Actividad 3: Interpolación con Python

Hugo de Jesús Valenzuela Chaparro

2 de abril del 2016

1. Interpolación

La interpolación sirve para obtener nuevos puntos partiendo de un conjunto discretos de puntos conocidos. Se hace encontrando una función que se ajuste y pase por dichos puntos, pudiendo así predecir los demás.

Por ejemplo, una aplicación en la física sería para las efémerides, las cuales (en astronomía y navegación celeste) nos proporciona la posición de un objeto celeste o satélite artificial en un determinado tiempo. Para saber la posición, lo que se hacía antes era utilizando tablas con intervalos de tiempos iguales, lo que se hace ahora para saber la posición, son modelos matemáticos obtenidos mediante interpolación, así a partir de datos conocidos podemos predecir y aproximar las posiciones en otro tiempo.

2. Interpolación usando Python

En Python se puede hacer interpolación usando la función SciPy interpolate, la cuál utiliza un conjunto de datos para generar ajustar una función a los puntos. Puede hacerse en una dimensión con interp1d y en dos dimensiones con interp2d; en este actividad lo hicimos con una dimensión.

3. Actividades

La actividad consistió en hacer interpolación (lineal, cuadrática y cúbica) de 4 distintos casos, en base a datos generados aleatoriamente y una función, en un dominio determinado. Para generar los números aleatorios utilizamos la función random de numpy (np.random.random) la cual nos genera números aleatorios entre 0 y 1, y ya dependiendo del intervalo que ocupemos podemos multiplicar y sumar.

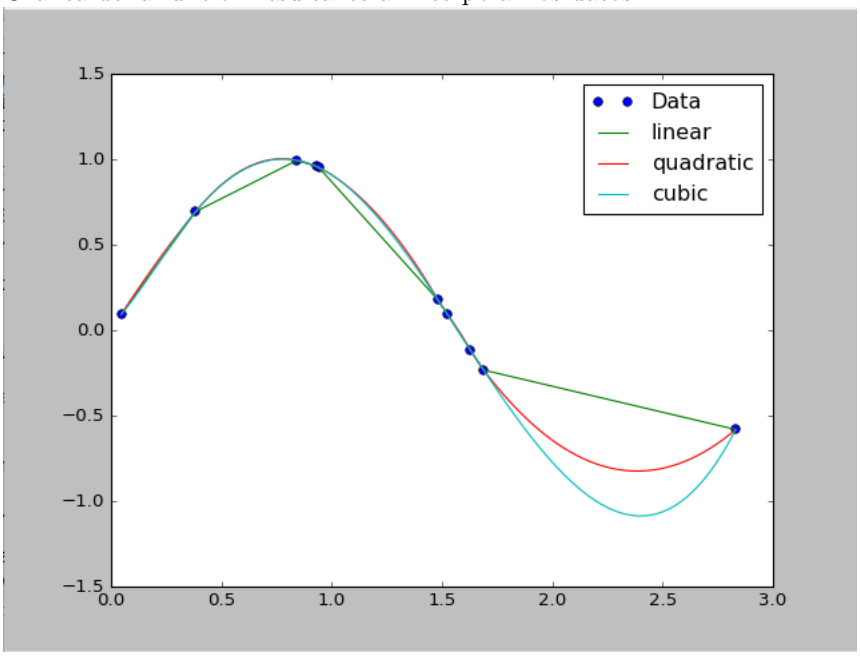
3.1. Primer caso

Dados 10 puntos aleatorios entre x=0 y x=3 para la función f(x) = sin(2 x)

A continuación el código en Python:

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

# Original "data set" --- 10 random numbers between 0 and 3.
n = np.random.random(10)
x0 = 3.0*n
y0 = np.sin(2.0*x0)

plt.plot(x0, y0, 'o', label='Data')

# Array with points in between those of the data set for interpolation.
x = np.linspace(min(x0),max(x0),101)

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

for o in options:
    f = interp1d(x0, y0, kind=o)    # interpolation function
    plt.plot(x, f(x), label=o)      # plot of interpolated data

plt.legend()
plt.show()
```

Gráfica de la función resultante al interpolar los datos:



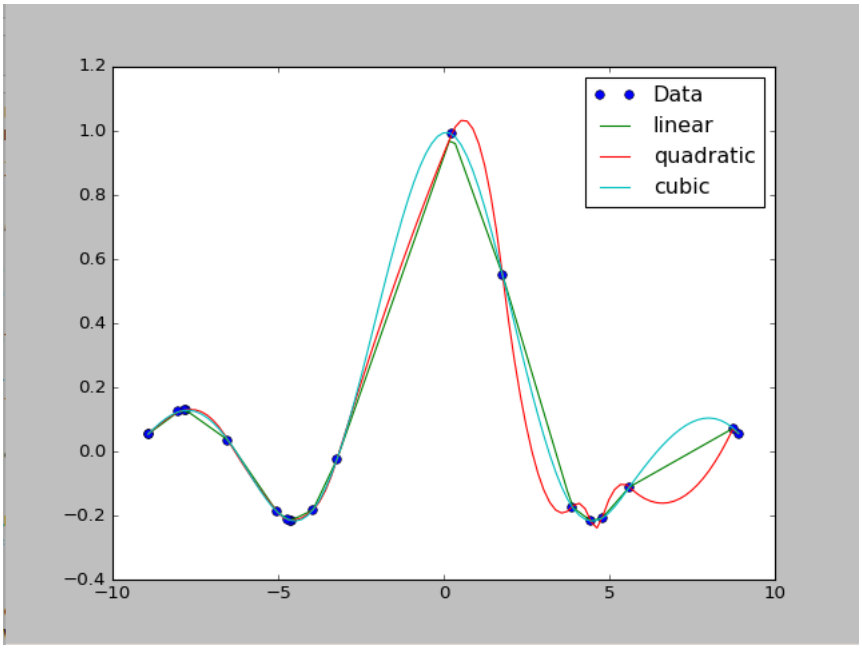
3.2. Segundo caso

Dados 20 puntos aleatorios entre x=-10 y x=10 para la función f(x) = sin(x)/x

Código:

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

# Original "data set" --- 20 random numbers between -10 and 10.
n = np.random.random(20)
x0 = 20.0*n - 10.0
y0 = np.sin(x0)/x0

plt.plot(x0, y0, 'o', label='Data')

# Array with points in between those of the data set for interpolation.
x = np.linspace(min(x0),max(x0),101)

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

for o in options:
    f = interp1d(x0, y0, kind=o)    # interpolation function
    plt.plot(x, f(x), label=o)      # plot of interpolated data

plt.legend()
plt.show()
```

Gráfica:



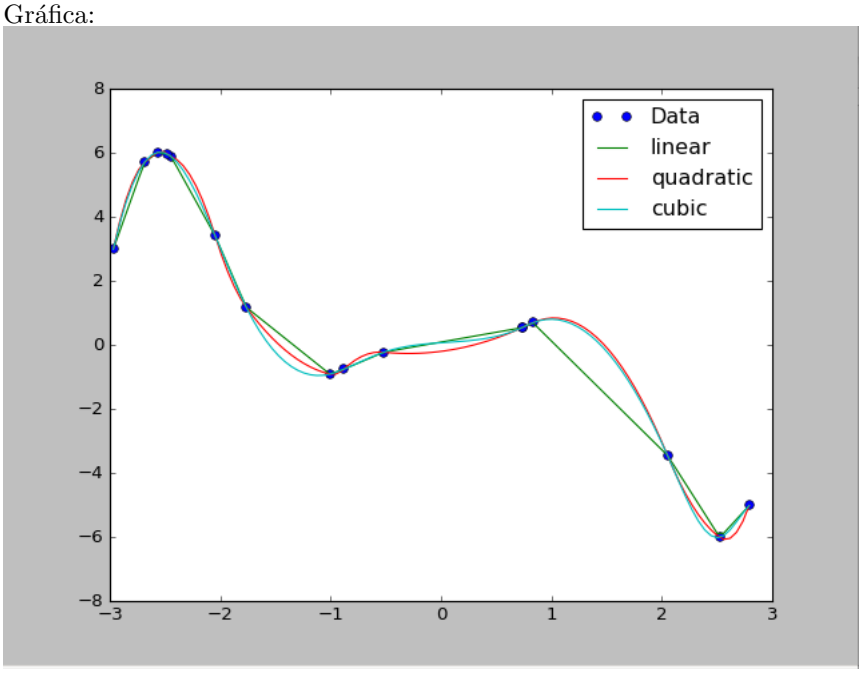
3.3. Tercer caso

Dados 16 puntos aleatorios entre x=-3 y x=3 para la función f(x) = $x^2 sin(2x)$
Código:

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

# Original "data set" --- 16 random numbers between -3 and 3.
n = np.random.random(16)
x0 = 6.0*n - 3.0
y0 = np.sin(2.0*x0)*x0*x0

plt.plot(x0, y0, 'o', label='Data')

# Array with points in between those of the data set for interpolation.
x = np.linspace(min(x0),max(x0),101)

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

for o in options:
    f = interp1d(x0, y0, kind=o)    # interpolation function
```

```
    plt.plot(x, f(x), label=o)      # plot of interpolated data

plt.legend()
plt.show()
```

Gráfica:



3.4. Cuarto caso

Dados 12 puntos aleatorios entre x=-2 y x=2 para la función f(x) = $x^3 sin(3x)$
Código:

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

# Original "data set" --- 12 random numbers between -2 and 2.
n = np.random.random(12)
x0 = 4.0*n - 2.0
y0 = np.sin(3.0*x0)*x0**(3.0)

plt.plot(x0, y0, 'o', label='Data')

# Array with points in between those of the data set for interpolation.
x = np.linspace(min(x0),max(x0),101)
```

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

for o in options:
    f = interp1d(x0, y0, kind=o)    # interpolation function
    plt.plot(x, f(x), label=o)      # plot of interpolated data

plt.legend()
plt.show()
```

Gráfica: