Data Integration

Rebecca Corley[†]

[1]University of North Georgia, Department of Physics and Astronomy

1. INTRODUCTION

In this project, we were given a set of data to use to compute and plot the indefinite, numerical integral. To calculate the indefinite integral, we used the cumulative trapezoidal rule. The trapezoidal rule is a technique used for approximating definite integrals. It is a more approximate Riemann summation method because instead of using rectangles to approximate area, the trapezoidal rule uses trapezoids. For a given curve, the trapezoidal rule approximates the region under the curve buy summing over infinitely small trapezoidal-shaped regions. The approximation of the integral becomes more accurate as the number of trapezoids increases, which makes the area of the trapezoids decrease. The cumulative trapezoidal rule is a similar numerical integration method. In practice, when given a set of (x,y) data, the cumulative trapezoidal method integrates y with respect to the coordinates or scalar spacing specified by x. For the provided data, we were given a matrix of nonuniform spacing, so the cumulative trapezoid method integrates each row independently to find the cumulative sum over each pair.

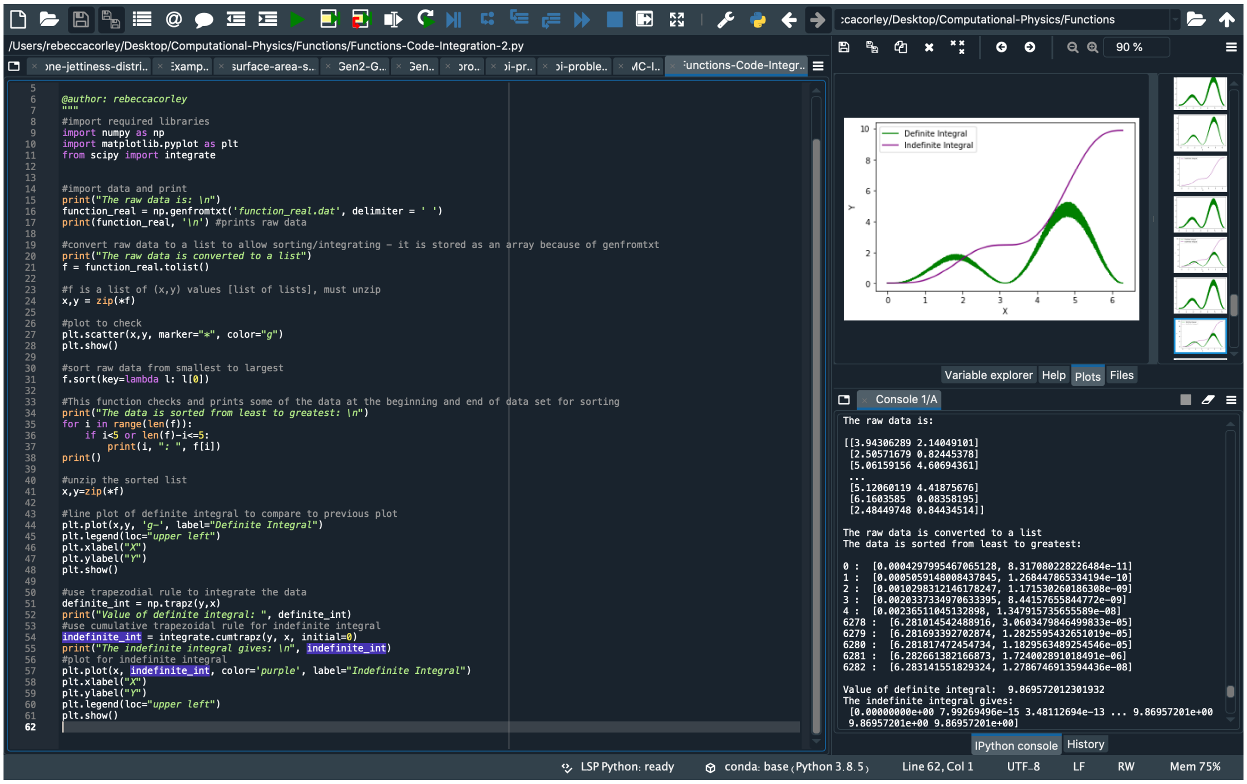
2. THE CODE

The code is structured as follows and can be seen in Fig. 1. First import required libraries and raw data. Numpy's genfromtxt saves data as an array, so we must convert it to a list, then unzip the data. We plot the data in a scatter plot to visualize the data. For integration, we order the data's x-values from smallest to largest. Then a simple for loop checks a few of the first and last values of the data set to see that it is ordered correctly. The data is unzipped again then plotted to compare to the previous unordered data. The trapezoidal rule is then used to calculate the floating point value of the definite integral. The cumulative trapezoidal rule is then used to calculate and plot the indefinite integral.

