Trabalho Prático 3

Bruno Mota, José Torres e Maria Lourenço

30 de dezembro de 2021

Introdução

O objetivo deste trabalho é construir o polinómio interpolador e o spline cúbico natural em dois casos: no primeiro queremos apenas encontrar o polinómio e o spline que passam num dado conjunto de pontos; no segundo queremos aproximar uma função dada, da qual se conhecem os valores de um dado conjunto de pontos, através do polinómio e do spline que passam nesses pontos. Pretende-se ainda, no último caso, calcular e comparar o valor do erro absoluto do polinómio e do spline obtidos relativamente à função original, bem como estimar o valor dessa função em pontos não conhecidos, usando ambas as aproximações. A linguagem de programação utilizada foi Python. Todos os programas implementados estão em apêndice.

Exercício 1

Polinómio Interpolador

Para a construção do polinómio interpolador, utilizamos o método de Newton em diferenças divididas. Para isso, implementamos um programa em Python que calcula a tabela das diferenças dividas e, a partir desta, define a função do polinómio. Assim, obtivemos a seguinte tabela:

i	$\mathbf{x_i}$	$\mathbf{f_i}$	$\mathbf{f_i}[,]$	$\mathbf{f_i}[,,]$	$\mathbf{f_i}[,,,]$	$\mathbf{f_i}[,,,,]$	$\mathbf{f_i}[,,,,,]$
0	0	1.4	-0.8	0.6	-0.56	0.4533	-0.2022
1	1	0.6	0.4	-0.8	0.8	-0.3556	
2	2	1.0	-0.8	0.8	-0.2667		
3	2.5	0.6	0	0.2667			
4	3	0.6	0.4				
5	4	1.0					

Tabela 1: Método de Newton em diferenças divididas.

e o polinómio interpolador (a simplificação foi feita com auxílio do Maxima):

$$\begin{aligned} p(x) &= 1.4 - 0.8x + 0.6x(x-1) - 0.56x(x-1)(x-2) + 0.4533x(x-1)(x-2)(x-2.5) \\ &\quad - 0.2022x(x-1)(x-2)(x-2.5)(x-3) = \\ &= -0.2022x^5 + 2.1722x^4 - 8.3111x^3 + 13.3611x^2 - 7.8200x + 1.4 \end{aligned}$$

Verificamos que o polinómio passa nos nós:

$\mathbf{x_i}$	0	1	2	2.5	3	4
$\mathbf{f_i}$	1.4	0.6	1.0	0.6	0.6	1.0
$\mathbf{p(x_i)}$	1.4	0.6	1.0	0.60000	0.60000	1.00000

Tabela 2: Valor do polinómio nos x_i.

Spline Cúbico Natural

Para encontrar o spline cúbico natural, substituímos os valores dos h_i e f_i no seguinte sistema:

$$\frac{h_i}{6}M_{i-1} + \frac{h_i + h_{i+1}}{3}M_i + \frac{h_{i+1}}{6}M_{i+1} = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i}, \quad i = 1, ..., 4 \tag{1}$$

e obtivemos a seguinte equação matricial, que foi resolvida em Python:

$$\begin{bmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 \\ 0 & \frac{1}{6} & \frac{1}{2} & \frac{1}{12} & 0 & 0 \\ 0 & 0 & \frac{1}{12} & \frac{1}{3} & \frac{1}{12} & 0 \\ 0 & 0 & 0 & \frac{1}{12} & \frac{1}{2} & \frac{1}{6} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \end{bmatrix} = \begin{bmatrix} 1.2 \\ -1.2 \\ 0.8 \\ 0.4 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

obtendo-se:

$\mathbf{M_0}$	$\mathbf{M_1}$	$\mathbf{M_2}$	$\mathbf{M_3}$	$\mathbf{M_4}$	$\mathbf{M_5}$
0	2.76846473	-3.87385892	3.30622407	0.24896266	0

Tabela 3: Valores dos M_i.

