Universidad del Estado de Sonora

División de Ciencias Exactas y Naturales.

Licenciatura en Física.

Física Computacional 1

Hedwin Aaron Encinas Acosta

1 Introduccion

En la física, es muí común que al hacer experimentos u observar un fenómeno, se termine con información que solo nos muestra una parte muí pequeña de lo que se quiere observar.
Esto es muí común al momento de gratificar datos que se obtienen de observar un fenómeno, y terminemos con una serie de puntos sin una relación aparente. Por si solos, los puntos nos dan muí poca información pues no se puede saber que ocurrió entre cada intervalo en los cuales se recolecto la información, aquí es donde entran en juego las matemáticas.
El análisis numérico es una herramienta muí útil en estos casos, ya que por medio de la interpolación, que es la obtención de nuevos puntos partir del conocimiento de una serie de puntos, podemos buscar la función que mejor modele el fenómeno que estamos observando.[1]
Ay diferente forma de interpolar datos, se puede hacer de forma lineal, polinomial o por un procedimiento de Gauss. Claro que cada método tiene sus ventajas y desventajas y dependiendo de los datos que se tenga se elegirá el que mejor se ajuste a estos.

2 Actividad

L que se hizo en esta actividad fue modificare el programa que se encuentra a continuación, el cual genera una seria de puntos aleatorios y después se les ajusta diferentes tipos de curvas:

```
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()
```

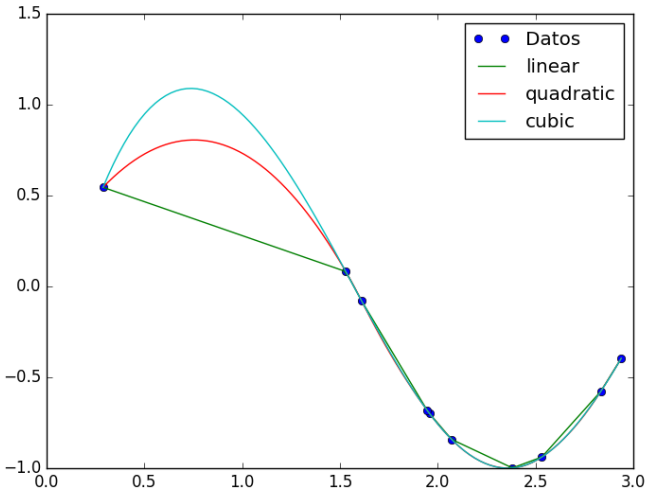
3 Ejercicio 1

Generar 10 puntos aleatorios entre x=0 y x=3 para la función $f(x) = \sin(2x)$.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from math import sin
# Original "data set" --- 21 random numbers between 0 and 1.

x0 = np.random.random(10)
x1 = 3*x0
y0 = np.sin(2*x1)
plt.plot(x1, y0, 'o', label='Datos')

x = np.linspace(min(x1),max(x1),100)

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

for o in options:
    f = interp1d(x1, y0, kind=o)    # interpolation function
    plt.plot(x, f(x), label=o)      # plot of interpolated data

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

3.1

resultando en la siguiente grafica:



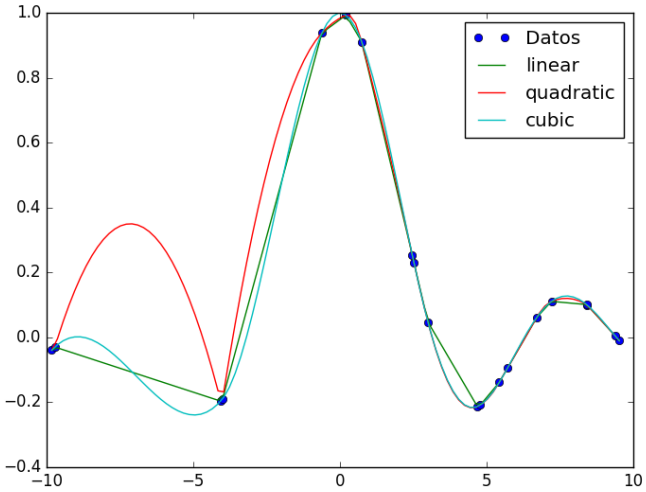
4 Ejercicio 2

Generar20 puntos aleatorios entre x=-10 y x=10 para la función $(x) = \frac{sin(x)}{x}$

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from math import sin
# Original "data set" --- 21 random numbers between 0 and 1.

x0 = np.random.random(20)
x1 = -10+(x0*20)
y0 = np.sin(x1)/x1
plt.plot(x1, y0, 'o', label='Datos')

x = np.linspace(min(x1),max(x1),100)

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

for o in options:
    f = interp1d(x1, y0, kind=o)    # interpolation function
    plt.plot(x, f(x), label=o)      # plot of interpolated data

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

4.1

resultando en la siguiente grafica:



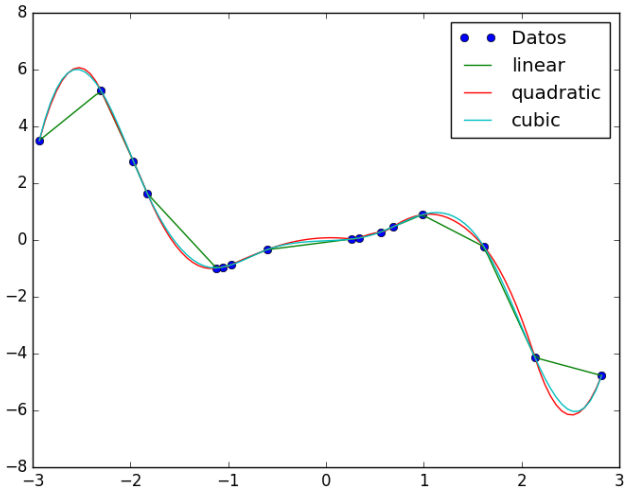
5 Ejercicio 3

Generar 16 puntos aleatorios entre x=-3 y x=3 para la función $(x) = x^2 sin(2x)$

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from math import sin
# Original "data set" --- 21 random numbers between 0 and 1.

x0 = np.random.random(16)
x1 = -3+(x0*6)
y0 = np.sin(x1*2)*(x1**2)
plt.plot(x1, y0, 'o', label='Datos')

x = np.linspace(min(x1),max(x1),100)

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

for o in options:
    f = interp1d(x1, y0, kind=o)    # interpolation function
    plt.plot(x, f(x), label=o)      # plot of interpolated data

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

5.1

resultando en la siguiente grafica:



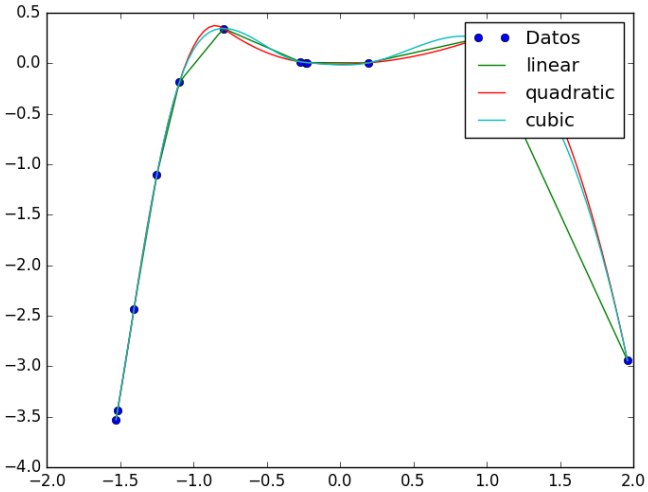
6 Ejercicio 4

Generar 12 puntos aleatorios entre x=-2 y x=2 para la función $(x) = x^3 sin(3x)$

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from math import sin
# Original "data set" --- 21 random numbers between 0 and 1.
x0 = np.random.random(12)
x1 = -2+(x0*4)
y0 = np.sin(x1*3)*(x1**3)
plt.plot(x1, y0, 'o', label='Datos')

x = np.linspace(min(x1),max(x1),100)

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

for o in options:
    f = interp1d(x1, y0, kind=o)    # interpolation function
    plt.plot(x, f(x), label=o)      # plot of interpolated data

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

6.1

resultando en la siguiente grafica:



Referencias

[1] Wikipedia, The free encyclopedia; Iterpolation; 2016