Report

November 27, 2023

0.0.1 Prepared By

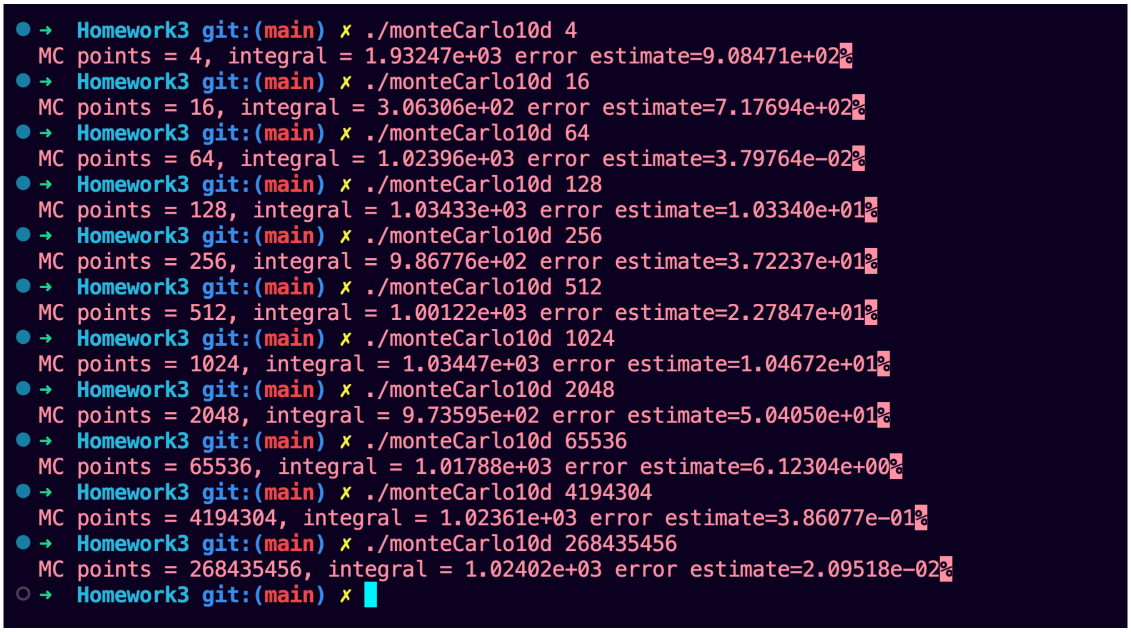
- Ayush Tiwari
- Kanishka Patre
- Sinchan Samajdar

The below screenshot solution is for Problem 1 As we can see the value of the integral is converging to 1024 and the error is decreasing as we keep on increasing the size of N.

Error estimate is calculated following way.

Error estimate=|I(4^k)-1024.0|

The code can be found in the file "monteCarlo10d.c".



```
● → Homework3 git:(main) ✗ ./monteCarlo10d 4
  MC points = 4, integral = 1.93247e+03 error estimate=9.08471e+02%
● → Homework3 git:(main) ✗ ./monteCarlo10d 16
  MC points = 16, integral = 3.06306e+02 error estimate=7.17694e+02%
● → Homework3 git:(main) ✗ ./monteCarlo10d 64
  MC points = 64, integral = 1.02396e+03 error estimate=3.79764e-02%
● → Homework3 git:(main) ✗ ./monteCarlo10d 128
  MC points = 128, integral = 1.03433e+03 error estimate=1.03340e+01%
● → Homework3 git:(main) ✗ ./monteCarlo10d 256
  MC points = 256, integral = 9.86776e+02 error estimate=3.72237e+01%
● → Homework3 git:(main) ✗ ./monteCarlo10d 512
  MC points = 512, integral = 1.00122e+03 error estimate=2.27847e+01%
● → Homework3 git:(main) ✗ ./monteCarlo10d 1024
  MC points = 1024, integral = 1.03447e+03 error estimate=1.04672e+01%
● → Homework3 git:(main) ✗ ./monteCarlo10d 2048
  MC points = 2048, integral = 9.73595e+02 error estimate=5.04050e+01%
● → Homework3 git:(main) ✗ ./monteCarlo10d 65536
  MC points = 65536, integral = 1.01788e+03 error estimate=6.12304e+00%
● → Homework3 git:(main) ✗ ./monteCarlo10d 4194304
  MC points = 4194304, integral = 1.02361e+03 error estimate=3.86077e-01%
● → Homework3 git:(main) ✗ ./monteCarlo10d 268435456
  MC points = 268435456, integral = 1.02402e+03 error estimate=2.09518e-02%
○ → Homework3 git:(main) ✗
```