Assim, tem-se a equação do spline cúbico natural:

$$s(x) = \begin{cases} \frac{M_1}{6}x^3 + 1.4(1-x) + (0.6 - \frac{M_1}{6})x, & 0 \leq x \leq 1 \\ \frac{M_1}{6}(2-x)^3 + \frac{M_2}{6}(x-1)^3 + (0.6 - \frac{M_1}{6})(2-x) + (1.0 - \frac{M_2}{6})(x-1), & 1 \leq x \leq 2 \\ \frac{M_2}{3}(2.5-x)^3 + \frac{M_3}{3}(x-2)^3 + (1.0 - \frac{M_2}{24})\frac{(2.5-x)}{0.5} + (0.6 - \frac{M_3}{24})\frac{(x-2)}{0.5}, & 2 \leq x \leq 2.5 \\ \frac{M_3}{3}(3-x)^3 + \frac{M_4}{3}(x-2.5)^3 + (0.6 - \frac{M_3}{24})\frac{(3-x)}{0.5} + (0.6 - \frac{M_4}{24})\frac{(x-2.5)}{0.5}, & 2.5 \leq x \leq 3 \\ \frac{M_4}{6}(4-x)^3 + (0.6 - \frac{M_4}{6})(4-x) + 1.0(x-3), & 3 \leq x \leq 4 \end{cases} \tag{3}$$

Substituindo e simplificando:

$$s(x) = \begin{cases} 0.4614x^3 - 1.2614x + 1.4, & 0 \leq x \leq 1 \\ -1.1071x^3 + 4.7054x^2 - 5.9668x + 2.9685, & 1 \leq x \leq 2 \\ 2.3934x^3 - 16.2971x^2 + 36.0382x - 25.0349, & 2 \leq x \leq 2.5 \\ -1.0191x^3 + 9.2963x^2 - 27.9452x + 28.2846, & 2.5 \leq x \leq 3 \\ -0.0415x^3 + 0.4979x^2 - 1.5502x + 1.8896, & 3 \leq x \leq 4 \end{cases} \tag{4}$$

Verifica-se que a s(x) é contínua e que os valores do spline nos x_i coincidem exatamente com os valores da função:

$\mathbf{x_i}$	0	1	2	2.5	3	4
$\mathbf{f_i}$	1.4	0.6	1.0	0.6	0.6	1.0
$\mathbf{s(x_i)}$	1.4	0.6	1.0	0.6	0.6	1.0

Tabela 4: Valor do spline nos x_i (nas abcissas interiores, o valor é idêntico nos dois ramos).

Verifica-se, ainda, que a primeira derivada[1] do spline é contínua:

$\mathbf{x_i}$	1		2		2.5		3	
$\mathbf{s'(x_i)}$	0.12	0.12	-0.43	-0.43	-0.57	-0.57	0.32	0.32

Tabela 5: Valores da primeira derivada do spline nas abcissas interiores. O valor à direita/esq. corresponde ao valor obtido pelo ramo da direita/esq., em cada x_i.

e que a segunda derivada[1] é contínua e coincide, nos x_i, com o valor M_i correspondente.

$\mathbf{x_i}$	1		2		2.5		3	
$\mathbf{M_i}$	2.76846		-3.87386		3.30622		0.24896	
$\mathbf{s''(x_i)}$	2.77	2.77	-3.87	-3.87	3.31	3.31	0.25	0.25

Tabela 6: Valores da segunda derivada do spline nas abcissas interiores. O valor à direita/esq. corresponde ao valor obtido pelo ramo da direita/esq., em cada x_i.

Assim, verificamos que o spline foi corretamente construído.

