UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Taller 4: Polinomios de Chebyshev

Néstor Heli Aponte Ávila
Métodos Númericos
Cód. 20182167052
2022 - I

July 7, 2022

1 Ejercicios

1. Realice las modificaciones pertinentes al programa socializado para que compile correctamente en cualquier intervalo $[a, b]$ con $a < b$.

Construire el programa desde cero basandome en las siguientes fórmulas:

.

Nodos. los cuales provienen de la sustitución $x = \left(\frac{b-a}{2}\right)t + \frac{a+b}{2}$ y también $t = 2\frac{x-a}{b-a} - 1$

$$x_k = \frac{b-a}{2}\cos\left(\frac{(2k+1)\pi}{2N+2}\right) + \frac{a+b}{2} \quad k = 0, 1, \cdots, N \tag{1}$$

Aproximación de Chebyshev.

$$P_N(x) = \sum_{j=0}^{N} c_j T_j\left(2\frac{x-a}{b-a} - 1\right) \tag{2}$$

donde

$$c_0 = \frac{1}{N+1}\sum_{k=0}^{N} f(x_k) \quad y \quad c_j = \frac{2}{N+1}\sum_{k=0}^{N} f(x_k)T_j(x_k) \;\; j = 1, 2, \cdots, N \tag{3}$$

```
from cmath import cos, e, exp, pi, sin # Libreria de la que se sacan cositas
    matematicas como funciones, numeros especiales, etc.
import numpy as np # Esencial para trabajar matematicas
import sympy as sym # Tambien trae muchas funciones matematicas
import matplotlib.pyplot as plt # Libreria para construir las graficas
from lagrangeinter import interpol

# PREVIOUSLY
x = sym.Symbol('x') # Declara 'x' como variable simbolica

# INPUT
```

```python
fun = 1 / (1+x**2)# Funcion que se desea aproximar
n = 20 # Grado del polinomio de aproximacion
a = -5 # Extremo inferior del intervalo
b = 5 # Extremo superior del intervalo
evalf = sym.lambdify(x, fun, 'numpy') # evalf() evalua la funcion fun

# OBTENCION DE LOS NODOS DE Chebyshev
X = [] # Vector que recoge los nodos
for i in range(n+1): # Recorrido
    X.insert(0, (b-a)/2*cos(((2*i+1)*pi)/(2*n+2))+(a+b)/2)# Calculo con la sustitucion a bordo y ordenados de una vez porque .sort() no me sirvio

# Evaluacion de la funcion en los nodos
Y = evalf(np.array(X))

# OBTENCION DE LOS COEFICIENTES c_j DE Chebyshev
C = [(1/(n+1))*sum(Y)] # c_0 primera iteracion
for j in range(n):
    c = 0 # Variable auxiliar para recoger las sumatorias
    for k in range(n+1):
        c += (2/(n+1))*evalf(X[k])*cos((j+1)*np.arccos(2*((X[k]-a)/(b-a))-1)) # Formula, usando la sustitucion y tambien la construccion alternativa de T_j(x) = cos(jarccos(x))
    C.append(c) # guarda lo calculado

# CALCULO DE LOS POLINOMIOS T_j(x)
T = [1, x] # Vector que recoge los polinomios T_j(x) con la construccion natural
for i in range(n-1): # Recorrido hasta el polinomio que necesito
    T.append(2*x*T[-1]-T[-2]) # Formulilla

# OBTENCION DEL POLINOMIO DE Chebyshev
P = [] # Vector recogedor
for i in range(n+1): # Recorrido
    P.append(0) # Anado 0's para poder acceder a esas posiciones en la iteracion
    if i != 0: # T(0) = 1 (No lo puedo evaluar en x)
        T[i] = T[i].subs(x, 2*((x-a)/(b-a))-1) # Calculo los polinomios con la sustitucion
    else:
        P[i] = C[i]
    P[i] = C[i]*T[i] # Formulilla

polinomio = sum(P).expand() # .sum() suma todo en el vector P y .expand() simplifica mi polinomio

# OUTPUT
print(f"Nodos: \n{X} \nImagen en esos nodos: \n{Y} \nPolinomio Interpolador: \n{polinomio}")
```

