Universidad del Estado de Sonora

División de Ciencias Exactas y Naturales.

Licenciatura en Física.

Física Computacional 1

Efemérides.

Olga María Fimbres Morales

8 de Febrero de 2016

Introducción.

En gran parte del desarrollo de modelos matemáticos para expresar diferentes fenómenos nos encontramos con la inexistencia de estos y enfrentamos el hecho de que lo único que poseemos son datos con los cuales podemos trabajar sin nada que los defina dentro de una misma función, para ello tenemos las herramientas de la interpolación, el cuál es un método mediante el cual construyen nuevos datos partiendo del rango de datos previamente conocidos.
Estos datos pueden obtener se diferente forma, por ejemplo por medio de una experimentación o una simple muestra obtenida por medio de la observación de algún fenómeno; y por medio de este método es que resulta posible tener una aproximación a la descripción de dicho fenómeno y de esta forma conocer mas datos con una aproximación razonable.
Esta puede ser lineales, polinomiales o incluso mediante un procedimiento de Gauss. La confianza que se puede tener en cada uno de ellos varia dependiendo de los datos que poseemos, ya que los que parecieran ser los mas confiables no siempre lo son.[1]

Efemérides

En astronomía, las efemérides permiten conocer la posición de los cuerpos astronómicos naturales como de los artificiales en un cierto periodo de tiempo. La posición astronómica que nos permiten conocer están dadas en coordenadas esféricas; algunos de los fenómenos que podemos encontrar son los eclipses, el movimiento de planetas, tiempo sideral, fases de la luna y cometas.[2]
Este es un caso en el que la interpolación puede utilizarse para adelantarnos a las fechas en que estos fenómenos ocurrirán, como lo es el cometa Halley, y de este modo poder llevar un registro de los avistamientos para un mejor desarrollo de la interpolación.

Problema.

En esta práctica se nos proporciono el siguiente código:

```
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()
```

el cuál realizaba múltiples interpolaciones por medio de 21 puntos equidistantes en el intervalo de -1 a 1. Las interpolaciones que realiza mas muestra de una forma suave ya que se utilizan una gran cantidad de puntos para trazarla, lo cual puede cambiar a consideración de cada persona.
Con dicho código se nos pidió modificarlo de manera que fuera posible obtener diferentes resultados basándonos en una cierta cantidad de puntos en un determinado rango; realizando siempre una interpolación lineal, cuadrática y cúbica.

Problema 1

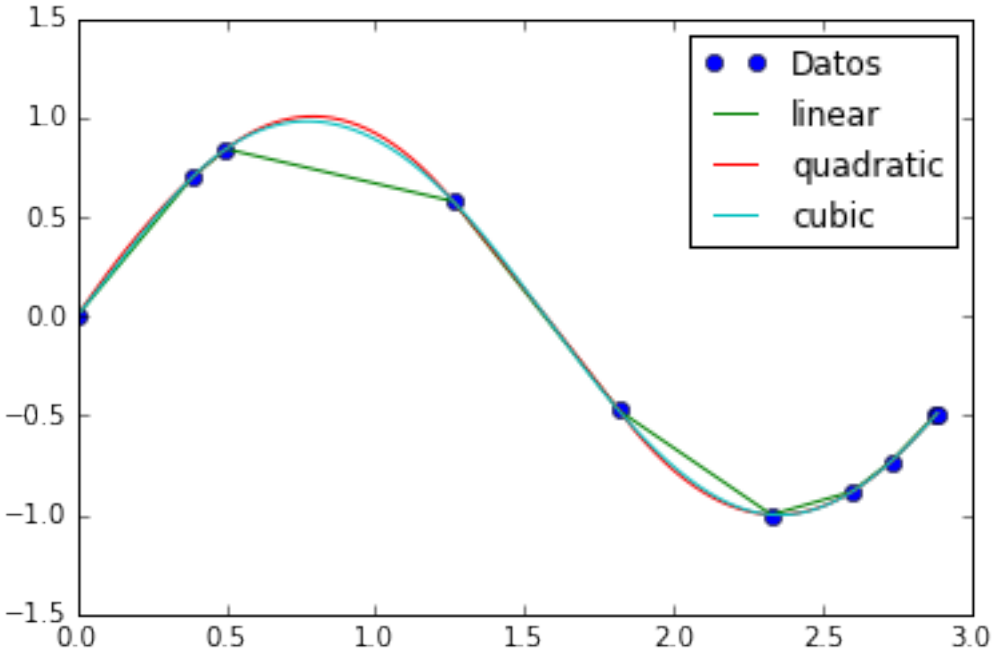
En este problema se nos pide utilizar 10 puntos aleatorios en el rango de X=0 a X=3 dentro de la función $f(x) = \sin(2x)$:

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

x0 = np.random.random(10)
x00 = 3*x0
y0 = np.sin(2*x00)

plt.plot(x00, y0, 'o', label='Datos')
x = np.linspace(min(x00),max(x00),1000)
options = ('linear', 'quadratic', 'cubic')
for o in options:
    f = interp1d(x00, y0, kind=o)
    plt.plot(x, f(x), label=o)
plt.legend()
plt.show()
```

Nótese que fue necesario cambiar ciertos aspectos para que el código funcionara de la forma en que lo deseamos. Inicialmente debemos de definir a $x0$ como números aleatorios, los cuales solo de nos dan entre 0 y 1; por lo que para cubrir el rango que deseamos, el cual es de 0 a 3 debemos multiplicar el resultado que nos arroje por 3, definiendo así una nueva $x00$ que es la que utilizaremos para obtener sus valores en la función y la que graficaremos. Fuera de ese detalle el código mantiene la misma estructura del original.
Mostrando como resultado la siguiente gráfica:


Datos
linear
quadratic
cubic
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0

Problema 2

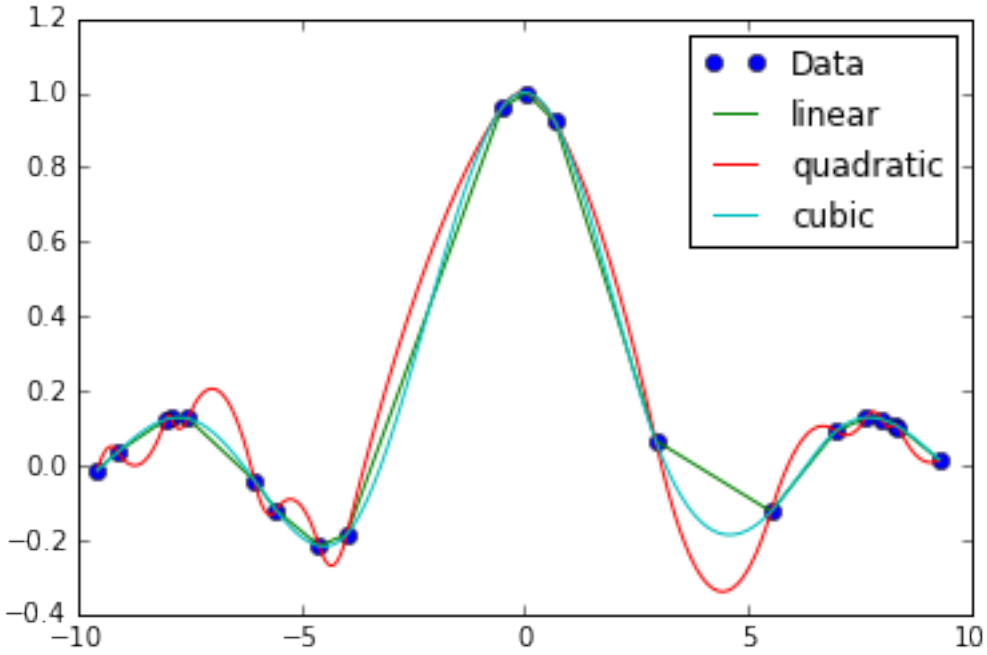
Esta vez se nos pidieron 20 puntos dentro del rango x=-10 y x=10 dentro de la función $f(x) = \frac{\sin x}{x}$:

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

x0 = np.random.random(20)
x00 = -10 + 20*x0
y0 = np.sin(x00)/x00

plt.plot(x00, y0, 'o', label='Data')
x = np.linspace(min(x00),max(x00),1000)
options = ('linear', 'quadratic', 'cubic')
for o in options:
    f = interp1d(x00, y0, kind=o)
    plt.plot(x, f(x), label=o)
plt.legend()
plt.show()
```

Nuevamente, note que fue necesario redefinir una nueva $x00$ para cubrir el rango que se desea.
La gráfica fue la siguiente:


Data
linear
quadratic
cubic
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−10
−5
0
5
10

Problema 3

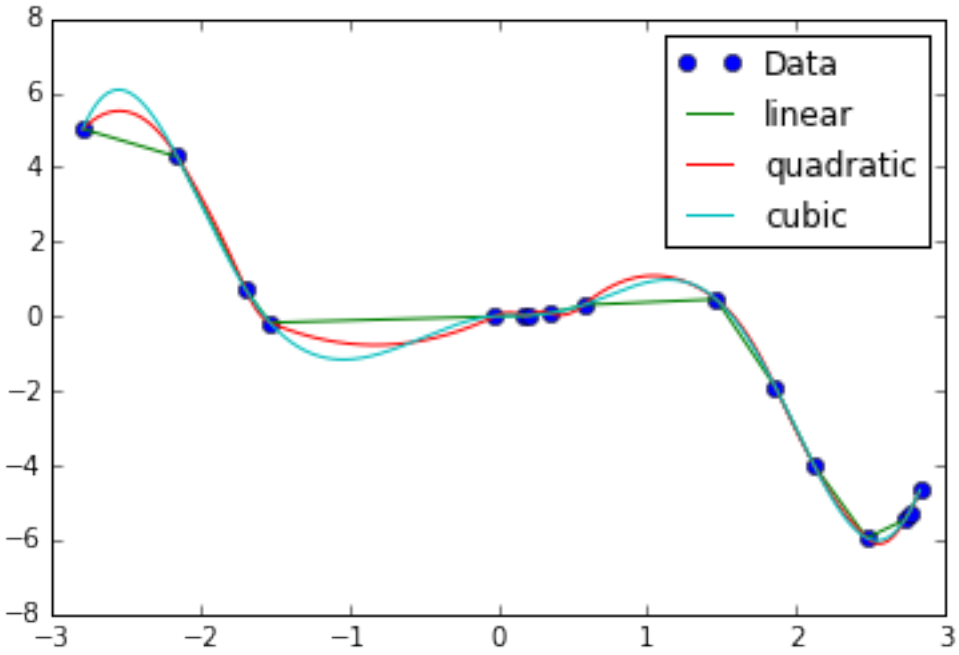
Este problema nos pedía ahora un rango entre x=-3 y x=3, con una cantidad de 16 puntos en la función $f(x) = x^2 \sin 2x00$:

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

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

plt.plot(x00, y0, 'o', label='Data')
x = np.linspace(min(x00),max(x00),1000)
options = ('linear', 'quadratic', 'cubic')
for o in options:
    f = interp1d(x00, y0, kind=o)
    plt.plot(x, f(x), label=o)
plt.legend()
plt.show()
```

Obteniendo como resultado la siguiente gráfica:


Data
linear
quadratic
cubic
8
6
4
2
0
−2
−4
−6
−8
−3
−2
−1
0
1
2
3

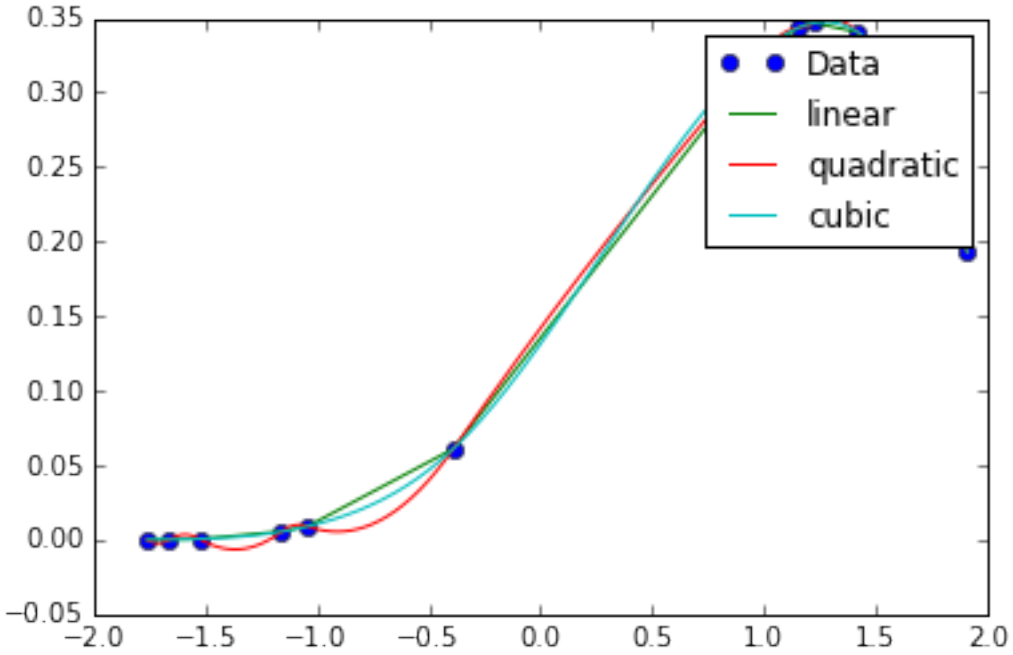
Problema 4

Finalmente, es necesario una cantidad de 12 puntos aleatorios entre x=-2 y x=2, dentro de la función $f(x) = x^3 \sin 3x$:

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

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

plt.plot(x00, y0, 'o', label='Data')
x = np.linspace(min(x00),max(x00),1000)
options = ('linear', 'quadratic', 'cubic')
for o in options:
    f = interp1d(x00, y0, kind=o)
    plt.plot(x, f(x), label=o)
plt.legend()
plt.show()
```


Data
linear
quadratic
cubic
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Referencias

[1] Wikipedia, The free encyclopedia; "Interpolation"; 2016

[2] Wikipedia, The free encyclopedia; "Ephemeris"; 2016