -6/x^4$$

$$M = \max_{x \in [0.4, 0.7]} |f^{(iv)}(x)| = \max | -6/x^4 | = 6/(0.4)^4 = 234.375$$

Então, o erro máximo de todo o intervalo é:

$$\text{Erro } P_n(x) \leq \left( \frac{234.375(0.1)^4}{4 \cdot 4} \right) \leq 0.00146484375$$

O erro de truncamento máximo de todo o intervalo, dado pelo **Corolário 2**,  $\text{Erro } P_n(x) \leq 0.00146484375$ , é sempre maior do que o erro máximo em algum ponto do intervalo, como em  $x = 0.65$ ,  $\text{Erro } P_n(0.65) \leq 9.15 * 10^{-4}$ , do **Exemplo 5.6**.

**Exemplo 5.8:** obtenha o erro de truncamento máximo cometido ao aproximarmos  $f(x) = e^x$  e  $x \in [2, 2.4]$  via  $P_n(x)$  com os  $x_i$  ordenados e igualmente espaçados em  $n = 4$  intervalos. Avalie o erro exato em  $x = 2.33$  e compare se ele fica abaixo do erro máximo do intervalo.

**Solução:**

$$n = 4 \Rightarrow h = \frac{2.4 - 2}{4} = 0.1$$

Temos  $f(x) = e^x \Rightarrow f'(x) = e^x \Rightarrow f''(x) = e^x \Rightarrow f'''(x) = e^x \Rightarrow f^{iv}(x) = e^x \Rightarrow f^v(x) = e^x$

$$M = \max_{x \in [2, 2.4]} |f^{(5)}(x)| = |f^{(5)}(2.4)| = e^{2.4}$$

$$\text{Então, o erro máximo } \text{Erro } P_n(x) \leq \left( \frac{e^{2.4} (0.1)^5}{4 * 5} \right) \leq 5.51159 * 10^{-6}$$

Verificação:

$$P_4(2.33) = 10.2779431277041$$

$$e^{2.33} = 10.2779415330434$$

$$\text{Erro exato } (x = 2.33) = |10.2779431277041 - 10.2779415330434| = 1.59466 * 10^{-6}.$$

Observe que o erro exato em  $x = 2.33$  fica abaixo do erro máximo:  $1.59466 * 10^{-6} < 5.51159 * 10^{-6}$ , como era esperado.

**Exemplo 5.9:** calcule o grau  $n$  mínimo do interpolador polinomial  $P_n(x)$  necessário para que o erro máximo entre o interpolador  $P_n(x)$  e  $f(x) = \ln(x)$  seja menor do que  $1 * 10^{-6}$ , em todo intervalo  $x \in [1, 2]$ , dividindo-o em  $n$  partes iguais.

**Solução:**

Uma possibilidade é usar o **Corolário 2** e estabelecer tentativas com valores de  $n$  (inteiros) para calcular o valor do erro máximo de truncamento, facilitando a obtenção das derivadas  $f^{(n+1)}(x)$ :

**Primera tentativa:**  $n = 3 \Rightarrow h = \frac{2-1}{3} = 0.333333333\dots$

$$f(x) = \ln(x) \Rightarrow f'(x) = 1/x \Rightarrow f''(x) = -1/x^2 \Rightarrow f'''(x) = 2!/x^3 \Rightarrow f^{(iv)}(x) = -3!/x^4$$

$$f^{(n+1)}(x) = (-1)^n n! x^{-(n+1)}$$

$$M = \max_{x \in [1, 2]} |f^{(4)}(x)| = 3! 1^{-4} = 6$$

Com  $\text{Erro } P_n(x) \leq \left( \frac{6 (0.33333333)^4}{4 * 4} \right) \leq 4.6496 * 10^{-3}$ , precisamos aumentar o  $n$  para baixar esse limite do erro, então vamos tentar  $n = 6$ .

**Segunda tentativa:**  $n = 6 \Rightarrow h = \frac{2-1}{6} = 0.16666666\dots$

$$M = \max_{x \in [1, 2]} |f^{(7)}(x)| = 6! 1^{-7} = 720$$

Com  $\text{Erro } P_n(x) \leq \left( \frac{720(0.1666667)^7}{4 * 7} \right) \leq 9.1858 * 10^{-5}$ , ainda precisamos aumentar o  $n$ , então vamos tentar  $n = 8$ .

**Terceira tentativa:**  $n = 8 \Rightarrow h = \frac{2-1}{8} = 0.125$

$$M = \max_{x \in [1, 2]} |f^{(9)}(x)| = 8! 1^{-9} = 40320$$

Com  $\text{Erro } P_n(x) \leq \left( \frac{40320 (0.125)^9}{4 * 9} \right) \leq 8.3447 * 10^{-6}$ , também precisamos aumentar o  $n$ ,

por isso vamos tentar  $n = 10$ .

**Quarta tentativa:**  $n = 10 \Rightarrow h = \frac{2-1}{10} = 0.1$

Temos:

$$M = \max_{x \in [1, 2]} |f^{(11)}(x)| = 10! 1^{-11} = 3628800$$

Então,  $\text{Erro } P_n(x) \leq \left( \frac{3628800 (0.1)^{11}}{4 * 11} \right) \leq 8.2473 * 10^{-7}$  é menor do que  $10^{-6}$ .

Logo, podemos usar grau  $n = 10$  para calcular o polinômio interpolador  $P_n(x)$  representativo de  $f(x) = \ln(x)$  em  $x \in [1, 2]$ , e o erro máximo de truncamento não ultrapassará  $10^{-6}$ . Com  $n = 9$ , esse erro máximo de truncamento será  $2.6018 * 10^{-6}$ , maior do que  $1 * 10^{-6}$ , mas ainda é da ordem de  $O(10^{-6})$ .

Outra possibilidade é calcular o valor do **erro exato** de truncamento em um número grande de pontos do intervalo e tomar o seu valor máximo, uma vez que a função

exata  $f(x) = \ln(x)$  está disponível para comparação através de várias bibliotecas de funções e em calculadoras. Também podemos estabelecer um algoritmo de busca do menor grau  $n$  enquanto o erro máximo exato estiver acima de  $1 \cdot 10^{-6}$ , calculando o erro para graus  $n$  crescentes do polinômio interpolador. Nesse caso, concluímos que com  $n=7$  os erros em todo o intervalo  $x \in [1, 2]$  são menores do que  $8.7302 \cdot 10^{-7}$ , conforme o arquivo **Cap5exem5.9\_busca\_menor\_n.m** do **Caderno de Algoritmos**.

## 5.2 Interpolação de Funções com Várias Variáveis Independentes

Até agora efetuamos interpolações de funções com uma única variável independente, unidimensionais; entretanto, na modelagem matemática da maioria dos fenômenos, ocorre a influência de várias variáveis independentes sobre o valor observado. Uma aplicação típica é a modelagem de superfícies de uma área geográfica, a partir das cotas (alturas) nos pontos de uma grade retangular, para fins de cálculos topográficos, de computação gráfica, entre outros.

Por exemplo, como estimar o valor de  $w = f(u, v)$  quando a função aproximada depende de duas variáveis independentes  $x$  e  $y$ ?

Seja uma função com duas variáveis independentes a ser aproximada por um polinômio interpolador:

$$\begin{cases} f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \\ (x, y) \rightarrow z = f(x, y) \end{cases} \text{ com } x \in [a, b] \text{ e } y \in [c, d] \text{ (domínio retangular).}$$

Para determinar o interpolador, acompanhe o passo a passo a seguir.

**Primeiro passo:** dividimos  $[a, b]$  em  $n$  partes gerando  $(n+1)$   $x_i$  e

$[c, d]$  em  $m$  partes gerando  $(m+1)$   $y_j$ .

**Segundo passo:** obtemos  $(n+1)(m+1)$  valores amostrais  $z_{ij} = f(x_i, y_j)$  da aproximanda, conforme a matriz a seguir:

$x_i$	$x_1$	$x_2$	$x_3$	...	$x = u$	...	$x_{n+1}$
$y_j$	$z_{11}$	$z_{21}$	$z_{31}$		$\downarrow$		$z_{(n+1)1}$
$y_2$	$z_{12}$	$z_{22}$	$z_{32}$		$\downarrow$		$z_{(n+1)2}$
$\vdots$					$\downarrow$		
$y = v$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$f(u, v) = ?$		
$\vdots$							
$y_{m+1}$	$z_{1(m+1)}$	$z_{2(m+1)}$	$z_{3(m+1)}$				$z_{(n+1)(m+1)}$

**Terceiro passo:** fixamos uma das variáveis (por convenção será fixada o  $x$ ).

**Quarto passo:** para cada coluna de  $x_i$  fixada,  $i = 1$  a  $n+1$ , separamos os pontos:

$y_j$	$y_1$	$y_2$	$y_3$	...	$v$	...	$y_{m+1}$
$z_{ij}$	$z_{i1}$	$z_{i2}$	$z_{i3}$		$t_i = PY_m(y = v)$		$z_{1(m+1)}$

Interpolando em  $y = v$ , geramos um  $t_i = PY_m(y = v)$  através de seu interpolador simples de grau  $m$ ,  $PY_m(y)$ .

**Quinto passo:** do passo anterior resulta o conjunto  $t_i$  em função de  $x_i$ :

$x_i$	$x_1$	$x_2$	$x_3$	...	$x = u$	...	$x_{n+1}$
$t_i$	$t_1$	$t_2$	$t_3$		$PX_n(x = u)$		$t_{(n+1)}$

Agora, obtemos o seu interpolador de grau  $n$ ,  $PX_n(x)$  e estimamos  $f(u, v) \approx PX_n(x = u)$ .

Observe que a estimativa via interpolação de um único  $f(u, v)$  envolve  $(n+1)+1=(n+2)$  interpolações simples. Podemos estender esse mesmo raciocínio para três, quatro, ...,  $k$  variáveis independentes. Por exemplo, a estimativa de  $f(u, v, w)$ , em três dimensões, envolve  $(n+1)$  interpolações duplas e uma simples no final  $\Rightarrow (n+1)(n+2)+1=n^2+3n+3$  interpolações simples. Logo, a interpolação múltipla provoca um crescimento exponencial no volume de operações a serem executadas. Por isso, é muito importante usar um interpolador simples envolvendo o menor número de operações aritméticas possível.

Com o [Exemplo 5.10](#), vamos apresentar um caso de interpolação bidimensional via

## Gregory-Newton.

**Exemplo 5.10:** monte um algoritmo que aproxime a  $f(x, y) = x^4 + y^4$  em  $xt = 8.7$  e  $yt = 5.5$  via interpolação dupla com Gregory-Newton, dados 5 valores de  $x$  e 6 valores de  $y$  independentes a seguir (na forma de vetor do Octave):

$x_i$	8.5	8.9	9.3	9.5	9.6
$y_j$					
5.0					
5.7					
6.2					
6.7					
7.0					
7.5					

O algoritmo do **Exemplo 5.10** está disponível no arquivo **Cap5interpolacao2D.m** do **Caderno de Algoritmos**.

Essa forma multidimensional de interpolação pode ser estendida através de outras **bases** dos polinômios interpoladores, como de Lagrange, em cada direção (RICE, 1983). Por Lagrange, devemos aplicar o primeiro e segundo passos da forma de interpolação anterior, definir os valores de  $z_{ij} = f(x_i, y_j)$  e obter o valor interpolado em qualquer  $x$  e  $y$  diretamente por meio de polinômio duplo  $P_{nm}(x, y)$ , de grau  $n$  em  $x$  e  $m$  em  $y$ , conforme a equação a seguir,

$$P_{nm}(x, y) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} z_{ij} L_i(x) L_j(y)$$

Em que,

$$L_i(x) = \prod_{\substack{k=1 \\ i \neq k}}^{n+1} \frac{(x - x_k)}{(x_i - x_k)}, \quad \forall i = 1, \dots, n+1$$

$$L_j(y) = \prod_{\substack{k=1 \\ j \neq k}}^{m+1} \frac{(y - y_k)}{(y_j - y_k)}, \quad \forall j = 1, \dots, m+1$$

Os polinômios de Lagrange  $L_i(x)$  e  $L_j(y)$  geram fatores peso fixos para um conjunto bidimensional fixo de pontos, ou seja, para uma mesma malha de pontos discretos  $x_i$ ,  $y_j$  e uma mesma posição  $(x, y)$  de interpolação, o que torna essa forma de interpolação mais eficiente para conjuntos de pontos fixos em que

tenhamos apenas diferentes valores de  $z_{ij}$ .

