Laboratório 4 - Interpolação

Instituto Tecnológico de Aeronáutica - ITA
Matemática Computacional CCI - 22
Nicholas Scharan Cysne
Turma 22.1

1. Introdução

Gerando 100 pontos, espaçados igualmente entre o intervalo $[-1,\ 0]$, encontramos $f(x) = x^{10} - 3x^2 - 2$ para cada um dos pontos e $g(x) = -1,004x^2 + 0.996x - 2$ igualmente. Como primeira tarefa do laboratório, deve-se encontrar o polinômio de ordem 2 $p(x)$ que melhor aproxima os 100 pontos gerados por $f(x)$. Posteriormente calcula-se o maior erro absoluto neste intervalo de valores entre as funções $f(x)$ e $g(x)$, bem como $f(x)$ e $p(x)$.

A segunda tarefa do laboratório consiste em obter o nós de Chebyshev no intervalo $[-3,\ 3]$ dada a função $f(x) = \frac{4}{5 + 2x^2}$ e 9 polinômios, variando de grau 2 a grau 10 sendo aplicados em 100 pontos espaçados igualmente dentro deste intervalo. Posteriormente, calcular o maior erro absoluto entre $f(x)$ e $p(x)$ para cada polinômio encontrado.

2. Resultados

2.1 Interpolação

A interpolação de 100 pontos com coordenadas no eixo X entre o intervalo $[-1,\ 0]$ e eixo Y dados por $f(x) = x^{10} - 3x^2 - 2$ resultou no polinômio $p(x)$ dado abaixo.

$$p(x) = -1.38258688\,x^2 + 1.14632412\,x - 1.8736144$$

Iterando pelos 100 pontos x_i de estudo, podemos encontrar o maior erro absoluto entre $f(x)$ e $g(x)$, bem como $f(x)$ e $p(x)$, sendo mostrados na Tabela 1.

Tabela 1. Erros Absolutos.

Maior Erro Absoluto entre $f(x)$ **e** $g(x)$	0.4243073205848007
Maior Erro Absoluto entre $f(x)$ **e** $p(x)$	0.4025253980942205

A Figura 1 mostra o plot de $f(x)$, $g(x)$ e $p(x)$ entre o intervalo $[-1,\ 0]$ e a Figura 2 mostra o output do algoritmo feito em Python3.

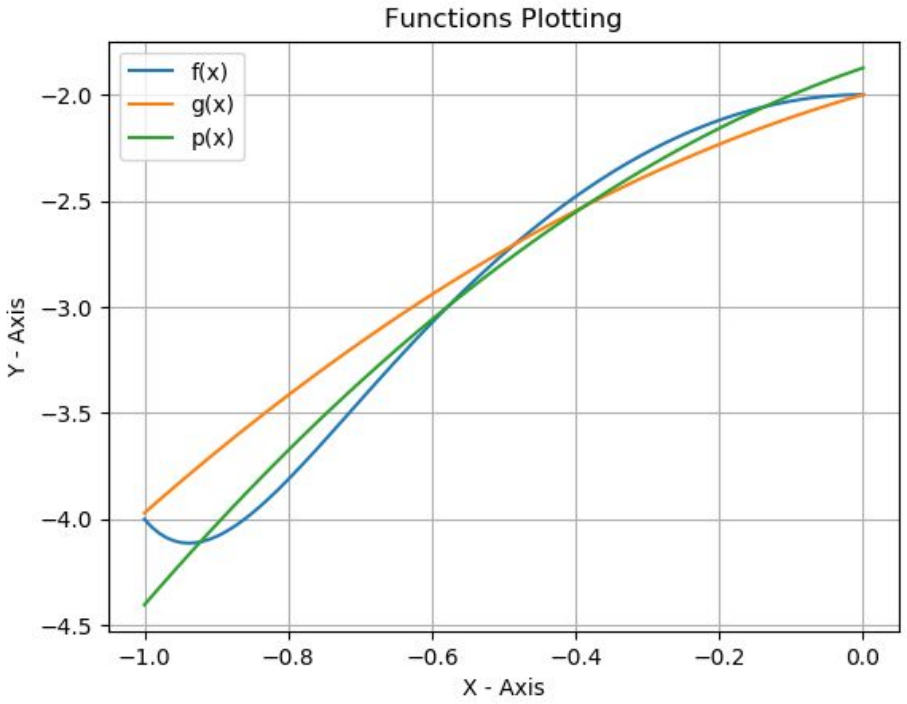

Figura 1. Plot de $f(x)$, $g(x)$ e $p(x)$.

```
Polinom p(x) fitted in vector: [-1.38258688  1.14632412 -1.8736144 ]
Maximum Absolute Error Between f(x) and g(x): 0.4243073205848007
Maximum Absolute Error Between f(x) and p(x): 0.4025253980942205
```

Figura 2. Output dado pelo algoritmo feito em Python3.

