UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Taller 3: Polinomios de Lagrange

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

June 29, 2022

1 Ejercicios

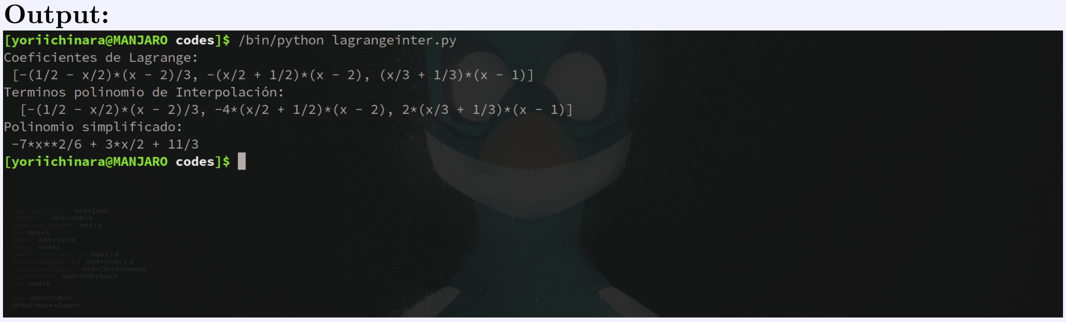
1. Escriba un código para cálcular los polinomios coeficientes de Lagrange y los coeficientes del polinomio de interpolación P_n. Recuerde comentar cuales son las variables de entrada, las variables de salida y el significado de las funciones usada. Verifique el programa con el ejercicio realizado en clase.

```python
# LIBRERIAS REQUERIDAS
import numpy as np # Libreria para expresiones y cosas matematicas
import sympy as sym # Para usar variables simbolicas y poder evaluar polinomios y funciones
import matplotlib.pyplot as plt # Libreria que me permite realizar plots

def interpol(X, Y):
    # INPUT
    # X ---> Vector que contiene las abscisas de entrada
    # Y ---> Ordenadas para las abscisas anteriores

    # CONSIDERACIONES PREVIAS
    x = sym.Symbol('x') # Consulte y la funcion .Symbol() de la libreria sym me sirve para indicar que x es una variable simbolica
    L = [] # Vector que recoge los coeficientes de Lagrange
    P = [] # Vector que recoge los terminos de mi Polinomio de Interpolacion

    # CONSTRUCCION
    for i in range(len(X)): # Recorrido para cada coeficiente del polinomio .len() devuelve la dimension del vector X
        L.append(1) # Agrego 1's para acceder a ellos en la posicion i, pongo 1 para facilitar la manipulacion numerica (multiplicaciones)

        # COEFICIENTE DE LAGRANGE
        for j in range(len(X)): # Recorrido por cada abscisa x_j
            if i != j: # Condicional en la formula i distinto de j
                L[i] = L[i]*(x - X[j]) / (X[i]-X[j]) # Contruye el factorial

        # COEFICIENTE DEL POLINOMIO DE INTERPOLACION
        P.append(0) # De nuevo agrego 0's para poder acceder a esas posiciones con la iteracion, sino Python no me deja
        P[i] = L[i] * Y[i] # Formulilla
```

```python

    # CONSOLIDACION DE RESULTADOS
    polinomio = sum(P).expand() # sum() suma todos los elementos de P (terminos de mi polinomio) y .expand() simplifica la expresion
    eval = sym.lambdify(x,polinomio) # Consultado, funcion .lambdify() de la libreria sym, Ahora eval() me permite evaluar esa expresion simbolica 'polinomio'

    # CONSTRUCCION DE LA GRAFICA
    dom = np.linspace(start = X[0], stop=X[-1],num = 100) # Dominio de la grafica .linspace() de la libreria np construye un vector equidistante entre los valores el primer y ultimo valor de X, en este casos de 100 numeros
    plt.plot(dom,eval(dom)) + plt.plot(X, Y, "o") # plot del polinomio evaluado en el dominio + plot de los puntos del INPUT (X e Y)
    plt.show() # Lanza la interfaz para visualizar la grafica

    # OUTPUT
    print("Coeficientes de Lagrange: \n",
        L,
        "\nTerminos polinomio de Interpolacion: \n ",
        P,
        "\nPolinomio simplificado: \n",
        polinomio)


# Usando la funcion
interpol(
  [-1, 1, 2],
  [1, 4, 2]
)
```