Representação Gráfica

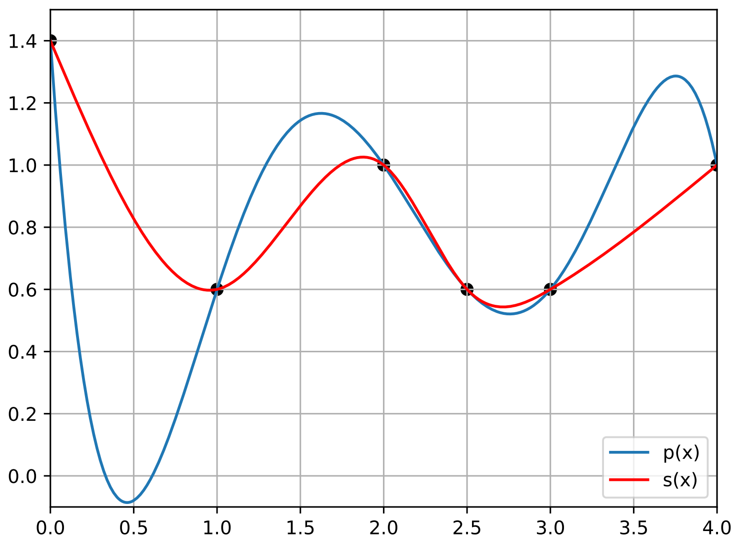

Figura 1: Polinómio interpolador e spline cúbico natural no conjunto de pontos.

[1] O cálculo das derivadas do spline neste exercício (e no seguinte) foi feito através do WolframAlpha, utilizando a expressão obtida na equação 4 (7), com coeficientes aproximados à 4ª casa decimal.

Exercício 2

$$f(x) = x^2 + \sin^2(9x), \quad 0 \le x \le 1 \tag{5}$$

Para obter 9 pontos de abcissas igualmente espaçadas no intervalo $[0, 1]$, basta que a distância entre abcissas adjacentes seja $\frac{1}{8}$. Calculando os valores de f nesses pontos, temos:

$\mathbf{x_i}$	0.0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1.0
$\mathbf{f_i}$	0.0	0.82971	0.66790	0.19412	1.20557	0.76478	0.76504	1.76518	1.16984

Tabela 7: Pontos da função em abcissas igualmente espaçadas.

Polinómio Interpolador

Da mesma forma que no exercício anterior, implementamos um programa em Python para construir o polinómio, pelo método de Newton em diferenças divididas. Obtivemos a seguinte tabela:

i	$\mathbf{x_i}$	$\mathbf{f_i}$	$\mathbf{f_i}[,]$	$\mathbf{f_i}[,,]$	$\mathbf{f_i}[,,,]$	$\mathbf{f_i}[,,,,]$	$\mathbf{f_i}[,,,,,]$	$\mathbf{f_i}[,,,,,,]$	$\mathbf{f_i}[,,,,,,,]$	$\mathbf{f_i}[,,,,,,,,]$
0	0.0	0.0	6.64	-31.73	57.99	190.74	-1598.05	5613.38	-12960.77	21314.38
1	0.125	0.83	-1.29	-9.98	153.36	-808.04	2611.98	-5727.3	8353.61	
2	0.25	0.67	-3.79	47.53	-250.66	824.44	-1683.5	1582.1		
3	0.375	0.19	8.09	-46.47	161.56	-227.74	-496.92			
4	0.5	1.21	-3.53	14.11	47.69	-538.32				
5	0.625	0.76	0.0	32.0	-221.47					
6	0.75	0.77	8.0	-51.06						
7	0.875	1.77	-4.76							
8	1.0	1.17								

Tabela 8: Método de Newton em diferenças divididas.

e o polinómio interpolador (a simplificação foi feita com auxílio do Maxima):

$$\begin{aligned}
p(x) = {} & 6.64x - 31.73x(x-0.125) + 57.99x(x-0.125)(x-0.25) + 190.74x(x-0.125)(x-0.25)(x-0.375) \\
& - 1598.05x(x-0.125)(x-0.25)(x-0.375)(x-0.5) \\
& + 5613.38x(x-0.125)(x-0.25)(x-0.375)(x-0.5)(x-0.625) \\
& - 12960.77x(x-0.125)(x-0.25)(x-0.375)(x-0.5)(x-0.625)(x-0.75) \\
& + 21314.38x(x-0.125)(x-0.25)(x-0.375)(x-0.5)(x-0.625)(x-0.75)(x-0.875) = \\
& = 21314.38x^8 - 87561.1x^7 + 146873.38x^6 - 129156.85x^5 + 63473.25x^4 - 17106.44x^3 + 2271.12x^2 - 106.56x
\end{aligned}$$

Representação Gráfica:

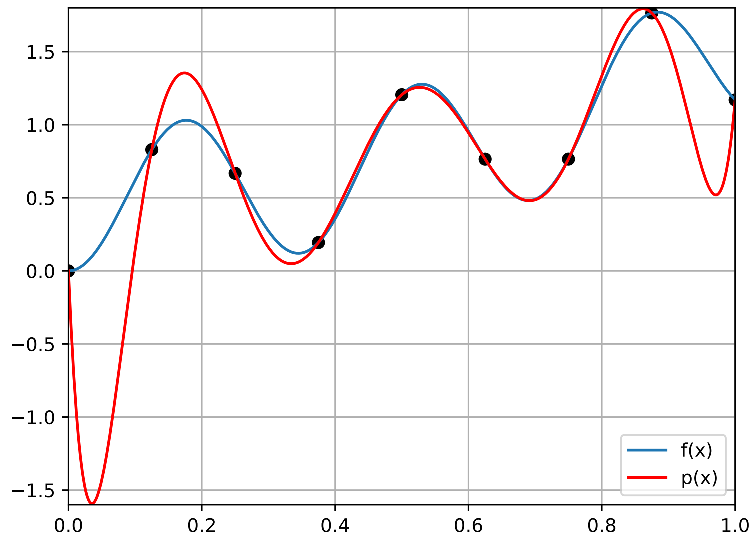

Figura 2: Polinómio interpolador no conjunto de pontos de f.

Spline Cúbico Natural

Para encontrar os valores $M_0, ..., M_8$, chegamos à equação matricial seguinte:

$$\begin{bmatrix} \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \\ M_6 \\ M_7 \\ M_8 \end{bmatrix} = \begin{bmatrix} \frac{f_2-2f_1+f_0}{h} \\ \frac{f_3-2f_2+f_1}{h} \\ \frac{f_4-2f_3+f_2}{h} \\ \frac{f_5-2f_4+f_3}{h} \\ \frac{f_6-2f_5+f_4}{h} \\ \frac{f_7-2f_6+f_5}{h} \\ \frac{f_8-2f_7+f_6}{h} \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

com $h = \frac{1}{8}$. Resolvendo em Python, obtivemos:

M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
0	-79.53736498	-62.59642	210.12957	-207.59761	62.60521	126.53784	-184.80105	0

Tabela 9: Valores dos M_i.

Assim, o spline é dado por:

$$s(x) = \begin{cases} \frac{4M_1}{3}x^3 + 8(f_1 - \frac{M_1}{6\times 8^2})x, & 0 \le x \le 1 \\ \frac{4M_1}{3}(0.25-x)^3 + \frac{4M_2}{3}(x-0.125)^3 + 8(f_1 - \frac{M_1}{6\times 8^2})(0.25-x) + 8(f_2 - \frac{M_2}{6\times 8^2})(x-0.125), & 0.125 \le x \le 0.25 \\ \frac{4M_2}{3}(0.375-x)^3 + \frac{4M_3}{3}(x-0.25)^3 + 8(f_2 - \frac{M_2}{6\times 8^2})(0.375-x) + 8(f_3 - \frac{M_3}{6\times 8^2})(x-0.25), & 0.25 \le x \le 0.375 \\ \frac{4M_3}{3}(0.5-x)^3 + \frac{4M_4}{3}(x-0.375)^3 + 8(f_3 - \frac{M_3}{6\times 8^2})(0.5-x) + 8(f_4 - \frac{M_4}{6\times 8^2})(x-0.375), & 0.375 \le x \le 0.5 \\ \frac{4M_4}{3}(0.625-x)^3 + \frac{4M_5}{3}(x-0.5)^3 + 8(f_4 - \frac{M_4}{6\times 8^2})(0.625-x) + 8(f_5 - \frac{M_5}{6\times 8^2})(x-0.5), & 0.5 \le x \le 0.625 \\ \frac{4M_5}{3}(0.75-x)^3 + \frac{4M_6}{3}(x-0.625)^3 + 8(f_5 - \frac{M_5}{6\times 8^2})(0.75-x) + 8(f_6 - \frac{M_6}{6\times 8^2})(x-0.625), & 0.625 \le x \le 0.75 \\ \frac{4M_6}{3}(0.875-x)^3 + \frac{4M_7}{3}(x-0.75)^3 + 8(f_6 - \frac{M_6}{6\times 8^2})(0.875-x) + 8(f_7 - \frac{M_7}{6\times 8^2})(x-0.75), & 0.75 \le x \le 0.875 \\ \frac{4M_7}{3}(1-x)^3 + 8(f_7 - \frac{M_7}{6\times 8^2})(1-x) + 8f_8(x-0.875), & 0.875 \le x \le 1 \end{cases}$$

