Assignment-2

EE2703-Applied Programming Lab 1 Feb 2018

Vector Operations and Functions in Python

Mohammed Khandwawala **EE16B117**

Índice

1. Introduction

Python is the most commonly used tool for scientic computing, engineering, and research. It is comparable with well-established softwares like MATLAB and even better in some aspects. A major cause is its open-source nature and huge base of python contributor. This is the reason why python tends to be very quick at incorporating useful features from other languages. Python is best among other programming languages in string manipulation. Some useful python scientific packages include scipy , numpy and matplotlib. These packages are taken together an bundled in pylab module which has all kind of matlab functionality. This assignment introduces vector operations provided by numpy and the calculation of definite integrals using the quad function.

1.1. Numpy arrays for vector operation

Numpy module in python consist of important numerical functions. One of the feature it has is numpy arrays which are much easier to use than C arrays and hence can handle much complex tasks. Numpy arrays are better for numerically extensive jobs than native lists.

1.2. Scipy : The Library for Scientific Computations

Scipy is an extensive package containing functions for scientific uses. In this assignment we are using quad function from scipy.integrate to calculate integral.

1.3. Matplotlib : The Plotting Library

It is a very powerful plotting package. It allows to plot various type of multidimensional plots , of various types and offer great amount of customizations on how the plot looks unlike MATLAB.

2. Function Defination

The function defined is a general python function and like all python function returns a vector if a vector is passed to it. *Format: x = arange([start,] stop[, step], dtype = None)* Above line shows the format for generating vectors. plt.title adds the title to the plot and plt.scatter plots the scatter plot between first and second argument passed to the function.

Listing 1: To generate and plot $1/(1+t^2)$

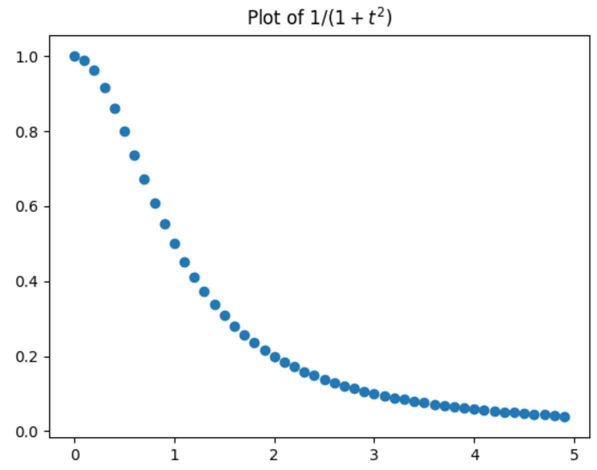
```
import matplotlib.pyplot as plt
import numpy as np
import math

def taninf(x):
        return 1/(1+x**2)

x = np.arange(0,5.1,0.1)

val = taninf(x)

plt.scatter(x,val)
plt.title(r"Plot␣of␣$1/(1+t^{2})$")
plt.show()
```



3. Integration Using Built-in quad function

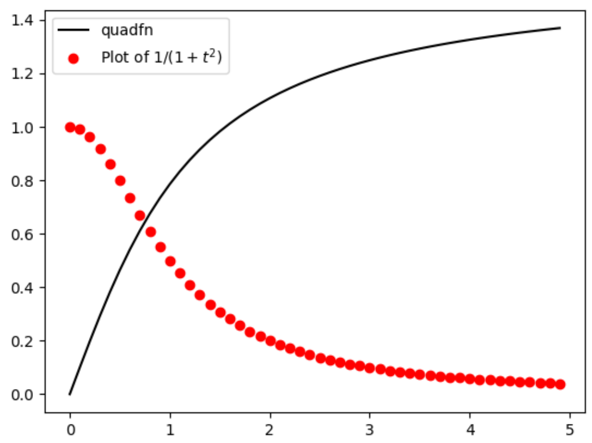
To compute integral using quad function function definition and the limits are need to be passed. The function returns two values for a point. First value is the value of the integral and second is the error.

Listing 2: Using quad function to integrate $1/(1+t^2)$

```
from scipy.integrate import quad
integration = []
```

```
for i in range(0,len(x)):
        temp = quad(taninf,0,x[i])
        integration.append(temp[0])

plt.scatter(x,val,c='red')
plt.plot(x,integration ,c="black")
plt.legend(("quadfn",r"Plot␣of␣...
...$1/(1+t^{2})$"))
plt.show()
```



```
+-----------------+--------------------+-----+
|   Quad Integral |    arctan(x) value |   x |
+-----------------+--------------------+-----|
|         0.00000 |            0.00000 | 0.0 |
|         0.09967 |            0.09967 | 0.1 |
|         0.19740 |            0.19740 | 0.2 |
|         0.29146 |            0.29146 | 0.3 |
|         0.38051 |            0.38051 | 0.4 |
|         0.46365 |            0.46365 | 0.5 |
|         0.54042 |            0.54042 | 0.6 |
|         0.61073 |            0.61073 | 0.7 |
|         0.67474 |            0.67474 | 0.8 |
|         0.73282 |            0.73282 | 0.9 |
|         0.78540 |            0.78540 | 1.0 |
|         0.83298 |            0.83298 | 1.1 |
|         0.87606 |            0.87606 | 1.2 |
|         0.91510 |            0.91510 | 1.3 |
|         0.95055 |            0.95055 | 1.4 |
|         0.98279 |            0.98279 | 1.5 |
|         1.01220 |            1.01220 | 1.6 |
|         1.03907 |            1.03907 | 1.7 |
|         1.06370 |            1.06370 | 1.8 |
|         1.08632 |            1.08632 | 1.9 |
|         1.10715 |            1.10715 | 2.0 |
```

