Facultad de Matemática, Astronomía, Física y Computación, U.N.C.

Computación

Aula Virtual: https://famaf.aulavirtual.unc.edu.ar/course/view.php?id=747

Resguardo tutoriales: https://www.famaf.unc.edu.ar/~moreschi/docencia/Computacion/

Tutorial Problema 11 de la Guía N° 4

Problema 11:

Utilice las funciones `ilagrange` e `inewton` definidas en los problemas anteriores para interpolar y graficar las funciones según se indica a continuación.

a) Función Lorentziana en grilla uniforme:

$$y = f(x) = \frac{1}{1+16x^2},$$

interpolada en $N+1$ puntos uniformemente distribuidos en el intervalo $[-1,1]$, o sea con $h = 2/N$, $x_i = -1 + hi,\ i = 0, 1, \ldots N$. Utilice $N = 4$, $N = 10$ y $N = 20$. Grafique la función interpolada (en azul) y el polinomio interpolatorio $p_N(x)$ (en rojo) superpuestos en un mismo gráfico, utilizando al menos 201 puntos como grilla de graficación. ¿Qué observa cerca de los extremos del intervalo? Realice un segundo gráfico con el error $\epsilon(x) = p_N(x) - f(x)$.

b) Repita el caso anterior pero interpolando la función en los puntos de Chebyshev dados por

$$x_j = -\cos\Big(\frac{\pi}{N}j\Big), \qquad j = 0, 1, 2 \ldots N.$$

c) Repita los dos puntos anteriores pero usando $f(x) = \sin(x)$, interpolada en el intervalo $[0, \pi]$. Las grillas uniforme y de Chebyshev vienen dadas por

$$x_j = \frac{\pi}{N}j, \qquad j = 0, 1, 2, \ldots, N. \qquad \text{(uniforme)}$$

$$x_j = \frac{\pi}{2}\left(1 - \cos\Big(\frac{\pi}{N}j\Big)\right), \qquad j = 0, 1, 2, \ldots, N. \qquad \text{(Chebyshev)}$$

En ambos casos utilice $N = 4$ y $N = 10$

Tutorial:

- Guarde en el archivo p11.py las siguientes instrucciones:

```python
# Interpolación de Lagrange y de Newton y fenómeno de Runge

import numpy as np
import matplotlib.pyplot as plt

def puntos_interpolacion(N, xmin, xmax, points):
    """ elige los puntos de Interpolación """
    if points == "equiesp":
        x = np.linspace(xmin, xmax, N)
```

```
    elif points == "cheby":
        # nodos de Chebyshev
        x = 0.5*(xmin + xmax) + \
            0.5*(xmax - xmin)*np.cos(np.arange(N)*np.pi/(N-1))
    return x

# función a estudiar ------------------------------------
def f(x):
    return 1./(1. + 16. * x**2)

# -------------------------------------------------------------
# definición de la función de python que calcula la Interpolación
# con el método de Lagrange
def ilagrange(x, xp, fp):
    pol = 0.0
    # suma sobre los puntos
    for m in range(len(xp)):
        l = 1.
        for n in range(len(xp)):
            if n == m:
                continue
            l *= (x - xp[n])/(xp[m] - xp[n])
        pol += fp[m]*l
    return pol


# -------------------------------------------------------------
# definición de la funciones de python que calculan la Interpolación
# con el método de Newton

def poli_newton_coeficientes(x, y):
    """
    coeficientes del método de Newton
    x: np array conteniendo los puntos x
    y: np array conteniendo los puntos y
    """
    m = len(x)
    x = np.copy(x)
    a_newt = np.copy(y)
    for k in range(1, m):
        a_newt[k:m] = (a_newt[k:m] - a_newt[k - 1])/(x[k:m] - x[k - 1])
    return a_newt

def inewton(x, x_datos, y_datos):
    """
    interpolación por el método de Newton
    x_datos: puntos x
    y_datos: puntos y
    x: punto de evaluación
    """
    a_newt = poli_newton_coeficientes(x_datos, y_datos)
    n = len(x_datos) - 1  # grado del polinomio
    p = a_newt[n]
    for k in range(1, n + 1):
        p = a_newt[n - k] + (x - x_datos[n - k])*p
    return p
```

```
66
67
68 # ------------------------------------------------------------------
69 # cálculos
70
71 xmin = -1.0
72 xmax =  1.0
73
74 neval = 200
75
76 ####################################################################
77 # distribución de puntos equiespaciados
78
79 points = "equiesp"  # puede ser: cheby o equiesp
80
81 # ------------------------------------------------
82 npts = 5
83
84 # xp, fp  son los puntos de donde se construye la Interpolación
85 xp = puntos_interpolacion(npts, xmin, xmax, points)
86 fp = f(xp)
87
88 # xx grilla sonde se calculará la Interpolación
89 # ff es el valor de la Interpolación
90 xx = np.linspace(xmin, xmax, neval)
91 ff5 = np.zeros(len(xx))
92
93 for n in range(len(xx)):
94     ff5[n] = ilagrange(xx[n], xp, fp)
95
96 nn5 = np.zeros(len(xx))
97
98 for n in range(len(xx)):
99     nn5[n] = inewton(xx[n], xp, fp)
100
101 # ------------------------------------------------
102 npts = 11
103
104 # xp, fp  son los puntos de donde se construye la Interpolación
105 xp = puntos_interpolacion(npts, xmin, xmax, points)
106 fp = f(xp)
107
108 # xx grilla sonde se calculará la Interpolación
109 # ff es el valor de la Interpolación
110 xx = np.linspace(xmin, xmax, neval)
111 ff11 = np.zeros(len(xx))
112
113 for n in range(len(xx)):
114     ff11[n] = ilagrange(xx[n], xp, fp)
115
116 nn11 = np.zeros(len(xx))
117
118 for n in range(len(xx)):
119     nn11[n] = inewton(xx[n], xp, fp)
120
121 # ------------------------------------------------
```

```
122 npts = 21
123
124 # xp, fp  son los puntos de donde se construye la Interpolación
125 xp = puntos_interpolacion(npts, xmin, xmax, points)
126 fp = f(xp)
127
128 # xx grilla sonde se calculará la Interpolación
129 # ff es el valor de la Interpolación
130 xx = np.linspace(xmin, xmax, neval)
131 ff21 = np.zeros(len(xx))
132
133 for n in range(len(xx)):
134     ff21[n] = ilagrange(xx[n], xp, fp)
135
136 nn21 = np.zeros(len(xx))
137
138 for n in range(len(xx)):
139     nn21[n] = inewton(xx[n], xp, fp)
140
141 # valor de la función exacta en los puntos de Interpolación
142 fexact = f(xx)
143
144 # error
145 e = fexact-ff21
146
147 plt.figure(figsize=(10,8))
148 plt.subplot(211)
149 plt.ylabel('funciones')
150 #plt.xlabel('x')
151 plt.title('Aproximacion de Lagrange, puntos equiespaciados')
152 plt.plot(xx, fexact, color="b")
153 plt.scatter(xp, fp, marker="x", color="r")
154 plt.plot(xx, ff5, label='5 puntos')
155 plt.plot(xx, ff11, label='11 puntos')
156 plt.plot(xx, ff21, label='21 puntos')
157 plt.legend(loc='best')
158 plt.grid()
159 plt.ylim(0-0.05, 1+0.05)
160 plt.xlim(xmin-0.05, xmax+0.05)
161 plt.subplot(212)
162 plt.ylabel('error')
163 plt.xlabel('x')
164 plt.grid()
165 #plt.legend(loc='best')
166 plt.title('Error en la aproximacion de Lagrange de 21 puntos')
167 plt.plot(xx, e)
168
169 plt.xlim(xmin-0.05, xmax+0.05)
170 plt.savefig("graficos/p11-lagrange-equiespaciados.png")
171 plt.show()
172
173
174 plt.figure( figsize=(10, 7.5) )
175 plt.title('Diferencia entre polinomio de Lagrange y de Newton (
    equiespaciados)')
176 plt.xlabel('x')
```

```
plt.ylabel('diferencia')
plt.grid()
plt.semilogy(xx , abs(ff21-nn21), label='plagrange 21p - pnewton 21p')
plt.legend(loc="best")
plt.savefig('graficos/p11-diferencia-equiespaciados.png', dpi=100)
plt.show()


#####################################################################
# va de nuevo con espaciamiento de Chebyshev

points = "cheby"

# ------------------------------------------------
npts = 5

# xp, fp  son los puntos de donde se construye la Interpolación
xp = puntos_interpolacion(npts, xmin, xmax, points)
fp = f(xp)

# xx grilla sonde se calculará la Interpolación
# ff es el valor de la Interpolación
xx = np.linspace(xmin, xmax, neval)
ff5 = np.zeros(len(xx))

for n in range(len(xx)):
    ff5[n] = ilagrange(xx[n], xp, fp)

nn5 = np.zeros(len(xx))

for n in range(len(xx)):
    nn5[n] = inewton(xx[n], xp, fp)

# ------------------------------------------------
npts = 11

# xp, fp  son los puntos de donde se construye la Interpolación
xp = puntos_interpolacion(npts, xmin, xmax, points)
fp = f(xp)

# xx grilla sonde se calculará la Interpolación
# ff es el valor de la Interpolación
xx = np.linspace(xmin, xmax, neval)
ff11 = np.zeros(len(xx))

for n in range(len(xx)):
    ff11[n] = ilagrange(xx[n], xp, fp)

nn11 = np.zeros(len(xx))

for n in range(len(xx)):
    nn11[n] = inewton(xx[n], xp, fp)

# ------------------------------------------------
npts = 21