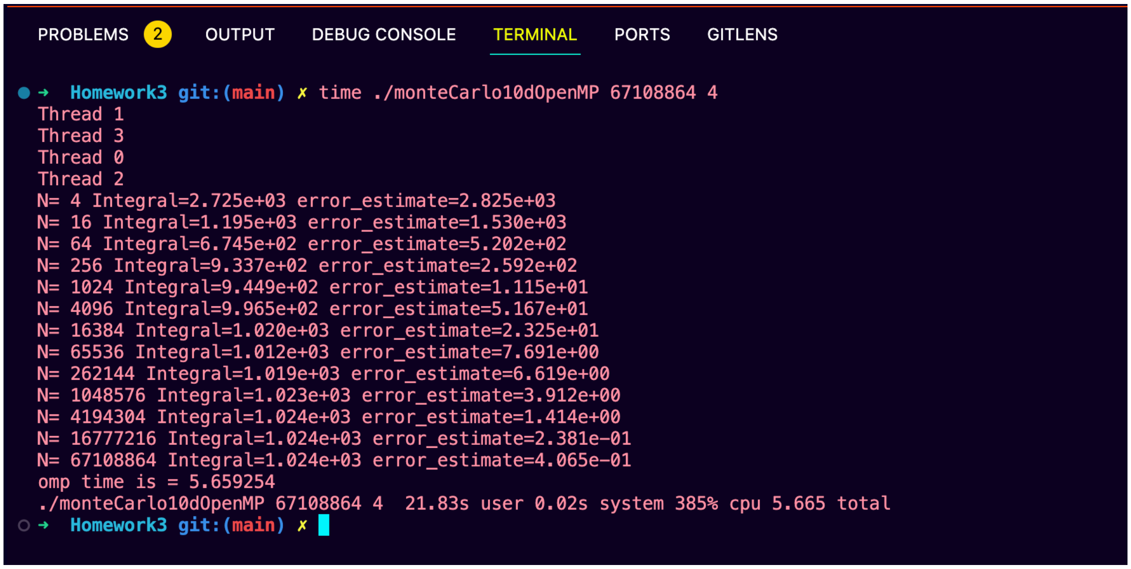
The below screenshot is solution for problem 2

a.) We have paralleize the for-loop and also there is evidence of 4 threads running.

b.)For the b part we have used omp__get__wtime() at the beginning and end of the program and startingtime is being subtracted by the ending time at the end of the program.The output can be seen in the screenshot

omp time is=5.659254 seconds for 4 threads and N=67108864 unix_time is=5.665 seconds