Agora confira o algoritmo do [Exemplo 5.10](#) com interpolação bidimensional via polinômios de Lagrange no arquivo [Cap5interpolacao2D.m](#).

**DESTAQUE** A interpolação de Lagrange bidimensional também só é válida para um domínio retangular,  $x \in [a,b]$  e  $y \in [c,d]$ , ou seja, não é válida para domínios de pontos aleatoriamente distribuídos no espaço. Neste último caso, podemos aplicar uma interpolação com erro de 2<sup>a</sup> ordem conhecida como **pseudolaplaciano**, proposta por Holmes e Connell (1989). Nesse método, calculamos um valor  $P_{nm}(x,y)$  aproximado usando apenas alguns valores de  $z_{ij} = f(x_i, y_j)$  conhecidos, baseado no cálculo adequado de fatores peso aplicado a cada valor discretizado  $z_{ij}$ , de tal forma que as aproximações geradas são exatas para funções lineares. Essa metodologia por ser aplicada para domínios multidimensionais. FIM DO DESTAQUE

Lembramos que a função aproximadora  $z = g(x)$  pode pertencer a outras famílias de funções, como vimos no início deste capítulo, não restringindo-se apenas às polinomiais, como o exemplo no exercício 4.5 do Capítulo 4. O importante é que a condição de aproximação seja mantida, ou seja, os **erros ou desvios locais**, sobre os pontos escolhidos para definir a aproximadora, **devem ser nulos**.

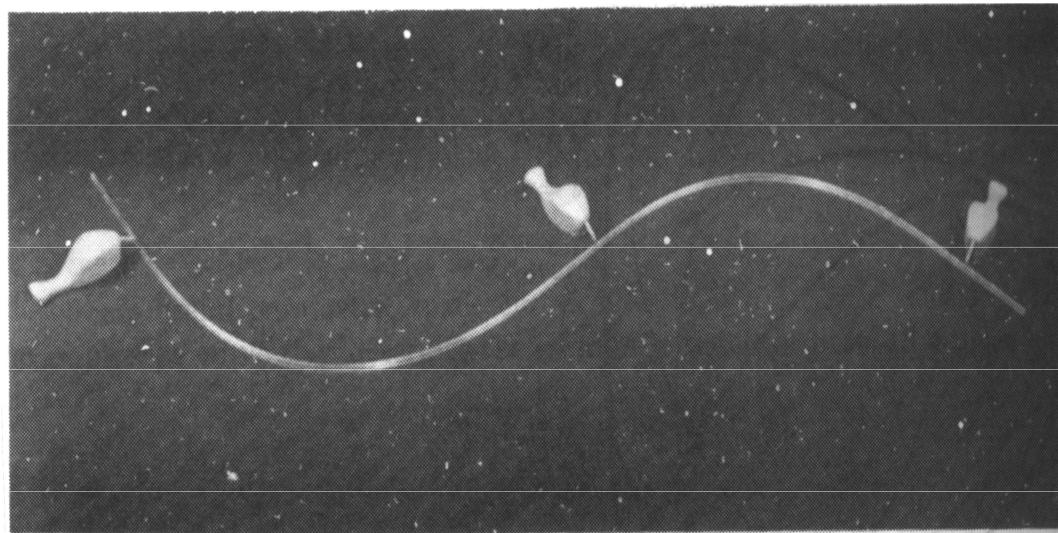
[Na próxima seção, vamos apresentar alternativas de aproximação mais adequadas para desenhar curvas.](#)

### 5.3 Aproximação por Interpolação *Spline*

Originalmente, *splines* eram réguas flexíveis, de madeira ou plástico, que podiam ser curvadas de forma a passar por um dado conjunto de pontos  $(x_i, y_i)$  chamados de “nós”. Essas *splines* originais usavam pesos (*ducks*) que eram fixados nas áreas de interesse causando a deformação da estrutura de madeira ou plástico de acordo com a curva desejada (Figura 5.1). Foram muito utilizadas em desenhos de engenharia nos tempos em que não havia recursos computacionais. Apesar de serem usadas desde o século XIX, somente no fim da década de 1960 foi desenvolvida uma formulação matemática desse problema. Tal formalização possibilitou o desenvolvimento de vários sistemas computadorizados que utilizam

aproximações gráficas de funções como os conhecidos CAD/CAM.

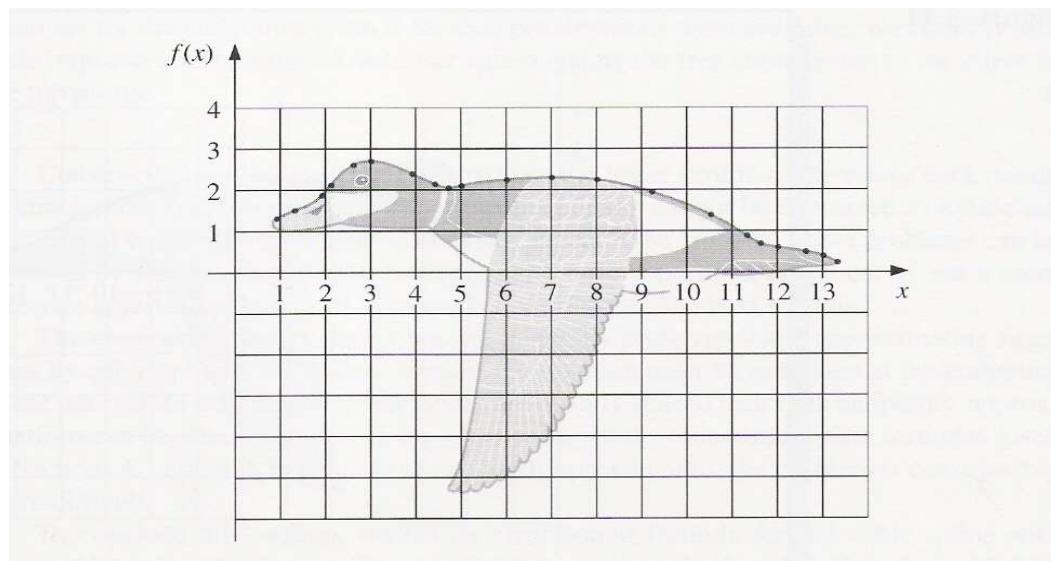
Figura 5.1 – *Spline* física original e pesos



Fonte: Instituto de Computação UFF (2016)

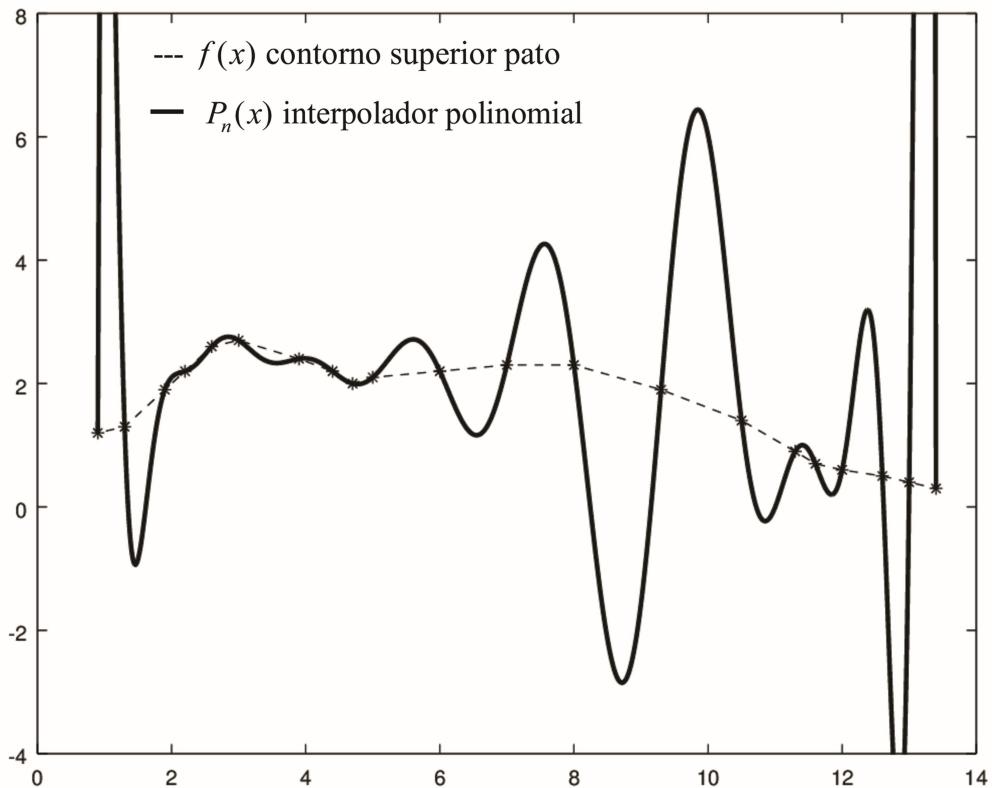
As maiores dificuldades relacionadas à interpolação polinomial convencional (Lagrange, Gregory-Newton etc.) ocorrem quando tomamos um polinômio interpolante de grau pequeno, que gera elevados erros de truncamento, ou um polinômio interpolador de grau elevado, que tende a ter gráficos sinuosos. Veja a representação do contorno superior do Gráfico 5.6 usando interpolação polinomial conforme o Gráfico 5.7.

Gráfico 5.6 – Representação do contorno superior de um pato voando



Fonte: Burden e Faires (2011)

Gráfico 5.7 – Contorno superior do pato voando representado por 21 pontos, conectados e destacados com marcador \*, e interpolador polinomial de grau  $n = 20$ , em linha contínua



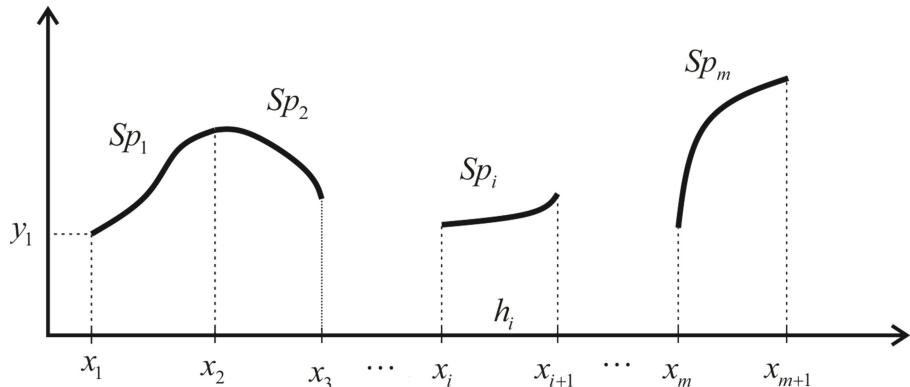
Fonte: Elaboração própria

Nas aplicações em que a aproximanda muda de comportamento no domínio de interesse, como a do Gráfico 5.6, a aproximadora deve também acompanhar essas mudanças e efetuar as junções de modo mais suave possível, evitando as sinuosidades como as da interpoladora do Gráfico 5.7.

