Actividad 3

Carolina Valenzuela Córdova

9 de Febrero de 2016

1. Introducción

Esta actividad se centró en utilizar la herramienta computacional para realizar interpolaciones a funciones dadas. El concepto de interpolación se manejó por primera vez en en curso de análisis numérico y consiste en hallar un dato dentro de un intervalo en el que conocemos los valores en los extremos. El problema general de la interpolación se nos presenta cuando nos dan una función de la cual solo conocemos una serie de puntos de la misma. Se pide hallar el valor de un punto x de esta función.

La interpolación se dirá lineal cuando solo se tomen dos puntos, cuadrática cuando se tomen tres, y así sucesivamente. Se sabe que entre mayor es el grado del polinomio que interpola a la función, se aproxima de manera más precisa a ella, fomando uan curva más suave. En la actividad se nos solicitó interpolar cuatro funciones con tres grados diferentes (lineal, cuadrática, cúbica),y ciertas cantidades de puntos. Para poder lograr esto, fue necesario recurrir al comando *SciPyinterpolate*, el cual (en nuestro caso),nos permite aproximar una función por medio de puntos aleatorios en un intervalo. Para comprender cómo se maneja, se nos proporcionó un código ejemplo, el cual debimos modificar para cumplir con las instrucciones planteadas:

```
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()
```

Las instrucciones consistieron en lo siguiente:
Modifica el código anterior para realizar una interpolación lineal, cuadrática y cúbica los siguientes casos:

- Dados 10 puntos aleatorios entre $x = 0$ y $x = 3$ para la función $f(x) = \sin 2x$.
- Dados 20 puntos aleatorios entre $x = -10$ y $x = 10$ para la función $f(x) = \frac{\sin x}{x}$
- Dados 16 puntos aleatorios entre $x = -3$ y $x = 3$ para la función $f(x) = x^2 \sin 2x$
- Dados 12 puntos aleatorios entre $x = -2$ y $x = 2$ para la función $f(x) = x^3 \sin 3x$

2. Resultados

A continuación se muestran los códigos modificados para cada uno de estos casos

2.1. Código 1

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

x00 = perrito.random.random(10)
x0=3*x00

y0 = perrito.sin(2*x0)

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

x=perrito.linspace(min(x0),
max(x0), 101)

options = ('linear', 'quadratic',
 'cubic')

for o in options:
f = interp1d(x0, y0, kind=o)
plt.plot(x, f(x), label=o)

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

.

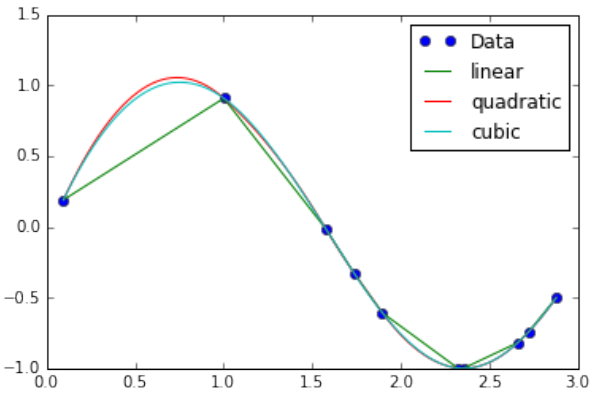

Figura 1: $\sin 2x$

2.2. Código 2

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

```
x00 = perrito.random.random(20)
x0=20*x00-10
y0 = perrito.sin(x0)/x0

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

x=perrito.linspace(min(x0),
max(x0), 101)

options = ('linear', 'quadratic'
, 'cubic')

for o in options:
f = interp1d(x0, y0, kind=o)
plt.plot(x, f(x), label=o)
plt.legend()
plt.show()
```

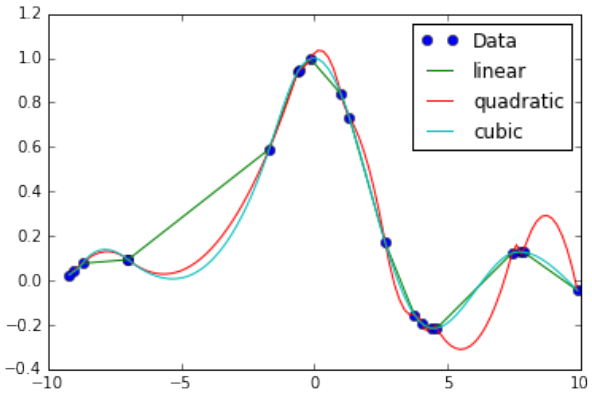

Figura 2: $\frac{\sin x}{x}$

2.3. Código 3

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

x00 = perrito.random.random(16)
x0=6*x00-3
y0 = x0**2*perrito.sin(2*x0)

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


x=perrito.linspace(min(x0), max(x0), 101)
options = ('linear', 'quadratic', 'cubic')

for o in options:
f = interp1d(x0, y0, kind=o)
plt.plot(x, f(x), label=o)

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

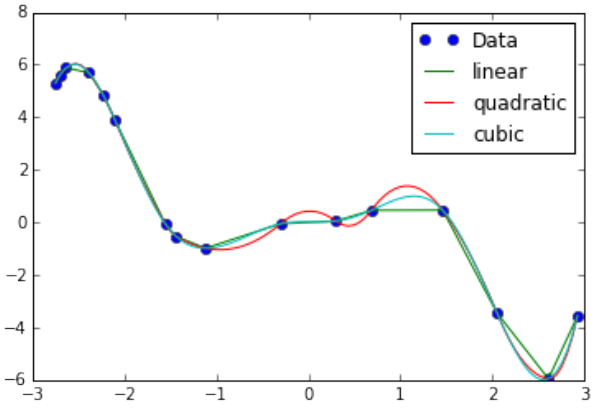

Figura 3: $x^2 \sin 2x$

2.4. Código 4

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

x00 = perrito.random.random(12)
x0=4*x00-2
y0 = x0**3*perrito.sin(3*x0)

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

x =perrito.linspace(min(x0)
, max(x0), 101)

options = ('linear', 'quadratic'
, 'cubic')

for o in options:
f = interp1d(x0, y0, kind=o)
plt.plot(x, f(x), label=o)

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

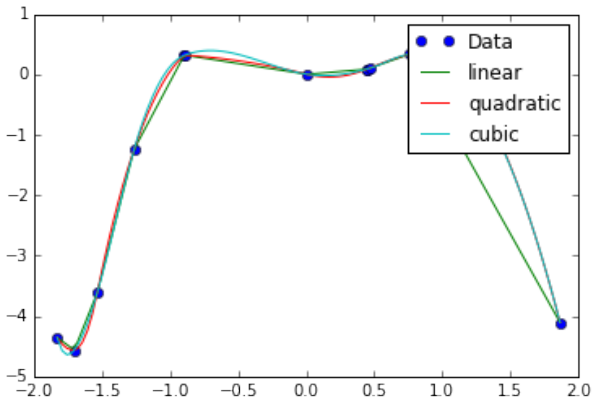

Figura 4: $x^3 \sin 3x$

3. Conclusiones

Me pareció muy interesante conocer una herramienta que nos ayude a realizar interpolaciones de manera rápida y precisa, pues el tema lo vimos el semestre pasado en el curso de Análisis numérico, el cual fue presentado de manera manual, haciendo todo muy tedioso y complicado. Sin embargo, conociendo un método más eficiente como el de utilizar las herramientas vistas en esta actividad, nos permite convertir el proceso de interpolación en algo muy sencillo.