2.2 Nós de Chebyshev

A obtenção dos nós de Chebyshev em um intervalo $[-A, A]$ é dado pela equação 1. Assim, variando o grau da função de 3 a 10, obtém-se o seguintes nós de Chebyshev, dados pela Tabela 2.

$$C_n = A\ cos\ (n\pi/2)$$

Tabela 2. Nós de Chebyshev.

Grau do Polinômio	**Nós de Chebyshev**
Grau 2	[3.0, 1.84e-16, -3.0]
Grau 3	[3.0, 1.84e-16, -3.0, -5.51e-16]
Grau 4	[3.0, 1.84-16, -3.0, -5.51e-16, 3.0]
Grau 5	[3.0 1.84e-16 -3.0 -5.51e-16 3.0 9.18e-16]

Grau 6	[3.0 1.84e-16 -3.0 -5.51e-16 3.0 9.18e-16 -3.0]
Grau 7	[3.0, 1.84e-16, -3.0, -5.51e-16, 3.0, 9.18e-16, -3.0, -1.28e-15]
Grau 8	[3.0, 1.84e-16, -3.0, -5.51e-16, 3.0, 9.18e-16, -3.0, -1.28e-15, 3.0]
Grau 9	[3.0, 1.84e-16, -3.0, -5.51e-16, 3.0, 9.18e-16, -3.0, -1.28e-15, 3.0, 1.65e-15]
Grau 10	[3.0, 1.84e-16, -3.0, -5.51e-16, 3.0, 9.18e-16, -3.0, -1.28e-15, 3.0, 1.65e-15, -3.0]

A Figura 3 mostra o output do programa na obtenção dos Nós de Chebyshev.

```
Chebyshev Nodes of degree 2: [ 3.0000000e+00  1.8369702e-16 -3.0000000e+00]

Chebyshev Nodes of degree 3: [ 3.0000000e+00  1.8369702e-16 -3.0000000e+00 -5.5109106e-16]

Chebyshev Nodes of degree 4: [ 3.0000000e+00  1.8369702e-16 -3.0000000e+00 -5.5109106e-16
  3.0000000e+00]

Chebyshev Nodes of degree 5: [ 3.00000000e+00  1.83697020e-16 -3.00000000e+00 -5.51091060e-16
  3.00000000e+00  9.18485099e-16]

Chebyshev Nodes of degree 6: [ 3.00000000e+00  1.83697020e-16 -3.00000000e+00 -5.51091060e-16
  3.00000000e+00  9.18485099e-16 -3.00000000e+00]

Chebyshev Nodes of degree 7: [ 3.00000000e+00  1.83697020e-16 -3.00000000e+00 -5.51091060e-16
  3.00000000e+00  9.18485099e-16 -3.00000000e+00 -1.28587914e-15]

Chebyshev Nodes of degree 8: [ 3.00000000e+00  1.83697020e-16 -3.00000000e+00 -5.51091060e-16
  3.00000000e+00  9.18485099e-16 -3.00000000e+00 -1.28587914e-15
  3.00000000e+00]

Chebyshev Nodes of degree 9: [ 3.00000000e+00  1.83697020e-16 -3.00000000e+00 -5.51091060e-16
  3.00000000e+00  9.18485099e-16 -3.00000000e+00 -1.28587914e-15
  3.00000000e+00  1.65327318e-15]

Chebyshev Nodes of degree 10: [ 3.00000000e+00  1.83697020e-16 -3.00000000e+00 -5.51091060e-16
  3.00000000e+00  9.18485099e-16 -3.00000000e+00 -1.28587914e-15
  3.00000000e+00  1.65327318e-15 -3.00000000e+00]
```

Figura 3. Output do programa para os Nós de Chebyshev.