Substituindo e simplificando:

$$s(x) = \begin{cases} -106.0498x^3 + 8.2947x, & 0 \le x \le 1 \\ 22.5879x^3 - 48.2392x^2 + 14.3246x - 0.2512, & 0.125 \le x \le 0.25 \\ 363.6346x^3 - 304.0242x^2 + 78.2709x - 5.5801, & 0.25 \le x \le 0.375 \\ -556.9696x^3 + 731.6555x^2 - 310.1090x + 42.9674, & 0.375 \le x \le 0.5 \\ 360.2704x^3 - 644.2044x^2 + 377.8297x - 71.6876, & 0.5 \le x \le 0.625 \\ 85.2435x^3 - 128.5290x^2 + 55.5238x - 4.5424, & 0.625 \le x \le 0.75 \\ -415.1185x^3 + 997.2856x^2 - 788.8371x + 206.5479, & 0.75 \le x \le 0.875 \\ 246.4014x^3 - 739.2042x^2 + 730.5914x - 236.6188, & 0.875 \le x \le 1 \end{cases} \tag{7}$$

Verifica-se que s(x) é contínua e que os valores do spline nos x_i coincidem exatamente com os valores da função:

$\mathbf{x_i}$	0.0	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1.0
$\mathbf{f_i}$	0.0	0.82971	0.66790	0.19412	1.20557	0.76478	0.76504	1.76518	1.16984
$\mathbf{s(x_i)}$	0.0	0.82971	0.66790	0.19412	1.20557	0.76478	0.76504	1.76518	1.16984

Tabela 10: Valor do spline nos x_i (nas abcissas interiores, o valor é idêntico nos dois ramos).

Verifica-se, ainda, que a primeira derivada do spline é contínua:

$\mathbf{x_i}$	0.125		0.25		0.375		0.5		0.625		0.75		0.875	
$\mathbf{s'(x_i)}$	3.32	3.32	-5.56	-5.56	3.66	3.66	3.82	3.83	-5.24	-5.24	6.58	6.58	2.93	2.94

Tabela 11: Valores da primeira derivada do spline nas abcissas interiores. O valor à direita/esq. corresponde ao valor obtido pelo ramo da direita/esq., em cada x_i.

e que a segunda derivada é contínua e coincide, nos x_i, com o valor M_i correspondente.

$\mathbf{x_i}$	0.125		0.25		0.375		0.5		0.625		0.75		0.875	
$\mathbf{M_i}$	-79.537365		-62.59642		210.12957		-207.59761		62.60521		126.53784		-184.80105	
$\mathbf{s''(x_i)}$	-79.5	-79.5	-62.6	-62.6	210.1	210.1	-207.6	-207.6	62.6	62.6	126.5	126.5	-184.8	-184.8

Tabela 12: Valores da segunda derivada do spline nas abcissas interiores. O valor à direita/esq. corresponde ao valor obtido pelo ramo da direita/esq., em cada x_i.

Verificamos, então, que o spline foi corretamente construído.

