Actividad 3

Andrés Ignacio Rodríguez Mendoza

Introducción

Los datos obtenidos de experimentación representan valores discretos de una función. De éstos, se requiere interpolar la función modelo para obtener los valores intermediós entre los datos originales.

En Python, la clase *interp1d* de **scipy.interpolate** es un método para crear una función basada en un conjunto de datos fijos tal que pueda ser evaluada en cualquier punto dentro del dominio definido por los datos dados usando interpolación lineal.

La interpolación lineal, generalmente toma dos puntos de datos, y calcula la pentiente entre ellos tomando su valor como el coeficiente de la variable independiente en el intervalo de esos puntos.Las desventajas de ésta son que no es muy precisa y no es diferenciable en los puntos de los datos originales.

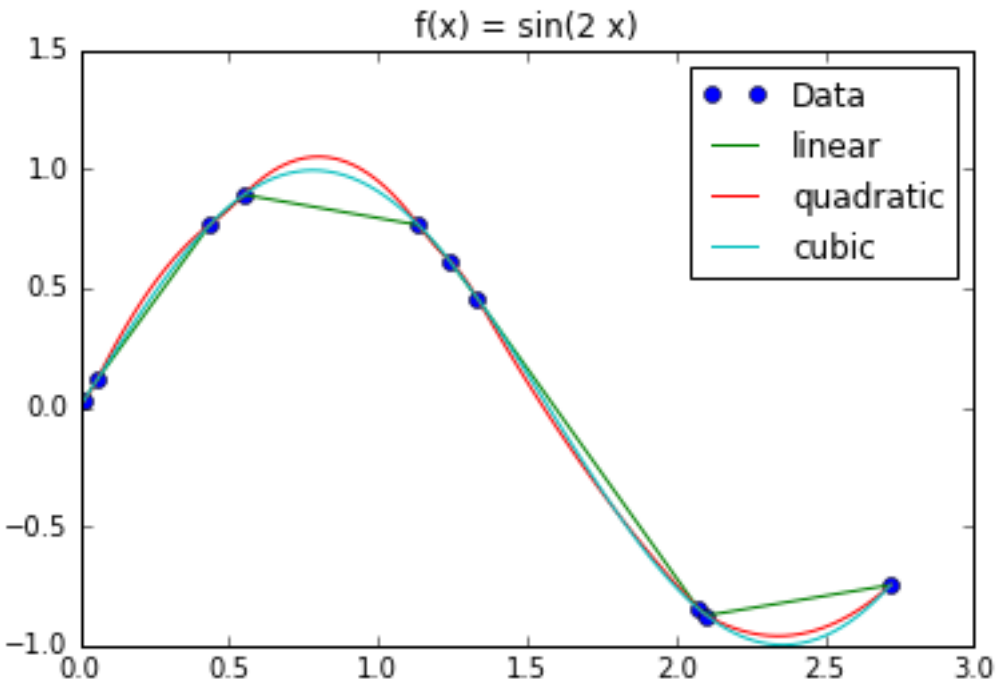
La interpolación polinomial es una generalización de la interpolación lineal. En ésta, se toma como modelo interpolante un polinomio de orden más alto que la lineal,por ser un polinomio es infinitamente diferenciable. Para obtener los coeficientes del polinomio se requiere de cálculo numérico.

