Introdução à Computação em Física

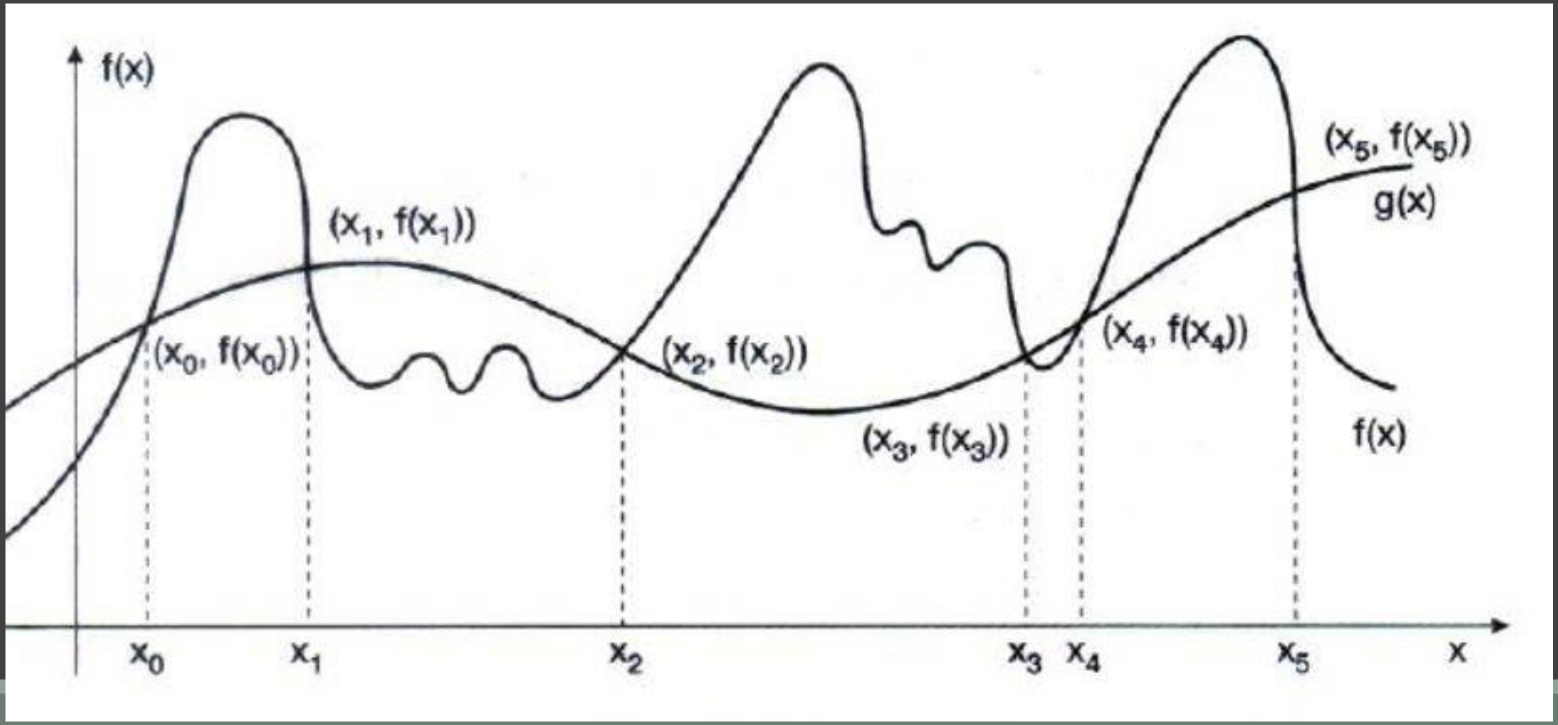
INTERPOLAÇÃO (PARTE 2)

PROF. WALBER

Refs.:
Cálculo numérico, aspectos teóricos e computacionais (2nd edição), M. A. G. Ruggiero, V. L. da Rocha Lopes
Numerical Python, Second Edition, Robert Johansson

Aula anterior: Interpolação polinomial

A interpolação consiste em "substituir" uma função f(x), da qual se tem um conjunto de pontos, por uma outra função g(x), com o objetivo de calcular o valor da função em um ponto não tabelado, além de facilitar certas operações (como diferenciação e integração).



Interpolação - métodos vistos

Vimos diferentes métodos para a obtenção do polinômio interpolador:

1. Interpolação polinomial:

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

2. Forma de Lagrange:

$$p_n(x) = y_0 L_0(x) + y_1 L_1(x) + \cdots + y_n L_n(x)$$

3. Forma de Newton:

$$p_n(x) = f(x_0) + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + \cdots + (x - x_0)(x - x_1) \ldots (x - x_{n-1}) f[x_0, x_1, \ldots, x_n]$$