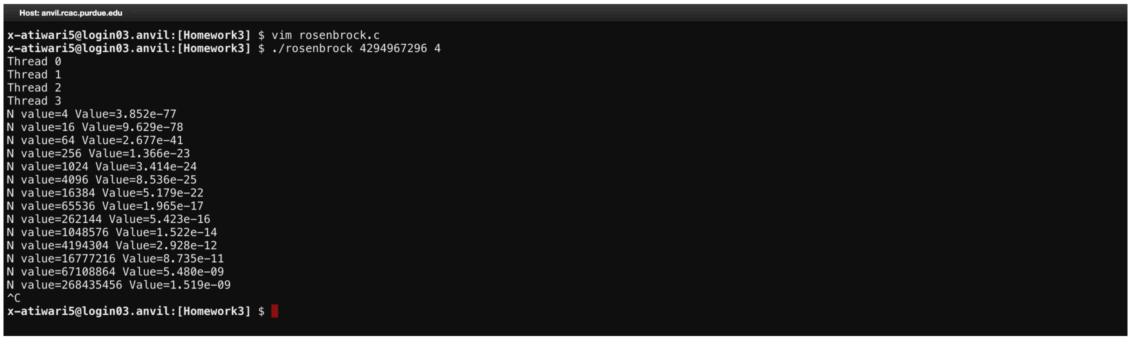
The code for this problem can be found in “monteCarlo10dOpenMP.c”



```
● ➜  Homework3 git:(main) ✗ time ./monteCarlo10dOpenMP 67108864 4
  Thread 1
  Thread 3
  Thread 0
  Thread 2
  N= 4 Integral=2.725e+03 error_estimate=2.825e+03
  N= 16 Integral=1.195e+03 error_estimate=1.530e+03
  N= 64 Integral=6.745e+02 error_estimate=5.202e+02
  N= 256 Integral=9.337e+02 error_estimate=2.592e+02
  N= 1024 Integral=9.449e+02 error_estimate=1.115e+01
  N= 4096 Integral=9.965e+02 error_estimate=5.167e+01
  N= 16384 Integral=1.020e+03 error_estimate=2.325e+01
  N= 65536 Integral=1.012e+03 error_estimate=7.691e+00
  N= 262144 Integral=1.019e+03 error_estimate=6.619e+00
  N= 1048576 Integral=1.023e+03 error_estimate=3.912e+00
  N= 4194304 Integral=1.024e+03 error_estimate=1.414e+00
  N= 16777216 Integral=1.024e+03 error_estimate=2.381e-01
  N= 67108864 Integral=1.024e+03 error_estimate=4.065e-01
  omp time is = 5.659254
  ./monteCarlo10dOpenMP 67108864 4  21.83s user 0.02s system 385% cpu 5.665 total
○ ➜  Homework3 git:(main) ✗ 
```

Problem 3

a.) Rosenbrock function is being implemented and tested on Anvil for N=4294967296 and #of thread=4