Listing 1: Interpolación de Lagrange

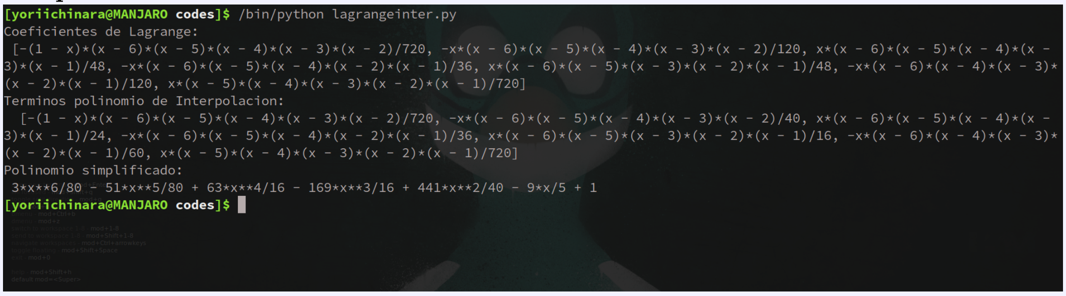
Output:



2. Usar el programa del primer punto para hallar el polinomio interpolador de Lagrange P_6 que contenga los siguientes puntos (0,1), (1,3), (2,2), (3,1), (4,3), (5,2), (6,1). Grafique el polinomio en el intervalo [0,6]. Recuerde adjuntar las salidas del programa.

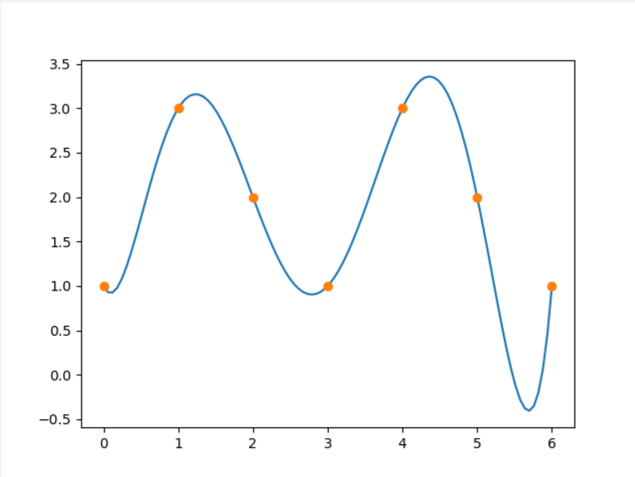
```python

interpol(
  [0, 1, 2, 3, 4, 5, 6],
  [1, 3, 2, 1, 3, 2, 1]
)

```

Output:



```
[yoriichinara@MANJARO codes]$ /bin/python lagrangeinter.py
Coeficientes de Lagrange:
 [-(1 - x)*(x - 6)*(x - 5)*(x - 4)*(x - 3)*(x - 2)/720, -x*(x - 6)*(x - 5)*(x - 4)*(x - 3)*(x - 2)/120, x*(x - 6)*(x - 5)*(x - 4)*(x - 3)*(x - 1)/48, -x*(x - 6)*(x - 5)*(x - 4)*(x - 2)*(x - 1)/36, x*(x - 6)*(x - 5)*(x - 3)*(x - 2)*(x - 1)/48, -x*(x - 6)*(x - 4)*(x - 3)*(x - 2)*(x - 1)/120, x*(x - 5)*(x - 4)*(x - 3)*(x - 2)*(x - 1)/720]
Terminos polinomio de Interpolacion:
  [-(1 - x)*(x - 6)*(x - 5)*(x - 4)*(x - 3)*(x - 2)/720, -x*(x - 6)*(x - 5)*(x - 4)*(x - 3)*(x - 2)/40, x*(x - 6)*(x - 5)*(x - 4)*(x - 3)*(x - 1)/24, -x*(x - 6)*(x - 5)*(x - 4)*(x - 2)*(x - 1)/36, x*(x - 6)*(x - 5)*(x - 3)*(x - 2)*(x - 1)/16, -x*(x - 6)*(x - 4)*(x - 3)*(x - 2)*(x - 1)/60, x*(x - 5)*(x - 4)*(x - 3)*(x - 2)*(x - 1)/720]
Polinomio simplificado:
 3*x**6/80 - 51*x**5/80 + 63*x**4/16 - 169*x**3/16 + 441*x**2/40 - 9*x/5 + 1
[yoriichinara@MANJARO codes]$
```