```

```
# xp, fp  son los puntos de donde se construye la Interpolación
xp = puntos_interpolacion(npts, xmin, xmax, points)
fp = f(xp)

# xx grilla sonde se calculará la Interpolación
# ff es el valor de la Interpolación
xx = np.linspace(xmin, xmax, neval)
ff21 = np.zeros(len(xx))

for n in range(len(xx)):
    ff21[n] = ilagrange(xx[n], xp, fp)

nn21 = np.zeros(len(xx))

for n in range(len(xx)):
    nn21[n] = inewton(xx[n], xp, fp)

fexact = f(xx)

# error
e = fexact-ff21

plt.figure(figsize=(10,8))
plt.subplot(211)
plt.ylabel('funciones')
plt.title('Aproximacion de Lagrange, puntos de Chebyshev')
plt.plot(xx, fexact, color="b")
plt.scatter(xp, fp, marker="x", color="r")
plt.plot(xx, ff5, label='5 puntos')
plt.plot(xx, ff11, label='11 puntos')
plt.plot(xx, ff21, label='21 puntos')
plt.legend(loc='best')
plt.grid()
plt.xlim(xmin-0.05, xmax+0.05)
plt.subplot(212)
plt.ylabel('error')
plt.xlabel('x')
plt.title('Error en la aproximacion de Lagrange de 21 puntos')
plt.plot(xx, e)
#plt.legend(loc='best')
plt.grid()
plt.xlim(xmin-0.05, xmax+0.05)
plt.savefig("graficos/p11-lagrange-cheby.png")
plt.show()


plt.figure( figsize=(10, 7.5) )
plt.title('Diferencia entre polinomio de Lagrange y de Newton (cheby)')
plt.xlabel('x')
plt.ylabel('diferencia')
plt.grid()
plt.semilogy(xx , abs(ff21-nn21), label='plagrange 21p - pnewton 21p')
plt.legend(loc="best")
plt.savefig('graficos/p11-diferencia-cheby.png', dpi=100)
plt.show()
```

- Desde la terminal ejecute:

  python3 p11.py

  e interprete el resultado.

  Alternativamente ejecute:

  python3

  y vaya agregando uno a uno los bloques del programa.

Obtendrá algo como esto:

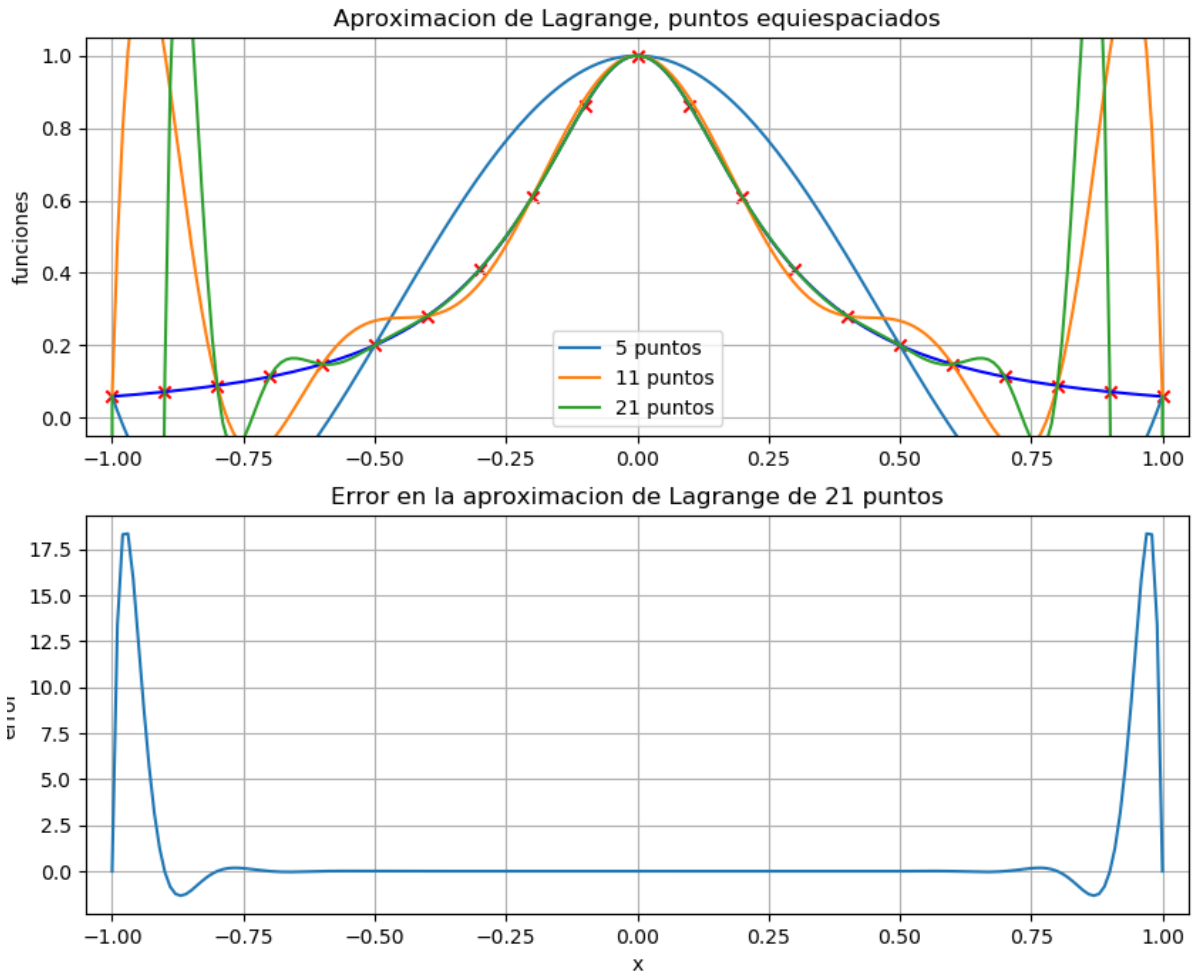

Figura 1: Funciones y errores.