3.1. Tabulated Values of quad function integral values of $1/(1+t^2)$ values against x

To tabulate the result built in python function tabulate can be used. It accepts table , 2-D array of data-frame. So to tabulate arctan(x) with quad function using pandas their respaective array is converted into data-frame which is than tabulated.

Listing 3: Tabulating Quad function integral of $1/(1+t^2)$ against x

```
import pandas as pd
from tabulate import tabulate

table = pd.DataFrame({"x":x,"Quad...
...Function␣Integral":integration})
print tabulate(table, headers='keys'...
...tablefmt='p')
```

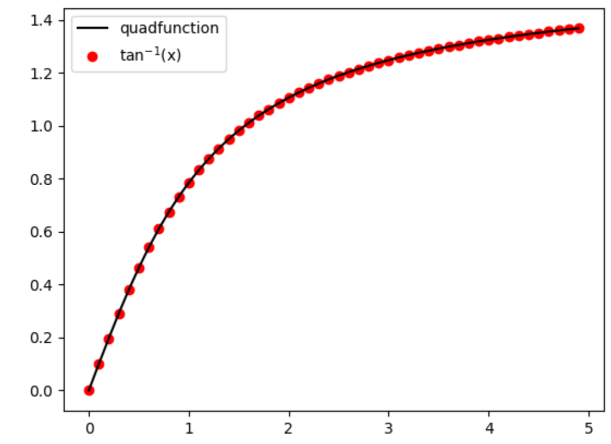
```
|         1.12638 |           1.12638 | 2.1 |
|         1.14417 |           1.14417 | 2.2 |
|         1.16067 |           1.16067 | 2.3 |
|         1.17601 |           1.17601 | 2.4 |
|         1.19029 |           1.19029 | 2.5 |
|         1.20362 |           1.20362 | 2.6 |
|         1.21609 |           1.21609 | 2.7 |
|         1.22777 |           1.22777 | 2.8 |
|         1.23874 |           1.23874 | 2.9 |
|         1.24905 |           1.24905 | 3.0 |
|         1.25875 |           1.25875 | 3.1 |
|         1.26791 |           1.26791 | 3.2 |
|         1.27656 |           1.27656 | 3.3 |
|         1.28474 |           1.28474 | 3.4 |
|         1.29250 |           1.29250 | 3.5 |
|         1.29985 |           1.29985 | 3.6 |
|         1.30683 |           1.30683 | 3.7 |
|         1.31347 |           1.31347 | 3.8 |
|         1.31979 |           1.31979 | 3.9 |
|         1.32582 |           1.32582 | 4.0 |
|         1.33156 |           1.33156 | 4.1 |
|         1.33705 |           1.33705 | 4.2 |
|         1.34230 |           1.34230 | 4.3 |
|         1.34732 |           1.34732 | 4.4 |
|         1.35213 |           1.35213 | 4.5 |
|         1.35674 |           1.35674 | 4.6 |
|         1.36116 |           1.36116 | 4.7 |
|         1.36540 |           1.36540 | 4.8 |
|         1.36948 |           1.36948 | 4.9 |
|         1.37340 |           1.37340 | 5.0 |
+-----------------+-------------------+-----+
```

4. Comparing Quad function integral with actual value of $\tan^{-1}$(x)

Plotting both actual value and the integral value via quad function show the accuracy of computation. Plots closely resemble.

Listing 5: Comparing Quad function integal with actual value of $\tan^{-1}$(x)

```
atan = np.arctan(x)
plt.scatter(x,integration ,c='red')
plt.plot(x,atan,c="black")
plt.legend(("quadfunction",r"tan$^{-1}$(x
plt.show()
```



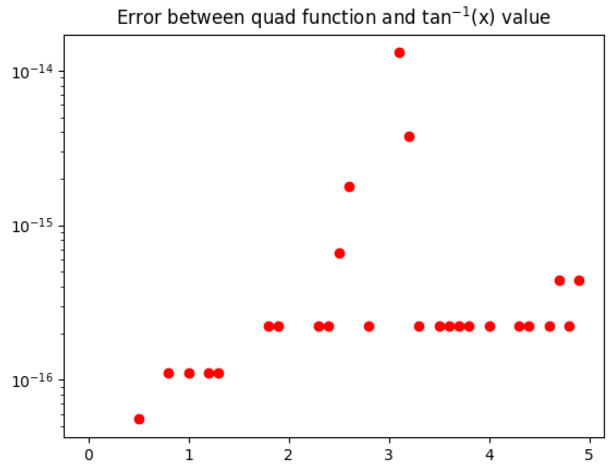
5. Computing difference between Quad function integal with actual value of tan^{-1}(x)

The difference plot between the both clearly shows the deviation in the range of 10^{-15}.

Listing 6: Comparing Quad function integal with actual value of tan^{-1}(x)

```
integration = np.array(integration)
error = abs(integration − atan)

plt.semilogy(x,error,"ro")
plt.title("Error␣between␣quad...
...function␣and␣tan$^{−1}$(x)␣value")
plt.show()
```



6. Integration using trapezoidal algorithm

6.1. About

According to the Trapezoidal Rule, if a function is known at points a, a + h, . . . , b, its integral is given by

$$\begin{cases} 0 & x = 0 \\ 0{,}5(f(a) + f(x_i) + \sum_{j=1}^{i-1} f(x_j)) & x = a + ih \end{cases}$$

or it can be rearranged as

$$I_i = h\ (\ \sum_{j=1}^{i} f(x_j)\ \text{-}\ \tfrac{1}{2}\ (\ f(x_j) + f(x_j)\ )\ )$$

The last equation can simply be computer in one line using cumsum() function which calculates the cumulative sum for every index of vector.

6.2. Computing trapezoidal Integral using a loop

Using the above defined formula trapezoidal integral can easily computed in a for loop.

```
def trapz(a,b,h):
    y=np.zeros(int((b−a)/h)+1)

    for i in range(1,len(y)):
        y[i]=y[i−1]+0.5*h*(taninf(a+i*h)+
    return y
```

6.3. Computing trapezoidal Integral by passing vector as argument

Code below shows a much simpler way of computing trapezoidal integral from the same formula using vectors. This implementation does not require to write a for loop.

Listing 7: Comparing Quad function integal with actual value of tan^{-1}(x)

```
h=0.1
n=5/h

pts = np.arange(0,5,h)
tanval = taninf(pts)
c_sum = np.cumsum(tanval)
```

```
trapz = h*(c_sum-(tanval[0]+tanval)/2)
plt.scatter(pts,trapz)
plt.title(r"Plot␣of␣$\int_{0}^{5}$␣...
...$1/(1+t^{2})$␣dx␣using␣trapz")
plt.show()
```

7. Computing Error

7.1. Exact Error

Exact error is (for a fixed h) computed by taking the maximum of the absolute difference between the value of $\tan^{-1}$(x) and integral value by trapezoidal function at each x.

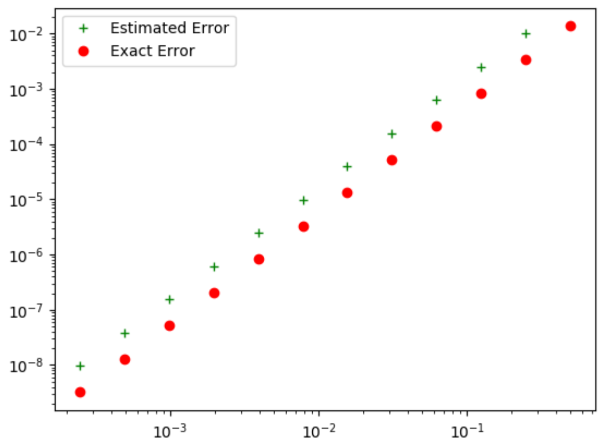
7.2. Estimated Error

Estimated error is calculates by taking the maximum value of absolute difference between the integral values (generated by trapezoidal function) between consecutive iteration of reduced step size (h) at common points . Here in this case starting value of stepsize is 0.5 and reduced by half in every iteration. Error computing loop in iterated 12 times to get desired tolerance of 10-8.

Listing 8: Computing estimated and exact error and plotting them.

```
H = []
estimated_error = []
exact_error = []

h=1
pts = np.arange(0,5,h)
tanval = taninf(pts)
c_sum = np.cumsum(tanval)
trapz = h*(c_sum-(tanval[0]+tanval)/2)
h = 0.5
for i in range(12):
        pts_2 = np.arange(0,5,h)
        tanval_2 = taninf(pts_2)
        act_val = np.arctan(pts_2)
        c_sum_2 = np.cumsum(tanval_2)
        trapz_2 = h*(c_sum_2-(tanval_2[0]+tanval_2)/2)
        if(i!=0):
                es_err = abs(trapz - trapz_2[::2])
                estimated_error.append(max(es_errs))
        ex_err = abs(trapz_2 - act_val)
        exact_error.append(max(ex_err))
        H.append(h)
        trapz = np.array(trapz_2)
        h=h/2
plt.loglog(H[1:],estimated_error,"g+")
plt.loglog(H,exact_error,"ro")
plt.legend(("Estimated␣Error","Exact␣Error"))
plt.show()
```



8. Conclusion

The python implementation of MATLAB like libraries i.e. scipy, numpy and matplotlib and accurate and powerful.Adopted matlab functions are improved in functionality in pythons implementation.