$$P_6(x) = \frac{3}{80}x^6 - \frac{51}{80}x^5 + \frac{63}{16}x^4 - \frac{169}{16}x^3 + \frac{441}{40}x^2 - \frac{9}{5}x + 1$$

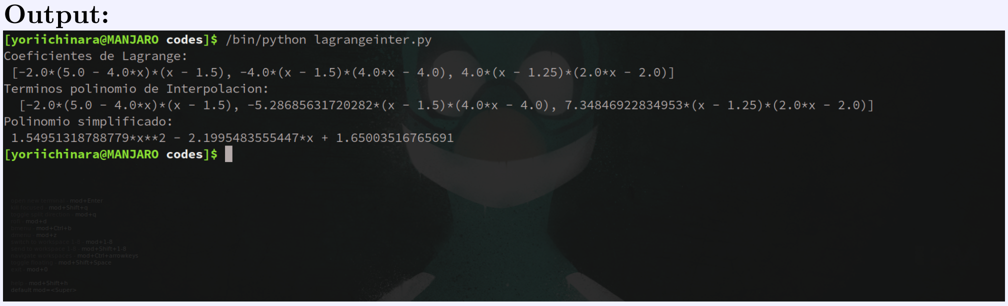


3. Sea $f(x) = x^x$

- Determine el polinomio interpolador de Lagrange cuadrático para $x_0 = 1$, $x_1 = 1.25$, $x_2 = 1.5$

```

interpol(
  [1, 1.25, 1.5],
  [1, 1.25**1.25, 1.5**1.5]
)

```

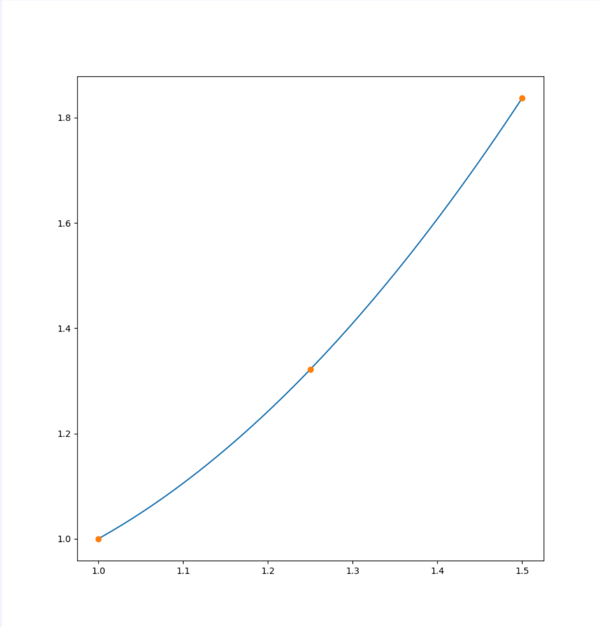
Output:



```
[yoriichinara@MANJARO codes]$ /bin/python lagrangeinter.py
Coeficientes de Lagrange:
 [-2.0*(5.0 - 4.0*x)*(x - 1.5), -4.0*(x - 1.5)*(4.0*x - 4.0), 4.0*(x - 1.25)*(2.0*x - 2.0)]
Terminos polinomio de Interpolacion:
  [-2.0*(5.0 - 4.0*x)*(x - 1.5), -5.28685631720282*(x - 1.5)*(4.0*x - 4.0), 7.34846922834953*(x - 1.25)*(2.0*x - 2.0)]
Polinomio simplificado:
 1.54951318788779*x**2 - 2.19954835555447*x + 1.65003516765691
[yoriichinara@MANJARO codes]$
```

$$P_2(x) = 1.54951318788779x^2 - 2.19954835555447x + 1.65003516765691$$

- Use el polinomio interpolador para estimar $f(x)$ en el intervalo [1,1.5]



