Winter Term
2016/2017

Prof. Dr. Christoph Pflaum
Christian Kuschel
Christoph Rettinger

Simulation and Scientific Computing
Assignment 2

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

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

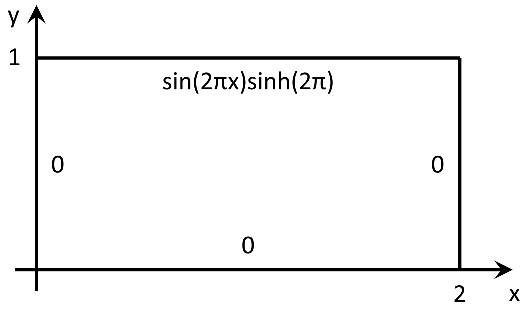
with $\Omega = [0,2] \times [0,1]$. In this assignment $k = 2\pi$ and the right-hand side is defined as

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

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. 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:

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

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

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].

2. Parallelize your RBGS implementation using OpenMP. Test your implementation with 1, 2, 4, 8, 16, and 32 threads on the `LSS cluster` [2]. Make sure your code scales!

   You can login to the front end of the `LSS cluster` via ssh with the command:

   ```
   ssh <login>@i10hpc.informatik.uni-erlangen.de
   ```

   You can start an interactive session on one compute node with the command:

   ```
   qsub -I -l nodes=1:ppn=32 -q siwir -l walltime=00:30:00
   ```

   Due to limited resources, it is only allowed to login to one node at a time!
   Most of the development can be done on the computers in the CIP pool. You do not have to login to the cluster for that. Once your program is almost finished, you can login to the cluster and do your final optimizations and measurements.

   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.

3. Theoretical part

   (a) Write down the discretization matrix $\mathbf{A}$ for (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.

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

   `./rbgs` n_x n_y c

Hand in your solution by Sunday, December 4, 2016, 23:59. Make sure the following requirements are met:

- The program must be compilable with a Makefile. Your program must compile without errors or warnings on LSS cluster nodes with the following g++ compiler flags:

```
-std=c++11 -Wall -Wextra -Wshadow -Werror -fopenmp -O3 -DNDEBUG
```

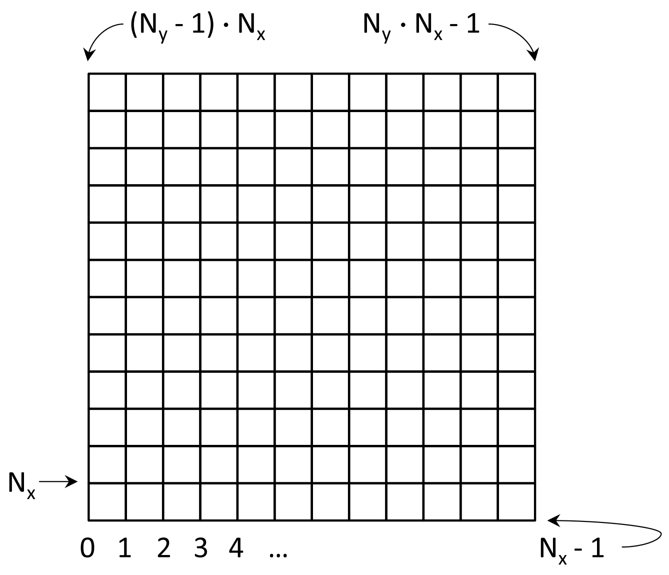


Figure 1: Grid point numbering.

You may add additional compiler flags. The reference compiler is GCC version 5.4.0 on the LSS cluster. You can check the version by typing `g++ --version`.

- The program must be callable as specified in 4.
- Check your solution to against the reference implementation `/software/siwir/rbgs`.
- Your code must be well-commented source files, a PDF file with performance graphs and, if necessary, instructions how to use your program (e.g. 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 32 threads and the parameters $n_x = n_y = 2049$ to determine the runtime. The fastest team will be awarded with an extra credit point.

References

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

[2] `https://www10.cs.fau.de/en/research/hpc-cluster/`