Universidade Federal de Juiz de Fora
Pós-Graduação em Modelagem Computacional
Métodos Numéricos

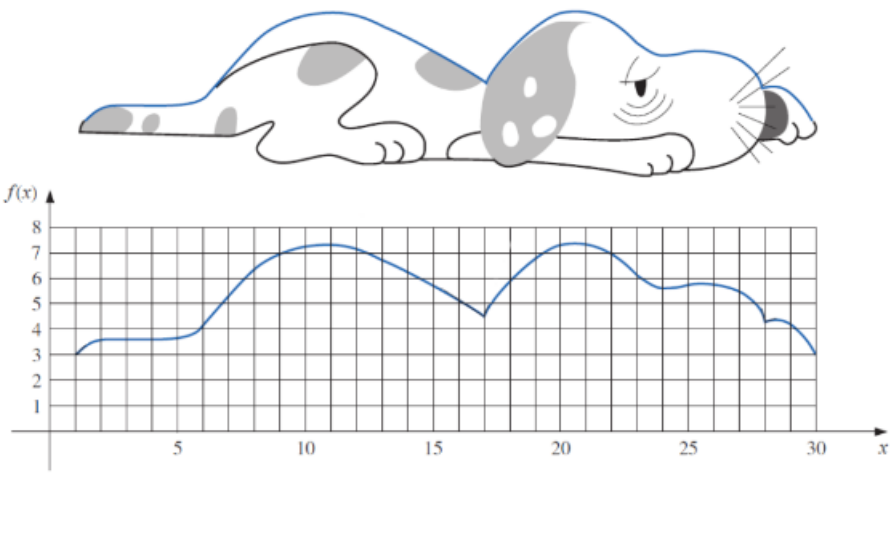
Eduardo Santos de Oliveira Marques

Atividade 5
Interpolação Polinomial

Juiz de Fora
2023

As figuras que seguem ilustram um cachorro e um gráfico que mostra como é a curva da sua parte superior. Os pontos apresentados na tabela que segue foram selecionados para gerar uma aproximação para essa parte superior.

x	$f(x)$
1	3,0
2	3,7
5	3,9
6	4,2
7	5,7
8	6,6
10	7,1
13	6,7
17	4,5
20	7,0
23	6,1
24	5,6
25	5,8
27	5,2
27,7	4,1
28	4,3
29	4,1
30	3,0



Questão 1. Utilize a Forma de Lagrange ou de Newton para montar um polinômio interpolador e desenhe um gráfico com o resultado. O resultado é adequado? Justifique.

Resolução:

A seguir serão apresentados os métodos de Lagrange e Newton para montar o polimônio interpolador.

Método de Lagrange:

O polinômio interpolador de Lagrange é dado pela seguinte fórmula:

$$P(x) = \sum_{i=0}^{n} f(x_i) \cdot L_i(x)$$

onde n é o número de pontos menos 1, e $L_i(x)$ é o termo de Lagrange para o ponto x_i:

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

Aplicando os pontos da tabela fornecida, obtêm-se:

$$P(x) = 3.0 \cdot L_0(x) + 3.7 \cdot L_1(x) + \ldots + 3.0 \cdot L_{18}(x)$$

Calculando os valores de $L_i(x)$ para cada ponto, é possível formar o polinômio interpolador.

Método de Newton:

O polinômio interpolador de Newton é dado pela seguinte fórmula:

$$P(x) = f[x_0] + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + \dots$$

onde $f[x_i]$ é a diferença dividida de ordem i e $f[x_i, x_{i+1}, \dots, x_j]$ é a diferença dividida de ordem j para $j \geq i$.

