Actividad 4: InterpolaciÃ^3n

Valenzuela Carrillo MarÃa InÃ©s

29 de Enero del 2016

1 Introdun

MÃnimos cuadrados es una tÃ©cnica de anÃ¡lisis numÃ©rico enmarcada dentro de la optimizaciÃ$^3nmatemÃ!$'$tica, enlaque, dadosunconjuntodeparesordenados : variableindependiente, variabledepe$
$nimoerrorcuadrÃ!$'$tico.$

2 Ajustes de datos en Python

Existen distintas maneras de ajustar funciones a datos, en esta actividad se utilizarÃ¡ la interpolaciÃ$^3nlineal, cuadrÃ!$'$ticaycubicaapoyadosenlafunciÃ^3nscipy.interpolatedePython.$

Se proporciona el siguiente cÃ$^3digopararealizarlemodificacionesyadecuarloaloquesepide$:

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

# Original "data set" --- 21 random numbers between 0 and 1.
x0 = np.linspace(-1,1,21)
y0 = np.random.random(21)

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

# Array with points in between those of the data set for interpolation.
x = np.linspace(-1,1,101)

# Available options for interp1d
options = ('linear', 'nearest', 'zero', 'slinear', 'quadratic', 'cubic', 10)

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()
```

2.1 Problema 1

Se pide darle a x 10 puntos aleatorios entre $x = 0$ y $x = 3$ para la funciÃ³nf(x)= $\sin(2x)$.

El cÃ³*digoes* :

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

# Original "data set" --- 21 random numbers between 0 and 1.
x00 = np.random.random(10)
x0=3*x00
y0 = np.sin(2*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 = ('slinear', '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()
```

La grÃ¡fica resultante es:

En esta grÃ¡fica de ve que la cuadrÃ¡tica y la cubica aproximan casi igual, mientras que la lineal solo en ciertos segmentos se parece a la curva real.

2.2 Problema 2

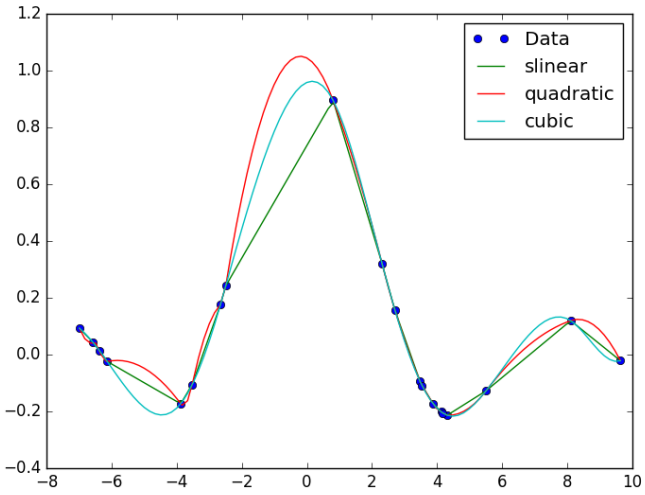
Se pide darle a x 20 puntos aleatorios entre $x = -10$ y $x = 10$ para la funciÃ³nf(x)=$\sin(2x)\frac{-}{x}$.

El cÃ³*digoes* :

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

# Original "data set" --- 21 random numbers between 0 and 1.
x00 = np.random.random(20)
x0=20*x00-10
```

```
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 = ('slinear', '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()
```

La grÃ¡fica resultante es:



Esta grÃ¡fica al contar con mayor cantidad de datos es mÃ¡s exacta y clara. La aproximaciÃ³n cuadrÃ¡tica y cubica no son tan parecidas como en el prolema 1.

2.3 Problema 3

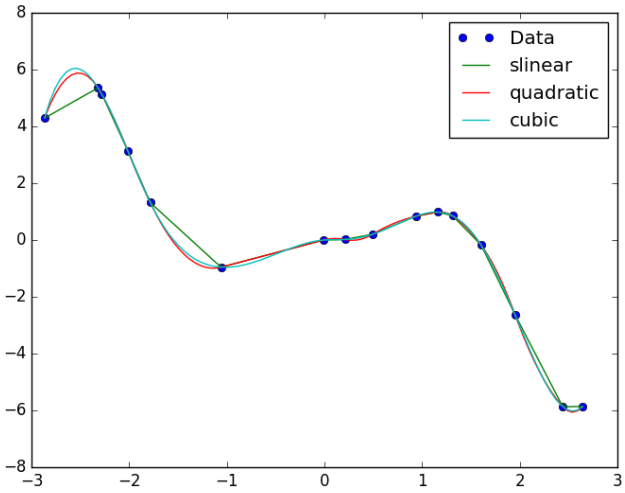
Se pide darle a x 16 puntos aleatorios entre $x = -3$ y $x = 3$ para la funciÃ³n f(x)=$x^2 \sin(2x)$.

El cÃ³digo es :

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

# Original "data set" --- 21 random numbers between 0 and 1.
x00 = np.random.random(16)
x0=6*x00-3
y0 = np.sin(2*x0)*(x0**2)

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 = ('slinear', '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()
```

La grÃ¡fica resultante es:



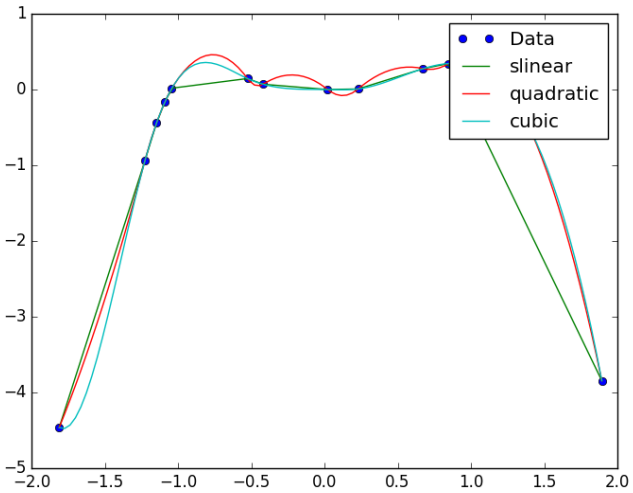
2.4 Problema 4

Se pide darle a x 12 puntos aleatorios entre $x = -2$ y $x = 2$ para la funciÃ³nf(x)=x^3 sin$(3x)$.

El cÃ³*digoes* :

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

# Original "data set" --- 21 random numbers between 0 and 1.
x00 = np.random.random(12)
x0=4*x00-2
y0 = np.sin(3*x0)*(x0**3)

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 = ('slinear', '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()
```

La grÃ¡fica resultante es:



3 ConclusiÃ³n

En esta actividad de pudo ver que la programaciÃ³n es una herramienta muy Ãºtil al realizar interpolaciones. M

Lo anterior tiene gran aplicaciÃ³n prÃ¡ctica ya que para toda persona que trabaje con datos experimentales es m

References

[1] Marco A. Cruz ChÃ¡vez,
http://www.gridmorelos.uaem.mx/ mcruz//cursos/mn/interpolacion.pdf.

[2] Wikipedia, the free encyclopedia, *Interpolation.*