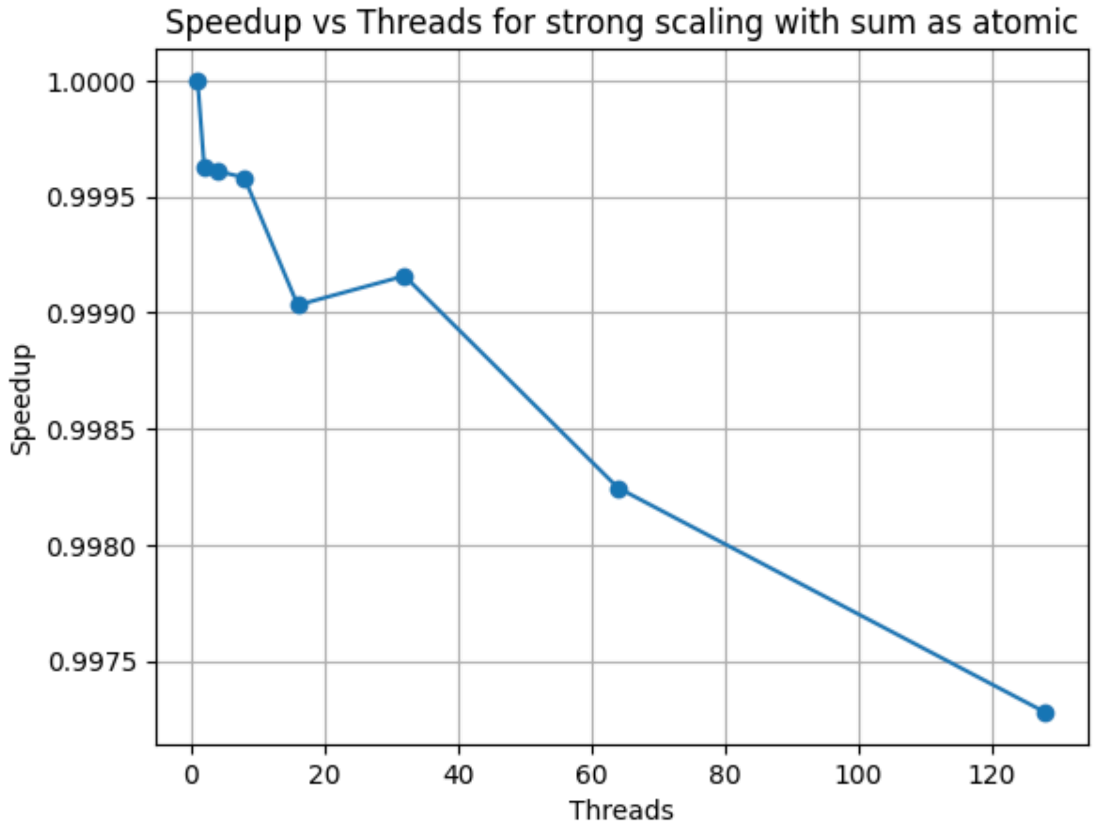


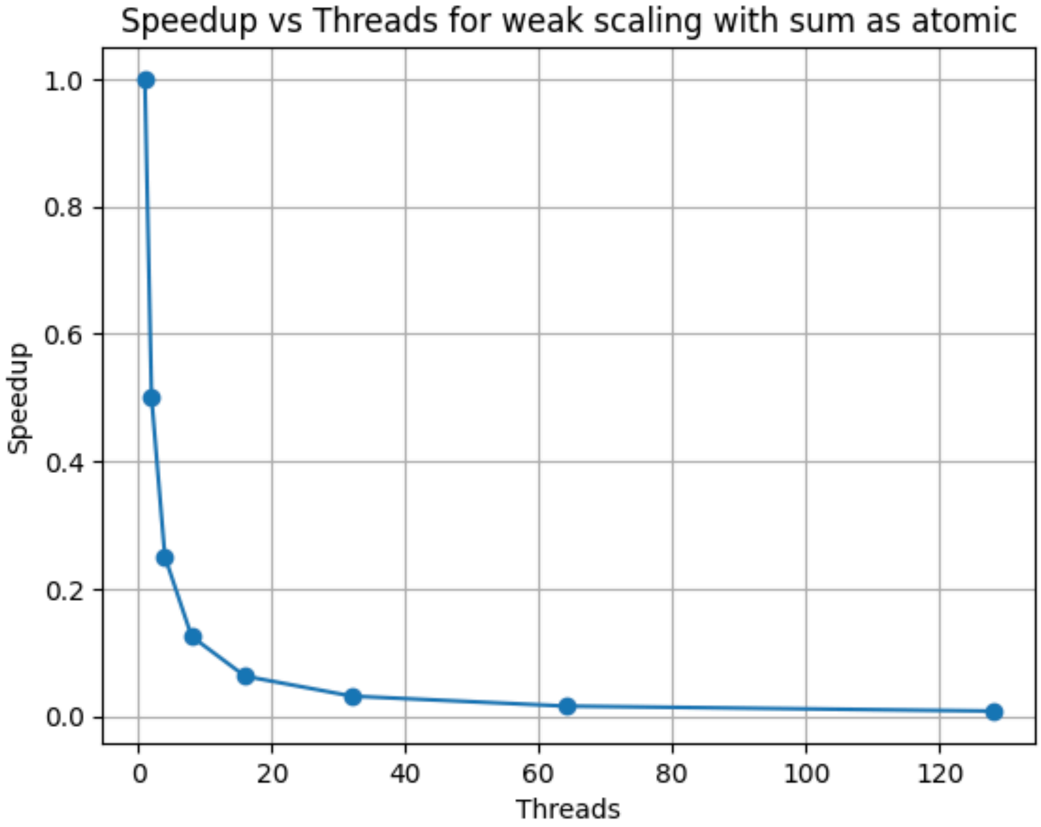
```
x-atiwari5@login03.anvil:[Homework3] $ vim rosenbrock.c
x-atiwari5@login03.anvil:[Homework3] $ ./rosenbrock 4294967296 4
Thread 0
Thread 1
Thread 2
Thread 3
N value=4 Value=3.852e-77
N value=16 Value=9.629e-78
N value=64 Value=2.677e-41
N value=256 Value=1.366e-23
N value=1024 Value=3.414e-24
N value=4096 Value=8.536e-25
N value=16384 Value=5.179e-22
N value=65536 Value=1.965e-17
N value=262144 Value=5.423e-16
N value=1048576 Value=1.522e-14
N value=4194304 Value=2.928e-12
N value=16777216 Value=8.735e-11
N value=67108864 Value=5.480e-09
N value=268435456 Value=1.519e-09
^C
x-atiwari5@login03.anvil:[Homework3] $ 
```

