Lab 7 MPI Documentation

Times for Lab5 with 4 threads: 10000 * 10000 are the sizes of the polynomial.

1. Sequential classic: 2065 ms
2. Sequential Karatsuba: 2091 ms
3. Parallel classic: 822 ms
4. Parallel Karatsuba: 625 ms

Times for Lab7: 10000 * 10000 are the sizes of the polynomial

1. Sequential with 4 threads: 1033 ms
2. Sequential with 8 threads: 812 ms
3. Karatsuba with 4 threads: 888 ms
4. Karatsuba with 8 threads: 873 ms

Regular algorithm

For the regular algorithm, we have a master node which will send the 2 polynomials and the start and end index to each of the 8 workers, which will create a result polynomial and send it back to the master which will add the results to obtain the final result.

Observations: - The MPI implementation is still much faster that the sequential one, but it is a bit slower than the one from Lab 5, as the polynomials are sent back and forth between nodes.

Karatsuba algorithm

As this is a divide and impera algorithm, which is a recursive algorithm, we need to think about when to stop using parallelism (reaching a maximum depth). The implementation has a master node, which will start the karatsuba algorithm by distributing the 3 necessary computations to 3 nodes, which will send back the results. Each of these 3 nodes will create a thread pool for other 3 computations, and this would be my max depth as I do not have so much computing power, but if we use multiple performant pcs, we would have better performance.

Observations: - The MPI implementation is still faster that the sequential one, but it is slower than the one from Lab 5, as transfering polynomials is still complicated and time consuming.