3. RESULTS AND ANALYSIS

Using the code in Fig. 1, I was able to calculate and plot the definite and indefinite integrals and plot the given data as seen in Fig. 2 and Fig. 3, respectively. The definite integral was computed using the trapezoidal rule and the indefinite integral implemented the cumulative trapezoidal method. In Fig. 4, the two curves are overlaid in the same plot, the green being the definite and purple being the indefinite. Looking at both curves in the same plot (Fig 4) we can see that the definite (green) curve is derivative of the indefinite (purple) curve. The results can also be compared numerically by looking at Fig. 5. The values of areas under the curves of the definite was printed to the screen and the indefinite is shown as the last number of the array at the bottom of the image. The areas under the curves were estimated to the same out to five decimal places. Comparing this to the square of pi, as we did in the in the previous project using Monte Carlo methods, is also the same value as the definite and indefinite integrals out to five decimals. I would conclude that this integration estimate was rather accurate based on a random, discontinuous set of data. As mentioned previously, using Riemann approximation is another method of numerical integration. However, it will be less accurate because the method sloppily approximates using rectangles instead of trapezoids, which leave more unaccounted for area under a curve. I would see a similar problem in the rectangle rule or midpoint formula. There are many ways to use numerical integration, as learned in my previous numerical analysis class.

E-mail: rlcorl0436@ung.edu



```
@author: rebeccacorley
"""
#import required libraries
import numpy as np
import matplotlib.pyplot as plt
from scipy import integrate


#import data and print
print("The raw data is: \n")
function_real = np.genfromtxt('function_real.dat', delimiter = ' ')
print(function_real, '\n') #prints raw data

#convert raw data to a list to allow sorting/integrating - it is stored as an array because of genfromtxt
print("The raw data is converted to a list")
f = function_real.tolist()

#f is a list of (x,y) values [list of lists], must unzip
x,y = zip(*f)

#plot to check
plt.scatter(x,y, marker="*", color="g")
plt.show()

#sort raw data from smallest to largest
f.sort(key=lambda l: l[0])

#This function checks and prints some of the data at the beginning and end of data set for sorting
print("The data is sorted from least to greatest: \n")
for i in range(len(f)):
    if i<5 or len(f)-i<=5:
        print(i, ": ", f[i])
print()

#unzip the sorted list
x,y=zip(*f)

#line plot of definite integral to compare to previous plot
plt.plot(x,y, 'g-', label="Definite Integral")
plt.legend(loc="upper left")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

#use trapezodial rule to integrate the data
definite_int = np.trapz(y,x)
print("Value of definite integral: ", definite_int)
#use cumulative trapezoidal rule for indefinite integral
indefinite_int = integrate.cumtrapz(y, x, initial=0)
print("The indefinite integral gives: \n", indefinite_int)
#plot for indefinite integral
plt.plot(x, indefinite_int, color='purple', label="Indefinite Integral")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend(loc="upper left")
plt.show()

```

```
The raw data is:

[[3.94306289 2.14049101]
 [2.50571679 0.82445378]
 [5.06159156 4.60694361]
 ...
 [5.12060119 4.41875676]
 [6.1603585  0.08358195]
 [2.48449748 0.84434514]]

The raw data is converted to a list
The data is sorted from least to greatest:

0 :  [0.0004297995467065128, 8.317080228226484e-11]
1 :  [0.0005059148008437845, 1.268447865334194e-10]
2 :  [0.0010298312146178247, 1.171530260186308e-09]
3 :  [0.0020337334970633395, 8.44157655844772e-09]
4 :  [0.00236511045132898, 1.347915735655589e-08]
6278 :  [6.281014542488916, 3.0603479846499833e-05]
6279 :  [6.281693392702874, 1.2825595432651019e-05]
6280 :  [6.281817472454734, 1.1829563489254546e-05]
6281 :  [6.282661382166873, 1.724002891018491e-06]
6282 :  [6.283141551829324, 1.2786746913594436e-08]

Value of definite integral:  9.869572012301932
The indefinite integral gives:
 [0.00000000e+00 7.99269496e-15 3.48112694e-13 ... 9.86957201e+00
 9.86957201e+00 9.86957201e+00]
```

Figure 1. shows a screenshot of the code used to compute and plot the definite integral and plot the indefinite integral. The plots and values to the right are described in further detain in the next section.