b)

Method 1 with sum as atomic

Strong and weak scaling test is being performed with sum as atomic. But couldn’t able to achieve the desired speedup. It can be seen from the graph that, with the increase of cores time also increases.

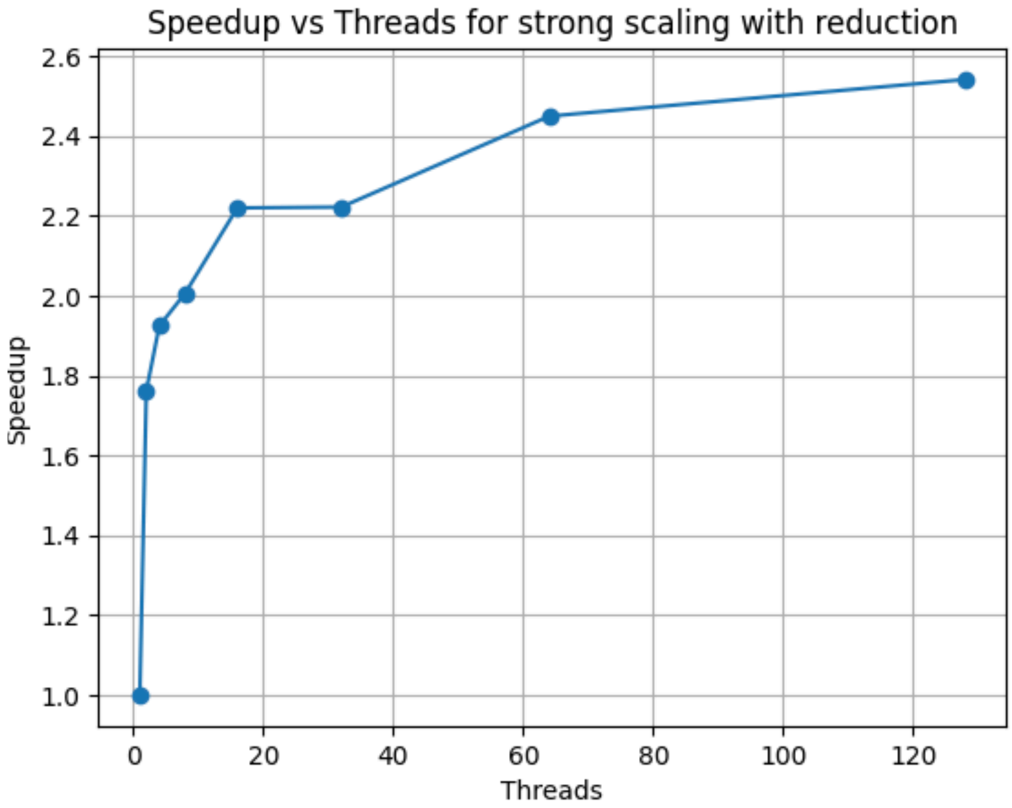

Speedup vs Threads for strong scaling with sum as atomic


Speedup vs Threads for weak scaling with sum as atomic

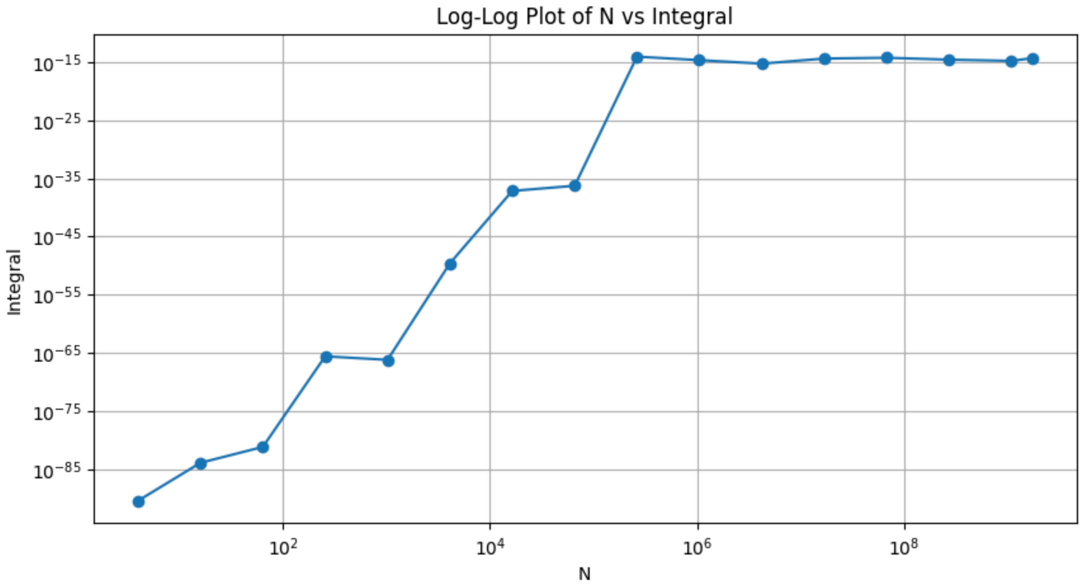
Method 2 with the usage of reduction for sum

Since the sum is shared among threads, so apart from using atomic and critical to achieve