Uma técnica de aproximação possível consiste em dividir o intervalo de interesse em vários subintervalos e aproximá-los separadamente com polinômio de grau pequeno, efetuando as junções destes polinômios da forma mais suave possível.

Definição 3: **interpolação spline** é uma técnica de aproximação que consiste em dividir o domínio  $[a, b]$  de interesse e interpolar separadamente cada subintervalo, efetuando as junções entre os interpoladores da forma mais suave possível, conforme o Gráfico 5.8.

Gráfico 5.8 – Splines genéricas aplicadas em cada subintervalo do domínio



Fonte: Elaboração própria

Por conveniência teórica e prática, e por questões de otimização, são utilizados interpoladores polinomiais cúbicos  $Sp_i(x)$ , com grau 3 fixo (*splines cúbicas*). Essa técnica e suas variantes constituem um dos fundamentos da computação gráfica.

Para aproximar um caminho funcional  $y=f(x)$ ,  $x \in [a, b]$  por **splines cúbicas**, procedemos da seguinte maneira:

**Primeiro passo:** dividimos  $[a, b]$  em  $m$  subintervalos convenientes  $[x_i, x_{i+1}]$  tal que  $x_{i+1} > x_i$  e comprimentos  $h_i = x_{i+1} - x_i$ , para  $i=1,2,\dots,m$ , e geramos  $m+1$  pontos amostrais  $(x_i, y_i)$  do caminho.

**Segundo passo:** obtemos um polinômio aproximador  $Sp_i(x)$  de grau 3 para cada um dos  $m$  subintervalos  $[x_i, x_{i+1}]$ , com a expressão:

$$Sp_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad \text{para } i=1,2,\dots,m \quad (12)$$

satisfazendo as seguintes **condições de aproximação**:

- a)  $Sp_i(x_i) = y_i$ , para todo ponto  $i=1,2,\dots,m$  e  $Sp_m(x_{m+1}) = y_{m+1}$ .
- b)  $Sp_{i-1}(x_i) = Sp_i(x_i) = y_i$ , para todo ponto interno  $i=2,\dots,m \Rightarrow$  Condição de **continuidade**.

Note que cada polinômio  $Sp_i(x)$ , relativo ao intervalo  $[x_i, x_{i+1}]$ , deve passar pelos seus dois pontos extremos  $(x_i, y_i)$  e  $(x_{i+1}, y_{i+1})$ . Logo, em cada ponto  $x_i$ , os valores dos dois polinômios que nele incidem são iguais.

c)  $Sp'_{i-1}(x_i) = Sp'_i(x_i)$ , para todo ponto  $x_i$ ,  $i=2,\dots,m \Rightarrow$  Condição de suavidade.

Ou seja, em cada ponto  $x_i$ , a inclinação dos dois polinômios que nele incidem são iguais.

d)  $Sp''_{i-1}(x_i) = Sp''_i(x_i)$ , para todo ponto  $x_i$ ,  $i=2,\dots,m \Rightarrow$  Velocidade de encurvamento.

Ou seja, em cada ponto interno, a velocidade de encurvamento dos dois polinômios que nele incidem são iguais.

---

**DESTAQUE** Num **spline mecânico**, conforme a Figura 5.1, essas propriedades correspondem a:

- a) o **spline** deve passar sobre os “nós”  $(x_i, y_i)$ ;
- b) o **spline** não quebra ou não forma ângulos agudos; e
- c) o **spline** assume a forma que minimiza a energia potencial.

FIM DESTAQUE

**Terceiro passo:** estimamos  $f(u)$  primeiro localizando o subintervalo que contenha o  $u$ . Definido que  $u \in [x_i, x_{i+1}] \Rightarrow f(u) \equiv Sp_i(u)$ , conforme o Gráfico 5.8.

Para definir os  $Sp_i(x)$  de cada subintervalo  $[x_i, x_{i+1}]$ , com  $i=1,2,\dots,m$ , expressos conforme a eq. (12), calculamos a primeira e segunda derivadas de  $Sp_i(x)$ :

$$Sp'_i(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i \quad (13)$$

$$Sp''_i(x) = 6a_i(x - x_i) + 2b_i \quad (14)$$

renomeamos as derivadas de segunda ordem da eq. (14):

$$\begin{aligned} S_i &= Sp''_i(x_i) \text{ e} \\ S_{i+1} &= Sp''_{i+1}(x_{i+1}) = Sp''_i(x_{i+1}). \end{aligned}$$

e aplicamos as condições estabelecidas nos itens (a), (b), (c) e (d) do **segundo passo**:

- a) Condição (a) aplicada na eq. (12):

$$Sp_i(x_i) = d_i = y_i \quad (15)$$

$$Sp_i(x_{i+1}) = a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i = y_{i+1} \quad (16)$$

b) Condição (d) aplicada na eq. (14):

$$Sp''_i(x_i) = 2b_i = S_i \Rightarrow b_i = S_i / 2 \quad (17)$$

$$Sp''_i(x_{i+1}) = 6a_i h_i + 2b_i = S_{i+1} \quad (18)$$

Substituindo a eq. (17) na eq. (18), temos:

$$a_i = \frac{(S_{i+1} - S_i)}{6h_i} \quad (19)$$

E substituindo as eqs. (19), (17) e (15) na eq. (16), temos:

$$\frac{(S_{i+1} - S_i)}{6h_i} h_i^3 + \frac{S_i}{2} h_i^2 + c_i h_i + y_i = y_{i+1}$$

Então,

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{S_{i+1} h_i + 2S_i h_i}{6} \quad (20)$$

Agrupando os coeficientes, temos:

$$\begin{cases} a_i = (S_{i+1} - S_i) / (6h_i) \\ b_i = S_i / 2 \\ c_i = (y_{i+1} - y_i) / h_i - (S_{i+1} + 2S_i) h_i / 6 \\ d_i = y_i \end{cases} \quad \forall i = 1, 2, \dots, m \quad (21)$$

Assim, temos os coeficientes das *splines* cúbicas em função dos valores das curvaturas  $S_i$  e  $S_{i+1}$  nas extremidade de cada subintervalo. Para obter esses valores, calculamos  $Sp'_i(x_i)$  e  $Sp'_{i-1}(x_i)$  através da eq. (13) aplicada em  $x_i$ :

$$\begin{aligned} Sp'_i(x_i) &= c_i \\ Sp'_{i-1}(x_i) &= 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1} \end{aligned}$$

e utilizamos a condição (c) do **segundo passo**,  $Sp'_i(x_i) = Sp'_{i-1}(x_i)$ , resultando em:

$$c_i = 3a_{i-1}h_{i-1}^2 + 2b_{i-1}h_{i-1} + c_{i-1} \quad (22)$$

Substituindo as expressões da eq. (21) na eq. (22), temos:

$$h_{i-1}S_{i-1} + 2(h_{i-1} + h_i)S_i + h_iS_{i+1} = 6[(y_{i+1} - y_i) / h_i - (y_i - y_{i-1}) / h_{i-1}] \quad (23)$$

para  $i = 2, \dots, m$  ( $m-1$  equações), resultando no seguinte sistema de equações lineares não quadrado:

$$\begin{bmatrix} h_1 & 2(h_1 + h_2) & h_2 \\ h_2 & 2(h_2 + h_3) & h_3 \\ \ddots & \ddots & \ddots \\ h_{m-1} & 2(h_{m-1} + h_m) & h_m \end{bmatrix} * \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ \vdots \\ S_m \\ S_{m+1} \end{bmatrix} = 6 \begin{bmatrix} \frac{y_3 - y_2}{h_2} - \frac{y_2 - y_1}{h_1} \\ \frac{y_4 - y_3}{h_3} - \frac{y_3 - y_2}{h_2} \\ \vdots \\ \vdots \\ \frac{y_{m+1} - y_m}{h_m} - \frac{y_m - y_{m-1}}{h_{m-1}} \end{bmatrix} \quad (24)$$

Como temos  $m+1$  pontos e consequentemente  $m+1$  incógnitas  $S_i$  e apenas  $m-1$  equações ( $i=2,3,\dots,m$ ) na eq. (24), então temos um sistema possível, mas indeterminado. Para que tenhamos solução única, isto é, sem ambiguidades, precisamos impor duas condições especiais (duas equações) diretamente nos pontos extremos de  $[a, b]$  envolvendo  $S_1$  e  $S_{m+1}$ . Dependendo de tais condições, podemos ter vários tipos de *splines* cúbicas:

- a) Supondo que o caminho tende a ser **linear nos extremos**, o que equivale a prescrever duas equações adicionais,  $S_1 = 0$  e  $S_{m+1} = 0$ , com soluções explícitas para  $S_1$  e  $S_{m+1}$ , que, substituídas diretamente na eq. (24), geram o sistema tridiagonal a seguir:

$$\begin{bmatrix} 2(h_1 + h_2) & h_2 \\ h_2 & 2(h_2 + h_3) & h_3 \\ \ddots & \ddots & \ddots \\ h_{m-1} & 2(h_{m-1} + h_m) & h_m \end{bmatrix} * \begin{bmatrix} S_2 \\ S_3 \\ \vdots \\ S_m \end{bmatrix} = 6 \begin{bmatrix} \frac{y_3 - y_2}{h_2} - \frac{y_2 - y_1}{h_1} \\ \frac{y_4 - y_3}{h_3} - \frac{y_3 - y_2}{h_2} \\ \vdots \\ \vdots \\ \frac{y_{m+1} - y_m}{h_m} - \frac{y_m - y_{m-1}}{h_{m-1}} \end{bmatrix} \quad (25)$$

- b) Supondo que o caminho tende a ser de **forma quadrática** nos extremos, o que equivale a prescrever duas equações adicionais,  $S_1 = S_2$  e  $S_{m+1} = S_m$ , que também podem ser substituídas diretamente na eq. (24), gerando o sistema linear tridiagonal a seguir:

$$\left[ \begin{array}{ccc} (3h_1 + 2h_2) & h_2 & \\ h_2 & 2(h_2 + h_3) & h_3 \\ & \ddots & \\ & & \ddots \\ h_{m-1} & & (2h_{m-1} + 3h_m) \end{array} \right] * \begin{bmatrix} S_2 \\ S_3 \\ \vdots \\ S_m \end{bmatrix} = 6 \begin{bmatrix} \frac{y_3 - y_2 - y_2 - y_1}{h_2} \\ \frac{y_4 - y_3 - y_3 - y_2}{h_3} \\ \vdots \\ \frac{y_{m+1} - y_m - y_m - y_{m-1}}{h_m} \end{bmatrix} \quad (26)$$

- c) Quando os valores das curvaturas nos **extremos** são **conhecidos** previamente, isto é,  $S_1 = u$  e  $S_{m+1} = v$ , basta substituí-los na eq. (24), gerando o sistema tridiagonal a seguir:

$$\left[ \begin{array}{ccc} 2(h_1 + h_2) & h_2 & \\ h_2 & 2(h_2 + h_3) & h_3 \\ & \ddots & \\ & & \ddots \\ h_{m-1} & & 2(h_{m-1} + h_m) \end{array} \right] * \begin{bmatrix} S_2 \\ S_3 \\ \vdots \\ S_m \end{bmatrix} = 6 \begin{bmatrix} -h_1 u + \frac{y_3 - y_2 - y_2 - y_1}{h_2} \\ \frac{y_4 - y_3 - y_3 - y_2}{h_3} \\ \vdots \\ -h_m v + \frac{y_{m+1} - y_m - y_m - y_{m-1}}{h_m} \end{bmatrix} \quad (27)$$

- d) Se assumirmos valores de curvatura para os extremos  $S_1$  e  $S_{m+1}$  obtidos por **extrapolação linear** de  $S_2$  e  $S_3$ , e de  $S_{m-1}$  e  $S_m$ , respectivamente, teremos também duas equações adicionais:

$$\frac{S_2 - S_1}{h_1} = \frac{S_3 - S_2}{h_2} \quad \text{e} \quad \frac{S_{m+1} - S_m}{h_m} = \frac{S_m - S_{m-1}}{h_{m-1}}$$

