ETH zürich

High-Performance Computing Lab for CSE **2020**

Student: Konrad Handrick Discussed with: -

Solution for Project 7 Due date: May 31, 2020, 12pm (midnight)

Project 7 – Numerical Mathematical Software for Extreme-Scale Science and Interactive Supercomputing with JupyterLab

1. Scientific Mathematical HPC Software Frameworks - The Poisson Equation [40 points]

I had a hard time installing all of the software I wanted to try but finally got PETSc to run and to interface with its python wrapper.

I used some example code found on bitbucket and added to it diagnostics to get loadup, communication and solution time. It works fairly well and gives the exact same results as the picture given in the project PDF.

It runs using ”mpirun -np $ranks python3 vers_petsc.py -ny $nx -ny $ny”. It performs fairly well and achieves linear scaling. The underlying parallel sparse matrix solver does a 1D decomposition of the matrices and solves with very low communication overhead.

I did try different programs also but I feel the python-petsc interface works well enough to fully solve the problem. Again, I really wanted to try out the solver on Euler but the centre is still down as of May 31st. That's why I'm leaving the table also empty and just putting in the scaling graphic as is.

Table 1: Wall-clock time (in seconds) and speed-up (in brackets) using multiple cores on Euler for solving the Poisson problem.

Problem	N	Number of Euler cores			
		1	8	16	32
Poisson	500^2				
Poisson	1000^2				
Poisson	1500^2				
Poisson	2000^2				
Poisson	2500^2				
Poisson	3000^2				