this, we can also use reductions. By using reductions all the threads declare a private variable and then perform reduction over all threads private variables.

As we can see that desired speedup is achieved by using reduction for sum, For the case of strong scaling.



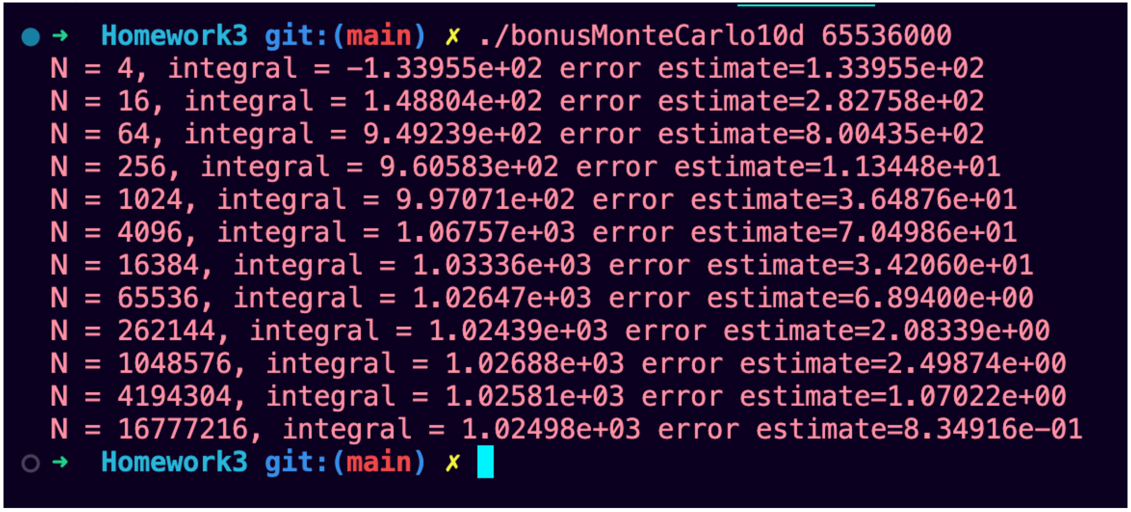
c.)After running code on anvil with 17 billion points below is the loglog graph that is obtained.We can see that after certain point of N the integral value is coming to constant.The data file obtained could be found in file “3c_anvil_data.dat” inside data folder