Representação Gráfica:

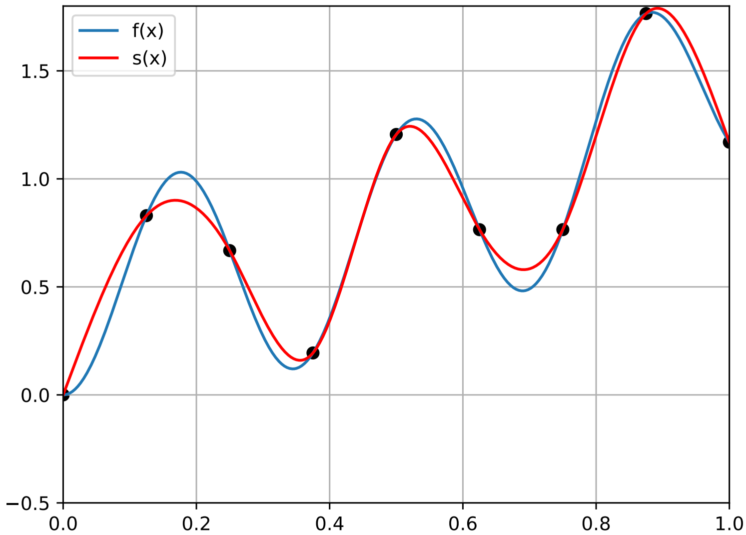

Figura 3: Spline cúbico natural no conjunto de pontos de f.

Aproximação à função e erros efetivos

Vimos pelos gráficos do polinómio e do spline, que este último parece ser, em geral, uma melhor aproximação à função dada. Apesar de, no intervalo $[0.375, 0.75]$, o polinómio estar bastante próximo da função, o mesmo apresenta grandes oscilações perto das extremidades do intervalo considerado, ao contrário do spline, que mantém uma boa aproximação à função ao longo do intervalo. Isto é confirmado pelo gráfico dos erros $|f - p|$ e $|f - s|$:

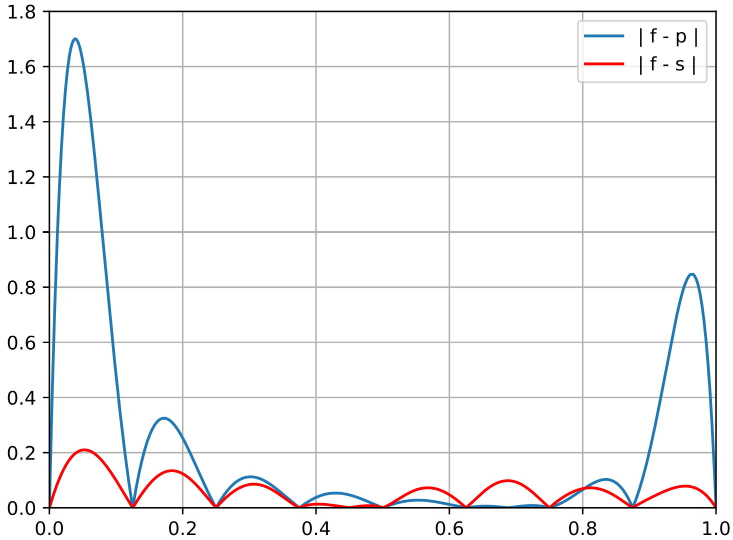

Figura 4: Erro absoluto (efetivo) de p e s relativamente a f.

f(0.12) e f(0.53)

Majorantes:

Como $f(x) \in C^9[0, 1]$, tem-se que $\forall x \in [0, 1]$:

$$|f(x) - p(x)| \leq \frac{1}{9!} M |\pi_9(x)| \tag{8}$$

onde $\pi_9(x) = (x - x_0)(x - x_1)...(x - x_8)$ e $M = max_{x \in [0,1]} |f^{(9)}(x)|$.

Calculando a derivada, obtivemos $f^{(9)}(x) = 99179645184 \sin(18x)$, logo $M = 99179645184$ e temos que $|f(x) - p(x)| \leq \frac{99179645184}{9!} |\pi_9(x)|$.

Por outro lado, $f(x) \in C^4[0,1]$, logo, $\forall x \in [0,1]$:

$$|f(x) - s(x)| \leq \frac{5}{384} M' h^4 \tag{9}$$

onde $M' = max_{x \in [0,1]} |f^{(4)}(x)|$ e $h = 1/8$ (intervalo entre as abcissas).