Para cada grau também obteve-se o polinômio associado que melhor aproxima os pontos dados pela aplicação de $f(x) = \frac{4}{5 + 2x^2}$ em 100 pontos igualmente espaçados no intervalo $[-3, 3]$. A cada um destes polinômios verificou-se o maior erro absoluto associado quando comparado à $f(x)$, podendo ser verificado na Tabela 3.

Grau do Polinômio Associado	Maior Erro Absoluto
Grau 2	0.1389666246413731
Grau 3	0.13896662464137377
Grau 4	0.056796554162190094
Grau 5	0.05679655416218998

Grau 6	0.020953911927931057
Grau 7	0.020953911927930724
Grau 8	0.007170990980141512
Grau 9	0.00717099098013696
Grau 10	0.002317859450271964

A Figura 4 mostra o output do programa na obtenção do maior erro absoluto de cada polinômio.

```
Maximum Absolute Error Between f(x) and p(x) of degree 2: 0.1389666246413731
Maximum Absolute Error Between f(x) and p(x) of degree 3: 0.13896662464137377
Maximum Absolute Error Between f(x) and p(x) of degree 4: 0.056796554162190094
Maximum Absolute Error Between f(x) and p(x) of degree 5: 0.05679655416218998
Maximum Absolute Error Between f(x) and p(x) of degree 6: 0.020953911927931057
Maximum Absolute Error Between f(x) and p(x) of degree 7: 0.020953911927930724
Maximum Absolute Error Between f(x) and p(x) of degree 8: 0.007170990980141512
Maximum Absolute Error Between f(x) and p(x) of degree 9: 0.00717099098013696
Maximum Absolute Error Between f(x) and p(x) of degree 10: 0.002317859450271964
```

Figura 4. Output do programa para o maior erro absoluto de cada polinômio.

O plot de cada polinômio em comparação com a $f(x)$ foi realizado, bem como a identificação dos Nós de Chebyshev nos gráficos a seguir, mostrando nos Gráficos 2.1 a 2.9.

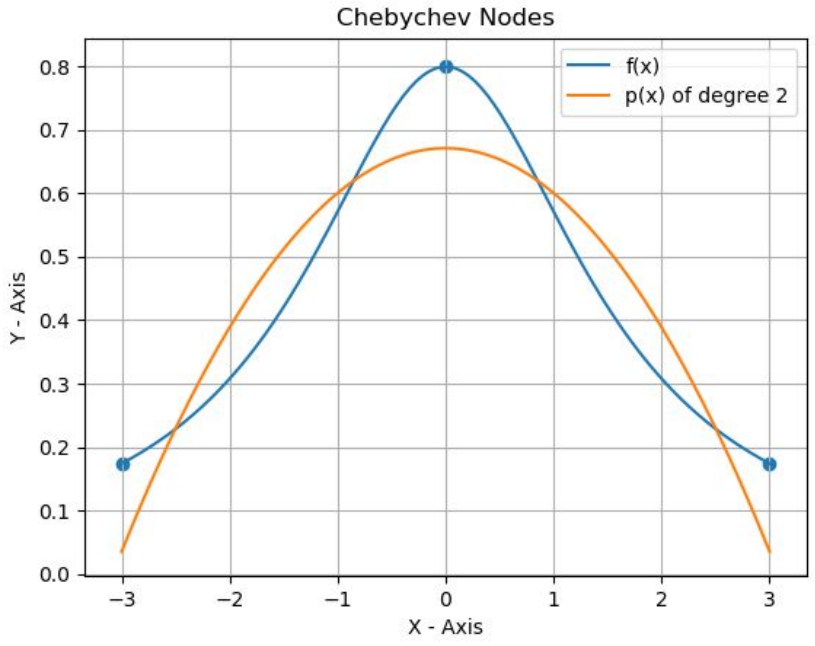

Gráfico 2.1. Plot do polinômio de grau 2.