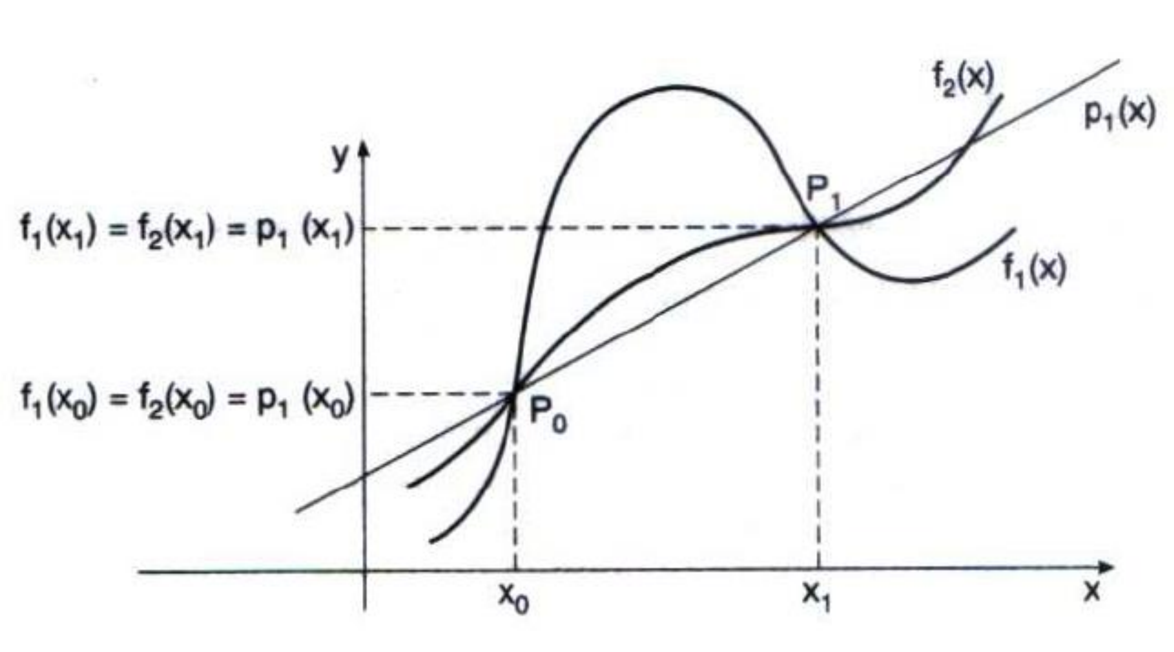
Erro na interpolação

Ao se aproximar uma função f(x) por um polinômio interpolador, tem-se o erro

$$E_n(x) = f(x) - p_n(x)$$

para todo x no intervalo $[x_0, x_n]$

Graficamente:



p_1 interpola $f_1(x)$ e $f_2(x)$,
O erro associado a f_2 e p_1 é maior.

Erro na interpolação

Podemos obter o erro através do seguinte teorema:

Sejam $x_0 < x_1 < \cdots < x_n$ (n+1) pontos. Seja $f(x)$ com derivadas até ordem $(n+1)$ para todo x pertencente ao intervalo $[x_0, x_n]$. Seja $p_n(x)$ o polinômio interpolador de $f(x)$ nos pontos $x_0, x_1, \ldots, x_n$. Então, em qualquer ponto x pertencente ao intervalo $[x_0, x_n]$, o erro é dado por

$$E_n(x) = f(x) - p_n(x) = (x - x_0)(x - x_1)(x - x_2)\ldots(x - x_n)\frac{f^{(n+1)}(\xi_x)}{(n+1)!},$$

onde $\xi_x \in (x_0, x_n)$.

Equação acima possui limitação prática, uma vez que nem sempre sabemos a expressão de f(x), suas derivadas e o ponto onde é calculada as derivada de ordem (n+1).

Erro na interpolação

Adicionalmente:

Se $f^{(n+1)}(x)$ for contínua em $[x_0, x_n]$, podemos escrever:

$$|E_n(x)| = |f(x) - p_n(x)| \leq |(x - x_0)(x - x_1)(x - x_2)\dots(x - x_n)|\frac{M_{(n+1)}}{(n+1)!},$$

onde $M_{n+1} = max|f^{(n+1)}(x)|$.

Caso a derivada não for conhecida, pode-se estimar o erro através da máxima diferença dividida:

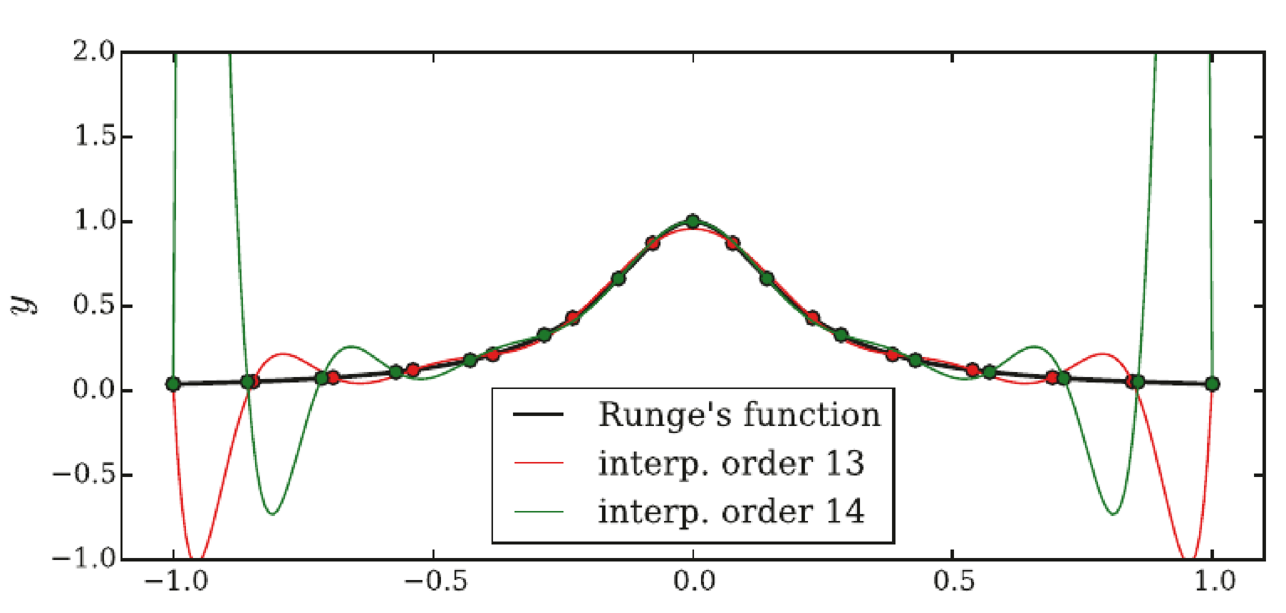
$$|E_n(x)| \approx |(x - x_0)(x - x_1)(x - x_2)\dots(x - x_n)|Md,$$

onde Md é o modulo da máxima diferença dividida de ordem n+1.

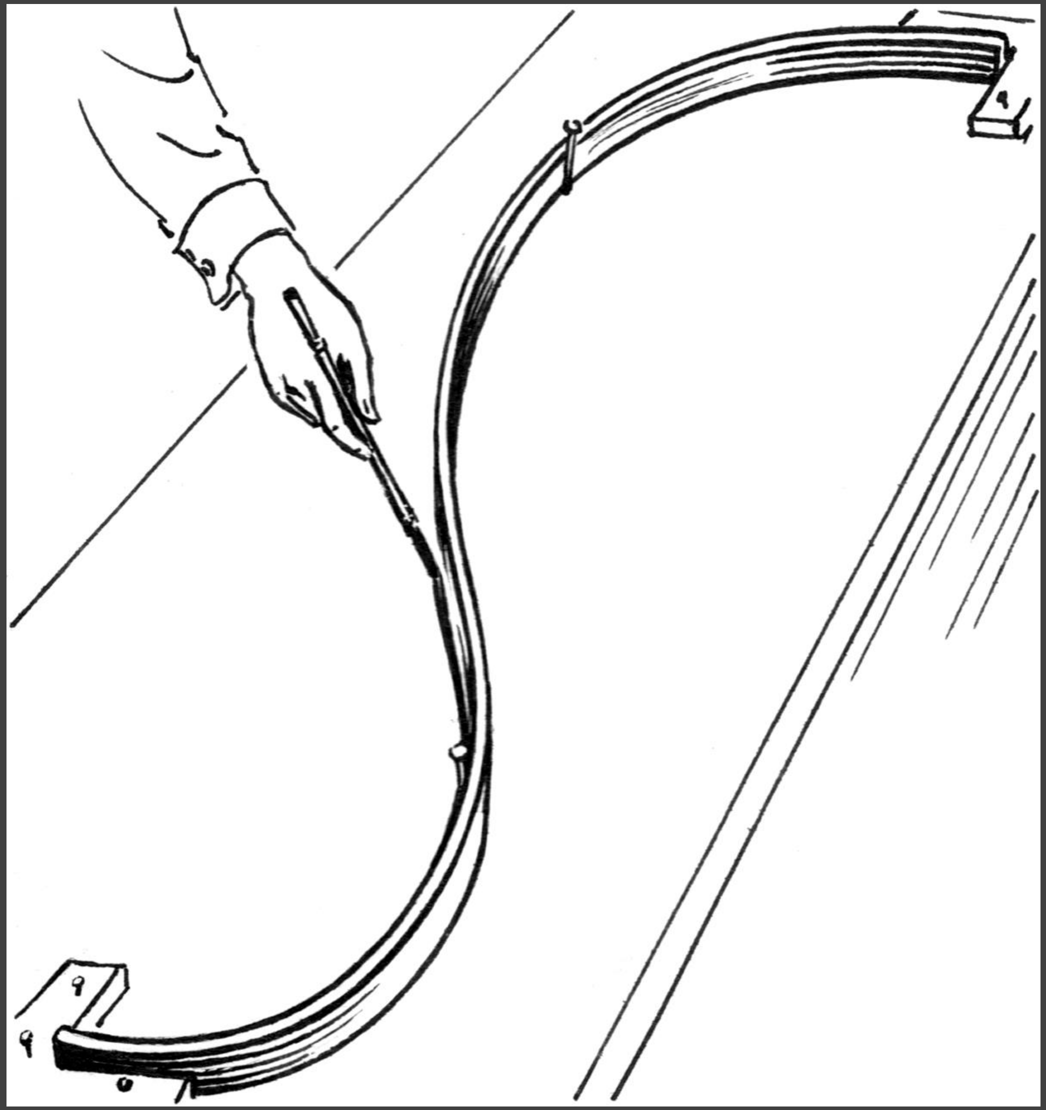
Sobre a ordem do polinômio

Um ponto importante sobre interpolação polinomial é a convergência do polinômio $p_n(x)$ para f(x) , no limite de termos n tendendo a ∞ (aumento consideravel no número de pontos no intervalo $[x_0, x_n]$.

Nesse sentido, polinômios de ordem superior podem introduzir oscilações indesejadas, chamado de fenômeno de Runge



Interpolação via funções Spline



Funções spline:

Iremos empregar uma abordagem alternativa para a interpolação dos pontos $x_0,x_1,\ldots, x_n$. A ideia é dividir o intervalo $[x_0,x_n]$ em uma coleção de subintervalos e obter um polinômio interpolador distinto para cada subintervalo. Esses diferentes polinômios são chamados de splines.

Função spline

Definição:

Sejam $a = x_0 < x_1 < x_2 < \cdots < x_n = b$ uma subdivisão do intervalo $[a, b]$. Uma função Spline de grau p com nós nos pontos $x_i, i = 0, 1, \ldots, n$ é uma função $S_p(x)$ com as propriedades:

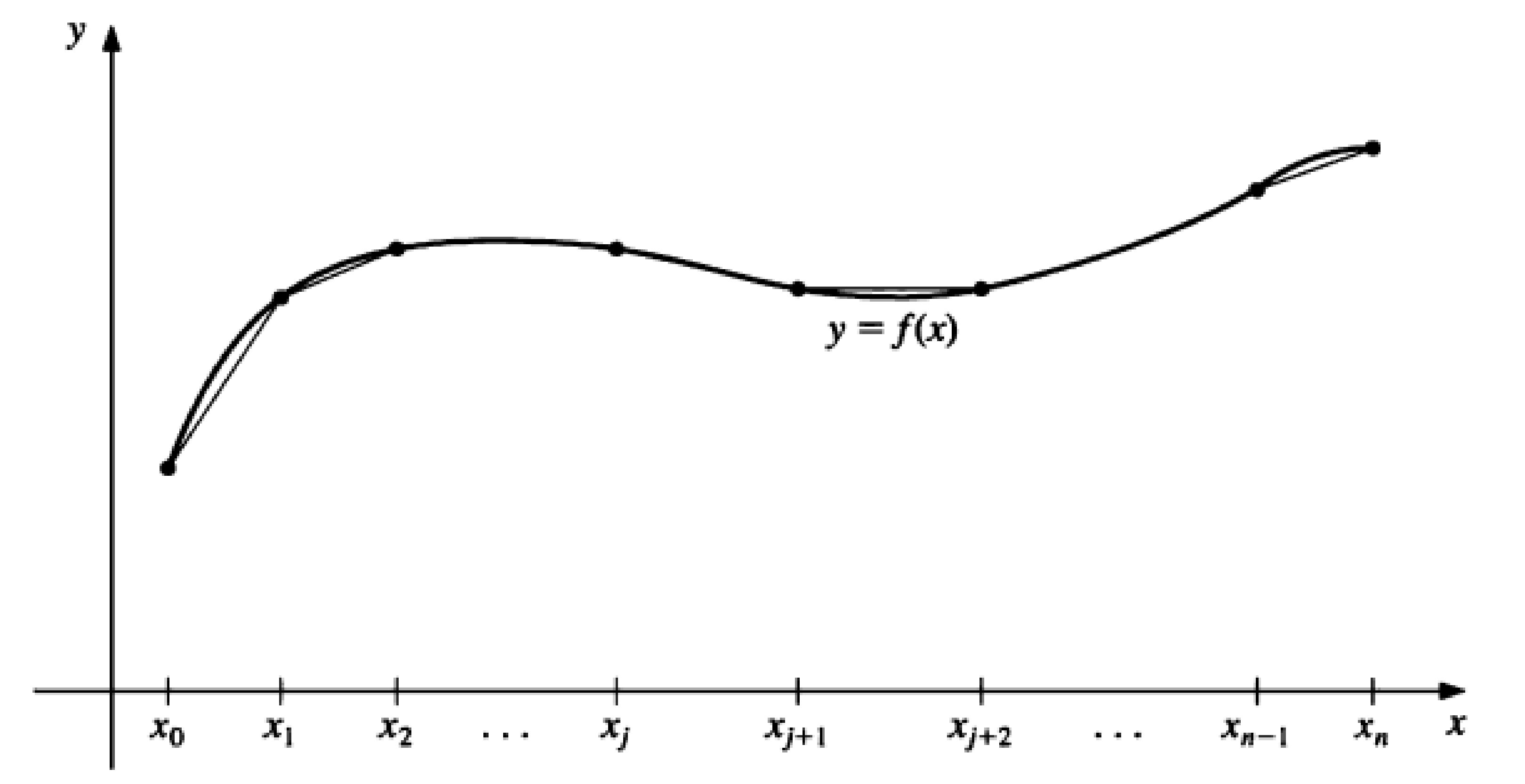
1. em cada subintervalo $[x_i, x_{i+1}], i = 0, 1, \ldots, n-1$, $S_p(x)$ é um polinômio de grau p.
2. $S_p(x)$ é contínua em $[a, b]$ e tem derivada contínua em $[a, b]$ até ordem $p - 1$.

A Spline interpolante é a função $S_p(x)$ da definição acima tal que $S_p(x_i) = f(x_i)$, i = 0, 1, ..., n.

Diferentes aproximações de acordo com grau das funções Spline, tendo então Spline lineares e cúbicas, por exemplo.

Splines lineares

Retas interligando os pontos.
Desvantagem associada a suavidade da curva (pode conter 'kinks') e sobre continuidade das derivadas.



Splines cúbicos

Uma aproximação bastante comum que iremos estudar consiste na utilização de Splines cúbicos. Teremos 4 constantes que nos dão uma certa flexibilidade, continuidade e uma segunda derivada contínua.

Definição:

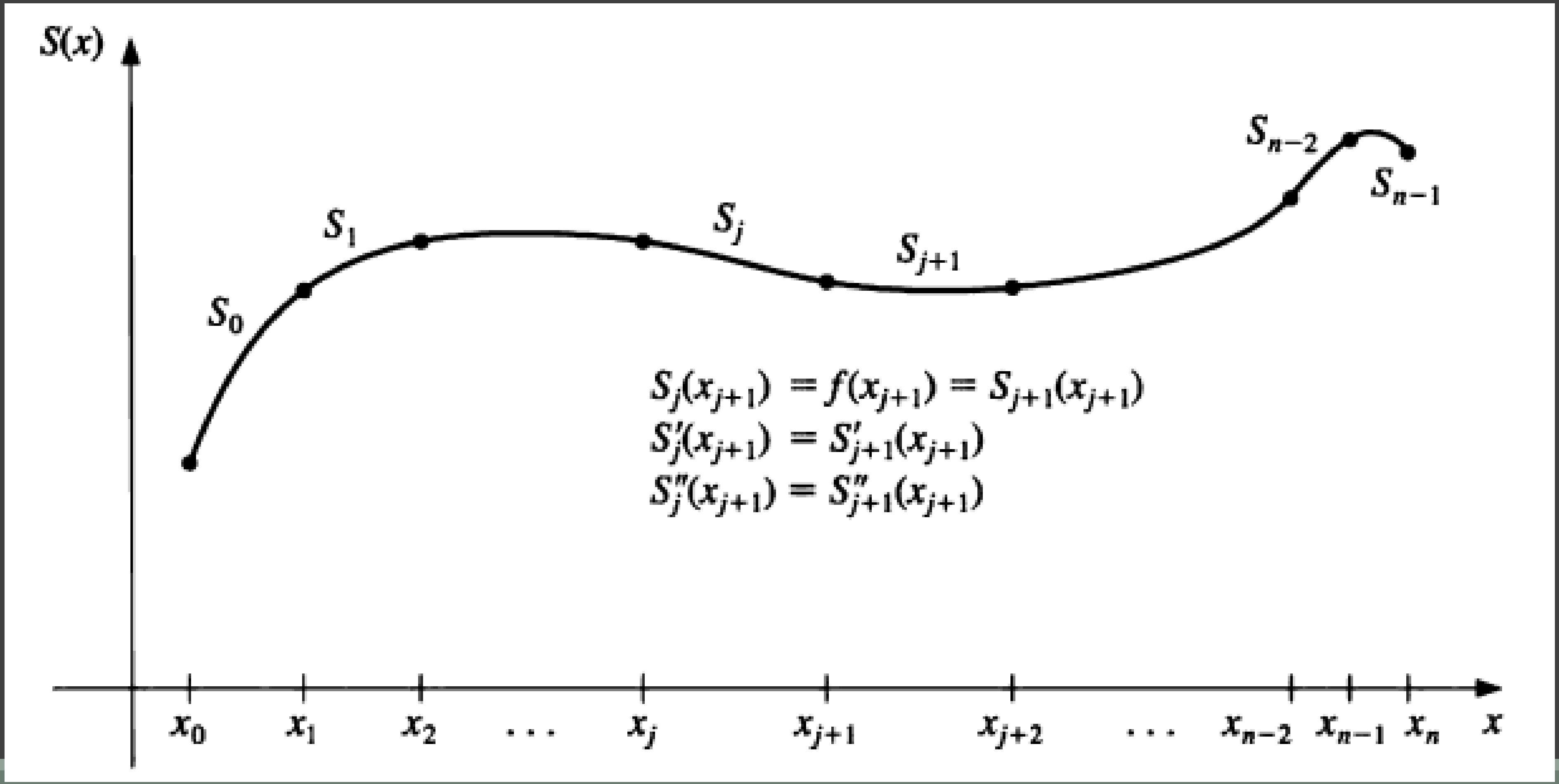
Dada uma função $f(x)$ definida em $[a, b]$ e um conjunto de pontos (nós) $a = x_0 < x_1 < x_2 < \cdots < x_n = b$, um Spline cúbico interpolador S para a função $f(x)$ é uma função que satisfaz as condições:

1. $S(x)$ é um polinômio cúbico, denotado por $S_j(x)$, no subintervalo $[x_j, x_{j+1}]$ para cada $j = 0, 1, \ldots, n-1$;
2. $S_j(x_j) = f(x_j)$ e $S_j(x_{j+1}) = f(x_{j+1})$ para cada $j = 0, 1, \ldots, n-1$;
3. $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ para cada $j = 0, 1, \ldots, n-2$;
4. $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ para cada $j = 0, 1, \ldots, n-2$;
5. $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ para cada $j = 0, 1, \ldots, n-2$;
6. Um dos conjuntos de condições de contorno abaixo é satisfeito:

   $S''(x_0) = S''(x_n) = 0$ (condições de contorno naturais);

   $S'(x_0) = f'(x_0)$ e $S'(x_n) = f'(x_n)$ (condições de contorno fixadas).

Splines cúbicos

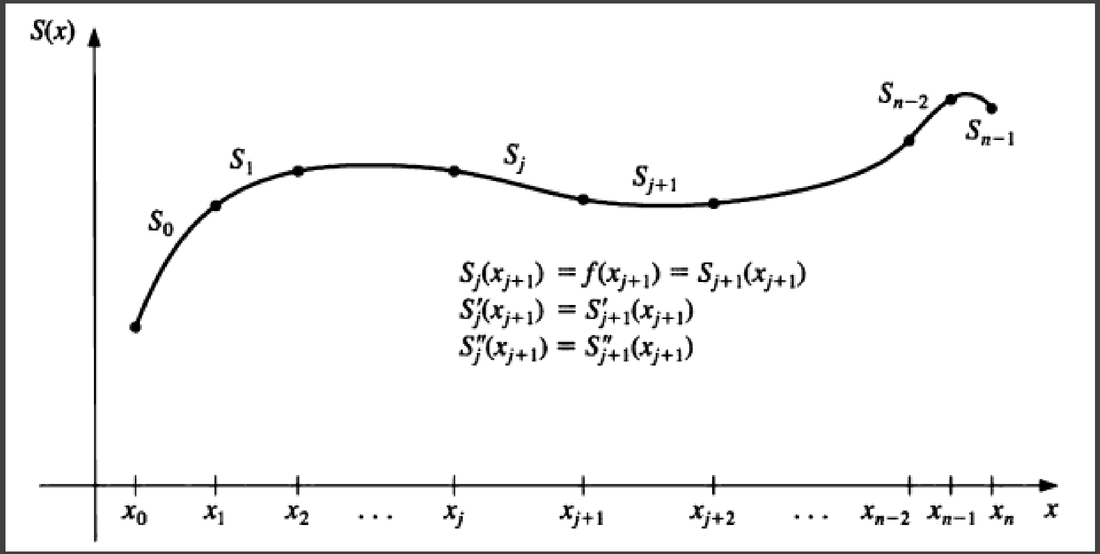


Construção das funções spline

Seja então a spline cúbica:

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

j= 0,1,2, ..., n-1



Queremos determinar os coeficientes a_j, b_j, c_j e d_j.

Construção das funções spline

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

Da condição 2 anterior,

$$\begin{aligned} S_j(x_j) &= f(x_j), \\ a_j &= f(x_j) \end{aligned}$$

temos que agora obter as equações para b_j,c_j e d_j.

Da condição 3,

$$S_{j+1}(x_{j+1}) = S_j(x_{j+1})$$

$$\Downarrow$$

$$a_{j+1} = a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 + d_j(x_{j+1} - x_j)^3$$

Construção das funções spline

Sendo: $h_j = x_{j+1} - x_j$

Temos que: $a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3$ j= 0,1,2, ..., n-1 (I)

Por outro lado, a primeira derivada:

$$S_j'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$$

Disso, vemos que: $S_j'(x_j) = b_j$

Da condição 4: $S_{j+1}'(x_{j+1}) = S_j'(x_{j+1}) \Rightarrow b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2$ j= 0,1,2, ..., n-1 (II)

Construção das funções spline

Definindo:

$$c_n = \frac{S''(x_n)}{2}$$

Da condição 5,

$$S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$$

$$S''_j(x) = 2c_j + 6d_j(x - x_j)$$

$$c_{j+1} = c_j + 3d_j h_j \qquad \text{(III)}$$

Isolando d_j:

$$d_j = \frac{c_{j+1} - c_j}{3h_j} \qquad \text{(III-a)}$$

Construção das funções spline

Substituindo (III-a) nas equações (I) e (II), obtemos que

$$a_{j+1} = a_j + b_j h_j + (2c_j + c_{j+1})\frac{h_j^2}{3} \qquad \text{(IV)}$$

$$b_{j+1} = b_j + h_j(c_j + c_{j+1}) \qquad \text{(V)}$$

Isolando b_j em (IV):

$$\text{(VI)} \quad b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) \quad \Rightarrow \quad b_{j-1} = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j)$$

Construção das funções spline

Utilizando a equação (V), reduzindo em um o índice

$$b_j = b_{j-1} + h_{j-1}(c_{j-1} + c_j)$$

Substituindo as equações de b_j e b_{j-1} encontradas no slide anterior, na equação acima , obtemos

$$\frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j) + h_{j-1}(c_{j-1} + c_j)$$

$$\Downarrow$$

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

j= 1,2, ..., n-1

Construção das funções spline

Como temos os valores de a_j, podemos obter os valores de c_j através do sistema linear gerado pela equação abaixo:

(VII)

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

j= 1,2, ..., n-1

Obtem-se os b_j's e d_j's

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1})$$

$$d_j = \frac{c_{j+1} - c_j}{3h_j}$$

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

Surge a questão da unicidade do spline interpolador

Splines naturais

Seguinte teorema, associado as condições de contorno 6 (contorno naturais) garantem a unicidade do spline

Se $f(x)$ for definida em $a = x_0 < x_1 < \cdots < x_n = b$, então f tem um spline $S(x)$ interpolador natural único nos pontos $x_0, x_1, \ldots, x_n$, isto é, um spline interpolador que satisfaz as condições de contorno $S''(a) = 0$ e $S''(b) = 0$.

Temos então:

$$c_0 = S''(x_0) = 0$$
$$c_n = S''(x_n) = 0$$

Logo, podemos agora reescrever as equações associadas aos coeficientes c_j

Splines naturais - Sistema Linear

Das equações oriundas de (VII), pode-se montar o sistema linear Ac = b:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \dots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \dots & 0 \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdot & \cdot & 0 & 0 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 0 \\ \frac{3}{h_1}(a_2-a_1) - \frac{3}{h_0}(a_1-a_0) \\ \cdot \\ \cdot \\ \cdot \\ \frac{3}{h_{n-1}}(a_n-a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) \\ 0 \end{pmatrix}$$

$$c = \begin{pmatrix} c_0 \\ c_1 \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{pmatrix}$$

Queremos encontrar então os valores c_0, c_1, ..., c_n.

Atividade prática

Implementar os splines cúbicos naturais considerando:

1. Gere 15 pontos igualmente espaçados entre –1 e 1 em x. Obtenha também os valores de $f(x) = 1/(1 + 25\ x^2)$ para esses pontos;

2. Obtenha a matriz A e vetor b. Calcule no final os coeficientes a_j, b_j, c_j e d_j.

3. Obtenha os splines para os pontos gerados em 1.

4. Plot o gráfico dos pontos conjuntamente com a curva interpoladora;

Obtendo funções splines através da SciPy

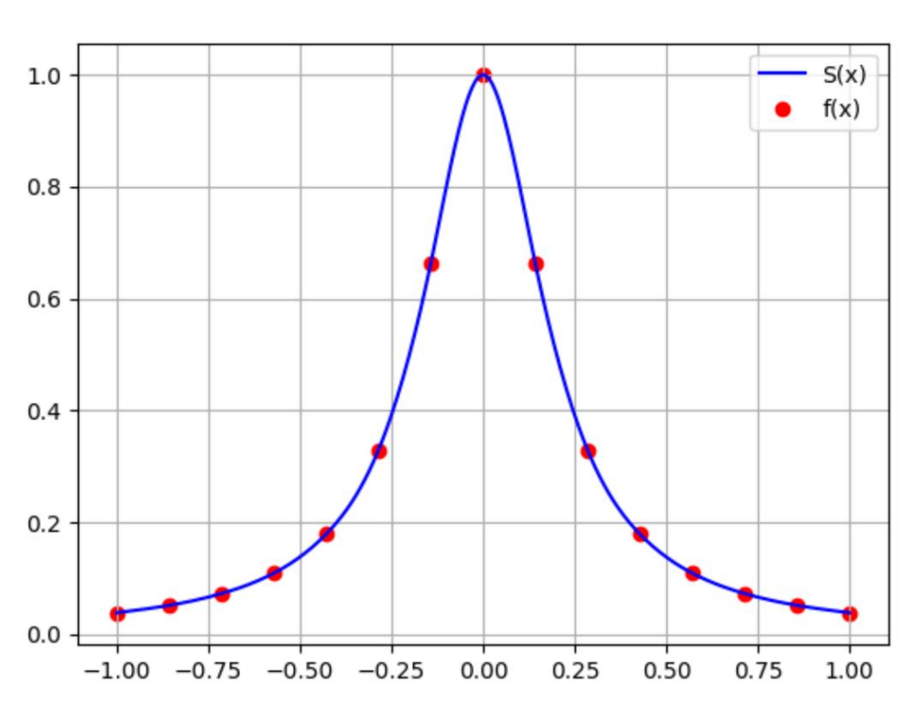
Spline cúbica natural via SciPy

Interpolação via spline cúbica natural pode ser feita através da biblioteca SciPy,

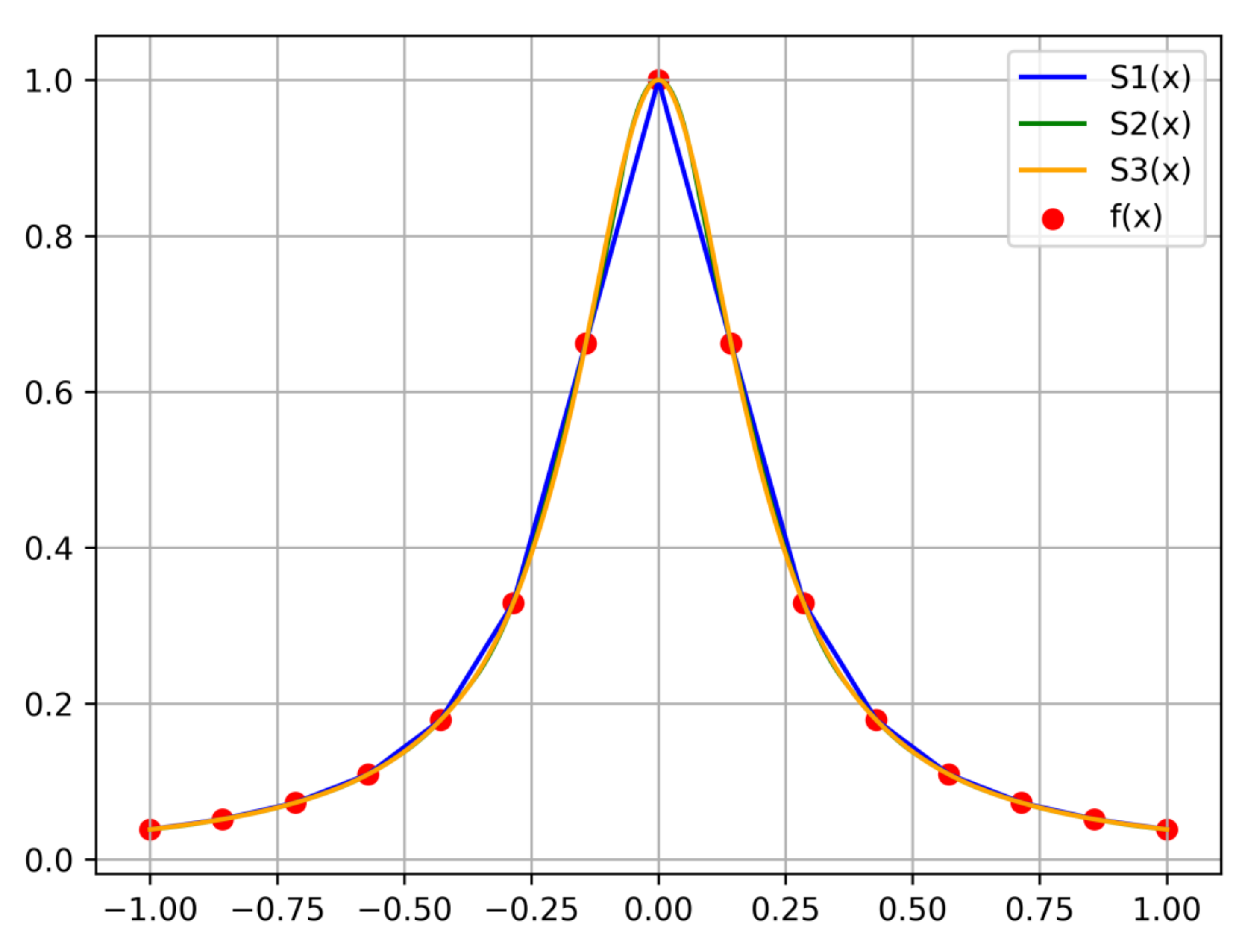
```
from scipy import interpolate

...

spl = interpolate.CubicSpline(x,f,axis=0, bc_type='natural')
```



Mais informações em:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.CubicSpline.html#scipy.interpolate.CubicSpline



Spline cúbica natural via SciPy (controlando o grau do spline)

Interpolação via spline de diferentes graus via SciPy,

```
from scipy import interpolate

...

spl1 = interpolate.interp1d(x,f, kind=1)
spl2 = interpolate.interp1d(x,f, kind=2)
spl3 = interpolate.interp1d(x,f, kind=3)
```