Yash Sharma

CSE13S, Winter 2021

Assignment 2 - WRITEUP.pdf

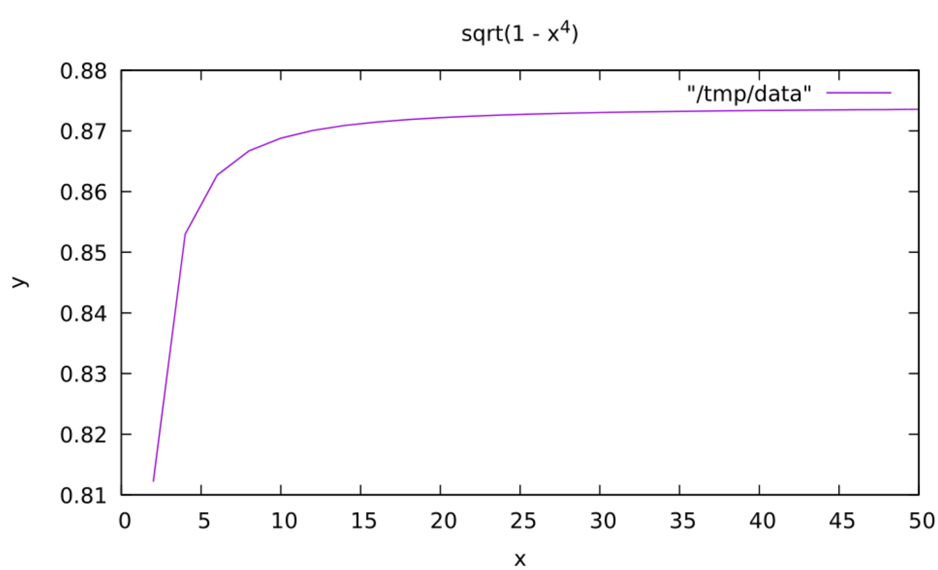
Introduction:

For this writeup, we'll be focusing on how to write a dedicated program, integrate, that computes the numerical integration of a function over a specified interval using the composite Simpson's rule. This program will use getopt() and a list of command line functions to execute the integration for each specified function. In addition, the program will allow us to set values for the lower bound and upper bound of a integral and how long we want to the integral to run (partitions).

The Integrate function computes the numerical integration of some function f over the interval [a,b]. This will be done with composite Simpson's ⅓ rule using n partitions. Within the function we are also using a function pointer (something that takes a single double as its sole argument and returns a double).

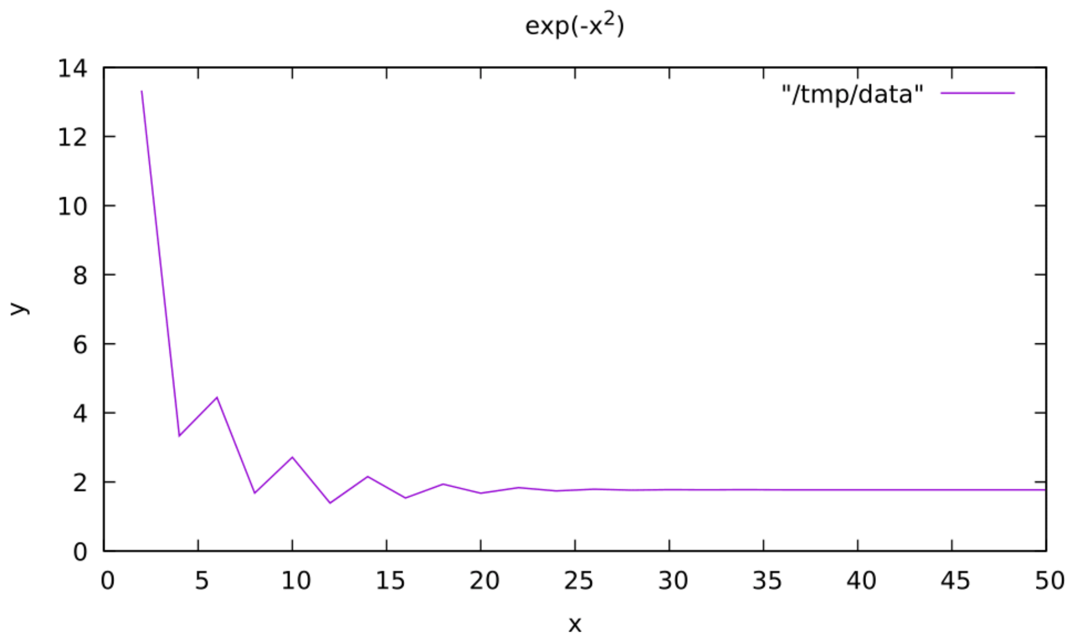
Integrate Function Graphs:

Integration of Sqrt(1-x^4):



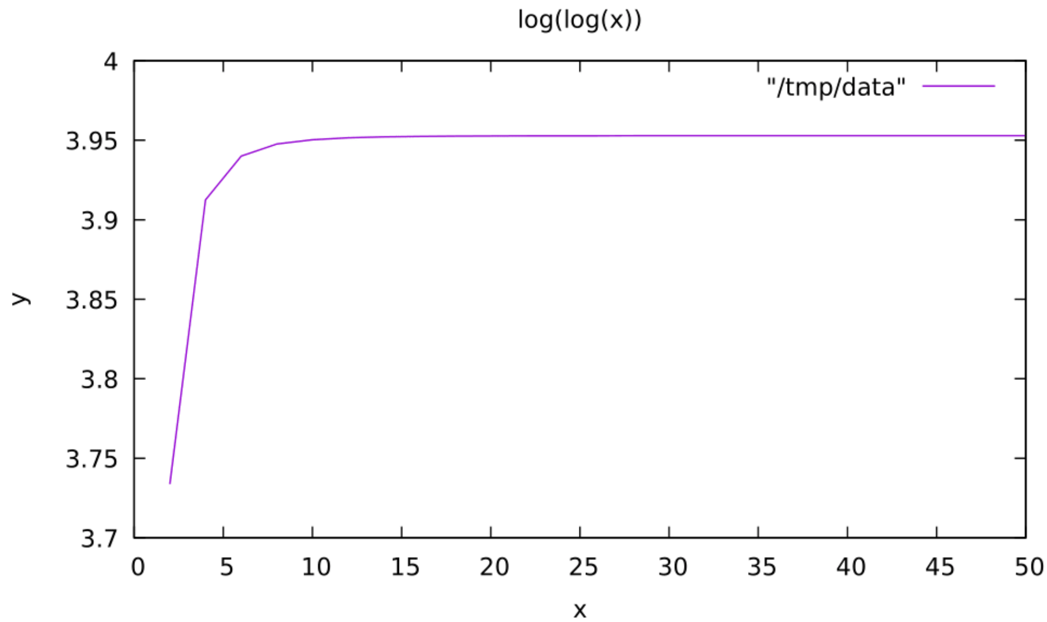
This was the data produced by the correlation between the x and y values of the integration of sqrt(1-x^4). To produce this graph, I wrote a while loop that checks whether the double input of x is greater than 1, then divide the integer by 4 and multiply another variable by 2. I then proceeded to write another while loop that checks whether the absolute value of y-z is greater than epsilon. When this condition is met, the value of y*f is returned. At the end, all this data is sent to a separate output file which then GNUPlot uses to create a graph. GNU Plot takes in the xlabel, ylabel and plots the graph according to the output file (sqrt.pdf). From this graph, we can conclude that as the # of partitions increase, the value of x and y start stabilizing.

