Assignment 2 Technical Document

The technical idea for parallelizing the given algorithm using MPI involves splitting the work of the two for-loops across multiple processes to reduce the overall computation time. Here's a detailed description of the approach:

Initialization

- The MPI environment is initialized at the beginning of the program using `MPI_Init`.
- Each process in the MPI program is assigned a unique rank, obtained using `MPI_Comm_rank`.
- The total number of processes is determined using `MPI_Comm_size`.

Data Distribution

- The iteration space of the two for-loops is divided among the available processes. This is done by assigning each process a range of iterations to execute. For example, in the first for-loop, each process calculates its own range of iterations based on its rank and the total number of processes.

Parallel Computation

- In the first for-loop, each process iterates over its assigned range of rows in the matrix and updates the `output` array based on the algorithm's logic. This loop does not require communication between processes since each iteration is independent.
- In the second for-loop, the algorithm checks for the least value in the `output` array and updates the array based on the least value found. This loop requires communication between processes to determine the global least value and to update the `output` array accordingly.

Communication

- To find the global least value, each process first finds the local least value within its assigned range. Then, an `MPI_Allreduce` operation is used to find the minimum of these local least values across all processes. The result is the global least value, which is then broadcast to all processes using `MPI_Bcast`.
- After determining the global least value, each process updates its portion of the `output` array based on this value. Since the `output` array is distributed across processes, an `MPI_Allgather` operation is used to gather the updated portions of the array from all processes and distribute the complete updated array to all processes.

Finalization

- After the computation is complete, the MPI environment is finalized using `MPI_Finalize`.
- By following this approach, the algorithm efficiently utilizes multiple processes to parallelize the computation, resulting in reduced execution time. The use of MPI's collective communication operations ensures that all processes have the necessary data to perform their computations and that the final results are consistent across all processes.