Project 1

Overlap of communication and computation

For this project, the process 0 generates the first 5000 data points and starts sending them with the non-blocking method *Isend.* While the process 0 is sending the data points, it is simultaneously generating the other 5000 data points, and when they are done, process 0 proceeds to calculate the second half (50 elements) of the *row\_sum* data*.* When process 0 finish calculating its part of the *row\_sum­,* the process waits for the ­*Isend­* call to finish.

On the other hand, process 1 uses the blocking method *Recv* to get the data that is being sent from process 0 with *Isend*. When it finally receives everything, process 1 proceeds to calculate 25 of the 50 *row\_sum* data that it needs to calculate. When it is done calculating these first 25 elements, process 1 calls the non-blocking method *Isend*, which will start to transfer the first 25 elements back to process 0. While this is happening, it will proceed to calculate the other 25 remaining elements from *row\_sum*. When it finishes calculating the remaining elements, process 1 calls the *Wait* method to block and wait for the previous called *Isend* method. When *Isend* finally finishes, process 1 proceeds and then calls the blocking method *Send*, which will send the remaining 25 calculated items from *row­­\_sum­* back to process 0.

Lastly, after process 0 finishes waiting to send its data to process 1 with the *Isend* method and then *Wait* method, it contains two consecutive *Recv* function calls, which are used to receive the first 25 items from process 1, and then the remaining 25 items from process 1. When process 0 receives all 50 elements back from process 1, process 0 proceeds to print the entire results, which consists of the first half of the *row­\_sum* that was calculated by process 1, and the second half of it that was calculated by itself (process 0).

In conclusion, to achieve an overlap of communication and computation in process 1, the *row\_sum­* buffer was split in half. While one half was being sent, the other half was being computed. Thus, achieving the desired overlap.