Prof. Dr. Christoph Pflaum
Dominik Thönnes

Winter Term
2023/2024

Simulation and Scientific Computing
Assignment 2

OpenMP–parallel solution of elliptic PDE

1. Compute the solution of the elliptic partial differential equation (PDE):

$$-\Delta u(x,y) + k^2 \cdot u(x,y) = f(x,y), \quad (x,y) \in \Omega, \tag{1}$$

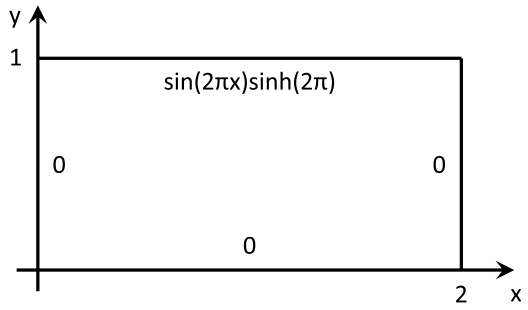
with $\Omega = [0,2] \times [0,1]$. In this assignment, $k = 2\pi$ and the right-hand side is defined as

$$f(x,y) = 4\pi^2 \cdot \sin(2\pi x) \cdot \sinh(2\pi y).$$

On the boundary $\partial\Omega$, use

$$\begin{aligned} u(x,1) &= \sin(2\pi x) \cdot \sinh(2\pi), \\ u(x,0) &= u(0,y) = u(2,y) = 0, \end{aligned}$$

which results in the following setup:



Use a finite difference discretization to solve (1). Choose the mesh sizes h_x and h_y of the grid with respect to the number of grid intervals in x-direction n_x and in y-direction n_y, which are passed on command line:

$$h_x = \frac{2}{n_x}, \qquad h_y = \frac{1}{n_y}.$$

Derive a linear system of equations (LSE) from (1) and choose a memory-efficient data structure to represent the LSE in your program (hint: stencil). Use zero values as an initial solution. For an efficient implementation, you should also carefully choose the data layout of the system's unknowns and right-hand side. Solve the LSE by the red-black Gauss-Seidel (RBGS) method. Use double precision for the computations. Perform a fixed number of iterations c (which

is passed on the command line) and measure the runtime with `Timer.h` from the previous assignment. Afterwards compute the discrete L2-norm of the residual:

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2 = \|\boldsymbol{r}\|_2 = \sqrt{\frac{1}{|\Omega_h|} \sum_{i \in \Omega_h} r_i^2},$$

with Ω_h being the set of interior points of the discretized domain Ω.

Then, print the time and the residual norm on the screen. Finally, write the computed solution to a file named `solution.txt`. Choose a file format, which can be visualized by the `gnuplot`'s `splot` function [1]. You can use the same format as in the reference solution provided on StudOn.

2. Parallelize your RBGS implementation using OpenMP. Test your implementation with 1, 2, 5, 10, 15 and 20 threads on the `Meggie cluster` [2]. Make sure your code scales!

   On assignment sheet 1 it was described how to access the Meggie front end via ssh and allocate your own node. Make sure you do not use the front end for your measurements!

3. Your program must be callable in the following way:

   `./rbgs` n_x n_y c

4. Provide a PDF file with *well–explained* performance graphs illustrating the speed–up of your parallelization. Assign the parallel efficiency to the ordinate and the number of CPUs to the abscissa. Perform 500 iterations for the following number of grid points in x- and y-direction: 32, 33, 1024, 1025, 2048, 2049.

5. Theoretical part

   (a) Write down the discretization matrix $\mathbf{A}$ for Eq. (1) for a grid point numbering according to Figure 1 and show whether it is (strictly) diagonal dominant or not.

   (b) Does the Jacobi algorithm converge for this problem? Prove your statement and if so give a boundary (< 1) for the convergence factor.

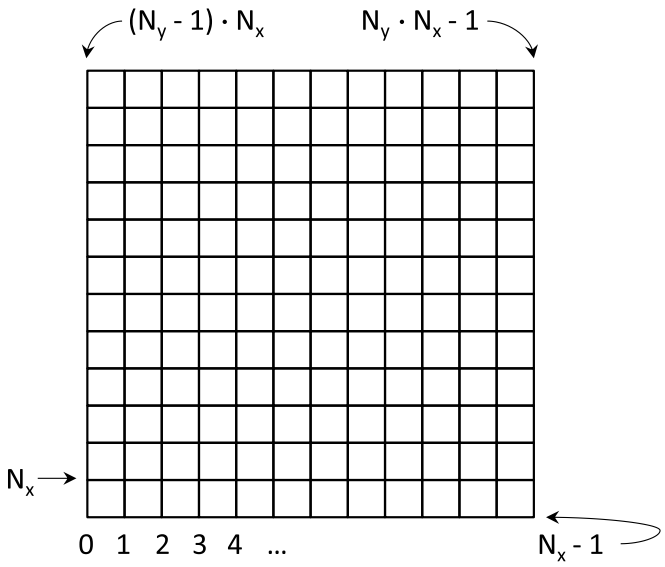


Figure 1: Grid point numbering. N_x and N_y denote the number of grid points in x and y direction. Note that they are not the same as n_x and n_y from before.

Hand in your solution on StudOn before the deadline. Make sure the following requirements are met:

- The program must be compilable with a Makefile or CMakeLists.txt that you have to provide. Your program must compile without errors or warnings on the Meggie cluster with the following g++ compiler flags:

  ```
  -std=c++17 -Wall -Wextra -Wshadow -Werror -fopenmp -O3 -DNDEBUG
  ```

  You may add additional compiler flags. The reference compiler is GCC version 8.5.0 on the Meggie cluster. You can check the version by typing `g++ --version`.

- The program must be callable as specified in 3.

- Check your solution against the reference implementation (see the binary compiled for Meggie provided on StudOn) to establish correctness of the method.

- Your submission must contain well–commented source files, a PDF file with performance graphs and instructions how to use your program in a **README file**. The theoretical part must be submitted in a read-only format, such as a high-quality scan (jpg) of your *readable* handwritten solution or a pdf. Upload your solution to StudOn as a team submission.

Performance Challenge

Your code is tested and the performance is measured. The program is run several times on the cluster with 20 threads and the parameters $n_x = n_y = 2049$ to determine the runtime. The fastest team will be awarded with an extra credit point.

Credits

In this assignment, points are awarded in the following way:

1. Up to four points can be obtained if your program correctly performs the above tasks and fulfills all of the above requirements. Submissions with compile errors will lead to zero points! The cluster nodes on the `Meggie cluster` act as reference environments.
2. One point is given for the correct answer of the theoretical task.
3. One bonus point is awarded to the winner of the performance challenge.

References

[1] `http://www.gnuplot.info/docs/gnuplot.pdf`

[2] `https://hpc.fau.de/systems-services/documentation-instructions/clusters/meggie-cluster/`