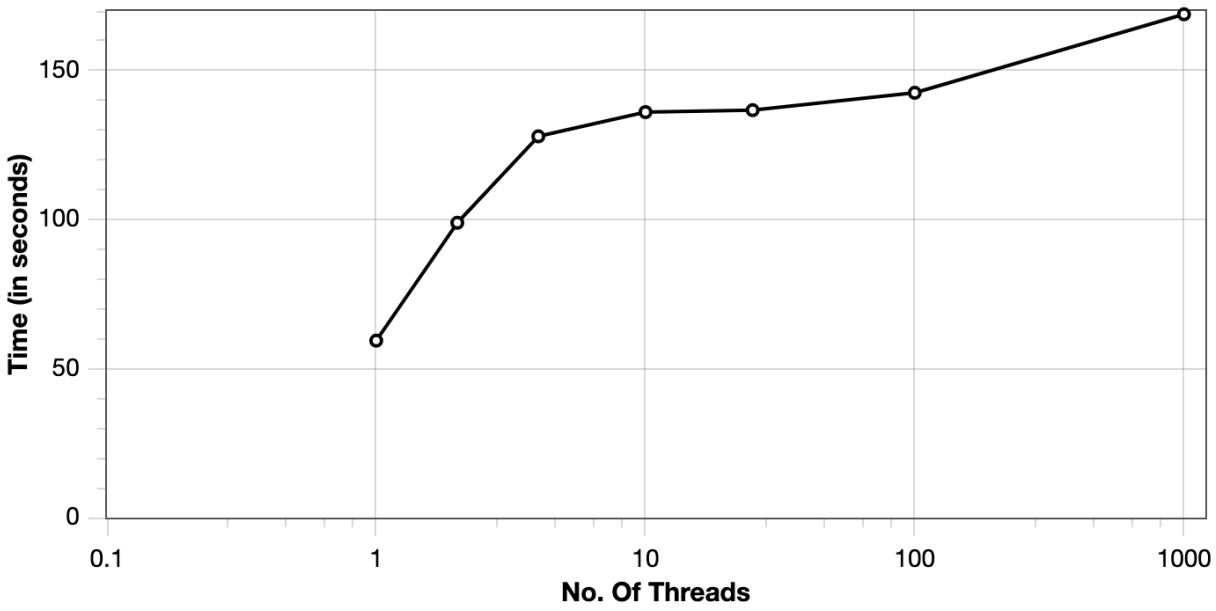
Summary

The purpose of this assignment is to use the application of multithreading to make the program run faster, especially to make the nested for loop provided in the determineAverageAngularDistance run faster and more efficiently while calculating the same value for mean, min, and max. The code should also provide a timing function, which calls any of the time functions to determine how long the program took to run. The user can run 1, 2, 4, 10, 25, 100, or 100 threads, and the program should print the same mean, min, and max values and should also print the time it took to run.

Timing Method

I chose to use the timing method using the clock() method, where I calculate the time that the program took to execute. It only required the use of time.h library, which was included in the main file. I used the clock() function at the beginning of the main and used clock() at the end of the function. Then I printed the value at the end to show how the function executes.

No. of Threads	Time (in seconds)
1	59.387163
2	98.913418
4	127.932692
10	136.05035
25	136.711563
100	142.531925
1000	168.888149



Anomalies

There have been multiple anomalies in this threading application. Ideally, the time should have decreased with an increase in the number of threads. But, in this scenario, we see that there is a significant increase in time with the increasing number of threads. This can be seen when the number of threads goes from 1 to 2, and the time it takes goes up significantly, from 58 seconds to 98 seconds. This is likely due to the limits that codespace has as it doesn't offer as much speed or memory as compared to running on a local machine. There is also a lot of shared memory that goes behind codespace and running a threaded program from it could lead to anomalies such as higher running time with a higher amount of threads.

Conclusion

The optimal number of threads required for this application is 1 as the program finishes its execution in approximately 58 seconds, much faster compared to any other threads in the program, and also takes the least amount of memory and CPU power to execute. Although this is not the ideal solution for a multithreading program, however, in this case, several limitations by the codespace environment require the program to have its best output by using only 1 thread.