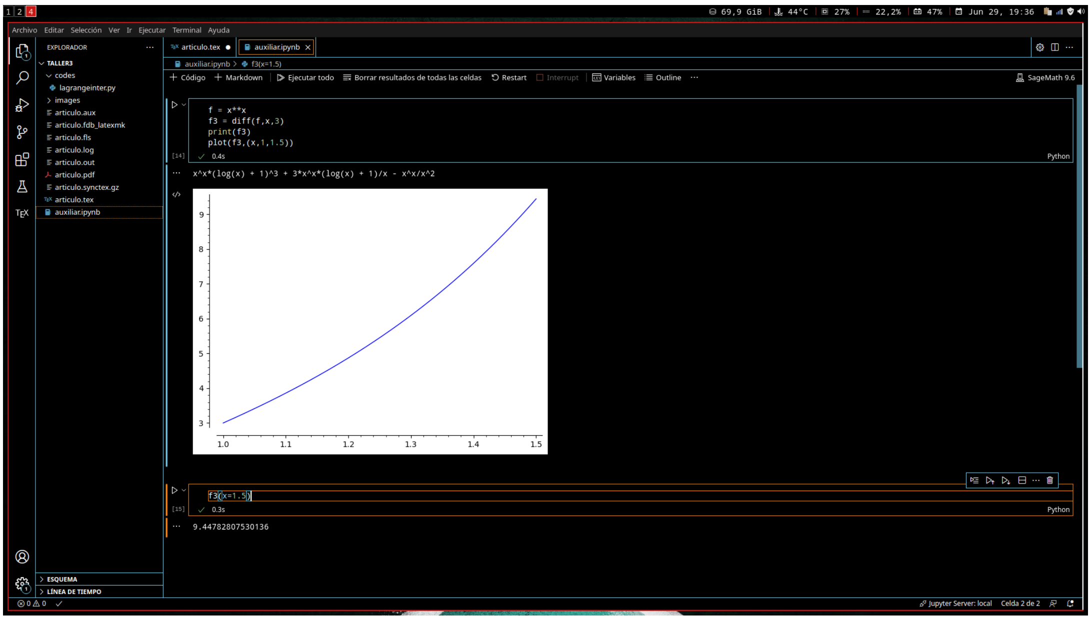
- Halle una cota del error para $|f(1.5) - P_2(1.5)|$

$$|f(x) - P_n(x)| \leq \left| \frac{M_{n+1}}{(n+1)!} \prod_{j=0}^{n} (x - x_j) \right|$$

$n+1$ son los puntos muestra, en este caso 3, por lo que me queda

$$\text{Error} \leq \left| \frac{M_3}{3!} \prod_{j=0}^{n} (x - x_j) \right|$$

$M_3 = \max_{\xi \in [1,1.5]} \left| f'''(\xi) \right|$. Realizare los calculos en SageMath.



Nuestra 3ra derivada:

$$f^{'''}(x) = x^x(\ln(x) + 1)^3 + 3x^{x-1}\ln(x) + 3x^{x-1} - x^{x-2}$$

Y el gráfico deja ver que el máximo está en el extremo derecho, por lo que $M_3 = f^{'''}(1.5) = 9.44782807530136$, seguidamente...

$$\text{Error} \leq \left| \frac{9.44782807530136}{6}(x-1)(x-1.25)(x-1.5) \right|$$

** La cota en 1.5 es cero (o deberia al menos, en la teoria) pues es uno de los puntos muestra que estoy tomando y por ende uno de los 0's de mi polinomio error. Si realizo en cálculo duro, no da 0 pero casi, es por la perdida de cifras cuando se evaluan los exponenciales tanto al calcular el error como al ingresar el dato para calcular el polinomio.

```

print(abs(1.5**1.5-eval(1.5)))

```

Output: 3.552713678800501e-15

References

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

[2] ESPOL, *Métodos Númericos - Curso con Python*, Guayaquil - Ecuador, (link)