Sheet 4 – Huan & Abraham (WinterSemester 2019/2020)

Part 3 – Global Operations

Ex4. Print in Order. We changed in MPI_Recv from AnySource to Ordered Rank I.
- Question: We created a funtion to print vectors of different lengths in order. For this
we exchanged the lenghts and then used received with the received length but we encounter deadlocks. To solve this issue, we found the global maximum of local vectors sizes and defined the max count in MPI_Receive to this number. The code seems to be stable.

Ex5. DebugVector. We print the vector stored in requested rank.
- Question: We tried to print the asked vector from the rank containing it but this lead to unexpected behaviour of the printing function. Some ranks were printing regardless of MPI_Barrier. We manage to correct this by sending the required data to RootRank and printing from there.

Ex6. Scalar Product. We distribute the vector among the ranks and let each rank calculate the local scalar product. Then, we reduce the result and print in it in rank 0. Rank 0 deals with the left over part.

Ex7. MinMax. We calculated the global max and min of a family of vectors and exchanged the first global max by a global min and the first global min by a global max.
- Question: We used MPI_Scatter to initialize the data via a global vector but we had issues with it.
We then define local vectors to avoid these issues. Why Scatter is not working?

Ex8. The required operation was performed.

Part 4 – Data Exchange

Ex9. Check VectorAccumulation and InnerProduct. We initialize a vector with different values (rank + 1) around the interface nodes in the subdomains and check the results with our pencil and paper calculations. We also initialize the vector with different functions and check that the graphs did not "jump" in the interface nodes, this was done by visual inspection. The results are satisfactory.

Ex10. VecAccuInt. We created a method for ParMesh called VecAccuInt that accumulates integer valued vectors and checked the result.

Ex11. Global Number Of Nodes. We created a method in ParMesh to recover the global number of nodes of the mesh by accumulating the local vectors initialized with 1 and then accumulted them which gives one in the internal nodes and the repetition of the interface nodes. Then, we transformed the vector by taking the inverses of these repetitions and made the sum of numbers. We reduce these local sums to get a global sum which by construction returns the global number of nodes of the non-distributed mesh.

Ex12. VectorAverage. We initialize a vector with different values (rank + 1) around the interface nodes in the subdomains and check the results with our pencil and paper calculations.

Ex13. Create Mesh and Run Code. We created a mesh with 6 subdomains in the form of a rectangle and runed the code succesfully. The funtions that were writen work well in this new domain as well.

Part 5 – Jacobi Iterative Solvers

Ex 14. JacobiSolverParallel. We created function JacobiSolveParralel that introduces the functions VectorAccumulation, ScalarProduct and AllReduction to get a global error of each iteration. This makes the iterations in each rank to be the same as well as avoiding the “jumps” in the interface nodes.-

- Question: We could not observe the jumps in the interface with the template provided online. The jumps could be observed by a version provided by Prof. Hasse but we could not find the meaninful difference that produced this jumps in the online version. Where is the problem?