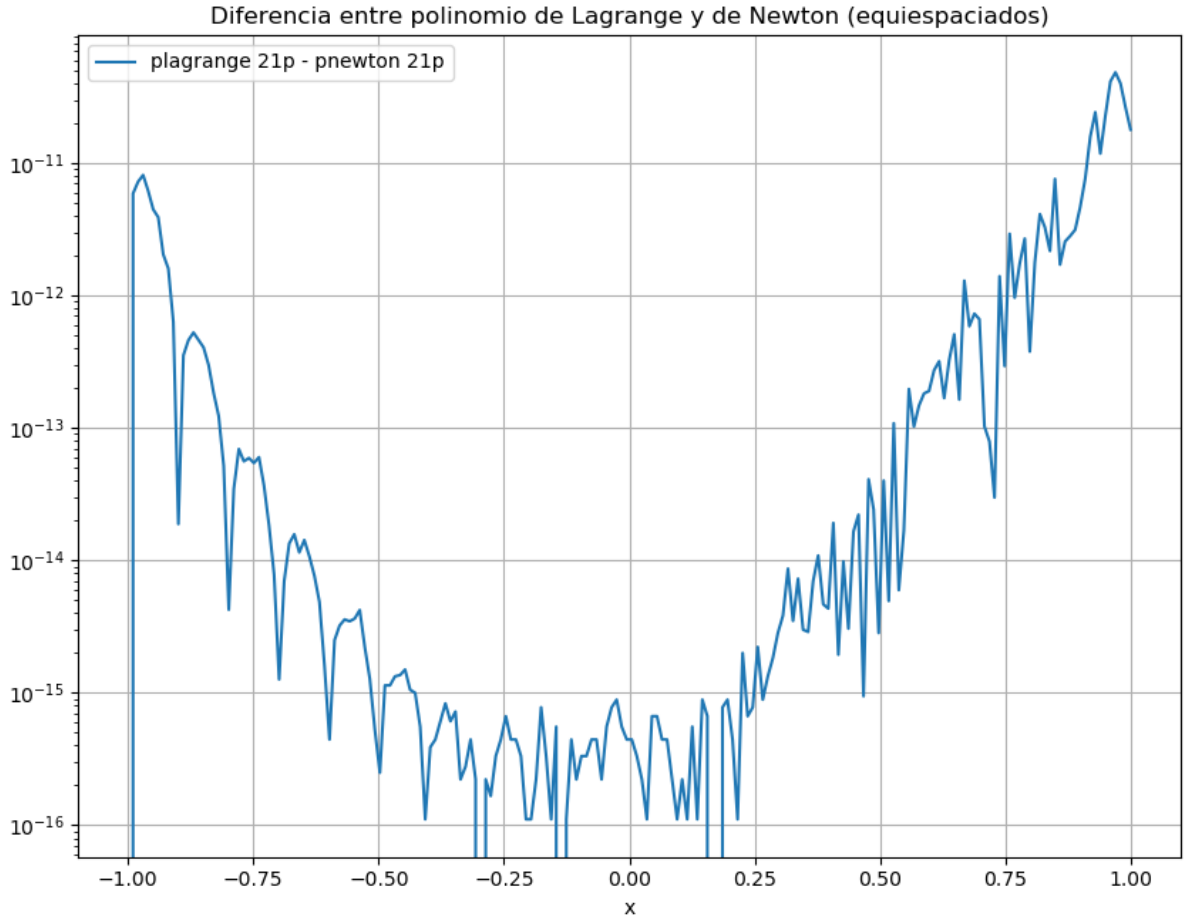

Figura 2: Diferencia entre las dos evaluaciones del polinomio con 21 puntos.