Integration of Exp(-x^2):



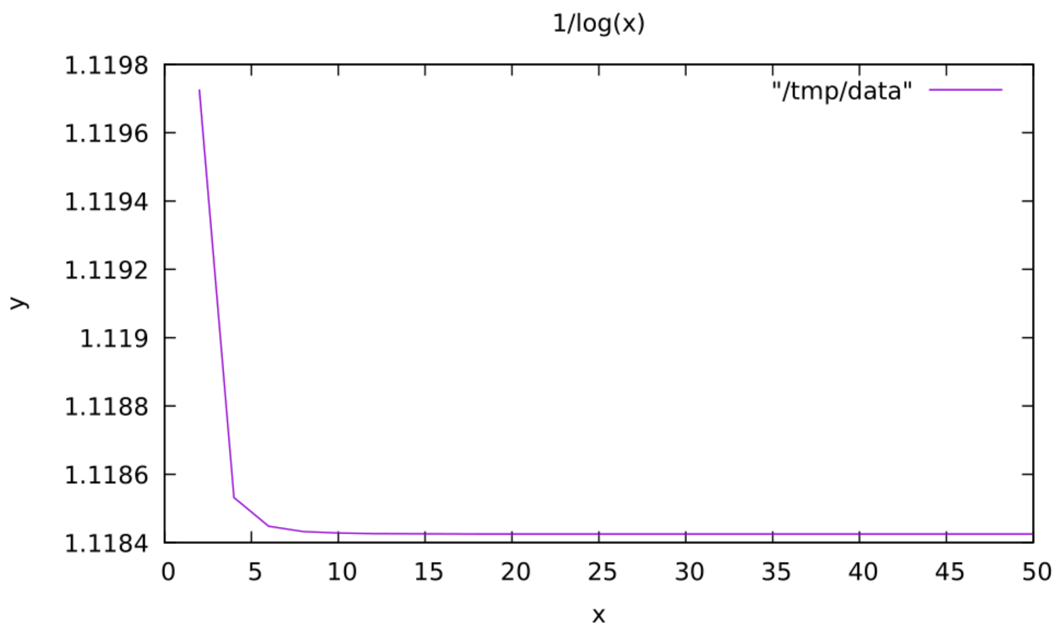
This was the data produced by the correlation between x and y for the integration of e^-x^2. To produce this graph, I initialized variables trm, sum, and k. These variables were then used within a while loop which computes the value of trm and sum if the term is greater than epsilon (e). Alongside this, I had an if statement that checks if the value of x is greater than 0, it will return sum or else 1 divided by sum. After this, the data is then sent through GNUPlot which plots this data based on the xlabel, ylabel, and a yrange [0:50] in the data file. From this graph, we can conclude that the exponential function is decreasing quickly in value on the x and y axes shows that the floating point numbers are not visible.

Integration of Log(log(x)):



This was the data produced by the correlation between x and y for the integration of log(log(x)). To produce this graph, I initialized variables y, p, e, and f. These variables were then used within a while loop which computes the value of x and f if the value of x is greater than e. Alongside this, I had another while loop that checks whether the absolute value of p-x is greater then epsilon. Once this condition is met, the value of y and p is calculated and f + y is returned to generate the values for log(log(x)). After this, the data is then sent through GNUPlot which plots this data based on the xlabel, ylabel, and a yrange [0:50] in the data file. From this graph, we can conclude that the logarithmic function is increasing quickly in value on the x and y axes shows that the floating point numbers are not visible and that the number of partitions increasing leads to a stabilization in x&y values.

Integration of 1/log(x)



This was the data produced by the correlation between x and y for the integration of 1/(log(x)). To produce this graph, I initialized variables y, p, e, and f. These variables were then used within a while loop which computes the value of x and f if the value of x is greater than e. Alongside this, I had another while loop that checks whether the absolute value of p-x is greater then epsilon. Once this condition is met, the value of y and p is calculated and f + y is returned to generate the values for log(log(x)). After this, the data is then sent through GNUPlot which plots this data based on the xlabel, ylabel, and a yrange [0:50] in the data file. From this graph, we can conclude that the logarithmic function decreases quickly in value on the x and y axes shows that the floating point numbers are not visible.

Conclusion:

From the data shown here, we can draw 3 major conclusions:

1. The data is not all represented with just floating numbers
2. As the number of partitions increase, the value of x and y begins to stabilize.
3. If values of floating numbers are not accurate in the real world, this could lead to serious problems as floating points and double values help keep our values accurate to the highest precision possible.