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 10 de la Guía N° 4

Problema 10:

Interpolación de Newton. Escriba un script de PYTHON similar al anterior, que defina una función `inewton`, que implemente la interpolación en forma de Newton. Pruebe su script realizando la interpolación de la función $\sin(x)$, evaluando en los puntos $[0, \frac{\pi}{4}, \frac{2\pi}{4}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{6\pi}{4}, \frac{7\pi}{4}, 2\pi]$. Calcule el valor del polinomio interpolatorio en el punto $x = \frac{\pi}{6}$.

Tutorial:

- Guarde en el archivo p10.py las siguientes instrucciones:

```python
import numpy as np
import matplotlib.pyplot as plt

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 = np.copy(y)
    for k in range(1, m):
        a[k:m] = (a[k:m] - a[k - 1])/(x[k:m] - x[k - 1])
    return a

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

xmin = 0.0
xmax = 2.0*np.pi

N = 9
```

```
xp = np.linspace(xmin, xmax, N)
fp = np.sin(xp)

xx = np.linspace(xmin, xmax, 200)
ff = np.zeros(len(xx))

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

fexact = np.sin(xx)

# error
e = fexact-ff

plt.figure( figsize=(12, 8))
plt.subplot(211)
plt.title('sin(x) y polinomio de Newton')
plt.scatter(xp, fp, marker="x", color="r", s=30, label='puntos')
plt.plot(xx, fexact, color="cyan",linewidth=3.5, label='exacta')
plt.plot(xx, ff, color="k", label='polinomio')
plt.legend(loc='best')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.xlim(xmin, xmax)
plt.axhline(y=0, color='#000000') # línea horizontal

plt.subplot(212)
plt.title('Error')
plt.plot(xx, e, label='error')
plt.legend(loc='best')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.xlim(xmin, xmax)
plt.axhline(y=0, color='#000000') # línea horizontal

plt.subplots_adjust(hspace=0.3)
plt.savefig("graficos/p10-newton.png")
plt.show()

print()
print('inewton(np.pi/6., xp, fp) =',inewton(np.pi/6., xp, fp))
print('          np.sin(np.pi/6.) =',np.sin(np.pi/6.))
print()
```

- Desde la terminal ejecute:

  python3 p10.py

  e interprete el resultado.

  Alternativamente ejecute:

  python3

  y vaya agregando uno a uno los bloques del programa.

Obtendrá algo como esto:

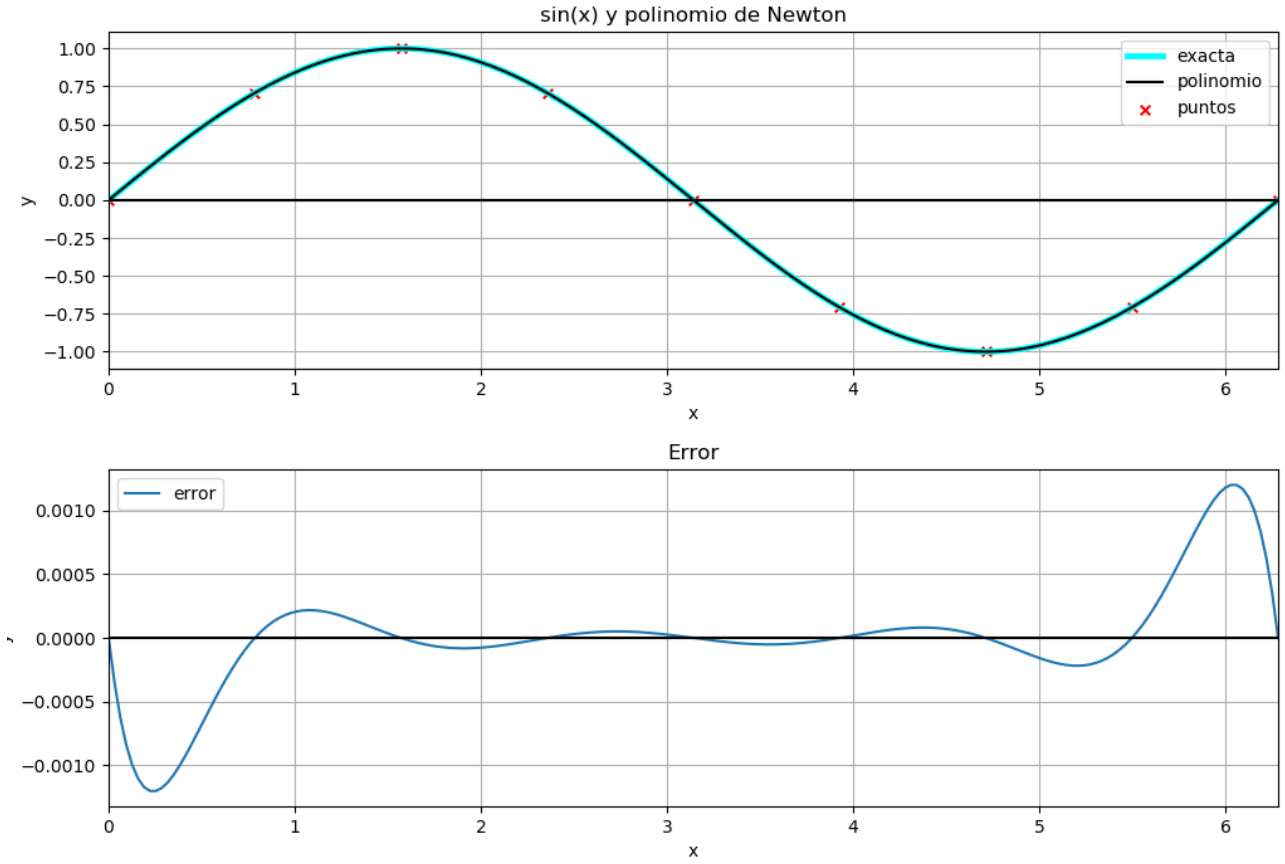

Figura 1: Funciones y error.

```
inewton(np.pi/6., xp, fp) = 0.5006139353456841
         np.sin(np.pi/6.) = 0.49999999999999994
```