Que também devem ser adicionadas ao sistema geral dado pela eq. (24), mas agora gerando um sistema com lei de formação diferenciada para as duas equações adicionais:

$$\text{Para } i=1 \Rightarrow h_2 S_1 - (h_1 + h_2) S_2 + h_1 S_3 = 0$$

$$\text{Para } i=m+1 \Rightarrow h_m S_{m-1} - (h_{m-1} + h_m) S_m + h_{m-1} S_{m+1} = 0$$

Assim, gerando um sistema de  $m+1$  equações para resolver diretamente as  $m+1$  incógnitas com dois coeficientes extras à matriz tridiagonal (destacados em negrito), temos:

$$\left[ \begin{array}{ccc|c} h_2 & -(h_1+h_2) & h_1 & \\ h_1 & 2(h_1+h_2) & h_2 & \\ h_2 & 2(h_2+h_3) & h_3 & \\ & \ddots & \ddots & \\ h_{m-2} & 2(h_{m-2}+h_{m-1}) & h_{m-1} & \\ h_{m-1} & 2(h_{m-1}+h_m) & h_m & \\ h_m & -(h_{m-1}+h_m) & h_{m-1} & \end{array} \right] * \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ \vdots \\ S_{m-1} \\ S_m \\ S_{m+1} \end{bmatrix} = 6 \begin{bmatrix} 0 \\ (y_3 - y_2)/h_2 - (y_2 - y_1)/h_1 \\ (y_4 - y_3)/h_3 - (y_3 - y_2)/h_2 \\ \vdots \\ (y_m - y_{m-1})/h_{m-1} - (y_{m-1} - y_{m-2})/h_{m-2} \\ (y_{m+1} - y_m)/h_m - (y_m - y_{m-1})/h_{m-1} \\ 0 \end{bmatrix} \quad (28)$$

Note que o sistema linear dado pela eq. (28) não é mais tridiagonal devido às duas equações adicionais.

Em todos os casos, depois de resolvidos os sistemas, temos uma solução única para  $S_1, S_2, S_3, \dots, S_{m+1}$ , já incluída a respectiva condição de extremos  $S_1$  e  $S_{m+1}$ . Substituindo os valores de  $S_i$  nas eqs. (21), obtemos os coeficientes  $a_i, b_i, c_i, d_i$ , das  $m$  *splines* cúbicas no intervalo  $[a, b]$ , conforme a eq. (12).

**Exemplo 5.11:** aproxime a função discretizada, a seguir, por *splines* cúbicas dos 4 tipos apresentados, em  $[0, 4]$ , com 4 subintervalos  $h=1$ , e estime  $f(x=2.5)$ .

$i$	1	2	3	4	5
$x_i$	0	1	2	3	4
$y_i$	-3	-2	5	24	61

(esse conjunto de pontos provém da  $f(x) = x^3 - 3$ )

**Solução:**

Então,  $h_i = 1$  para  $i=1,2,\dots,4$  ( $m=4$  intervalos e  $m+1=5$  pontos):

a) Com *splines* na forma linear nos extremos, conforme a eq. (25):

$$\left[ \begin{array}{ccc|c} 2(h_1+h_1) & h_2 & 0 & \\ h_2 & 2(h_2+h_3) & h_3 & \\ 0 & h_3 & 2(h_3+h_4) & \end{array} \right] * \begin{bmatrix} S_2 \\ S_3 \\ S_4 \end{bmatrix} = 6 \begin{bmatrix} (y_3 - y_2)/h_2 - (y_2 - y_1)/h_1 \\ (y_4 - y_3)/h_3 - (y_3 - y_2)/h_2 \\ (y_5 - y_4)/h_4 - (y_4 - y_3)/h_3 \end{bmatrix}$$

Calculando os valores, obtemos o sistema:

$$\begin{bmatrix} 4 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 4 \end{bmatrix} * \begin{bmatrix} S_2 \\ S_3 \\ S_4 \end{bmatrix} = \begin{bmatrix} 36 \\ 72 \\ 108 \end{bmatrix}$$

Sendo a solução  $S = \{S_2, S_3, S_4\} = \{6.42857, 10.28571, 24.42857\}$ , com  $S_1 = S_5 = 0$ .

Assim, para o primeiro intervalo,  $x \in [0, 1]$ ,  $i = 1$ , temos:

$$\begin{cases} a_1 = (S_2 - S_1) / (6h_1) = 1.07143 \\ b_1 = S_1 / 2 = 0 \\ c_1 = (y_2 - y_1) / h_1 - (S_2 + 2S_1)h_1 / 6 = -0.071429 \\ d_1 = y_1 = -3 \end{cases}$$

$$Sp_1(x) = 1.07143(x-0)^3 + 0(x-0)^2 - 0.071429(x-0) - 3 \quad \text{para } x \in [0, 1].$$

As demais *splines* são as seguintes:

$$Sp_2(x) = 0.64286(x-1)^3 + 3.21429(x-1)^2 + 3.142857(x-0) - 2, \quad x \in [1, 2]$$

$$Sp_3(x) = 2.35714(x-2)^3 + 5.14286(x-2)^2 + 11.50000(x-2) + 5, \quad x \in [2, 3]$$

$$Sp_4(x) = -4.07143(x-3)^3 + 12.21429(x-3)^2 + 28.857143(x-3) + 24, \quad x \in [3, 4]$$

Podemos obter uma aproximação de  $f(x=2.5)$  via  $Sp_3(x=2.5) = 12.330$ .

b) Com *splines* de extremos quadráticos, conforme a eq. (26):

$$\begin{bmatrix} (3h_1 + 2h_2) & h_2 & 0 \\ h_2 & 2(h_2 + h_3) & h_3 \\ 0 & h_3 & (2h_3 + 3h_4) \end{bmatrix} * \begin{bmatrix} S_2 \\ S_3 \\ S_4 \end{bmatrix} = 6 \begin{bmatrix} (y_3 - y_2) / h_2 - (y_2 - y_1) / h_1 \\ (y_4 - y_3) / h_3 - (y_3 - y_2) / h_2 \\ (y_5 - y_4) / h_4 - (y_4 - y_3) / h_3 \end{bmatrix}$$

Calculando os valores, obtemos o sistema:

$$\begin{bmatrix} 5 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 5 \end{bmatrix} * \begin{bmatrix} S_2 \\ S_3 \\ S_4 \end{bmatrix} = \begin{bmatrix} 36 \\ 72 \\ 108 \end{bmatrix}$$

Sendo a solução  $S = \{S_2, S_3, S_4\} = \{4.8, 12, 19.2\}$ , com  $S_1 = S_2 = 4.8$  e com  $S_{m+1} = S_m = 19.2$  ( $m = 4$ ).

Assim,

$$Sp_1(x) = 0(x-0)^3 + 2.4(x-0)^2 - 1.4(x-0) - 3 \quad \text{para } x \in [0, 1]$$

$$Sp_2(x) = 1.2(x-1)^3 + 2.4(x-1)^2 + 3.4(x-0) - 2 \quad \text{para } x \in [1, 2]$$

$$Sp_3(x) = 1.2(x-2)^3 + 6(x-2)^2 + 11.8(x-2) + 5 \quad \text{para } x \in [2, 3]$$

$$Sp_4(x) = 0(x-3)^3 + 9.6(x-3)^2 + 27.4(x-3) + 24 \quad \text{para } x \in [3, 4]$$

Podemos obter uma aproximação de  $f(x=2.5)$  via  $Sp_3(x=2.5) = 12.550$ .

c) Com *splines* de extremos conhecidos, conforme a eq. (27):

Por exemplo:  $S_1 = u = 0$  e  $S_2 = v = 24$  (que são os valores exatos de  $f(x) = x^3 - 3$  discretizada).

$$\begin{bmatrix} 2(h_1 + h_1) & h_2 & 0 \\ h_2 & 2(h_2 + h_3) & h_3 \\ 0 & h_3 & 2(h_3 + h_4) \end{bmatrix} * \begin{bmatrix} S_2 \\ S_3 \\ S_4 \end{bmatrix} = 6 \begin{bmatrix} -h_1 u + (y_3 - y_2)/h_2 - (y_2 - y_1)/h_1 \\ (y_4 - y_3)/h_3 - (y_3 - y_2)/h_2 \\ -h_m v + (y_5 - y_4)/h_4 - (y_4 - y_3)/h_3 \end{bmatrix}$$

Calculando os valores, obtemos o sistema:

$$\begin{bmatrix} 4 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 4 \end{bmatrix} * \begin{bmatrix} S_2 \\ S_3 \\ S_4 \end{bmatrix} = \begin{bmatrix} 36 \\ 72 \\ 84 \end{bmatrix}$$

Sendo a solução  $S = \{S_1, S_2, S_3, S_4, S_5\} = \{0, 6.0, 12.0, 18.0, 24.0\}$

Assim,

$$Sp_1(x) = 1(x-0)^3 + 0(x-0)^2 + 0(x-0) - 3 \quad \text{para } x \in [0, 1]$$

$$Sp_2(x) = 1(x-1)^3 + 3(x-1)^2 + 3(x-0) - 2 \quad \text{para } x \in [1, 2]$$

$$Sp_3(x) = 1(x-2)^3 + 6(x-2)^2 + 12(x-2) + 24 \quad \text{para } x \in [2, 3]$$

$$Sp_4(x) = 1(x-3)^3 + 9(x-3)^2 + 27(x-3) + 24 \quad \text{para } x \in [3, 4]$$

Podemos obter uma aproximação de  $f(x=2.5)$  via  $Sp_3(x=2.5)=12.625$ .

- d) Com *splines* de extremos extrapolados a partir das duas curvaturas internas vizinhas, conforme a eq. (28):

$$\begin{bmatrix} h_2 & -(h_1 + h_2) & h_1 \\ h_1 & 2(h_1 + h_2) & h_2 \\ h_2 & 2(h_2 + h_3) & h_3 \\ h_{m-1} & 2(h_{m-1} + h_m) & h_m \\ h_m & -(h_{m-1} + h_m) & h_{m-1} \end{bmatrix} * \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \end{bmatrix} = 6 \begin{bmatrix} 0 \\ (y_3 - y_2)/h_2 - (y_2 - y_1)/h_1 \\ (y_4 - y_3)/h_3 - (y_3 - y_2)/h_2 \\ (y_5 - y_4)/h_4 - (y_4 - y_3)/h_3 \\ 0 \end{bmatrix}$$

Observamos que esse sistema é quase tridiagonal, exigindo apenas duas alterações no algoritmo (ambos coeficientes adicionais destacados em negrito):

$$\begin{bmatrix} 1 & -2 & \mathbf{I} \\ 1 & 4 & 1 \\ 1 & 4 & 1 \\ 1 & 4 & 1 \\ \mathbf{I} & -2 & 1 \end{bmatrix} * \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 36 \\ 72 \\ 108 \\ 0 \end{bmatrix}$$

Sendo a solução:

$$S = \{S_1, S_2, S_3, S_4, S_5\} = \{-4.6957, 3.9130, 12.5217, 18.0000, 23.4783\}.$$

Assim,

$$Sp_1(x) = 1.43478(x-0)^3 - 2.3478(x-0)^2 + 1.9130(x-0) - 3 \quad \text{para } x \in [0, 1]$$

$$Sp_2(x) = 1.43478(x-1)^3 + 1.9565(x-1)^2 + 3.6087(x-0)-2 \quad \text{para } x \in [1, 2]$$

$$Sp_3(x) = 0.91304(x-2)^3 + 6.2609(x-2)^2 + 11.8261(x-2)+5 \quad \text{para } x \in [2, 3]$$

$$Sp_4(x) = 0.91304(x-3)^3 + 9.0000(x-3)^2 + 27.0870(x-3)+24 \quad \text{para } x \in [3, 4]$$

Podemos obter uma aproximação de  $f(x=2.5)$  via  $Sp_3(x=2.5)=12.592$ .