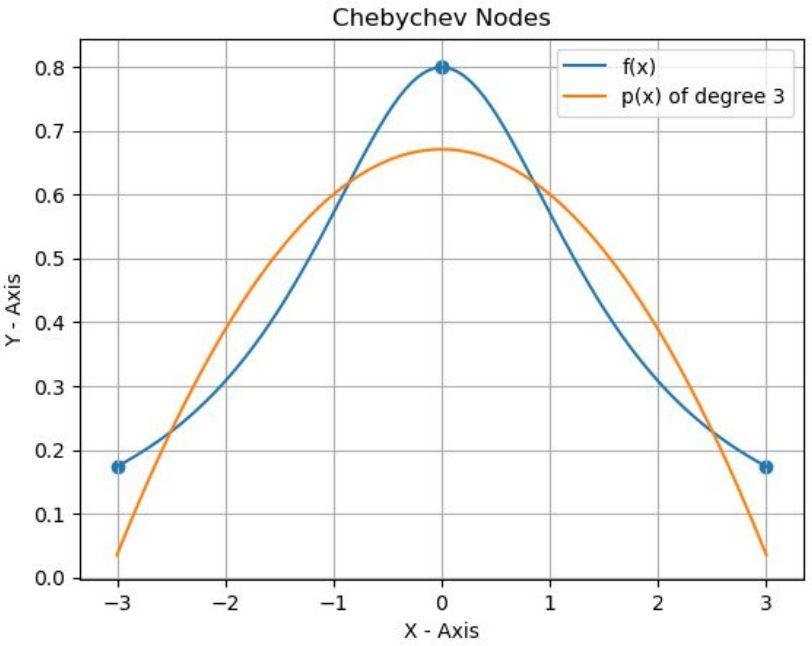

Gráfico 2.2. Plot do polinômio de grau 3.

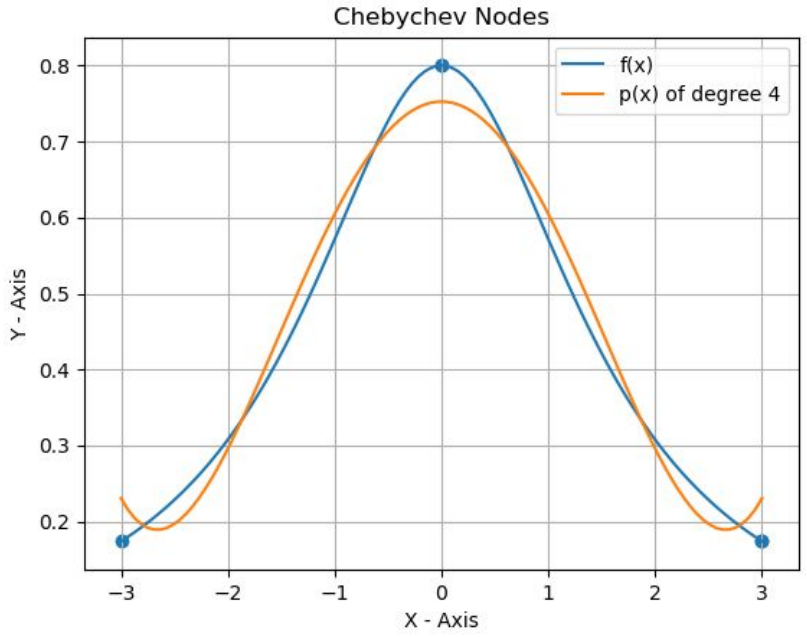

Gráfico 2.3. Plot do polinômio de grau 4.

