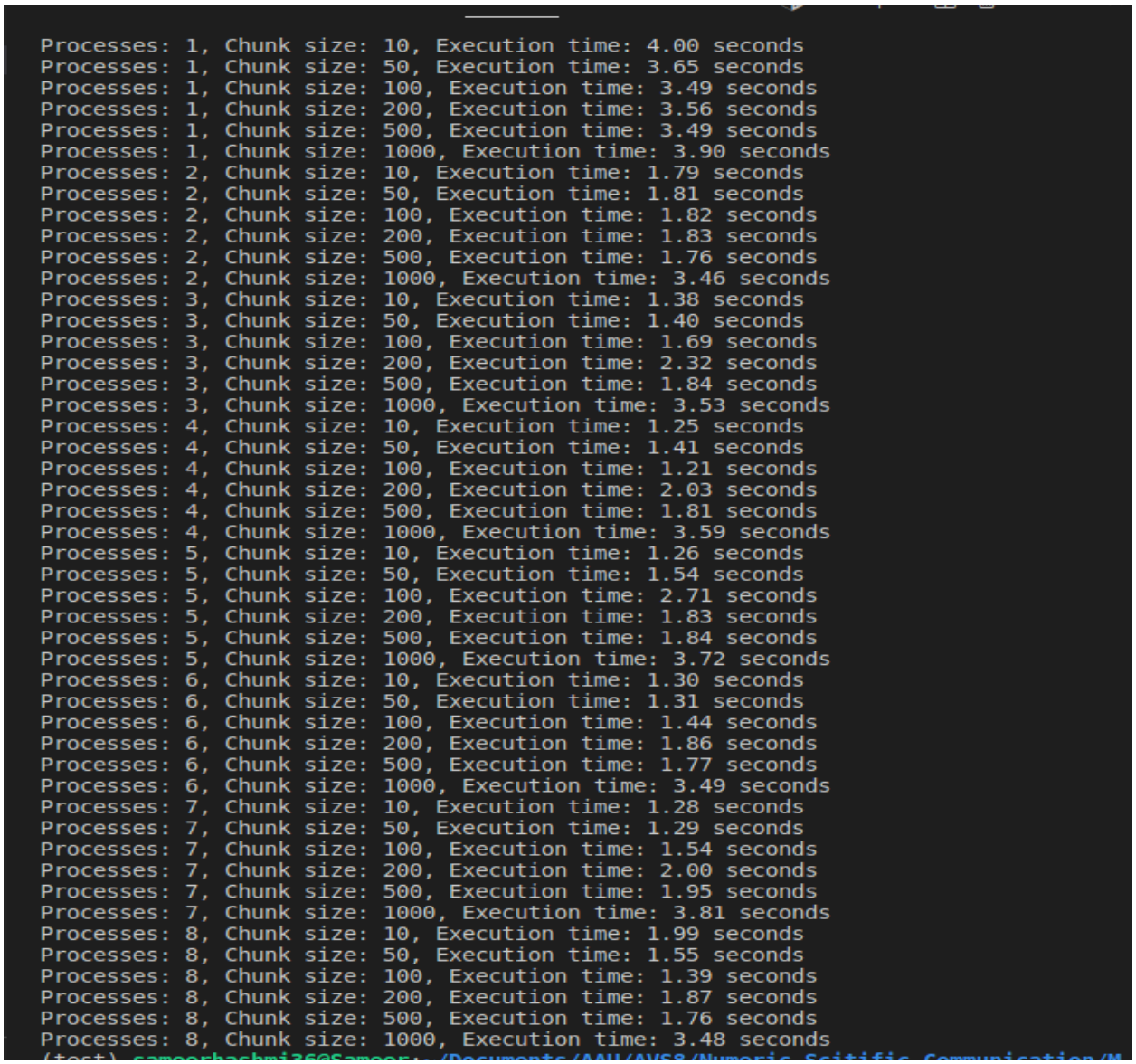
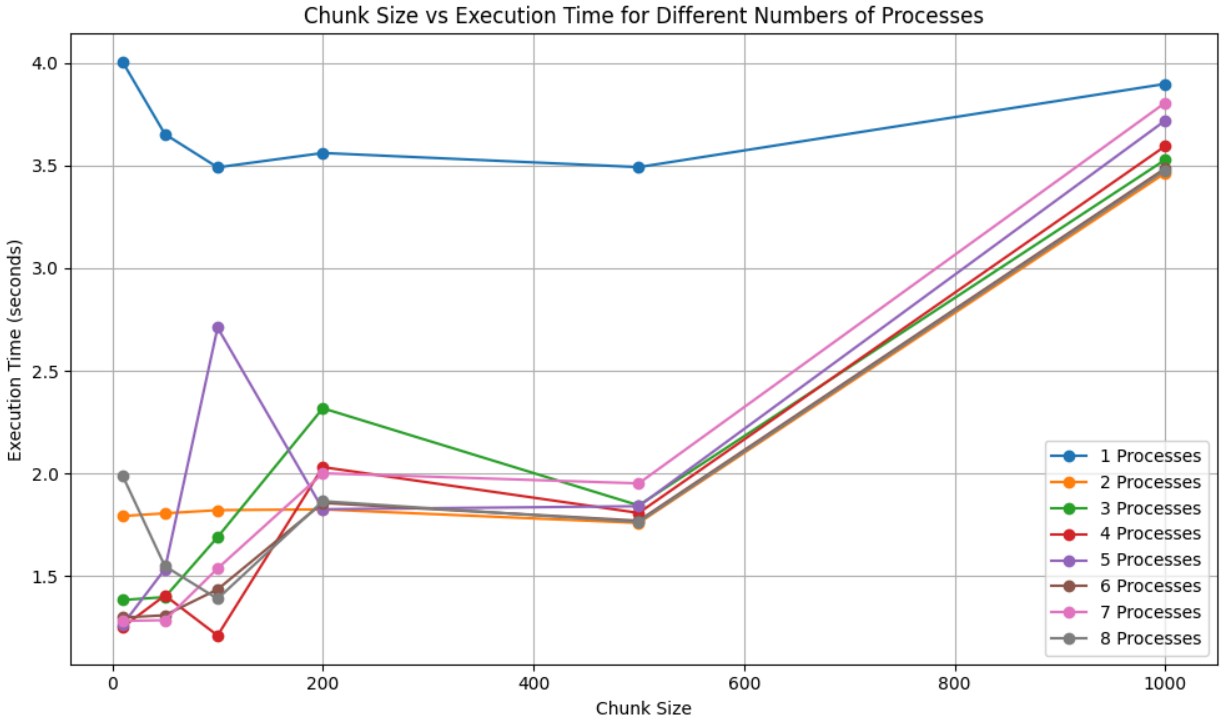
Different Techniques for optimizing Mandelbrot implementation:

- The fastest technique is the Numba-optimized Mandelbrot implementation, which took 1.12 seconds to execute.
- Following closely behind is the multiprocessing-based Mandelbrot implementation, which completed in 1.28 seconds.
- The naive Mandelbrot implementation, though straightforward, took 3.21 seconds to execute, making it slower than the optimized approaches.
- The NumPy-based Mandelbrot implementation, while efficient, completed in 3.40 seconds, making it slightly slower than the naive approach but faster than the multiprocessing implementation.

Different execution time and speed-up for different number of processes:





1. For 1 process: Chunk size of 100 exhibits the lowest execution time (3.49 seconds).
2. For 2 processes: Chunk size of 500 shows the lowest execution time (1.76 seconds).
3. For 3 processes: Chunk size of 10 yields the lowest execution time (1.38 seconds).
4. For 4 processes: Chunk size of 100 displays the lowest execution time (1.21 seconds).
5. For 5 processes: Chunk size of 10 results in the lowest execution time (1.26 seconds).
6. For 6 processes: Chunk size of 50 demonstrates the lowest execution time (1.30 seconds).
7. For 7 processes: Chunk size of 10 showcases the lowest execution time (1.28 seconds).
8. For 8 processes: Chunk size of 100 exhibits the lowest execution time (1.39 seconds).

Comparison of Execution Time and Speed-up for Different Number of Processes:
As the number of processes increases, the execution time generally decreases. However, beyond a certain point, increasing the number of processes may lead to diminishing returns in terms of speed-up. The optimal number of processes depends on the workload and available hardware resources. The choice of chunk size significantly impacts the overall execution time and efficiency of parallel processing.