Para efeito de comparação, temos que a  $f(x)$  discretizada corresponde a  $f(x) = x^3 - 3$ , cujo valor exato de  $f(x=2.5)=12.625$ .

Assim,

- a) com *spline* de extremos lineares, temos  $Sp_3(2.5)=12.330$ ;
- b) com *spline* de extremos quadráticos, temos  $Sp_3(2.5)=12.550$ ;
- c) com *spline* de curvatura de extremos definidos, temos  $Sp_3(2.5)=12.625$  (atribuídos os extremos exatos);
- d) com *spline* de curvatura extrapolada nos extremos, temos  $Sp_3(2.5)=12.592$ ; e
- e) valor exato  $f(x=2.5)=12.625$ .

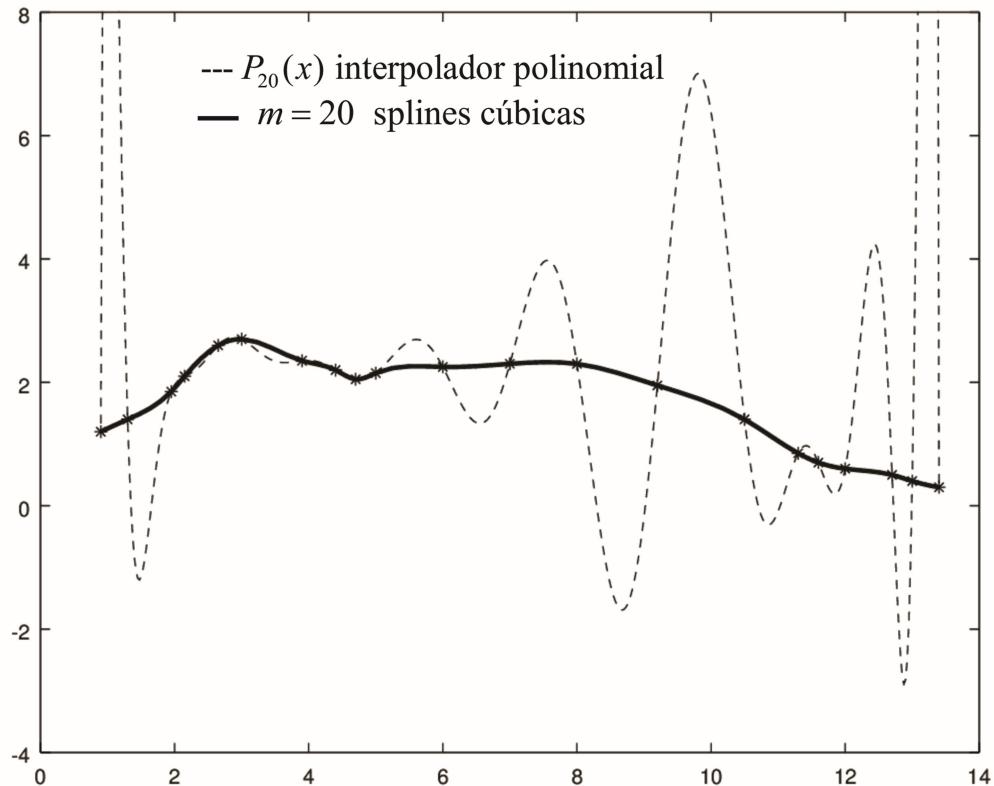
Logo, para esse exemplo com curvatura conhecida e exata nos extremos, a *spline* aproximadora e a função aproximanda  $f(x)$  são iguais. Genericamente, sempre que temos disponível os extremos S, é recomendável que eles sejam usados, conforme a eq. (27).

**DESTAQUE** Considerando um caso geral, quando as curvaturas nas extremidades não são conhecidas previamente, a interpolação com *spline* de curvatura extrapolada nos extremos gera um resultado mais realístico, com menor erro de truncamento. **FIM DO DESTAQUE**

O algoritmo de interpolação por *splines* cúbicas para os quatro tipos de extremidades – linear, quadrático, de extremos definidos e extremos extrapolados – está disponível no **Caderno de Algoritmos** no arquivo **Cap5Graf5.10Splines.seno.m**.

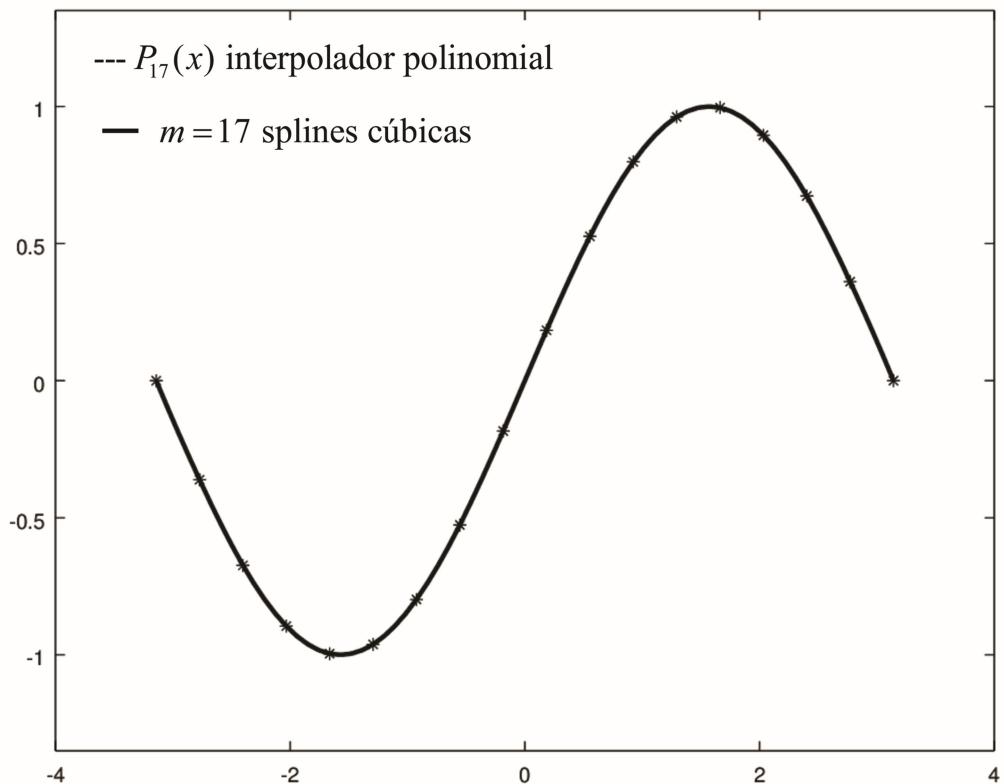
Apresentamos, nos Gráficos 5.9 e 5.10, dois exemplos de aplicação de *splines*: a representação do contorno superior do pato voando do Gráfico 5.7b e a da função  $\sin(x)$ , respectivamente.

Gráfico 5.9 – Curva do Gráfico 5.7 representado por meio de  $m = 20$  *splines* cúbicas com extremos quadráticos, em linha contínua, e interpolador  $P_{20}(x)$ , em linha tracejada



Fonte: Elaboração própria

Gráfico 5.10 – Função  $f(x) = \sin(x)$  em  $x \in [-\pi, +\pi]$  representada por meio de  $m = 17$  *splines* cúbicas com extremos quadráticos, em linha contínua, e  $P_{17}(x)$ , em linha tracejada.



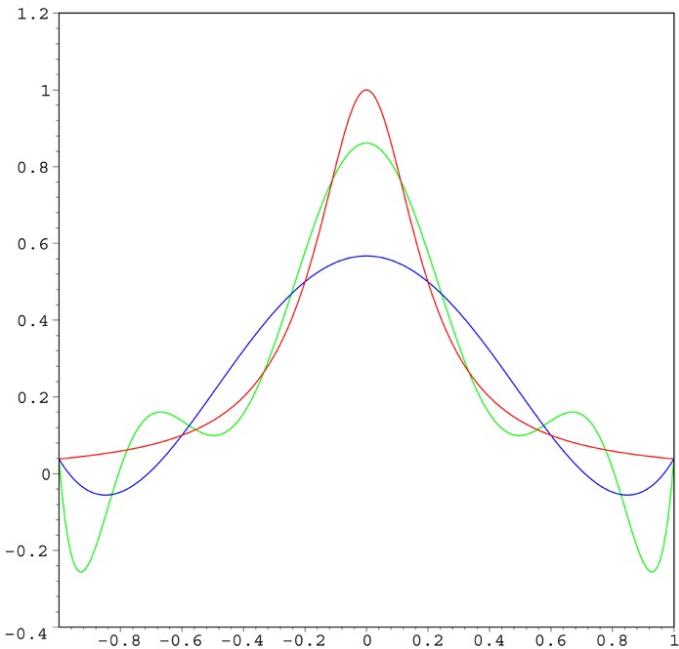
Fonte: Elaboração própria

Observe que as *splines* cúbicas formaram representações suaves nos Gráficos 5.9 e 5.10, gerando curvas com derivadas e curvaturas suaves. No Gráfico 5.10, a interpolação polinomial também gerou resultados suaves (linha tracejada, não visível), mas, no Gráfico 5.9, a interpolação gerou sinuosidades muito acentuadas, enquanto as *splines* geraram curvas suaves.

#### 5.4 Tratamento do Fenômeno de Runge

No campo específico da análise numérica, o **fenômeno de Runge** é um problema de oscilação em funções interpoladoras polinomiais que ocorre quando usamos interpolação com polinômios de ordem elevada, como mostra o Gráfico 5.11. Carl Runge descobriu esse fenômeno quando investigava erros na interpolação polinomial para aproximar certas funções, como a função assintótica  $f(x) = 1 / (25 * x^2 + 1)$ .

Gráfico 5.11 – Representação do fenômeno de Runge



Fonte: Fenómeno de Runge (2013)

Observe em  $x=0$  que a curva mais alta é a função exata  $f(x) = 1/(25x^2 + 1)$ ; a do meio é uma interpolação polinomial de 9º grau, com 10 pontos de interpolação igualmente espaçados; e a mais baixa é uma interpolação polinomial de 5º grau, com 6 pontos de interpolação igualmente espaçados.

Quando usamos  $m = n+1$  “nós” equidistantes na interpolação polinomial de grau  $n$ , o erro máximo aumenta quando a ordem do polinômio interpolador aumenta, em princípio, contrariando o teorema de Weirstrass, ou seja, as sinuosidades aumentam com o grau  $n$  do interpolador, como demonstram os Gráficos 5.11 e 5.12. Podemos amenizar essas sinuosidades usando  $n$  *splines* cúbicas ou interpolação polinomial de grau  $n$  sobre  $m = n+1$  “nós” dos polinômios Tchebyshev de grau  $m$  LINK No Capítulo 6, apresentaremos as propriedades dos polinômios Tchebyshev e seus “nós” (raízes dos polinômios de Tchebyshev), que, mostraremos, estão mais concentrados nas extremidades do intervalo, de modo que o polinômio interpolador fica mais “amarrado” à  $f(x)$ , gerando menos sinuosidades. FIM DO LINK, em vez de  $m$  “nós” equidistantes, para fixar o polinômio interpolador de grau  $n$  ( $n = m-1$ ).