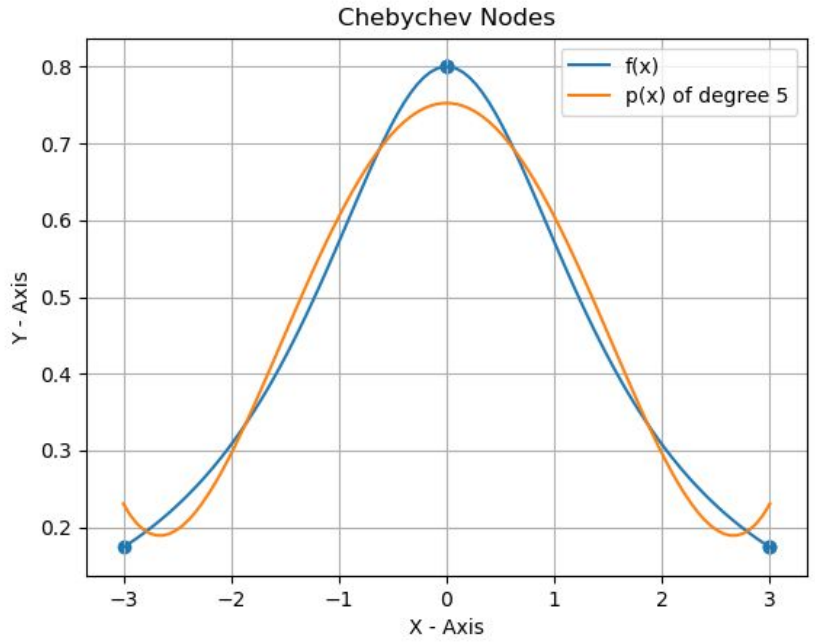

Gráfico 2.4. Plot do polinômio de grau 5.

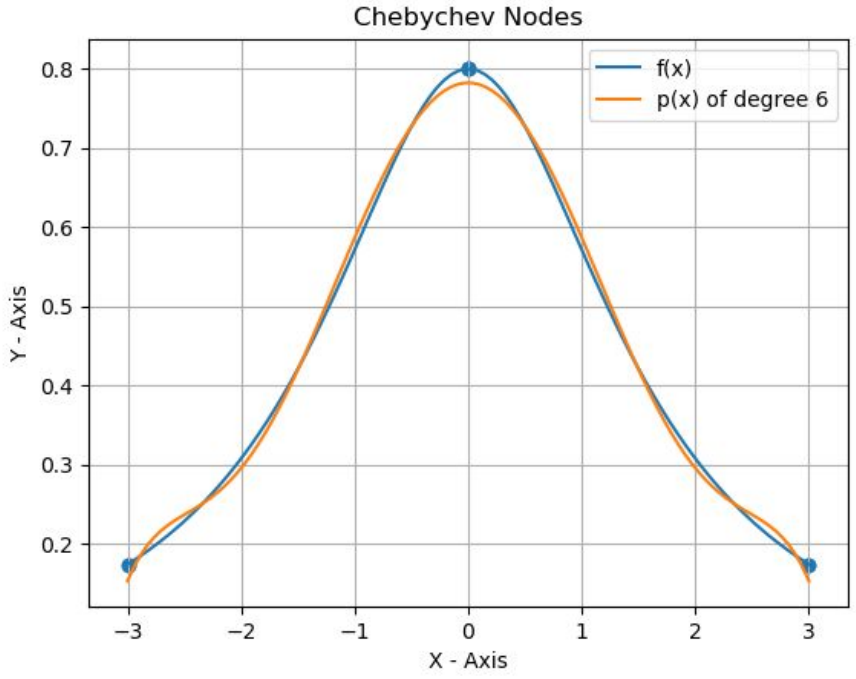

Gráfico 2.5. Plot do polinômio de grau 6.