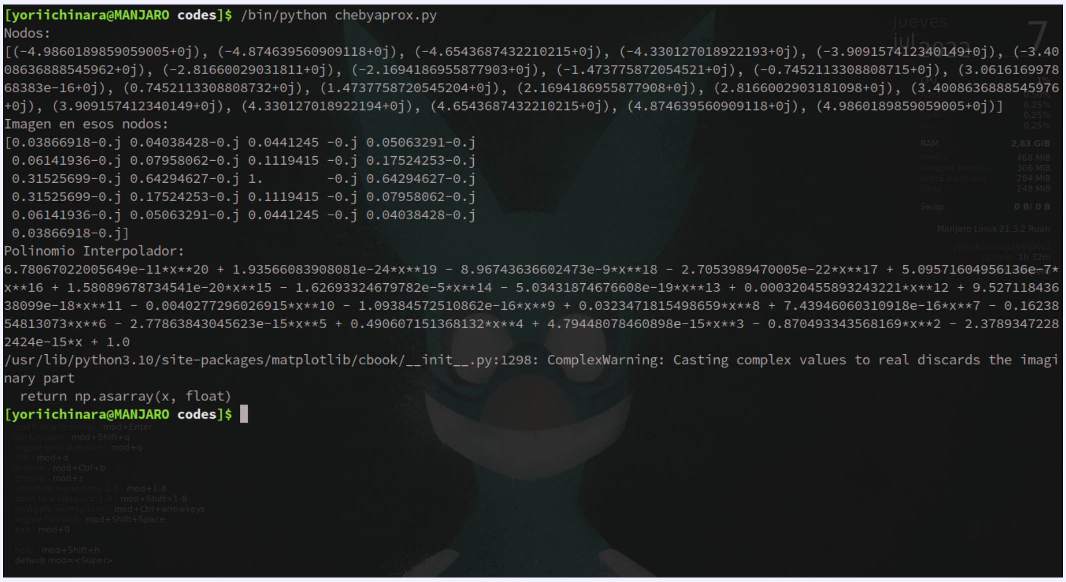
Listing 1: Aproximación de Chebyshev

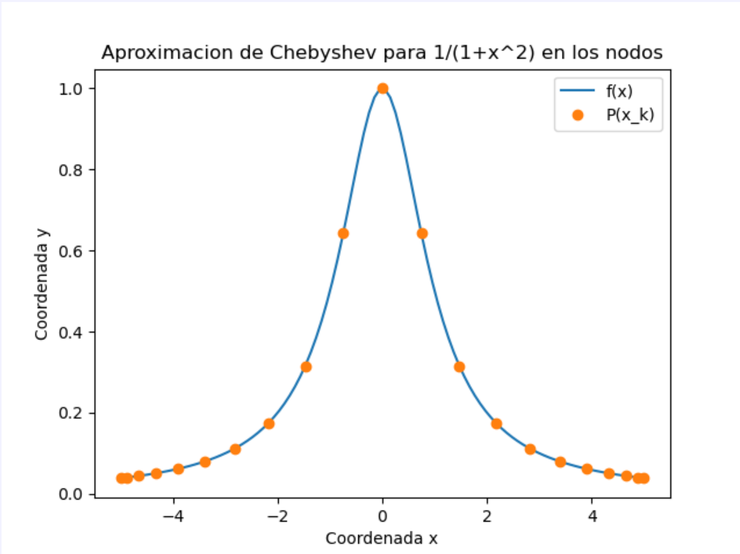
2. Grafique la función propuesta en clase en el intervalo $[-5, 5]$ y sobre esta el polinomio de grado 20 evaluado en los nodos. Adjunte la salida del programa. Agregue nombres a los ejes y título a la gráfica.

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

```python

# INPUT
fun = 1 / (1+x**2)# Funcion que se desea aproximar
n = 20 # Grado del polinomio de aproximacion
a = -5 # Extremo inferior del intervalo
b = 5 # Extremo superior del intervalo

#GRAPHICS
evalp = sym.lambdify(x, polinomio, 'numpy') # evalp() evalua en el polinomio interpolador
dom = np.linspace(start = a, stop = b, num = 100) # linspace() me ayuda a crear el dominio para los graficos
plt.plot(dom, evalf(dom), X, evalp(np.array(X)),"o") # Graficos de la funcion y el polinomio de aproximacion evaluado en los nodos
plt.legend(["f(x)", "P(x_k)"]) # Convenciones
plt.xlabel("Coordenada x") # Indicacion eje horizontal
```

```
plt.ylabel("Coordenada y") # Indicacion eje vertical
plt.title("Aproximacion de Chebyshev para 1/(1+x^2) en los nodos") # Titulo
plt.show() # Lanza la interfaz para la visualizacion del grafico

```

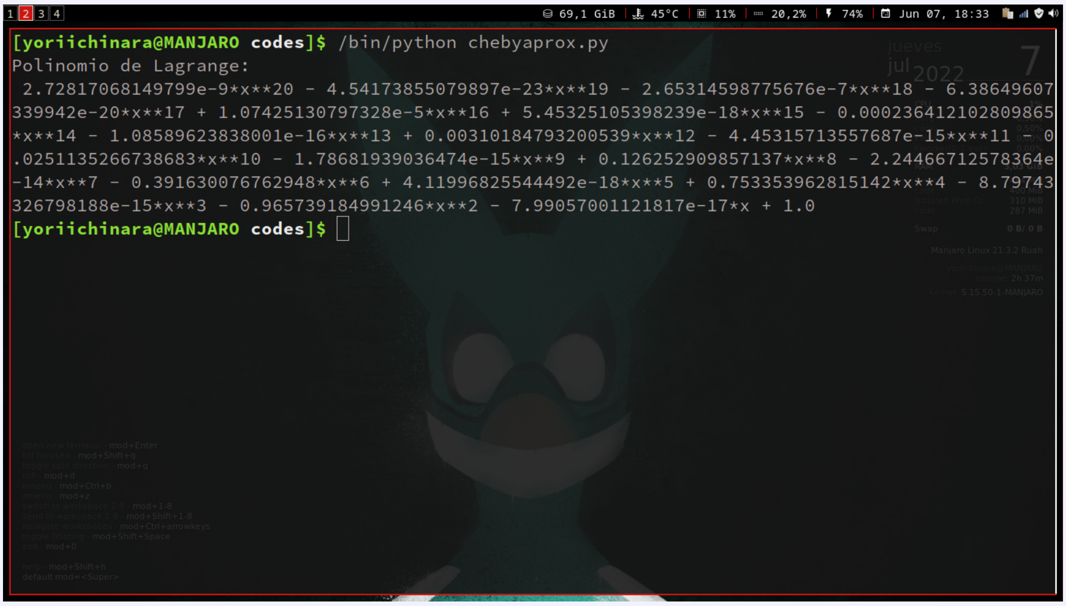




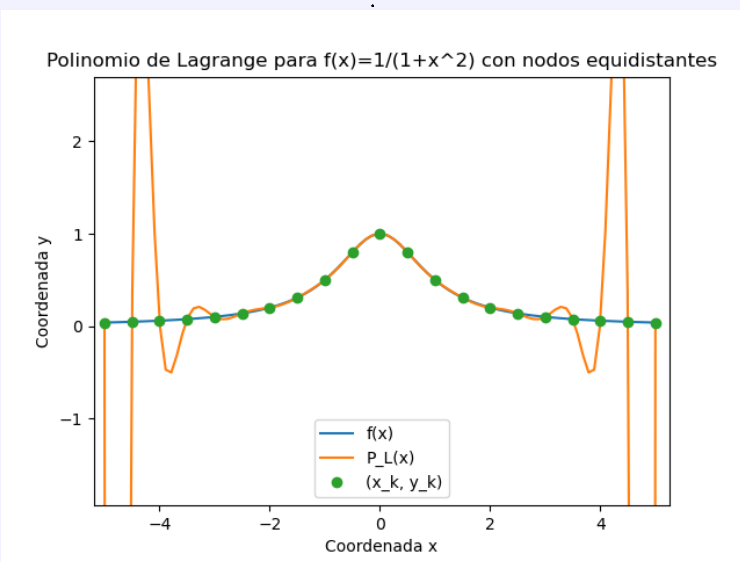
3. Con la misma función y misma cantidad de nodos pero tomados de manera que sean equidistantes aproxime la función con el polinomio P de Lagrange.

```

from lagrangeinter import interpol

# Usando la funcion con lagrange
X1 = np.linspace(start=-5, stop=5, num=21) # Crea el vector de nodos equiespaciados (21 porque necesito que el polinomio sea de grado 20)
lagpolinomio = interpol(X1, evalf(np.array(X1))) # Usa la funcion
evalplag = sym.lambdify(x, lagpolinomio, 'numpy') # evalplag() para evaluar en el polinomio de Lagrange
print("Polinomio de Lagrange:\n", lagpolinomio)

# Grafica para visualizar
plt.plot(dom, evalf(dom), dom, evalplag(dom), X1, evalplag(np.array(X1)),"o") # Graficos de la funcion y el polinomio de lagrange en dom
plt.legend(["f(x)", "P_L(x)", "(x_k,y_k)"]) # Convenciones
plt.xlabel("Coordenada x") # Indicacion eje horizontal
```

```
plt.ylabel("Coordenada y") # Indicacion eje vertical
plt.title("Interpolacion de Lagrange para f(x)=1/(1+x^2) con nodos equidistantes")
plt.show() # Lanza la interfaz para la visualizacion del grafico

```



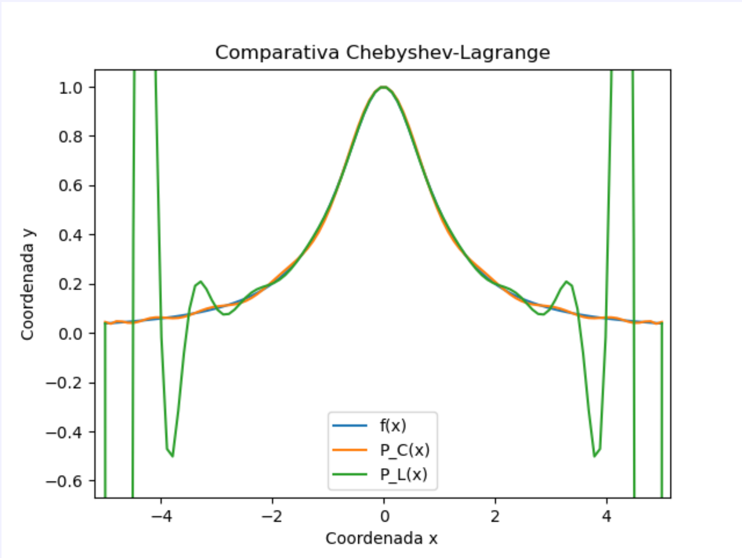
.



4. Ahora use los nodos de Chebyshev y halle el polinomio de grado 20 que aproxime la función. Escriba una conclusión de los puntos 3 y 4.

```

plt.plot(dom, evalf(dom), dom, evalp(dom), dom, evalplag(dom)) # Graficos de la
    funcion y el polinomio en dominio dom
plt.legend(["f(x)", "P_C(x)", "P_L(x)"]) # Convenciones
plt.xlabel("Coordenada x") # Indicacion eje horizontal
plt.ylabel("Coordenada y") # Indicacion eje vertical
plt.title("Comparativa Chebyshev-Lagrange") # Titulo
plt.show() # Lanza la interfaz para la visualizacion del grafico

```

Ya lo tenía así que construí el gráfico para comparar.



Reflexión. El error usando Chebyshev es practicamente inperceptible, mientras que Lagrange lo dispara en los extremos. Esto deja ver que cuando de funciones específicas se trata, donde tengo cierta autonomia para escoger los puntos muestra, es mejor invocar al amigo Chebyshev.

References

[1] Jhon H. Mathews - Kurtis D. Fink, *Métodos Númericos con MATLAB*, Pretince Hall, (2000)