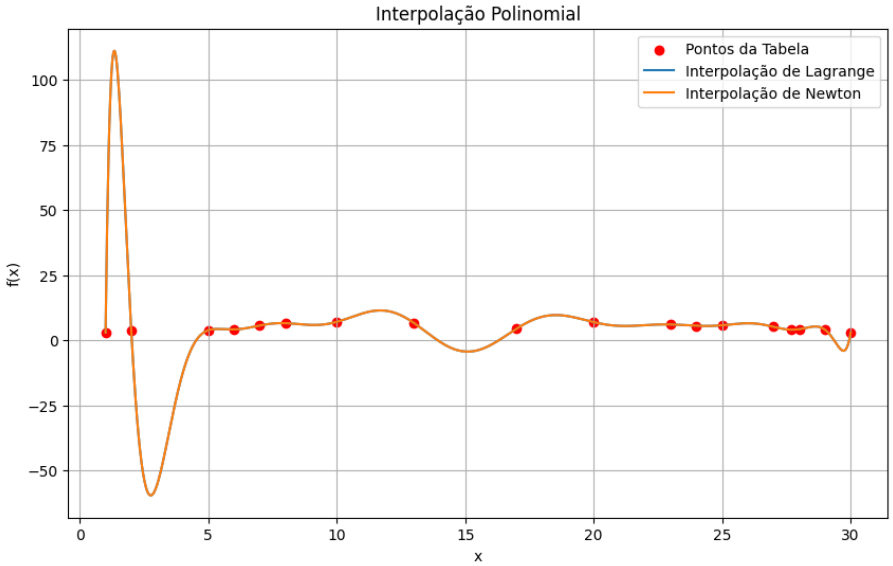
Gráfico Resultante:

Através do ambiente virtual Google Colab e da linguagem de programação Python, o gráfico com o polinômio interpolador foi desenvolvido usando ambos os métodos e os pontos da tabela para verificar o resultado. No entanto, é importante mencionar que, dados os pontos, nota-se que a distribuição não é uniforme e há variações bruscas. Isso pode levar a resultados de interpolação que parecem não muito suaves ou precisos. É possível que o polinômio interpolador tenha oscilações indesejadas entre os pontos. Abaixo é apresentado o código:

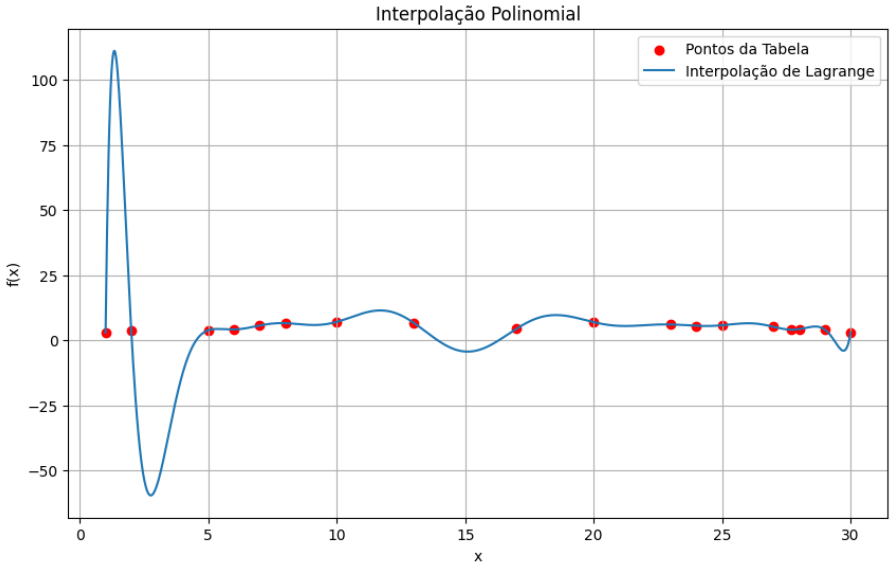
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange
from scipy.interpolate import BarycentricInterpolator

# Dados da tabela
x_values = [1, 2, 5, 6, 7, 8, 10, 13, 17, 20, 23, 24, 25, 27, 27.7, 28, 29,
    30]
f_values = [3.0, 3.7, 3.9, 4.2, 5.7, 6.6, 7.1, 6.7, 4.5, 7.0, 6.1, 5.6,
   5.8, 5.2, 4.1, 4.3, 4.1, 3.0]

# Criar uma funcao para interpolacao utilizando Lagrange
lagrange_interp = lagrange(x_values, f_values)

# Criar uma funcao para interpolacao utilizando Newton
newton_interp = BarycentricInterpolator(x_values, f_values)

# Gerar pontos para o gráfico
x_range = np.linspace(min(x_values), max(x_values), 1000)
lagrange_y = lagrange_interp(x_range)
newton_y = newton_interp(x_range)

# Plotar os resultados
plt.figure(figsize=(10, 6))
plt.scatter(x_values, f_values, color='red', label='Pontos da Tabela')
plt.plot(x_range, lagrange_y, label='Interpolacao de Lagrange')
plt.plot(x_range, newton_y, label='Interpolacao de Newton')
plt.xlabel('x')
```

