1-tarea1

February 19, 2024

0.1 Tarea 1: Interpolación, Derivación e Integración numérica

Nombre: Maximiliano Vaca Montejano

email: maximiliano.vaca@uabc.edu.mx

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import CubicSpline
```

$$f(x) = 9Sen(\frac{1}{10}e^{\frac{1}{7}x})$$

1. Seleccione alguna gráfica tipo x vs $f(x)$ que tenga en algún archivo PDF. Grabe la imagen en algún formato png o jpeg y suba la imagen (gráfica) en el siguiente enlace. Genere algunos datos de la figura y exportelos a un archivo de datos de extension csv o dat. https://automeris.io/WebPlotDigitizer/

[ ]:
```
df = pd.read_csv('datasets/Default_Dataset_4.csv', header=None)
df.head(5)
```

[ ]:

	0	1
0	0.815889	0.982022
1	2.426276	1.245419
2	4.036767	1.570889
3	5.669910	1.984467
4	7.258194	2.488583

3. Use la función de numpy “genfromtxt()” para leer los datos a un array y grafiquelos con “matplotlib.pyplot” para verlos.

[ ]:
```
arr = np.genfromtxt('datasets/Default_Dataset_4.csv', delimiter=",")

#for i in range(5):
#    print(arr[i])

#print([arr[i] for i in range(5)])

[arr[i] for i in range(5)]
```

[ ]:
```
[array([0.81588907, 0.98202205]),
 array([2.4262762 , 1.24541854]),
 array([4.03676699, 1.57088935]),
 array([5.66990954, 1.98446657]),
 array([7.258194  , 2.48858284])]
```

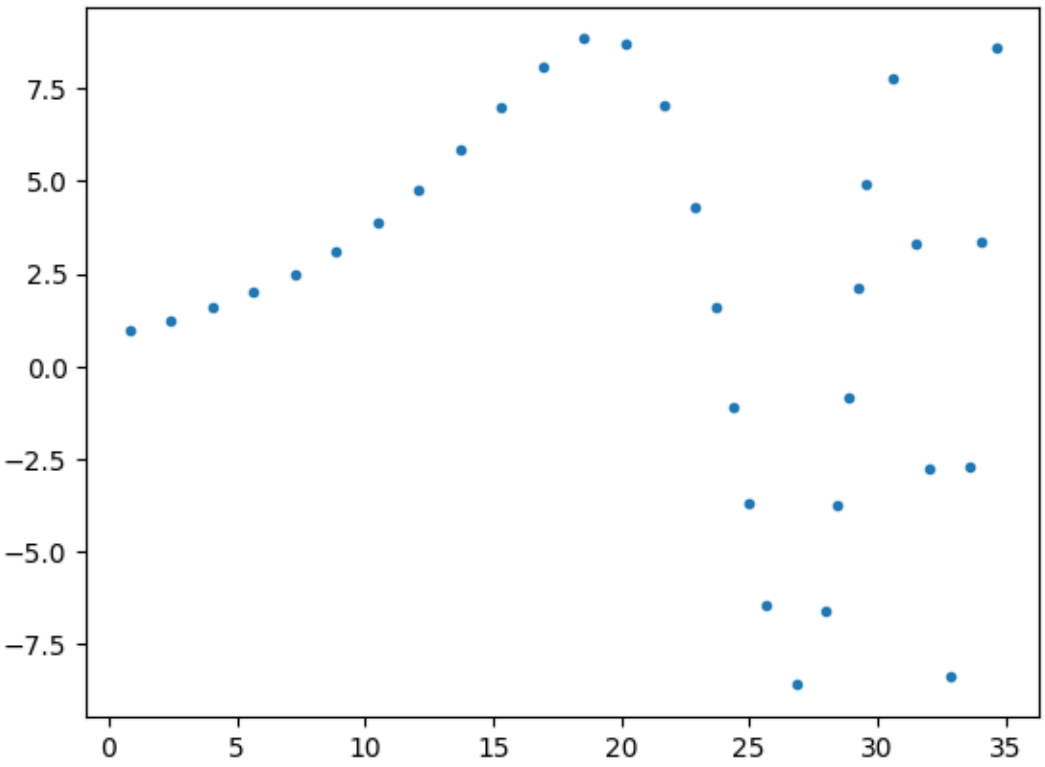
[ ]:
```
# Dividir el array en dos arrays separados para x y y
x_arr = arr[:, 0]
y_arr = arr[:, 1]

x_arr
```

[ ]:
```
array([ 0.81588907,  2.4262762 ,  4.03676699,  5.66990954,  7.258194  ,
        8.86918205, 10.48040262, 12.09187112, 13.70358055, 15.31543986,
       16.92721512, 18.53842169, 20.14817428, 21.70338169, 22.89327412,
       23.6937389 , 24.36441768, 24.98325077, 25.67980034, 26.87064794,
       27.96462596, 28.46275567, 28.85707832, 29.22552123, 29.59377752,
       30.58524873, 31.51266844, 32.02181059, 32.86936636, 33.64661652,
       34.03148953, 34.63750839])
```

[ ]:
```
plt.plot(x_arr, y_arr, 'o', markersize=3)
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f9f7d403760>]
```

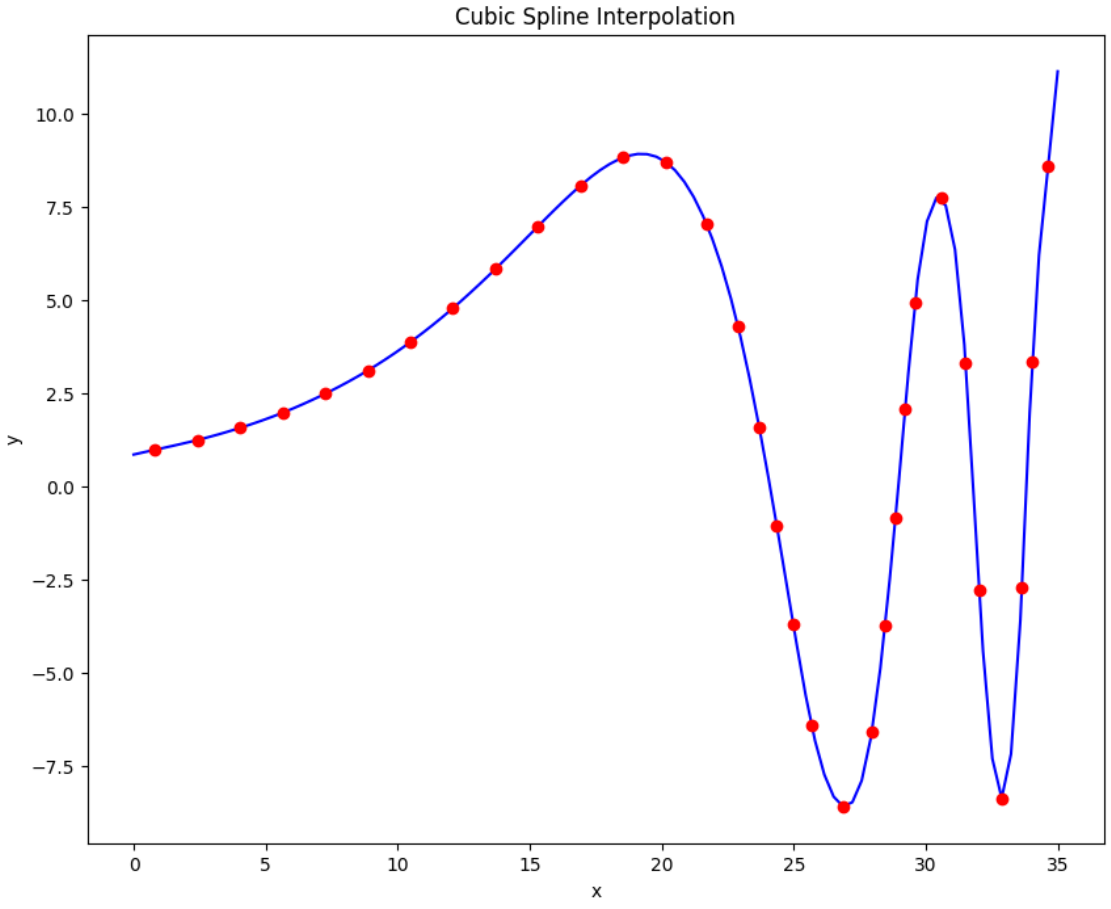


4. Del módulo "scipy.interpolate" importe y use la función "CubicSpline()" para crear una función interpolada de sus datos. Con la función anterior, cree un array de datos x, y (estos datos seran igualmente espaciados por lo que los podemos usar para derivar e integrar con los métodos discutidos en clase). Ejemplo de uso: https://pythonnumericalmethods.berkeley.edu/notebooks/chapter17.03-Cubic-Spline-Interpolation.html

[ ]:
```
#x_arr = np.sort(x_arr)