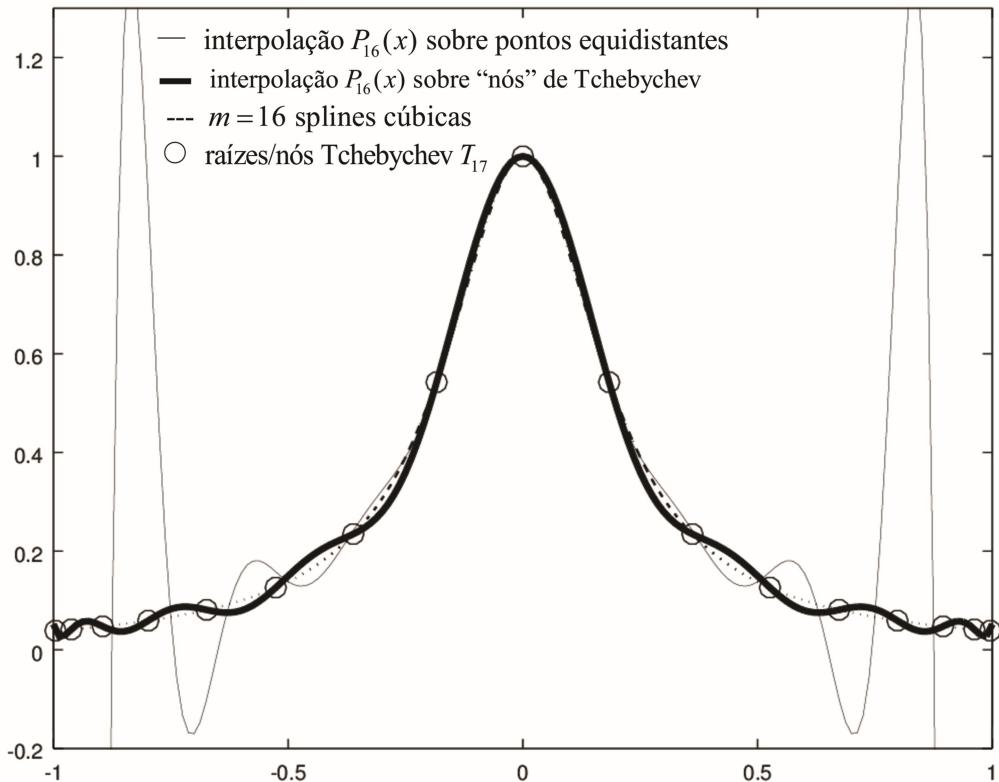
No Exemplo 5.12, vamos aplicar *splines* cúbicas e interpolações polinomiais à função  $f(x) = 1/(25x^2 + 1)$  visando reduzir os erros das aproximações.

**Exemplo 5.12:** aproxime a função  $f(x) = 1/(25*x^2 + 1)$  por meio da interpolação polinomial  $P_{16}(x)$  com  $n+1=17$  “nós” (pontos) equidistantes no intervalo normalizado  $[-1, +1]$ , por meio de interpolação polinomial  $P_{16}(x)$  com  $m=17$  pontos definidos pelos “nós” (raízes) dos polinômios de Tchebyschev  $T_{17}$  de grau  $m=17$  e por meio de  $n=16$  *splines* cúbicas. Apresente os resultados em um gráfico e calcule os erros máximos em cada caso.

**Solução:**

Obtida via algoritmo Cap5Graf5.12FenomenoRunge.m disponível no Caderno de Algoritmos:

Gráfico 5.12 – Representação das aproximações do **Exemplo 5.12**



Fonte: Elaboração própria

Observe que a interpolação polinomial com pontos  $x$  definidos pelos “nós” do polinômio de Tchebyschev reduz gradativamente as sinuosidades com o aumento do grau  $n$  (linha contínua espessa), mas as *splines* cúbicas conseguem a melhor representação (linha tracejada), que fica realmente muito próxima da função exata (linha não apresentada no gráfico), pois apresenta o menor erro de truncamento

máximo:

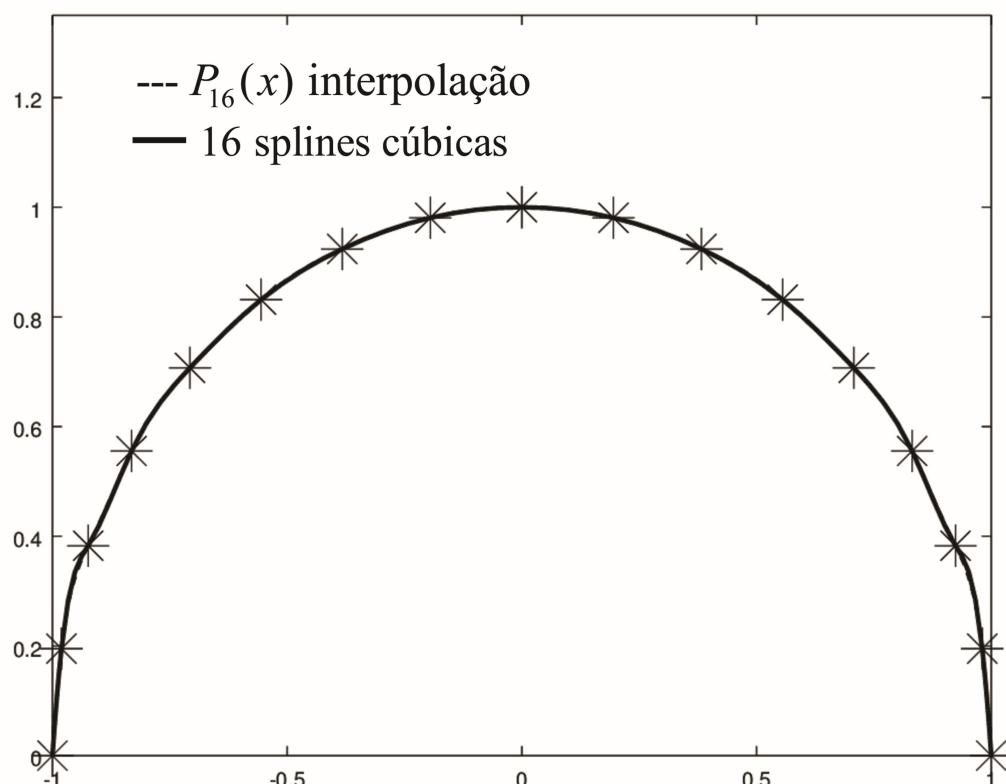
Erro máximo do polinômio com  $m$  “nós” (pontos) equidistantes é 14.370

Erro máximo do polinômio com  $m$  “nós” (pontos) de Tchebyschev é 0.032603

Erro máximo de  $n = m - 1$  splines é 0.0037367

As splines são essencialmente funções polinomiais (interpoladores), aplicadas em cada subintervalo, então podem apresentar deformações em regiões onde a curva não se comporta como “função”, por exemplo, quando uma curva fica muito vertical, como no semicírculo apresentado no Gráfico 5.13.

Gráfico 5.13 – Spline representando um semicírculo



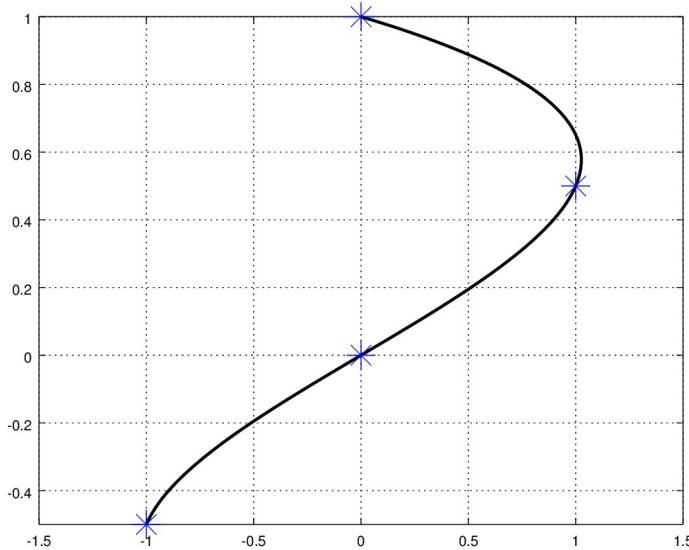
Fonte: Elaboração própria

Nesses casos, as curvas propostas por Pierre Bézier são mais adequadas, porque uma função  $y = f(x)$ , ou mesmo uma relação não funcional, é separada em duas funções, de modo parametrizado, com  $x = x(t)$  e  $y = y(t)$ , em que  $t$  é um parâmetro independente. Assim,  $x = x(t)$  e  $y = y(t)$  sempre serão funções bem comportadas.

## 5.5 Aproximação de Não Funções Via Parametrização

As técnicas de aproximação que abordamos até este momento possibilitam o traçado de caminhos que sejam funções, mas se o caminho, ou a curva, for não funcional, como o esboçado no Gráfico 5.14, precisaremos parametrizá-lo como duas funções.

Gráfico 5.14 – Curva não funcional



Fonte: Elaboração própria

Para aproximar um caminho como função, efetuamos estes passos:

**Primeiro passo:** escolhemos  $m+1$  pontos de referência do caminho

$x_i$	$x_1$	$x_2$	$\cdots$	$x_{m+1}$
$y_i$	$y_1$	$y_2$	$\cdots$	$y_{m+1}$

**Segundo passo:** adotamos um novo parâmetro independente  $t \in [0, 1]$ , tomamos  $m+1$  valores ordenados  $t_i$ , em que  $t_{i+1} = t_i + h$ ,  $h = (1-0)/m$ , com  $i=1,2,\dots,m+1$ , e geramos dois conjuntos de pontos, representando-os como duas funções discretizadas:

$t_i$	$t_1$	$t_2$	$\cdots$	$t_{m+1}$
$x_i$	$x_1$	$x_2$	$\cdots$	$x_{m+1}$

$t_i$	$t_1$	$t_2$	$\cdots$	$t_{m+1}$
$y_i$	$y_1$	$y_2$	$\cdots$	$y_{m+1}$

No Gráfico 5.14, tomando a relação não funcional:

$x_i$	0	1	0	-1
$y_i$	1	1/2	0	-1/2

E parametrizando-a, resulta em:

$t_i$	0	1/3	2/3	3/3
$x(t_i)$	0	1	0	-1

$t_i$	0	1/3	2/3	3/3
$y(t_i)$	1	1/2	0	-1/2

Note que essas duas novas tabelas de pontos anteriores agora são funções com variável independente  $t$ , isto é, cada par  $(x, y)$  do caminho é reescrito como função de  $t$ :  $(x, y) = (x(t), y(t))$ .

**Terceiro passo:** obtemos, pela técnica mais adequada ao problema, as duas aproximadoras de  $x(t)$  e  $y(t)$ . Por exemplo, aplicando a interpolação de Gregory-Newton nos dois conjuntos de pontos anteriores, resultam estes dois interpoladores de grau  $n = 3$ :

$$x(t) \cong PX_3(t) = 0 + 3(t - 0) - 3(t - 0)(3t - 1) + (t - 0)(3t - 1)(3t - 2)$$

$$y(t) \cong PY_3(t) = 1 - (3/2)(t - 0)$$

**Quarto passo:** plotamos os pontos de interesse  $p_k = (PX_n(t_k), PY_n(t_k))$ ,  $\forall t_k \in [0, 1]$  para gerar o caminho desejado. No Gráfico 5.14, usando os interpoladores do passo anterior, temos, por exemplo, para  $t_k = 1/2$ , o ponto  $p_k = (0.625, 0.25)$ .