```
27 plt.ylabel('f(x)')
28 plt.title('Interpolacao Polinomial')
29 plt.legend()
30 plt.grid(True)
31 plt.show()
```

Output:



Observa-se que o resultado não se mostrou adequado para o problema, visto que ficou bem diferente do gráfico apresentado no enunciado. O maior problema foi encontrado no início da interpolação, onde a curva sobe e desce bruscamente. Neste caso, as curvas de Lagrange e Newton apresentaram exatamente o mesmo comportamento, por esse motivo, só é possível observar a cor em laranja, que foi plotada depois, de acordo com o código desenvolvido. Para mostrar que as duas são iguais, abaixo é apresentado apenas a Interpolação de Lagrange.



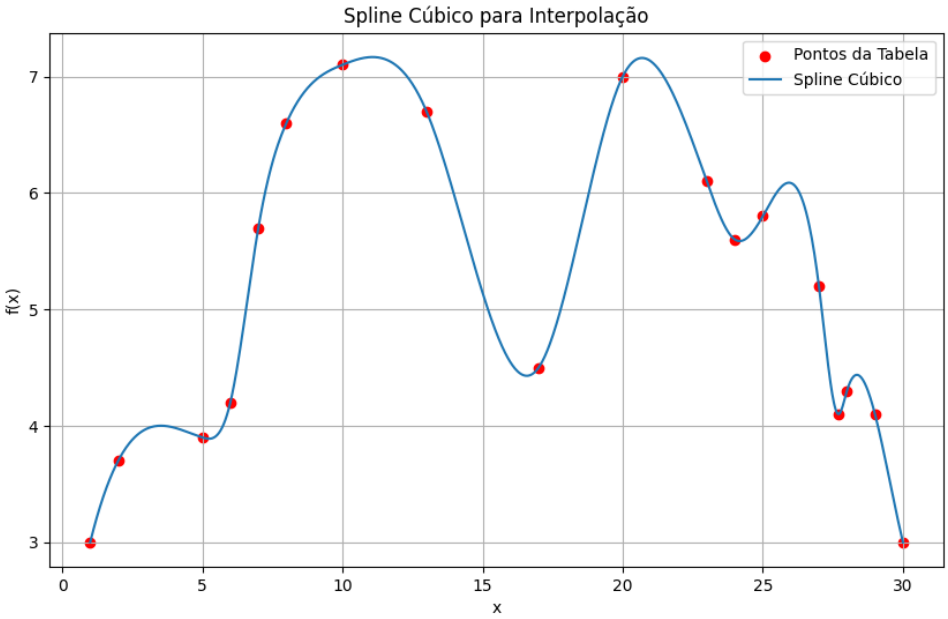
Questão 2. Monte uma curva o mais similar possível à ilustrada para a parte superior do desenho utilizando *Splines*. Desenhe o gráfico e compare com o resultado anterior.

Resolução:

Abaixo foi desenvolvido um código para criar um gráfico utilizando Splines cúbicos para se ajustar aos pontos da tabela e tentar obter uma curva suave que se assemelhe à ilustração da enunciado. Novamente, ele foi desenvolvido em Python no ambiente virtual Google Colab.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import CubicSpline

# Dados da tabela
x_values = [1, 2, 5, 6, 7, 8, 10, 13, 17, 20, 23, 24, 25, 27, 27.7, 28, 29,
    30]
f_values = [3.0, 3.7, 3.9, 4.2, 5.7, 6.6, 7.1, 6.7, 4.5, 7.0, 6.1, 5.6,
   5.8, 5.2, 4.1, 4.3, 4.1, 3.0]

# Criar uma funcao de spline cúbico
spline_interp = CubicSpline(x_values, f_values)

# Gerar pontos para o gráfico
x_range = np.linspace(min(x_values), max(x_values), 1000)
spline_y = spline_interp(x_range)

# Plotar os resultados
plt.figure(figsize=(10, 6))
plt.scatter(x_values, f_values, color='red', label='Pontos da Tabela')
plt.plot(x_range, spline_y, label='Spline Cúbico')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Spline Cúbico para Interpolacao')
plt.legend()
plt.grid(True)
plt.show()
```

Output:



O Spline cúbico forneceu uma curva suave que passa perto dos pontos da tabela, evitando as oscilações indesejadas que podem ocorrer com outros métodos de interpolação. Isso torna os Splines cúbicos especialmente úteis para criar curvas de ajuste suaves e realistas. Foi possível observar que esse método obteve resultados mais satisfatórios que os métodos de Lagrange e Newton.