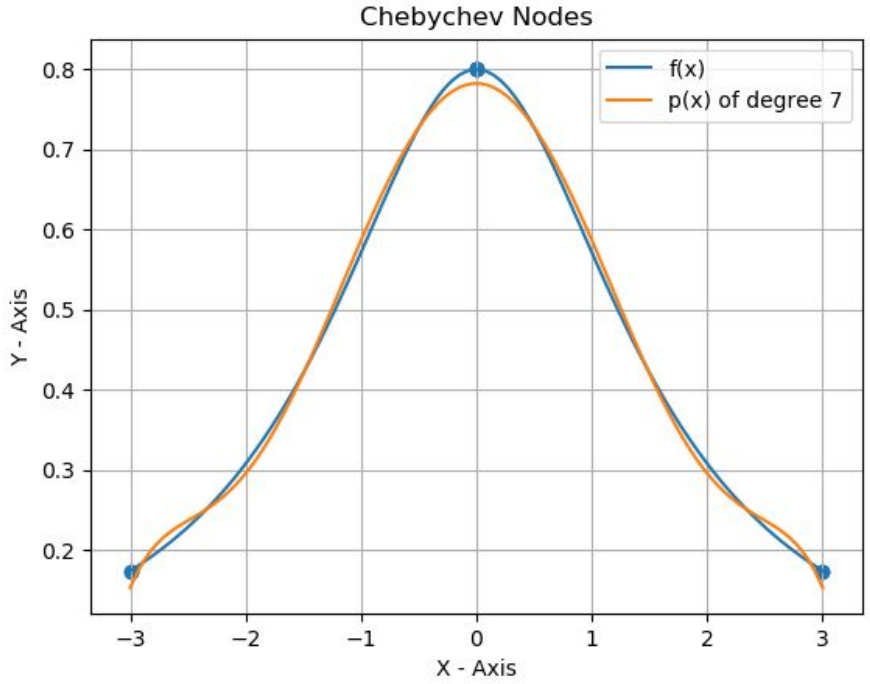

Gráfico 2.6. Plot do polinômio de grau 7.

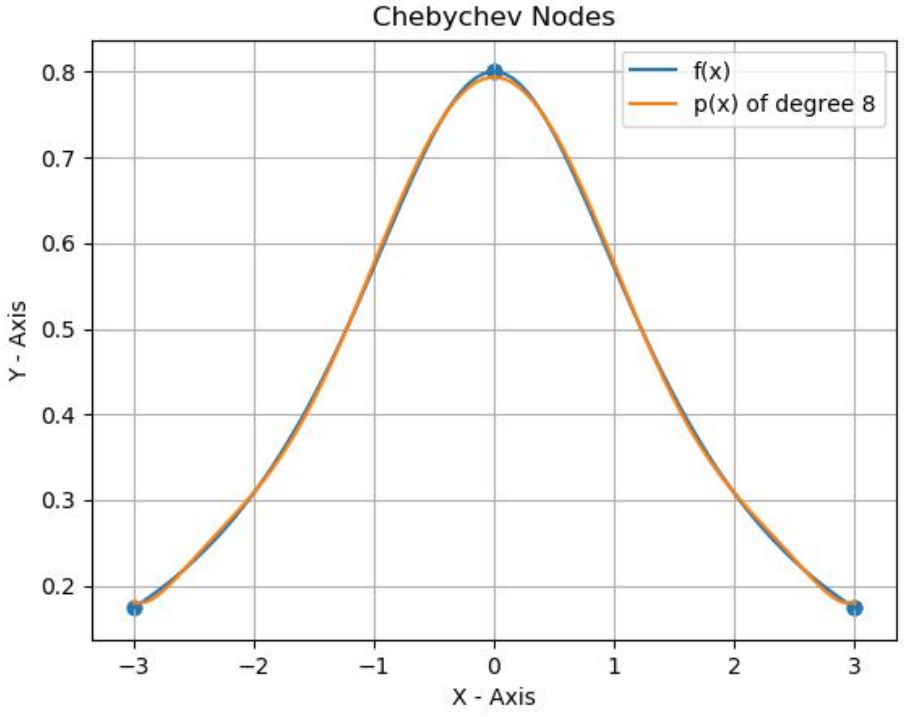

Gráfico 2.7. Plot do polinômio de grau 8.