4. FURTHER QUESTIONS

What if we wanted to preform numerical integration on a higher order than degree two polynomial? What kinds of methods could we implement other than MC methods? I came across Bayesian Quadrature, a statistical approach to computing integrals that I hope to explore in more detail. I have seen Bayesian statistics used all over physics, particularly in cosmology, so I am curious if they are similar.

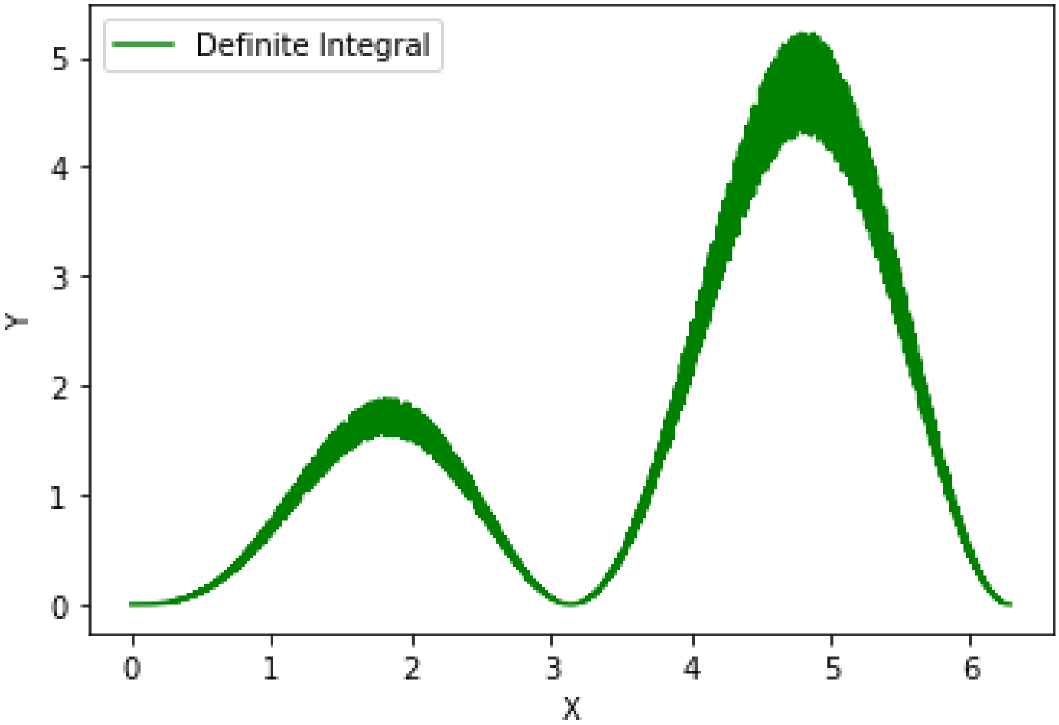


Figure 2. is the plot (green) of the definite integral. The X and Y axes correspond to data points provided in the random data set. This plot was created using the trapezoidal rule to approximate the area under the curve.

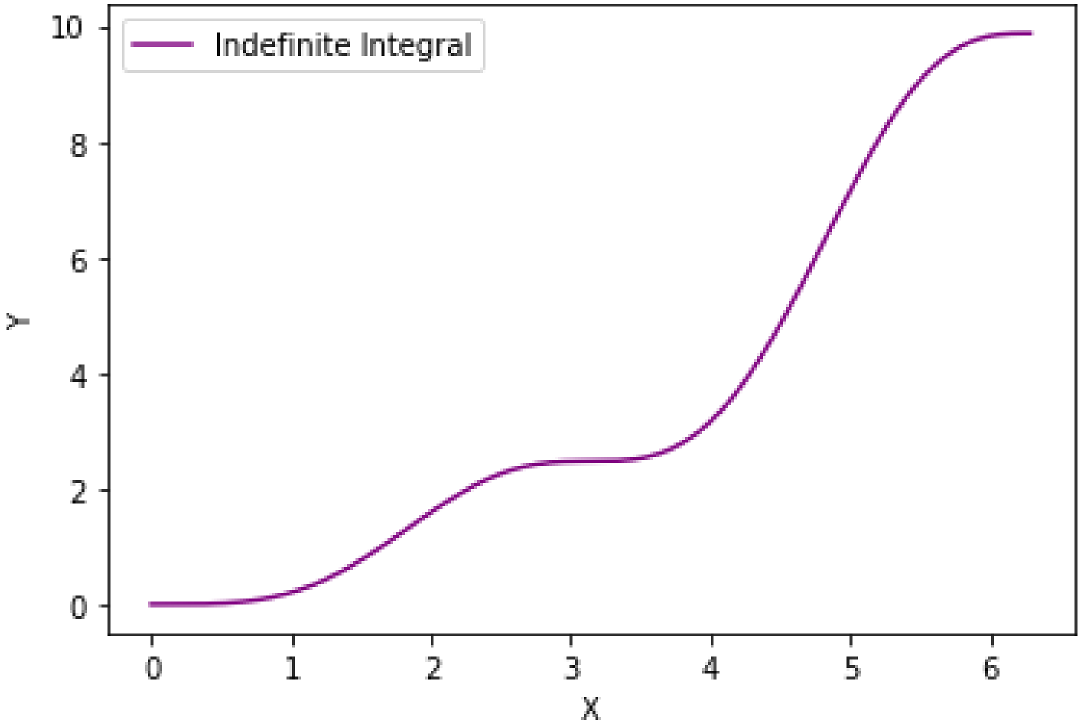


Figure 3. is the plot (purple) of the definite integral. The X and Y axes correspond to data points provided in the random data set. This plot was created using the cumulative trapezoidal rule to approximate the area under the curve.