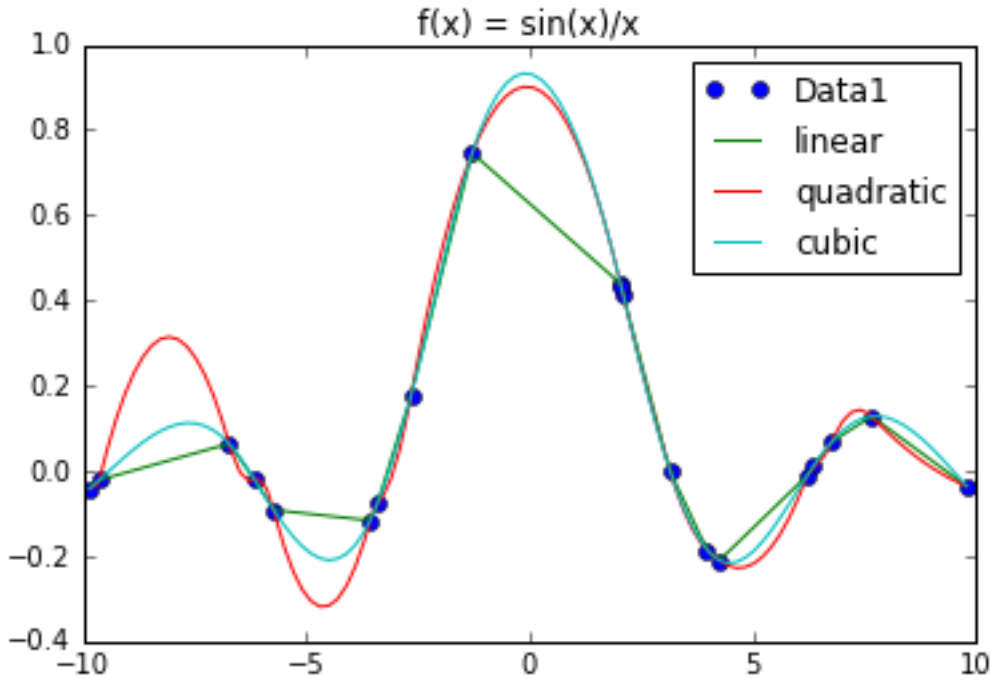
Código

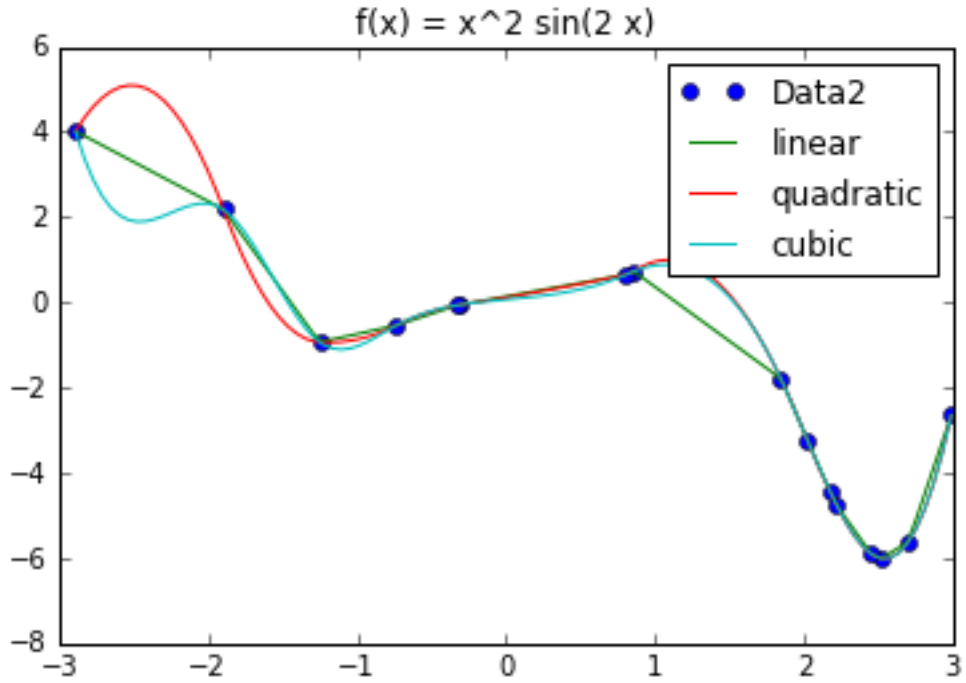
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d


#    Dados 10 puntos aleatorios entre x=0 y x=3 para la función f(x) = sin(2 x).
x0 = 3*np.random.random(10)
y0 = np.sin(2*x0)

#    Dados 20 puntos aleatorios entre x=-10 y x=10 para la función f(x) = sin(x)/x
x1 = 20*np.random.random(20) -10
y1 = np.sin(x1)/x1

#    Dados 16 puntos aleatorios entre x=-3 y x=3 para la función f(x) = x^2sin(2x)
x2 = 6*np.random.random(16) - 3
y2 = x2*x2 *np.sin(2*x2)

#    Dados 12 puntos aleatorios entre x=-2 y x=2 para la función f(x) = x^3sin(3x)
x3 = 4*np.random.random(12) -2
y3 =  x3*x3*x3*np.sin(3*x3)