$f^{(4)}(x) = -52488 \cos(18x)$, logo $M' = 52488$, e temos que $|f(x) - s(x)| \leq \frac{5 \times 52488}{384 \times 8^4} \approx 0.167$

f(0.12):

Usando o polinómio, obtemos que $f(0.12) = p(0.12) \pm \frac{99179645184}{9!} |\pi_9(0.12)| = 0.71 \pm 0.44$. Isto equivale a um erro relativo superior a 60%, logo a aproximação é muito pouco precisa.

Usando o spline, obtemos que $f(0.12) = s(0.12) \pm 0.17 = 0.81 \pm 0.17$. Como o erro absoluto é menor, esta aproximação é mais precisa.

Como conhecemos a função que estamos a aproximar, podemos calcular os valores dos erros efetivos:

	valor	maj. do erro	erro efetivo
p(0.12)	0.713	0.44	0.079
s(0.12)	0.812	0.17	0.020

Tabela 13: Estimativa de f(0.12) pelo polinómio e pelo spline, e respetivos erros (majorantes e efetivos).

Pela tabela, verificamos que os valores obtidos são, de facto, majorantes do erro. Como 0.12 está relativamente próximo de um dos nós (0.125), o valor efetivo do erro é muito pequeno, uma vez que, em ambas as aproximações, o erro efetivo tende a zero perto dos nós.

f(0.53):

Usando o polinómio, obtemos que $f(0.53) = p(0.53) \pm \frac{99179645184}{9!} |\pi_9(0.53)| = 1.26 \pm 0.26$.

Usando o spline, obtemos que $f(0.53) = s(0.53) \pm 0.17 = 1.24 \pm 0.17$. Como no caso anterior, o spline aproxima o valor da função em 0.53 com maior precisão.

Conhecendo a função que estamos a aproximar, podemos calcular os valores dos erros efetivos:

	valor	maj. do erro	erro efetivo
p(0.53)	1.255	0.26	0.022
s(0.53)	1.236	0.17	0.041

Tabela 14: Estimativa de f(0.53) pelo polinómio e pelo spline, e respetivos erros (majorantes e efetivos).

Assim, verificamos que os valores obtidos são, de facto, majorantes do erro. Como tínhamos visto pelo gráfico 2, o polinómio aproxima muito bem a função no intervalo $[0.375, 0.75]$, pelo que temos que $|f(0.53) - p(0.53)|$ é, de facto, muito pequeno.

Observação:

Na resolução do exercício, reparámos que o erro efetivo do spline excede o valor do majorante, em alguns pontos do intervalo $[0, 0.125]$:

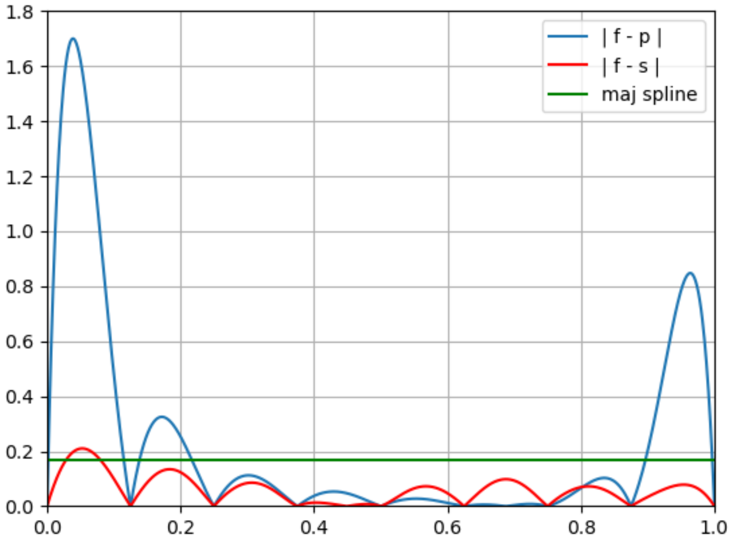

Figura 5: Comparar o erro efetivo com o majorante do spline.

Mesmo após verificarmos todos os cálculos e programas, não conseguimos encontrar razão aparente para este erro. Trabalhamos sempre com valores à precisão do computador, e as expressões utilizadas nos programas não foram tão complexas que justificassem um erro de propagação significativo.

Apêndice

Exercício 1:

```
import numpy as np
import matplotlib.pyplot as plt

n=5
xi=[0,1,2,2.5,3,4]
yi=[1.4,0.6,1.0,0.6,0.6,1.0]
hi=[0]+[xi[i]-xi[i-1] for i in range(1,n+1)]

#tabela de diferencas divididas:
T = [yi]
for j in range(n+1):
    T += [[(T[j][i+1]-T[j][i])/(xi[i+1+j]-xi[i]) for i in range(n-j)]]

#(x-x0)(x-x1)-...(x-xn):
def prod(x,n):
    Pr = 1
    for i in range(n+1):
        Pr *= (x-xi[i])
    return Pr

#polinomio interpolador:
def p(x):
    P = T[0][0]
    for i in range(n):
        P += T[i+1][0]*prod(x,i)
    return P


#sistema dos Ms para o spline:
A = np.array([[1/6,2/3,1/6,0,0,0], [0,1/6,1/2,1/12,0,0],
    [0,0,1/12,1/3,1/12,0],[0,0,0,1/12,1/2,1/6],[1,0,0,0,0,0],[0,0,0,0,0,1]])
B = np.array([1.2,-1.2,0.8,0.4,0,0])
M = np.linalg.solve(A,B)

#funcao spline
def si(x,i):
    return 1/(6*hi[i])*M[i-1]*(xi[i]-x)**3 + 1/(6*hi[i])*M[i]*(x-xi[i-1])**3 + (yi
    [i-1]-(M[i-1]*hi[i]**2)/6)*(xi[i]-x)/hi[i] + (yi[i]-(M[i]*hi[i]**2)/6)*(x-xi[i
    -1])/hi[i]

def s(x):
    for i in range(1,n+1):
        if xi[i-1]<=x<=xi[i]:
            return si(x,i)
```

Exercício 2:

```
import numpy as np
import matplotlib.pyplot as plt
from math import *

```

```python
def f(x):
    return x**2 + (np.sin(9*x))**2

n=8
xi = [i/n for i in range(n+1)]
yi = [f(xi[i]) for i in range(n+1)]
h = 1/n

#interpolacao polinomial:
T = [yi]
for j in range(n+1):
    T += [[(T[j][i+1]-T[j][i])/(xi[i+1+j]-xi[i]) for i in range(n-j)]]

def prod(x,n):
    Pr = 1
    for i in range(n+1):
        Pr *= (x-xi[i])
    return Pr

def p(x):
    P = T[0][0]
    for i in range(n):
        P += T[i+1][0]*prod(x,i)
    return P

#spline:
A = np.array([[h/6,2*h/3,h/6,0,0,0,0,0,0,0], [0,h/6,2*h/3,h/6,0,0,0,0,0,0], [0,0,h/6,2*h/3,h/6,0,0,0,0,0],[0,0,0,h/6,2*h/3,h/6,0,0,0,0],[0,0,0,0,h/6,2*h/3,h/6,0,0],[0,0,0,0,0,h/6,2*h/3,h/6,0],[0,0,0,0,0,0,h/6,2*h/3,h/6],[1,0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,1]])
B1 = [(yi[i+1]-yi[i])/h - (yi[i]-yi[i-1])/h for i in range(1,n)]+[0,0]
B = np.array(B1)
M = np.linalg.solve(A,B)

def si(x,i):
    return 1/(6*h)*M[i-1]*(xi[i]-x)**3 + 1/(6*h)*M[i]*(x-xi[i-1])**3 + (yi[i-1]-(M[i-1]*h**2)/6)*(xi[i]-x)/h + (yi[i]-(M[i]*h**2)/6)*(x-xi[i-1])/h

def s(x):
    for i in range(1,n+1):
        if xi[i-1]<=x<=xi[i]:
            return si(x,i)

#funcoes erro efetivo:
def erro_p(x):
    return abs(f(x)-p(x))

def erro_s(x):
    return abs(f(x)-S(x))

#majorantes do erro:
def erro_pM(x):
    return abs(prod(x,8))*9*18**8/factorial(9)

def erro_sM(x):
    return (52488)*5*h**4/384
```