**DESTAQUE** Todos os algoritmos das técnicas de aproximação de funções por interpolação polinomial e por *splines* cúbicas podem ser estendidos para

aproximação de não funções via parametrização. FIM DO DESTAQUE

## 5.6 Aproximação por Curvas de Bézier

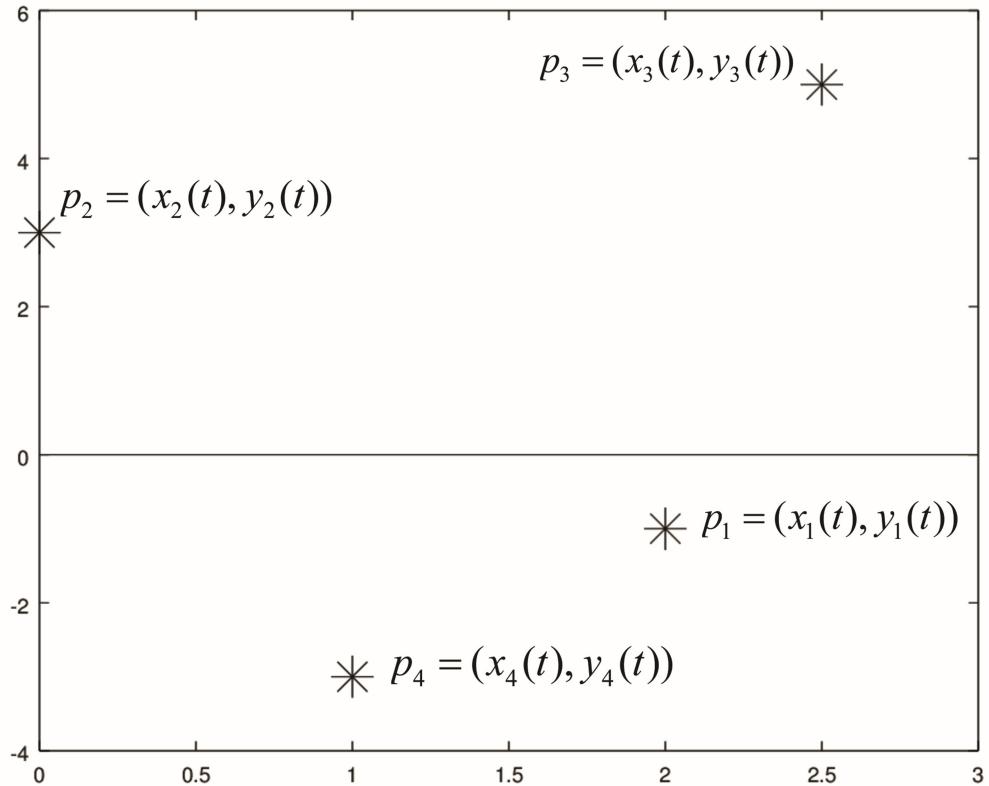
A curva de Bézier é a base matemática de sistemas de computação gráfica como o CorelDRAW® (CURVA DE BÉZIER, 2016). Foi desenvolvida pelo engenheiro Pierre Bézier, na década de 1960, então funcionário da Renault, para simulação de *layout* de automóveis.

A aproximação por curvas de Bézier foi concebida e algebrizada para representar (aproximar) caminhos que não sejam funções, não sendo necessária a parametrização prévia como nos casos de interpolação e de *splines*, pois já é uma aproximação parametrizada na sua origem.

Para seguir um caminho como o do Gráfico 5.15 usando a aproximação de Bézier, devemos efetuar os passos a seguir.

**Primeiro passo:** tomamos  $m+1$  pontos referenciais do desenho desejado  $\Rightarrow p_i = (x_i, y_i), i=1, \dots, m+1$  e os expressamos na forma parametrizada  $\Rightarrow p_i = (x_i(t), y_i(t))$ , em que  $t \in [0, 1]$ , conforme o Gráfico 5.15:

Gráfico 5.15 – Pontos na forma parametrizada



Fonte: Elaboração própria

**Segundo passo:** subdividimos os  $m+1$  pontos referenciais em subconjuntos com  $(k+1)$  pontos,  $2 \leq k < m+1$  (se necessário, nos casos de termos mais do que quatro pontos). Cada um desses subconjuntos irá representar determinado comportamento do caminho.

**Terceiro passo:** para cada subconjunto de pontos  $p_i = (x_i, y_i)$ ,  $i=1, \dots, k+1$ , obtemos o seu polinômio aproximador de grau  $k$  expresso na forma de Bernstein:

$$B_k(t) = \sum_{i=0}^k C_k^i (1-t)^{k-i} t^i p_{i+1} \Rightarrow \begin{cases} BX_k(t) = \sum_{i=0}^k C_k^i (1-t)^{k-i} t^i x_{i+1} \\ BY_k(t) = \sum_{i=0}^k C_k^i (1-t)^{k-i} t^i y_{i+1} \end{cases} \quad (29a)$$

Em que  $C_k^i = \frac{k!}{(k-i)!i!}$  são combinações e a eq. (29a) são os polinômios resultantes.

Expressando a eq. (29a) na sua forma matricial equivalente, temos:

$$B_k(t) = \begin{bmatrix} C_k^0 p_1 & C_k^1 p_2 & \dots & C_k^k p_{k+1} \end{bmatrix} \begin{bmatrix} (1-t)^k \\ (1-t)^{k-1} t^1 \\ \vdots \\ t^k \end{bmatrix} = \begin{cases} BX_k(t) = \begin{bmatrix} C_k^0 x_1 & C_k^1 x_2 & \dots & C_k^k x_{k+1} \end{bmatrix} \begin{bmatrix} (1-t)^k \\ (1-t)^{k-1} t^1 \\ \vdots \\ t^k \end{bmatrix} \\ BY_k(t) = \begin{bmatrix} C_k^0 y_1 & C_k^1 y_2 & \dots & C_k^k y_{k+1} \end{bmatrix} \begin{bmatrix} (1-t)^k \\ (1-t)^{k-1} t^1 \\ \vdots \\ t^k \end{bmatrix} \end{cases} \quad (29b)$$

De acordo com o grau do polinômio de Bernstein, podemos gerar vários tipos de curvas de Bézier:

$k = 2 \Rightarrow$  **Bézier quadrática (3 pontos referenciais):**

$$B_2(t) = 1(1-t)^{(2-0)} t^0 p_1 + 2(1-t)^{(2-1)} t^1 p_2 + 1(1-t)^{(2-2)} t^2 p_3 \quad (30)$$

que é equivalente a

$$\begin{cases} BX_2(t) = 1(1-t)^{(2-0)} t^0 x_1 + 2(1-t)^{(2-1)} t^1 x_2 + 1(1-t)^{(2-2)} t^2 x_3 \\ BY_2(t) = 1(1-t)^{(2-0)} t^0 y_1 + 2(1-t)^{(2-1)} t^1 y_2 + 1(1-t)^{(2-2)} t^2 y_3 \end{cases}$$

Desenvolvendo para aplicação computacional, temos:

$$\begin{cases} xx(t) = x_1 + t(bx + t * ax) \\ yy(t) = y_1 + t(by + t * ay) \end{cases} \Rightarrow \forall t \in [0, 1] \quad (31)$$

Em que

$$\begin{cases} bx = 2(x_2 - x_1) \\ ax = x_1 - 2x_2 + x_3 \end{cases}$$

$$\begin{cases} by = 2(y_2 - y_1) \\ ay = y_1 - 2y_2 + y_3 \end{cases}$$

$k = 3 \Rightarrow$  **Bézier cúbica (4 pontos referenciais):**

$$B_3(t) = 1(1-t)^{(3-0)} t^0 p_1 + 3(1-t)^{(3-1)} t^1 p_2 + 3(1-t)^{(3-2)} t^2 p_3 + 1(1-t)^{(3-3)} t^3 p_4 \quad (32)$$

que é equivalente a

$$\begin{cases} BX_3(t) = 1(1-t)^{(3-0)} t^0 x_1 + 3(1-t)^{(3-1)} t^1 x_2 + 3(1-t)^{(3-2)} t^2 x_3 + 1(1-t)^{(3-3)} t^3 x_4 \\ BY_3(t) = 1(1-t)^{(3-0)} t^0 y_1 + 3(1-t)^{(3-1)} t^1 y_2 + 3(1-t)^{(3-2)} t^2 y_3 + 1(1-t)^{(3-3)} t^3 y_4 \end{cases}$$

Desenvolvendo para aplicação computacional, temos:

$$\begin{cases} xx(t) = x_1 + t(cx + t(bx + t * ax)) \\ yy(t) = y_1 + t(cy + t(by + t * ay)) \end{cases} \Rightarrow \forall t \in [0, 1] \quad (33)$$

Em que

$$\begin{cases} cx = 3(x_2 - x_1) \\ bx = 3(x_3 - x_2) - cx \\ ax = (x_4 - x_1) - (cx + bx) \end{cases}$$

$$\begin{cases} cy = 3(y_2 - y_1) \\ by = 3(y_3 - y_2) - cy \\ ay = (y_4 - y_1) - (cy + by) \end{cases}$$

---

A plotagem de uma polinomial  $B_k(t)$ , com  $t \in [0, 1]$ , gera uma curva com as seguintes **condições de aproximação**:

**Propriedade 1:**

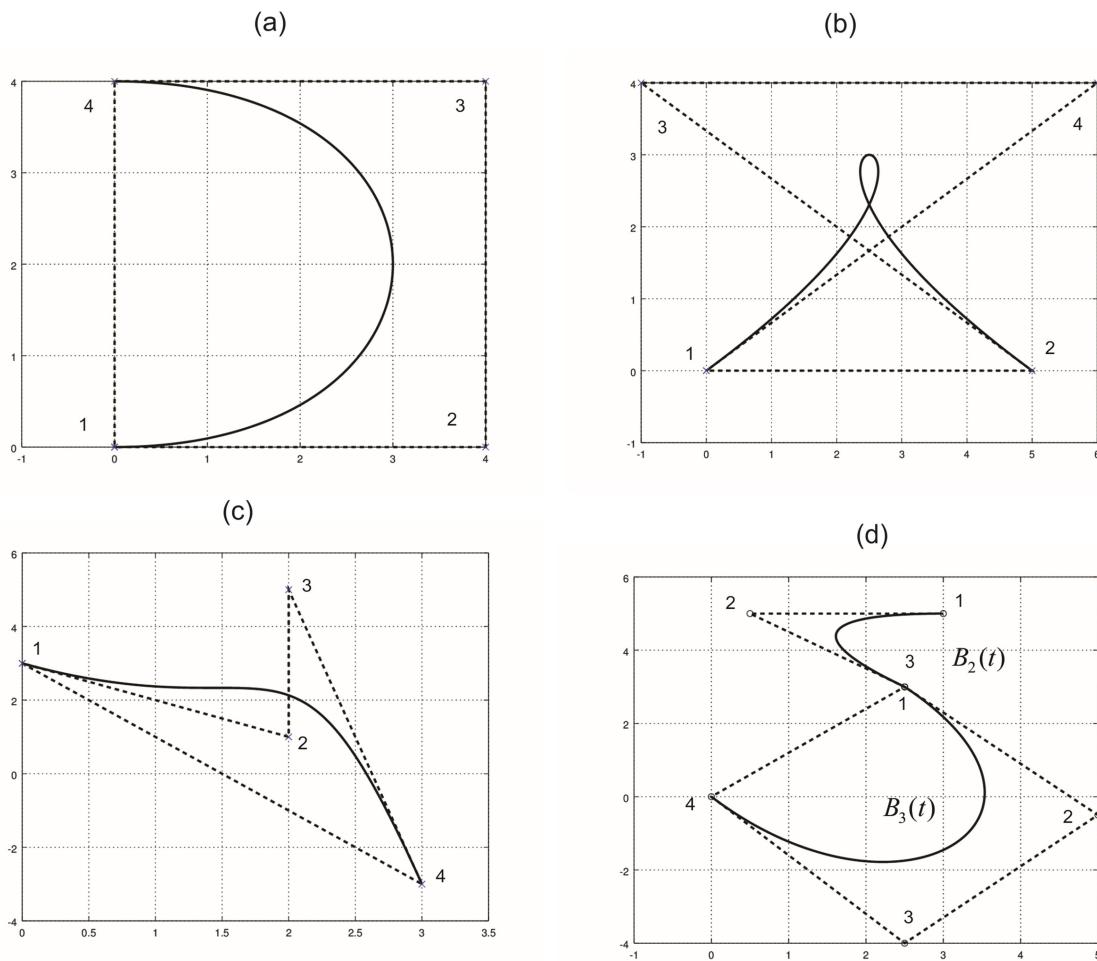
$$\begin{cases} BX_k(0) = x_1 \\ BY_k(0) = y_1 \end{cases} \quad \text{e} \quad \begin{cases} BX_k(1) = x_{k+1} \\ BY_k(1) = y_{k+1} \end{cases} \quad \text{isto é, } B_k(t) \text{ inicia em } p_1 \text{ e termina em } p_{k+1}.$$

**Propriedade 2:** a reta definida pelos pontos  $p_1$  e  $p_2$  é tangente a  $B_k(t)$  em  $p_1$ , e a reta definida por  $p_k$  e  $p_{k+1}$  é tangente a  $B_k(t)$  em  $p_{k+1}$ . Portanto, a curva de Bézier passa somente sobre os pontos  $p_1$  e  $p_{k+1}$  e usa o(s) ponto(s) intermediário(s) como controle para definir as inclinações no início e no final do segmento.

**Propriedade 3:** a curva gerada pela sequência de pontos  $\{BX_k(t), BY_k(t)\}$ ,  $t \in [0, 1]$ , está contida sempre no menor polígono convexo que contém os  $k+1$  pontos de referência da  $B_k(t)$ .

No Gráfico 5.16, apresentamos 3 curvas de Bézier  $B_3(t)$  em (a), (b) e (c) e, no quarto gráfico (d), uma composição de curvas  $B_2(t)$  e  $B_3(t)$ .

Gráfico 5.16 – Sequência de curvas de Bézier



Fonte: Elaboração própria

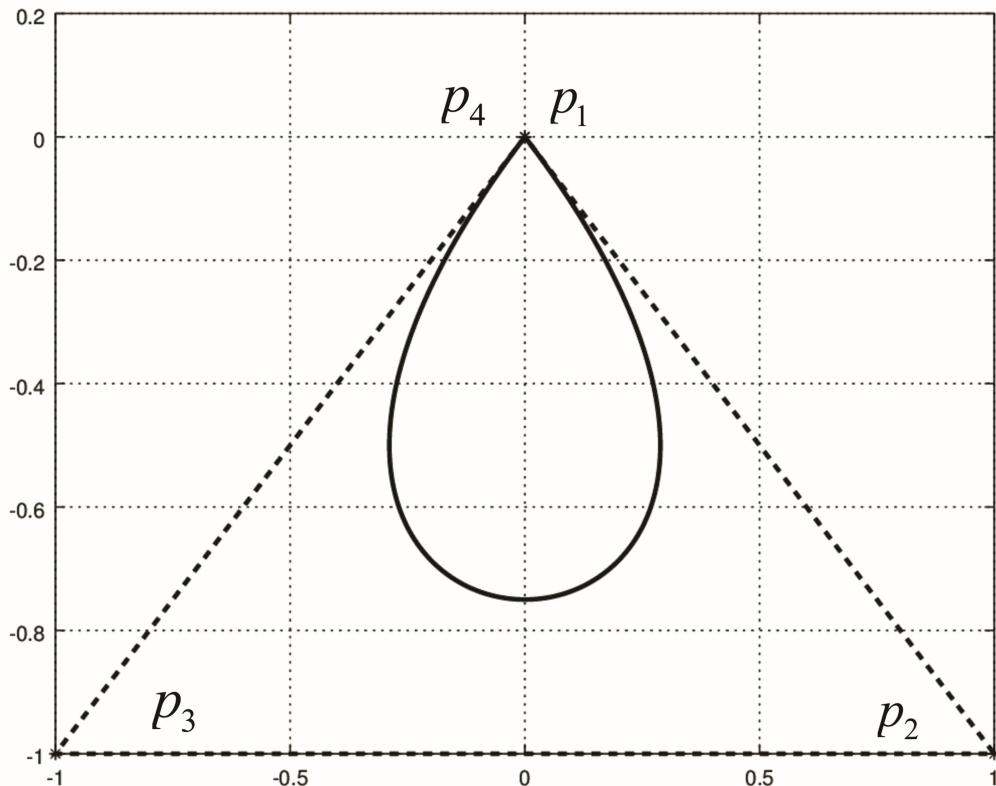
Note que, para traçar o gráfico da curva  $B_k(t)$ , podemos gerar tantos pontos quantos forem necessários e assim suprir a resolução gráfica do desenho desejado. Nesse caso, basta diminuir o passo  $h=1/np$  ( $np$  é o número de passos) com o qual se subdivide o domínio  $[0, 1]$  do parâmetro  $t$ . Já para ajustar a curva gerada por  $B_k(t)$  ao *layout* desejado, mantendo os extremos, basta arrastar o(s) ponto(s) de controle interno(s), o que gerará novas curvas.

Nos sistemas de computação gráfica que utilizam essa técnica de aproximação, a curva de Bézier mais utilizada é a  $B_3(t)$ , repetida tantas vezes quantas forem necessárias para formar um desenho de vários segmentos.

Confira no **Caderno de Algoritmos**, nos arquivos **Cap5Graf5.16.Bezier abc B3.m** e **Cap5Graf5.16.Bezier d B2 e B3.m**, os algoritmos que geram  $np+1=1001$  pontos de curvas  $B_3(t)$  e  $B_2(t)$  do Gráfico 5.16, na sua forma otimizada.

Tomando os pontos referenciais  $p_1 = (0, 0)$ ,  $p_2 = (1, -1)$ ,  $p_3 = (-1, -1)$  e  $p_4 = (0, 0)$ , esse algoritmo gera uma única curva de Bézier, tipo gota, conforme o Gráfico 5.17.

Gráfico 5.17 – Curva de Bézier tipo gota



Fonte: Elaboração própria

No Gráfico 5.18, a seguir, apresentamos um perfil superior de uma asa simples composta de três segmentos de curvas de Bézier que devem passar por quatro pontos de ancoragem, A, B, C e D, e com inclinação pré-definidas, conforme segue:

- A(0, 0) com inclinação de  $45^\circ$ ;
- B(2, 1) com inclinação de  $0^\circ$ ;
- C(8, 0.2) com inclinação de  $-10^\circ$ ;
- D(10, 0) com inclinação de  $-5^\circ$ ;

Verifique que somente conseguimos passar sobre os quatro pontos, A, B, C, D, se usarmos três curvas de Bézier separadas, devendo colocar dois pontos intermediários/auxiliares para cada curva, pois essas curvas passam sobre suas

duas extremidades e, com inclinação pré-definida pelas retas que as ligam aos seus dois pontos intermediários. Assim, sugerimos estes **pontos intermediários** LINK Explore variações desses pontos intermediários no exercício 5.3. FIM DO LINK em cada segmento, na notação do *Octave*:

% parte 1: segmento AB

```
x(1)=0; y(1)=0;  
x(2)=1; y(2)=1; %inclinação início = 45º entre 1 e 2  
x(3)=1.9; y(3)=1; %inclinação final = 0º entre 3 e 4  
x(4)=2; y(4)=1;
```

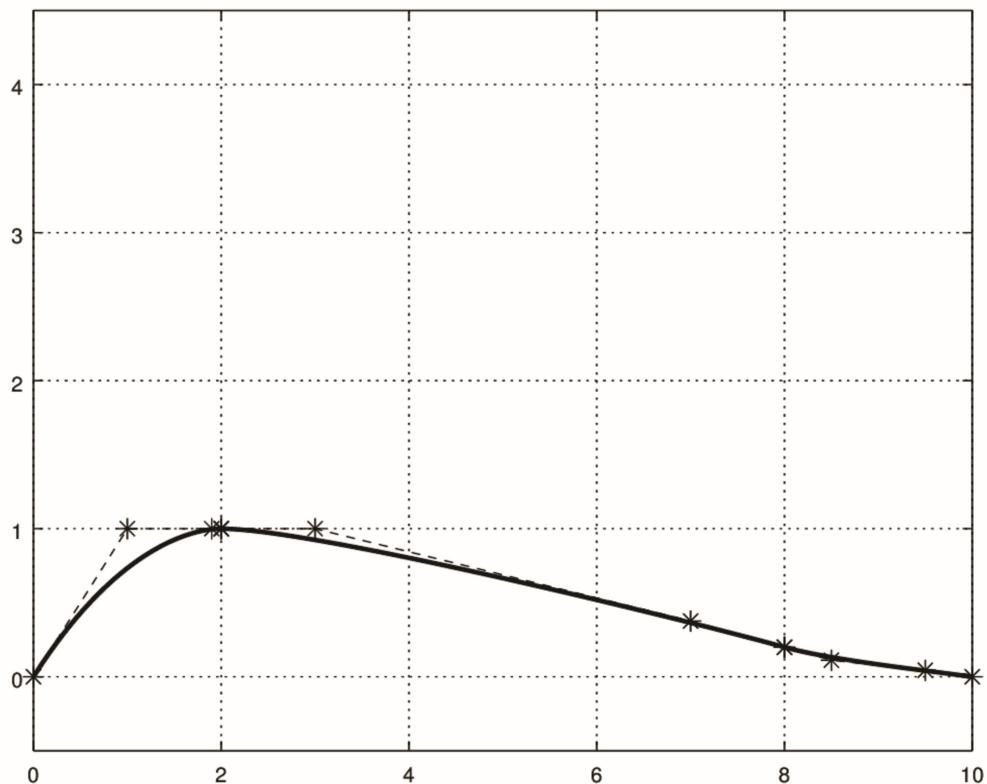
% parte 2: segmento BC

```
x(1)=2; y(1)=1;  
x(2)=3; y(2)=1; %inclinação início = 0º  
x(3)=8-1; y(3)=0.2+1*tan(10*pi/180); %inclinação final = 10º  
x(4)=8; y(4)=0.2;
```

% parte 3: segmento CD

```
x(1)=8; y(1)=0.2;  
x(2)=8+0.5; y(2)=0.2-0.5*tan(10*pi/180); %inclinação início = 10º  
x(3)=10-0.5; y(3)=0+0.5*tan(05*pi/180); %inclinação final = 5º  
x(4)=10; y(4)=0;
```

Gráfico 5.18 – **Perfil superior de uma asa**  
LINK No **Caderno de Algoritmos**, você encontra o arquivo  
Cap5Graf5.18.Bezier.aerofolio.m.FIM DO LINK



Fonte: Elaboração própria

Podemos encontrar exemplos práticos de aplicação das curvas de Bézier no CorelDRAW®, na ferramenta que converte uma forma qualquer em curvas, permitindo a manipulação de “nós”.

Lembre-se de que está disponível no [link <http://sergiopeters.prof.ufsc.br/exercicios-e-respostas/>](http://sergiopeters.prof.ufsc.br/exercicios-e-respostas/) o **Caderno de Exercícios e Respostas** para o aprofundamento dos estudos de cada capítulo deste livro.

## 5.7 Conclusões

Neste Capítulo, iniciamos o estudo da teoria da aproximação de funções abordando três das suas várias metodologias. A primeira metodologia, a interpolação polinomial, que tem como condição de aproximação um princípio simples e óbvio, possui aplicações genéricas. Já a segunda metodologia, a interpolação via *splines* cúbicas, e a terceira, as curvas de Bézier, possuem como fundamento princípios matemáticos mais sofisticados e são a base teórica de vários sistemas de computação gráfica. Nos próximos dois capítulos, o tema terá

continuidade com o estudo de outras metodologias de aproximação de funções que fundamentam matemática e computacionalmente outros tipos de aplicações.