f = CubicSpline(x_arr, y_arr, bc_type='natural')
x_new = np.linspace(0, 35, 100)
y_new = f(x_new)
```

[ ]:
```
plt.figure(figsize = (10,8))
plt.plot(x_new, y_new, 'b')
plt.plot(x_arr, y_arr, 'ro')
plt.title('Cubic Spline Interpolation')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



re: 4. Con la función anterior, cree un array de datos x, y (estos datos seran igualmente espaciados por lo que los podemos usar para derivar e integrar con los métodos discutidos en clase). Ejemplo de uso: https://pythonnumericalmethods.berkeley.edu/notebooks/chapter17.03-Cubic-Spline-Interpolation.html

resultados de la interpolacion:

$$x_{new}$$

$$y_{new}$$

5. Usando una derivada progresiva cree la derivada de la función y grafiquela. Adicionalmente, muestre la tabla de los datos con "DataFrame" de pandas para una mejor visualización

$$f(x) = 9Sen(\frac{1}{10}e^{\frac{1}{7}x})$$

[ ]:
```
x_new[1] - x_new[0]
```

[ ]:
```
0.35353535353535354
```

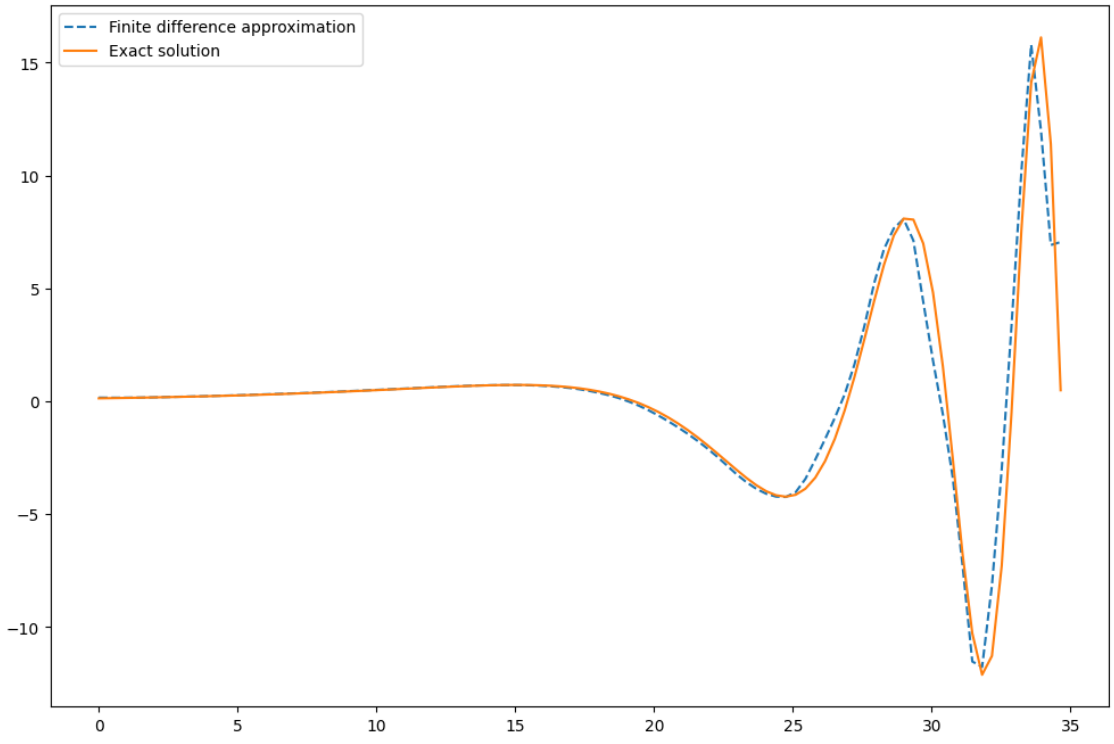
[ ]:
```
# step size
h = x_new[1] - x_new[0]
# define grid
x = x_new
# compute function
y = y_new

# compute vector of forward differences
forward_diff = np.diff(y)/h
# compute corresponding grid
x_diff = x[:-1:]
# compute exact solution
exact_solution = (9/70) * np.exp(x_diff/7) * np.cos((1/10) * np.exp(x_diff/7))

# Plot solution
plt.figure(figsize = (12, 8))
plt.plot(x_diff, forward_diff, '--', \
         label = 'Finite difference approximation')
plt.plot(x_diff, exact_solution, \
         label = 'Exact solution')
plt.legend()
plt.show()

# Compute max error between
# numerical derivative and exact solution
max_error = max(abs(exact_solution - forward_diff))
```

```
print(max_error)
```




```
6.562443877216377
```

utilizando el codigo del libro:

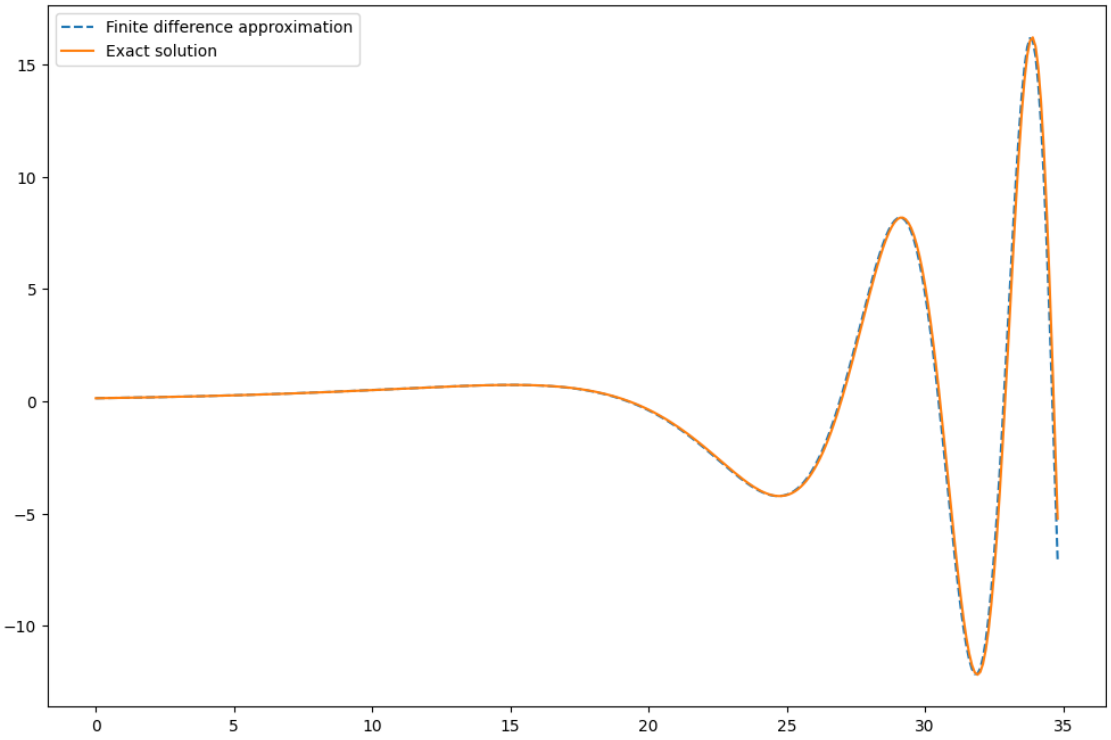
[ ]:
```
# step size
h = 0.1
# define grid
x = np.arange(0, 35, h)
# compute function
y = 9 * np.sin((1/10) * np.exp(x/7))

# compute vector of forward differences
forward_diff = np.diff(y)/h
# compute corresponding grid
x_diff = x[:-1:]
# compute exact solution
exact_solution = (9/70) * np.exp(x_diff/7) * np.cos((1/10) * np.exp(x_diff/7))

# Plot solution
plt.figure(figsize = (12, 8))
plt.plot(x_diff, forward_diff, '--', \
         label = 'Finite difference approximation')
```

```
plt.plot(x_diff, exact_solution, \
         label = 'Exact solution')
plt.legend()
plt.show()

# Compute max error between
# numerical derivative and exact solution
max_error = max(abs(exact_solution - forward_diff))
print(max_error)
```




```
1.8708747023711518
```

6. Realice una integración numérica de los datos usando la regla compuesta de Simpson.

re:

$$f(x) = 9Sen(\frac{1}{10}e^{\frac{1}{7}x})$$

resultados de la interpolacion:

$$x_{new}$$

$$y_{new}$$

[ ]:
```
a = 0
b = 35
n = 35

h = (b - a) / (n - 1)
x = x_new
f = y_new

I_simp = (h/3) * (f[0] + 2*sum(f[:n-2:2]) \
            + 4*sum(f[1:n-1:2]) + f[n-1])

err_simp = 95.17549943710374 - I_simp

print(I_simp)
print(err_simp)
```

```
81.9471278335225
13.228371603581238
```

[ ]:
```

```