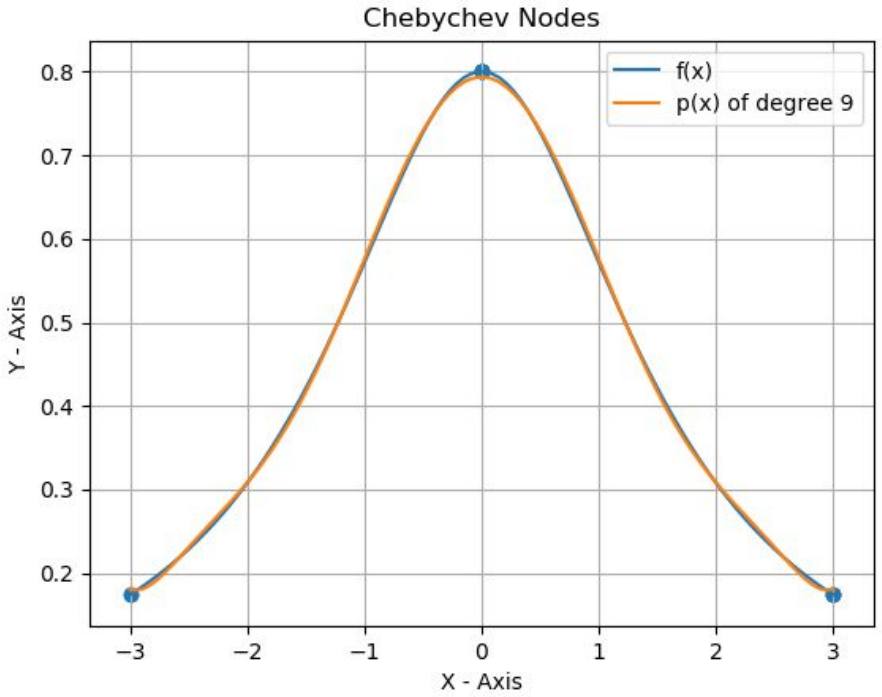

Gráfico 2.8. Plot do polinômio de grau 9.

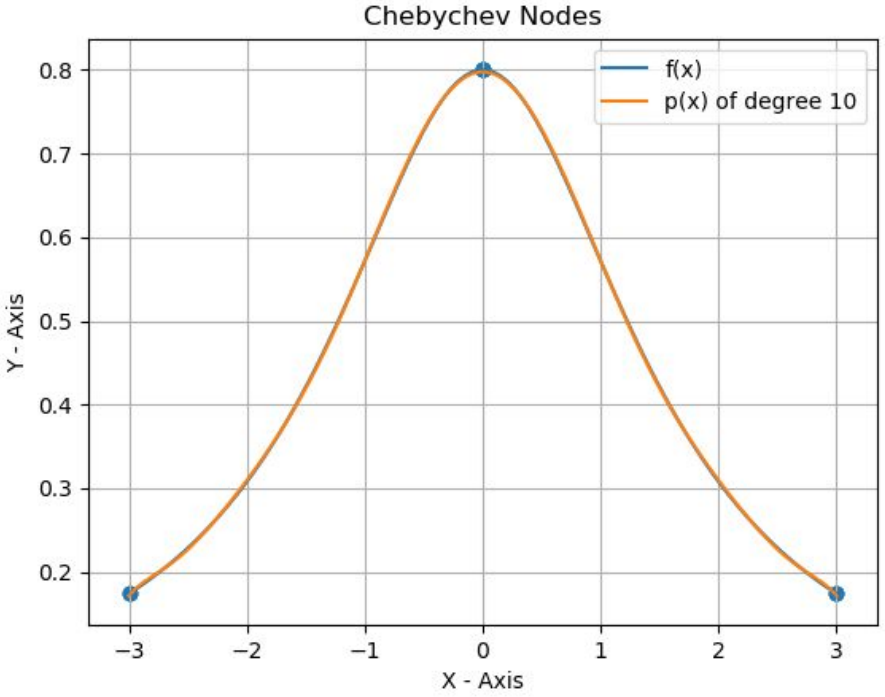

Gráfico 2.9. Plot do polinômio de grau 10.

3. Código

A implementação dos algoritmos foi feita utilizando Python3 utilizando a biblioteca *numpy* de representações numéricas, os códigos fonte estão em anexo no Apêndice A. Cada algoritmo foi feito em seu respectivo arquivo, sendo o método de Interpolação documentado no arquivo “interpolation.py” e os Nós de Chebyshev documentado no arquivo “chebyshev.py”.

4. Conclusão

O algoritmo de interpolação gerou um $p(x)$ que aproxima melhor os pontos que a $g(x)$ dada pelo problema, apresentando um erro absoluto menor. Isso ocorre pelo processo de regressão polinomial que tenta achar a curva que melhor aproxima um conjunto de pontos.

Para o segundo problema, podemos constatar que ao aumentar o grau de um polinômio, este, a cada acréscimo, possui um melhor fit para um conjunto de pontos, obtendo uma curva sobreposta chegando ao grau 10, possuindo um erro mínimo. Os nós de Chebyshev neste caso ficam alternando entre os pontos $(3,\ 4/23)$, $(-3,\ 4/23)$ e pontos muito próximos de $(0,\ 4/5)$.

APÊNDICE A

Arquivo “interpolation.py”:

```
#   Laboratório 4 - Interpolação
#
#   Nicholas Scharan Cysne      T22.1
#   Professor: Johnny
#   Data: 02/05/2019
#
#   Interpolação

import numpy as np
import matplotlib.pyplot as plt

# Size of interpolation
size = 100

# Vetor de [0, 1] com 100 valores equidistantes
vector = np.linspace(-1, 0, size)

# Hardcoded function f(x) = x^10 - 3x^2 - 2
function1 = [1, 0, 0, 0, 0, 0, 0, 0, -3, 0, -2]

# Hardcoded function f(x) = -1.004x^2 + 0.996x - 2
function2 = [-1.004, 0.966, -2]

# Vector of function f(x) = x^10 - 3x^2 - 2 applied in vector
vector_f1 = np.polyval(function1, vector)

# Vector of function f(x) = -1.004x^2 + 0.996x - 2 applied in vector
vector_f2 = np.polyval(function2, vector)

# Function p(x) of degree 2 for polifit vector and f(vector)
function1_fitted = np.polyfit(vector, vector_f1, 2)
vector_f1f = np.polyval(function1_fitted, vector)
print("Polinom p(x) fitted in vector: {}".format(function1_fitted))

# Maximum absolute error between f(x) and g(x)
error_vector = np.abs(vector_f1 - vector_f2)
```

```
error_vector.sort()
error = error_vector[size - 1]
print("Maximum Absolute Error Between f(x) and g(x): {}".format(error))

# Maximum absolute error between f(x) and p(x)
error_vector = np.abs(vector_f1 - vector_f1f)
error_vector.sort()
error = error_vector[size - 1]
print("Maximum Absolute Error Between f(x) and p(x): {}".format(error))

# Plotting
fig_format = 'png'
plt.figure()
plt.plot(vector, vector_f1)
plt.plot(vector, vector_f2)
plt.plot(vector, vector_f1f)
plt.legend(['f(x)', 'g(x)', 'p(x)'])
plt.xlabel('X - Axis')
plt.ylabel('Y - Axis')
plt.title('Functions Plotting')
plt.grid()
plt.savefig('plotting.%s' % fig_format, format=fig_format)
plt.show()
```

Arquivo “chebyshev.py”:

```
#   Laboratório 4 - Interpolação
#
#   Nicholas Scharan Cysne      T22.1
#   Professor: Johnny
#   Data: 02/05/2019
#
#   Nós de Chebyshev

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

# Size of interpolation
size = 100

# Vetor de [-3, 3] com 100 valores equidistantes
vector = np.linspace(-3, 3, size)

# Chebyshev Nodes for p(x) of degree 2 to 10
chebyshev = 9*[0]
for i in range (9):
   shape = np.linspace(0, i+2, i+3)
   chebyshev[i] = 3 * np.array([math.cos(a*math.pi/2) for a in shape])
   print("Chebyshev Nodes of degree {}: {}\n".format(i + 2, chebyshev[i]))

# Hardcoded function d(x) = 2x^2 + 5
function_d = [2, 0, 5]

# Vector of function d(x) = 2x^2 + 5 applied in vector
vector_d = np.polyval(function_d, vector)

# Vector of function f(x) = 4/d(x) applied in vector
vector_f = 4 / vector_d

# Function p(x) of degree 2 to 10 for polifit vector and f(vector)
function_fitted = 9*[0]
vector_p = 9*[0]
for i in range (9):
```

```
    function_fitted[i] = np.polyfit(vector, vector_f, i + 2)
    vector_p[i] = np.polyval(function_fitted[i], vector)

# Maximum absolute error between f(x) and p(x)
for i in range (9):
    error_vector = np.abs(vector_f - vector_p[i])
    error_vector.sort()
    error = error_vector[size - 1]
    print("Maximum Absolute Error Between f(x) and p(x) of degree {}:
          {}".format(i + 2, error))


# Plotting
for i in range (9):
    fig_format = 'png'
    plt.figure()
    plt.plot(vector, vector_f)
    plt.plot(vector, vector_p[i])
    plt.scatter(chebyshev[i], 4 / np.polyval(function_d, chebyshev[i]))
    plt.legend(['f(x)', 'p(x) of degree {}'.format(i + 2)])
    plt.xlabel('X - Axis')
    plt.ylabel('Y - Axis')
    plt.title('Chebychev Nodes')
    plt.grid()
    plt.savefig('plotting_{}.{}'.format(i+2, fig_format), format=fig_format)
    plt.show()
```