# Array with points in between those of the data set for interpolation.
x_0 = np.linspace(min(x0),max(x0),100)
x_1 = np.linspace(min(x1),max(x1),100)
```

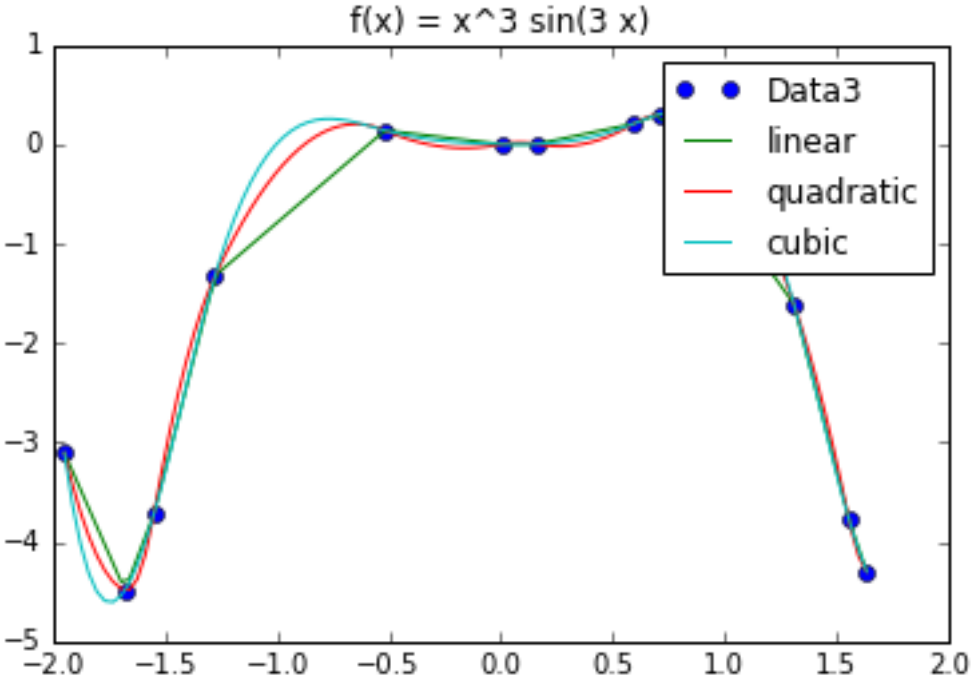
```
x_2 = np.linspace(min(x2),max(x2),100)
x_3 = np.linspace(min(x3),max(x3),100)

# Available options for interp1d
options = ('linear','quadratic','cubic')



#plot 1
plt.plot(x0, y0, 'o', label='Data')
for o in options:
    f = interp1d(x0, y0, kind=o)
    plt.plot(x_0, f(x_0), label=o)
plt.title('f(x) = sin(2 x)')
plt.legend()
plt.show()

#plot2
plt.plot(x1, y1, 'o', label='Data1')
for o in options:
    f = interp1d(x1, y1, kind=o)
    plt.plot(x_1, f(x_1), label=o)
plt.title('f(x) = sin(x)/x')
plt.legend()
plt.show()
#plot3
plt.plot(x2, y2, 'o', label='Data2')
for o in options:
    f = interp1d(x2, y2, kind=o)
    plt.plot(x_2, f(x_2), label=o)
plt.title('f(x) = x^2 sin(2 x)')
plt.legend()
plt.show()

#plot4
plt.plot(x3, y3, 'o', label='Data3')
for o in options:
    f = interp1d(x3, y3, kind=o)
    plt.plot(x_3, f(x_3), label=o)
plt.title('f(x) = x^3 sin(3 x)')
plt.legend()
plt.show()
```

Gŕaficas


f(x) = sin(2 x)
Data
linear
quadratic
cubic
1.5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0


f(x) = sin(x)/x
Data1
linear
quadratic
cubic
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−10
−5
0
5
10


f(x) = x^2 sin(2 x)
Data2
linear
quadratic
cubic
6
4
2
0
−2
−4
−6
−8
−3
−2
−1
0
1
2
3


f(x) = x^3 sin(3 x)
Data3
linear
quadratic
cubic
1
0
−1
−2
−3
−4
−5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0