Questão 3. Demonstre o teorema que segue: Seja $f(x)$ diferenciável $n \neq 1$ vezes no intervalo $[a, b]$. Dados $x_0, x_1, \ldots, x_n$ pontos distintos em $[a, b]$, então existe um ponto $c \in [\min_{i=0,\ldots,n}\{xi\}, \max_{i=0,\ldots,n}\{xi\}]$, tal que:

$$\frac{f^{(n)}(c)}{n!} = f[x_0, x_1, \ldots, x_n]$$

Resolução:

O teorema apresentado é uma forma do Teorema do Valor Médio para Diferenças Divididas. Ele será demonstrado abaixo.

Demonstração:

Considerando a função $g(x) = f(x) - P(x)$, onde $P(x)$ é o polinômio interpolador de Newton que passa pelos pontos $x_0, x_1, \ldots, x_n$ e é definido por:

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

onde $P(x)$ é um polinômio de grau n e $g(x)$ é a diferença entre a função original $f(x)$ e o polinômio interpolador $P(x)$.

Pela definição de diferenças divididas, têm-se:

$$f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(c)}{n!}$$

onde c é um ponto no intervalo $[x_0, x_n]$. Da mesma forma, as diferenças divididas são definidas para o polinômio interpolador $P(x)$:

$$P[x_0, x_1, \ldots, x_n] = \frac{P^{(n)}(c)}{n!}$$

onde c é um ponto no intervalo $[x_0, x_n]$. Agora, observando que $P^{(n)}(x)$ é igual a zero, uma vez que $P(x)$ é um polinômio de grau n. Portanto, $P[x_0, x_1, \ldots, x_n] = 0$.

Agora, retornando à função $g(x) = f(x) - P(x)$, aplica-se o Teorema do Valor Médio para Derivadas ao intervalo $[x_0, x_n]$ para a função $g(x)$. Isso diz que existe um ponto c em $[x_0, x_n]$ tal que:

$$g'(c) = \frac{g(x_n) - g(x_0)}{x_n - x_0}$$

No entanto, $g(x_0) = f(x_0) - P(x_0) = 0$ e $g(x_n) = f(x_n) - P(x_n) = 0$, uma vez que o polinômio interpolador passa pelos pontos extremos x_0 e x_n. Portanto, $g'(c) = 0$. Agora, a derivada $g'(x)$ é dada por:

$$g'(x) = f'(x) - P'(x)$$

Novamente, observa-se que $P'(x)$ é o polinômio derivado de $P(x)$ e, portanto, é um polinômio de grau $n-1$, o que implica que $P^{(n-1)}(x) = 0$. Portanto, $P'[x_0, x_1, \ldots, x_n] = 0$. Agora, voltando a $g'(c) = 0$, têm-se:

$$f'(c) - P'(c) = 0$$

O que leva a:

$$f'(c) = P'(c)$$

No entanto, a derivada $P'(c)$ é uma constante, uma vez que $P'(x)$ é um polinômio de grau $n-1$. Portanto, $f'(c) = P'[x_0, x_1, \ldots, x_n]$, o que, junto com a definição das diferenças divididas, dá a igualdade desejada:

$$\frac{f^{(n)}(c)}{n!} = f[x_0, x_1, \ldots, x_n]$$

Assim, o teorema foi demonstrado. Esse teorema é fundamental na interpolação polinomial e na construção de fórmulas de diferenças divididas para fins de interpolação e aproximação de funções. Ele fornece uma relação entre as derivadas da função original e as diferenças divididas, o que ajuda a entender como os coeficientes de interpolação estão relacionados às propriedades da função.

APÊNDICE A – Códigos da Atividade

Abaixo são apresentados os códigos realizados, desenvolvidos e testados na plataforma `https://colab.google/`. A seguir, segue o link do ambiente virtual com as questões: `https://colab.research.google.com/drive/1Ef-IEJPvp0f7udZExhBoTT4s4b5LmZm3?usp=sharing`

OBS: É importante ressaltar que a função que explicita códigos no Overleaf (*lstlisting*) apresenta erros quando alguns caracteres são inseridos, tais como: **ç**, **â** e **ã**. Então os comentários e *prints* dos códigos no relatório são diferentes do ambiente virtual, lembrando que apenas é feita a troca dos caracteres não reconhecidos.