The code for probelm 3 could be found in file “rosenbrock.c”

Bonus Questions

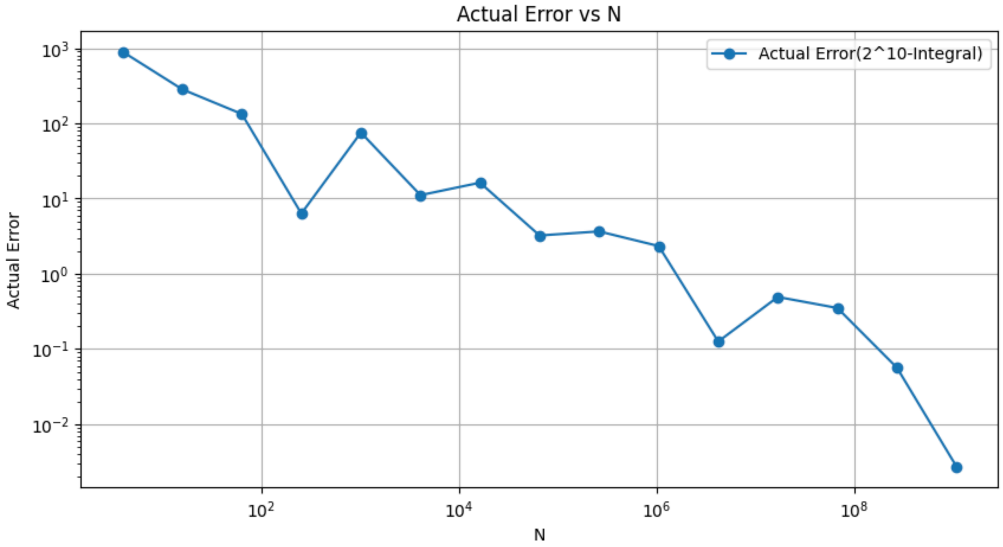
a.)Running the serial code with different values of N and calculating the error by saving the previous integral for every 4^k value. Below is the result obtained.