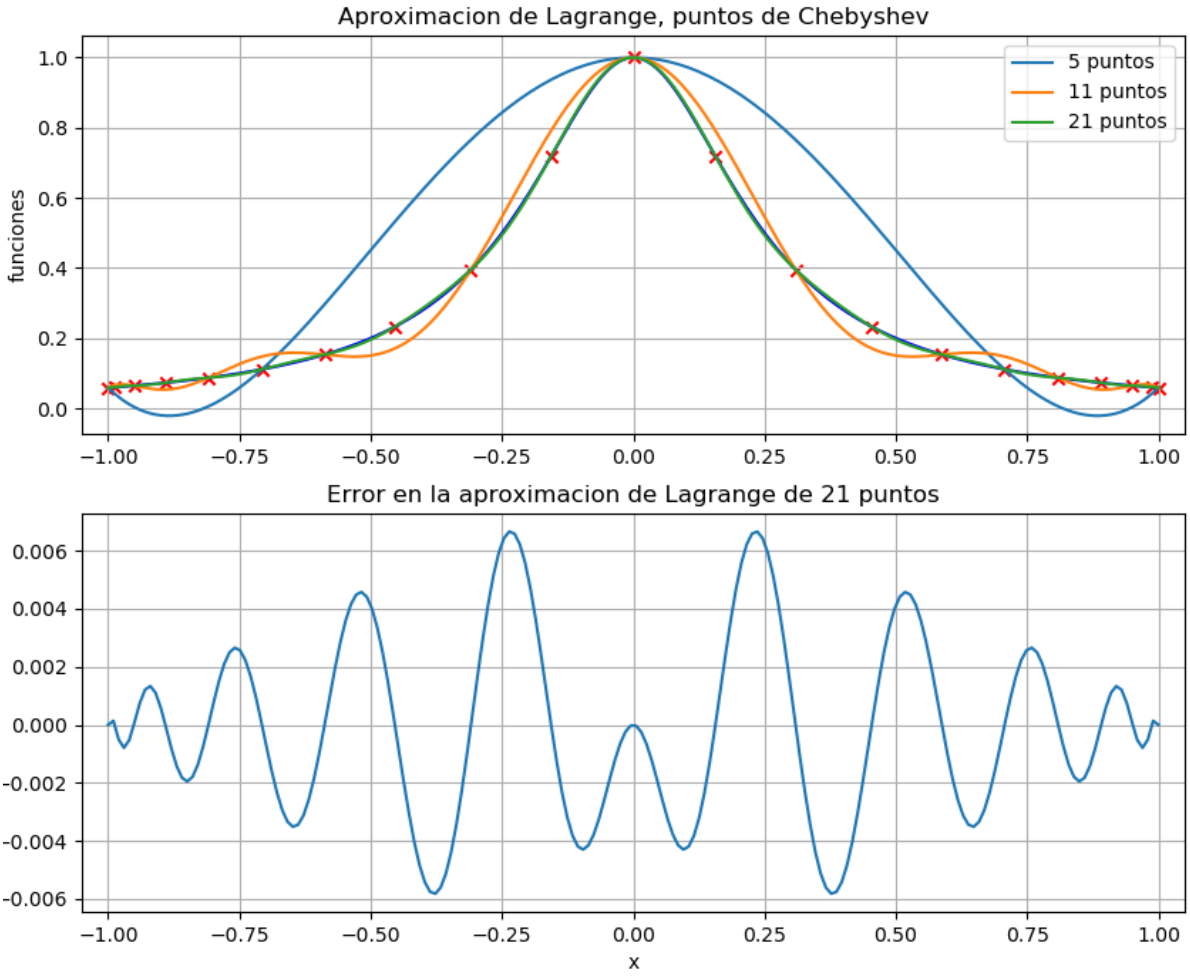

Figura 3: Funciones y errores.

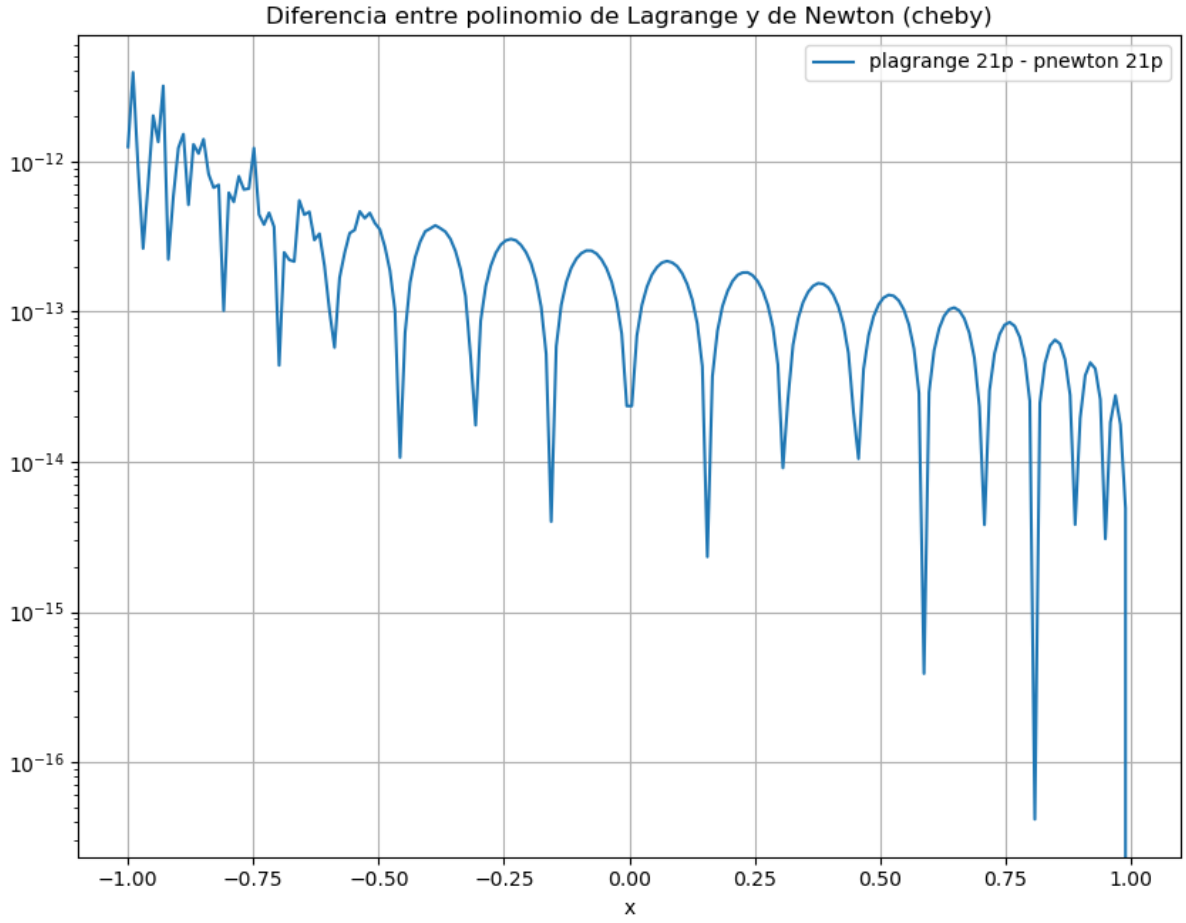

Figura 4: Diferencia entre las dos evaluaciones del polinomio con 21 puntos.