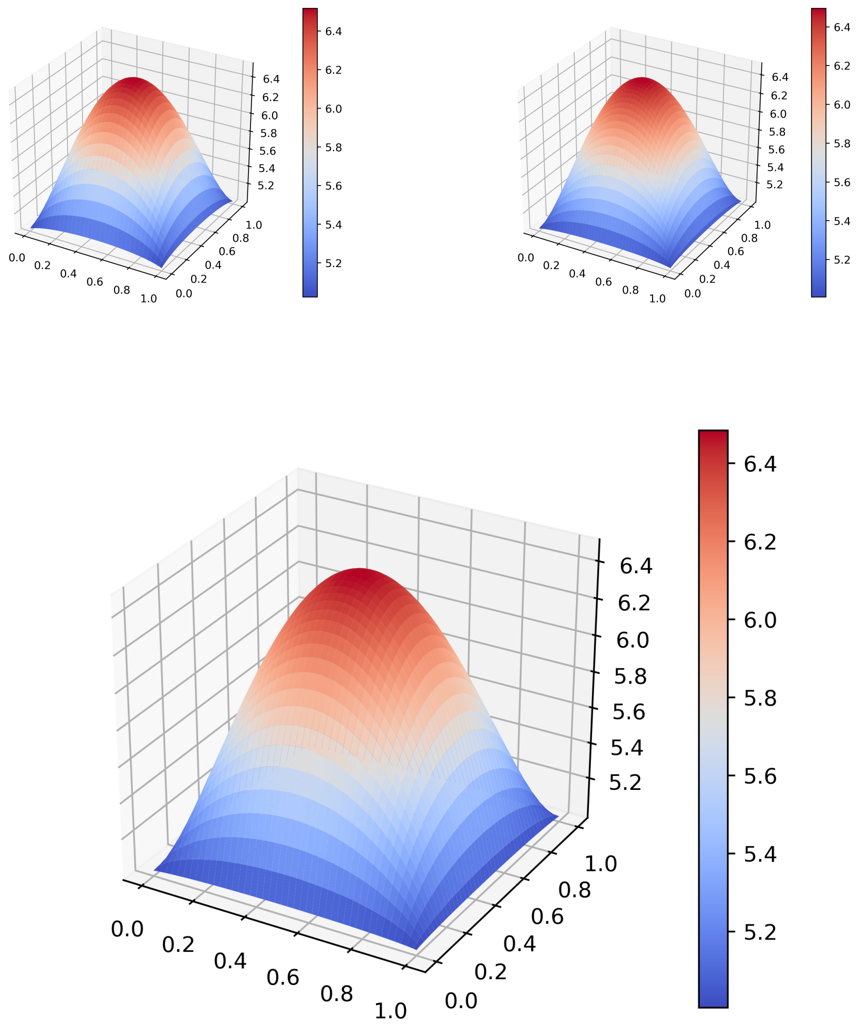

Figure 1: Solutions to the Poisson equations with N=64, 128, 256 respectively

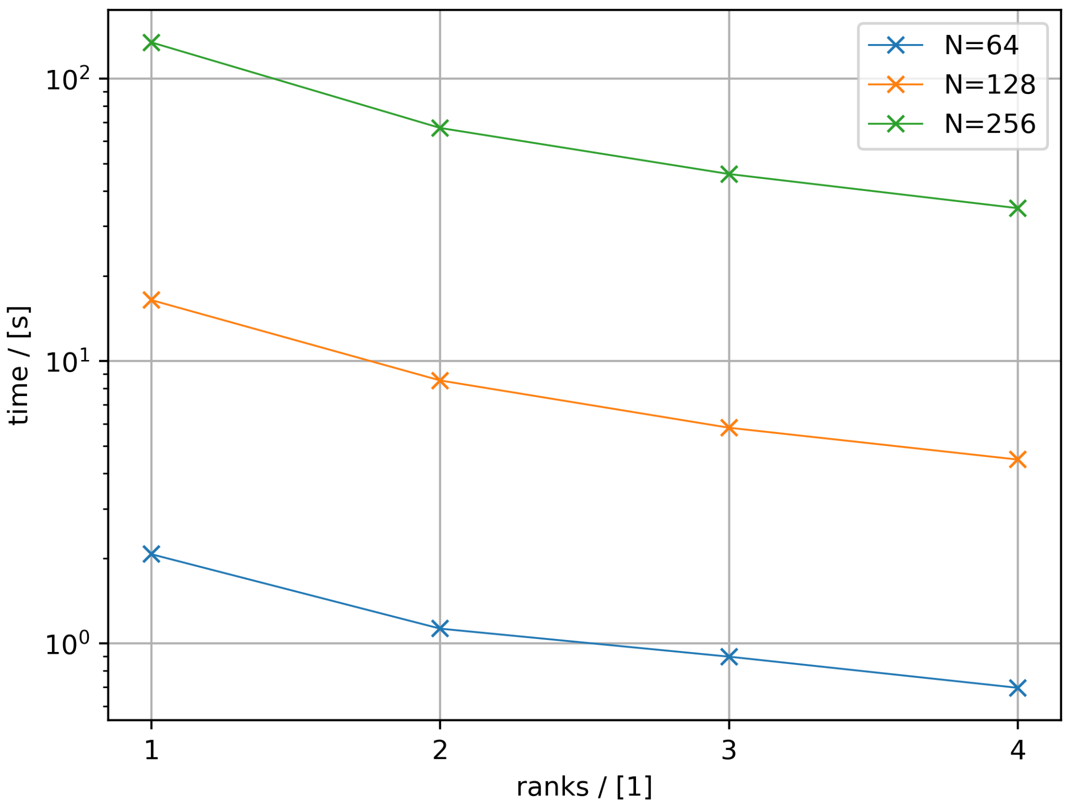

Figure 2: Linear scaling of poisson-petsc on quadcore i7 8th generation machine

2. Interactive Supercomputing using Jupyter Notebook [10 points]

I opted for a simple matrix multiplication. Please have a look at the corresponding folder and the associated script. Juypter Notebook would have equivalent features.

3. Jupyter Notebook - Parallel PDE-Constrained Optimization [50 points]

I thank Erick Schulz from ETH's Seminar for Applied Mathematics for helping me try to understand the problem and getting through the process of arriving at an 'on-paper' solution.

Even though we spent a tremendous time on the problem, the short and insufficient presentation of the mathematical background and the non-existence of IPOPT's python-interface documentation made it very hard to make progress solving this problem.

We did get to discretize the PDE, the functional and the according boundary condition but we did give up debugging the expression for the Jacobian and further derivatives.

I really do hope there will be an official solution to this problem since while interesting, we feel that there is not enough background provided to usefully understand the problem and the underlying intricacies.

I followed the example NLP provided in the project's directory but some dimensions are off and concludingly, it does not work and does not yield a sufficient solution.

2. An appropriate initial point for the optimizer would be the solution to the Poisson equation from the first problem with the control set to 0.

4. We can algebraically derive better expressions which eliminate the need to solve PDEs for this problem.

Table 2: Wall-clock time (in seconds) and speed-up (in brackets) using multiple cores on Euler for solving the PDE-constrained optimization problem.

Problem	N	Number of Euler cores			
		1	8	16	32
Inverse Poisson	500^2				
Inverse Poisson	1000^2				
Inverse Poisson	1500^2				
Inverse Poisson	2000^2				
Inverse Poisson	2500^2				
Inverse Poisson	3000^2				