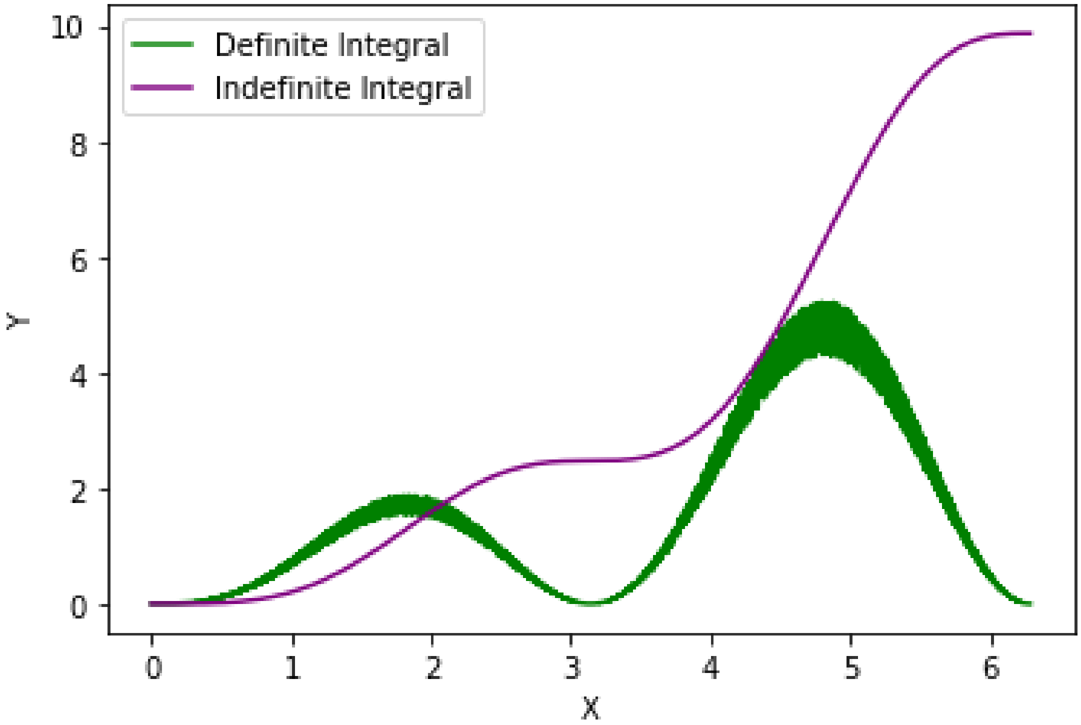
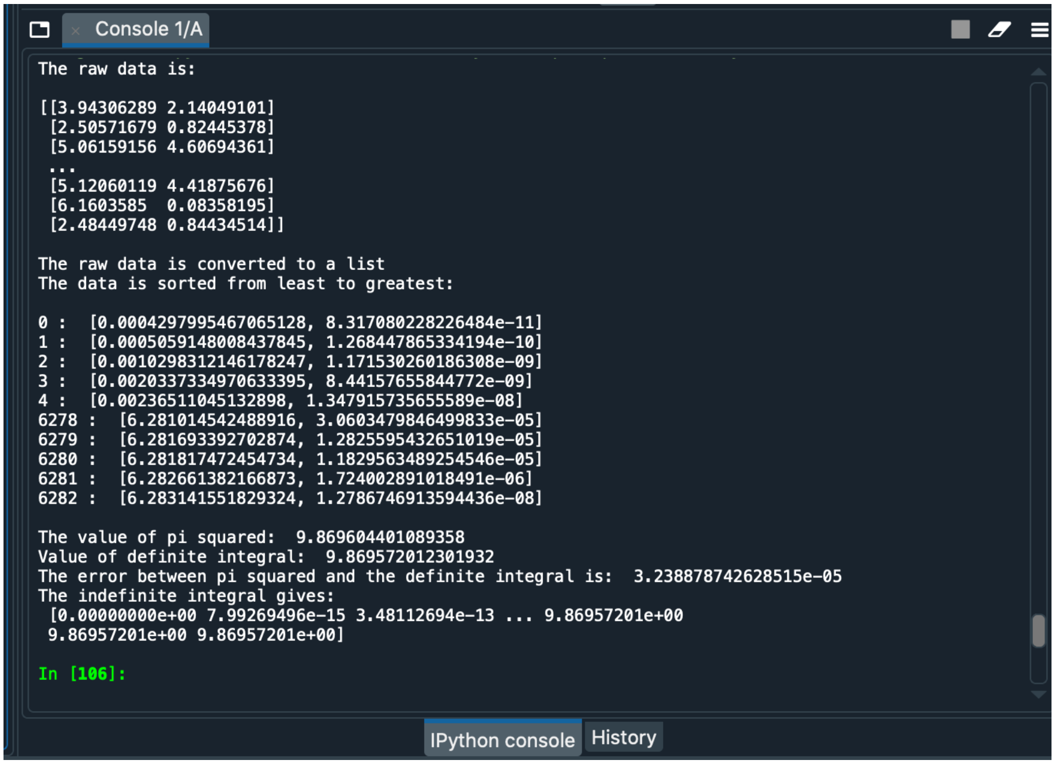


Figure 4. shows the definite (green) and indefinite (purple) curves on the same plot.



```
The raw data is:

[[3.94306289 2.14049101]
 [2.50571679 0.82445378]
 [5.06159156 4.60694361]
 ...
 [5.12060119 4.41875676]
 [6.1603585  0.08358195]
 [2.48449748 0.84434514]]

The raw data is converted to a list
The data is sorted from least to greatest:

0 :  [0.0004297995467065128, 8.317080228226484e-11]
1 :  [0.0005059148008437845, 1.268447865334194e-10]
2 :  [0.0010298312146178247, 1.171530260186308e-09]
3 :  [0.0020337334970633395, 8.44157655844772e-09]
4 :  [0.00236511045132898, 1.347915735655589e-08]
6278 :  [6.281014542488916, 3.0603479846499833e-05]
6279 :  [6.281693392702874, 1.2825595432651019e-05]
6280 :  [6.281817472454734, 1.1829563489254546e-05]
6281 :  [6.282661382166873, 1.724002891018491e-06]
6282 :  [6.283141551829324, 1.2786746913594436e-08]

The value of pi squared:  9.869604401089358
Value of definite integral:  9.869572012301932
The error between pi squared and the definite integral is:  3.238878742628515e-05
The indefinite integral gives:
 [0.00000000e+00 7.99269496e-15 3.48112694e-13 ... 9.86957201e+00
 9.86957201e+00 9.86957201e+00]

In [106]:
```

Figure 5. shows numerical results from definite and indefinite integration.