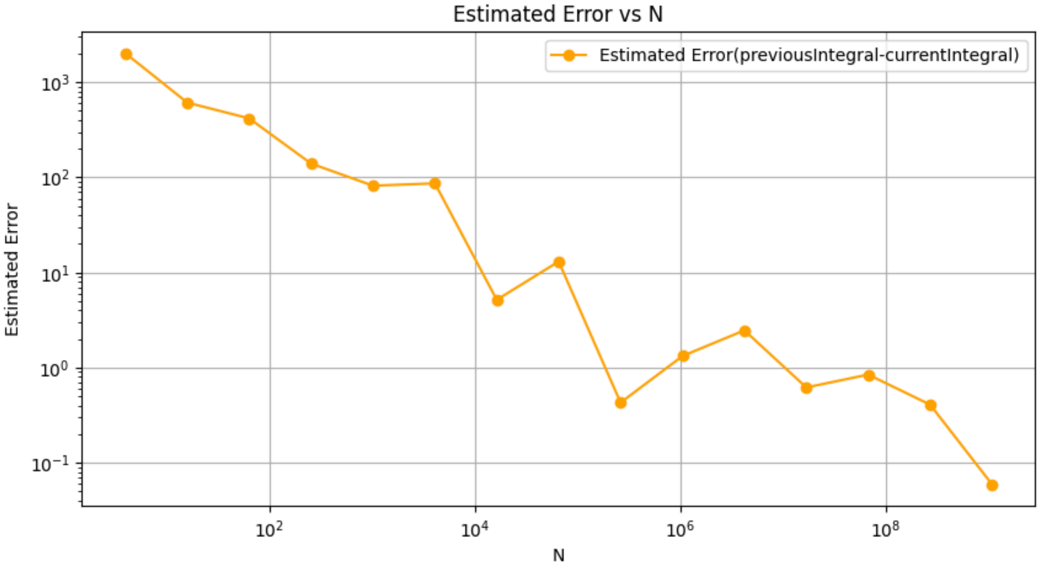


```
● ➜  Homework3 git:(main) ✗ ./bonusMonteCarlo10d 65536000
  N = 4, integral = -1.33955e+02 error estimate=1.33955e+02
  N = 16, integral = 1.48804e+02 error estimate=2.82758e+02
  N = 64, integral = 9.49239e+02 error estimate=8.00435e+02
  N = 256, integral = 9.60583e+02 error estimate=1.13448e+01
  N = 1024, integral = 9.97071e+02 error estimate=3.64876e+01
  N = 4096, integral = 1.06757e+03 error estimate=7.04986e+01
  N = 16384, integral = 1.03336e+03 error estimate=3.42060e+01
  N = 65536, integral = 1.02647e+03 error estimate=6.89400e+00
  N = 262144, integral = 1.02439e+03 error estimate=2.08339e+00
  N = 1048576, integral = 1.02688e+03 error estimate=2.49874e+00
  N = 4194304, integral = 1.02581e+03 error estimate=1.07022e+00
  N = 16777216, integral = 1.02498e+03 error estimate=8.34916e-01
○ ➜  Homework3 git:(main) ✗ 
```

code file->“bonusMonteCarlo10.c”

b.)Plotting the graph for Actual error and estimated error. Below is the graph obtained





data file is in "./data/bonus_b_acutualerror_esterror" The code of graph could be found in "graphs.ipynb"

c.)After running "monteCarlo10dOpenMP.c" code below is the result obtained

- N=4 Integral=1.918e+03 actual_estimate=8.939e+02 estimated_error=2.018e+03
- N=16 Integral=1.307e+03 actual_estimate=2.833e+02 estimated_error=6.105e+02
- N=64 Integral=8.908e+02 actual_estimate=1.332e+02 estimated_error=4.166e+02
- N=256 Integral=1.030e+03 actual_estimate=6.329e+00 estimated_error=1.396e+02
- N=1024 Integral=9.486e+02 actual_estimate=7.535e+01 estimated_error=8.168e+01
- N=4096 Integral=1.035e+03 actual_estimate=1.111e+01 estimated_error=8.646e+01
- N=16384 Integral=1.040e+03 actual_estimate=1.627e+01 estimated_error=5.158e+00
- N=65536 Integral=1.027e+03 actual_estimate=3.232e+00 estimated_error=1.304e+01
- N=262144 Integral=1.028e+03 actual_estimate=3.661e+00 estimated_error=4.282e-01
- N=1048576 Integral=1.026e+03 actual_estimate=2.338e+00 estimated_error=1.322e+00
- N=4194304 Integral=1.024e+03 actual_estimate=1.259e-01 estimated_error=2.464e+00

- N=16777216 Integral=1.024e+03 actual_estimate=4.923e-01 estimated_error=6.182e-01
- N=67108864 Integral=1.024e+03 actual_estimate=3.513e-01 estimated_error=8.437e-01
- N=268435456 Integral=1.024e+03 actual_estimate=5.651e-02 estimated_error=4.078e-01
- N=1073741824 Integral=1.024e+03 actual_estimate=2.741e-03 estimated_error=5.925e-02

[ ]: