Prof. Dr. Christoph Pflaum
Florian Schornbaum
Christian Kuschel

Winter Term
2015/2016

# Simulation and Scientific Computing
# Assignment 2

Exercise 2 (*Elliptic PDE with OpenMP* )

## Tasks

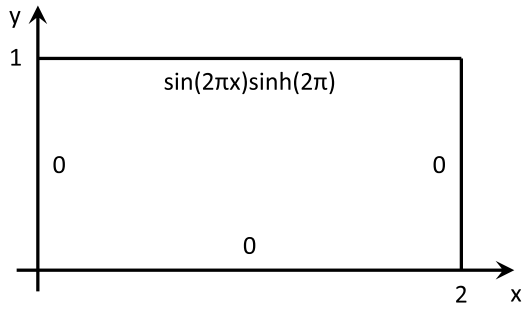
1. Your task is to compute a discrete solution of the following elliptic partial differential equation (PDE):

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

Where $k = 2\pi$ and the right-hand side is defined as $f(x,y) = 4\pi^2 \sin(2\pi x)\sinh(2\pi y)$. The domain is defined as $\Omega = [0,2] \times [0,1]$. On the boundary $\partial\Omega$, use

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

which results in the following setup:




To solve the PDE (1), apply a finite difference discretization. Choose the mesh sizes $h_x$ and $h_y$ of the grid according to the parameters $n_x$ and $n_y$ (number of grid intervals in x and y direction, respectively) supplied on the command line:

$$h_x = \frac{2}{n_x}, \qquad h_y = \frac{1}{n_y}. \tag{3}$$

Derive the resulting linear system of equations (LSE) 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. To solve the LSE, apply the red-black Gauss-Seidel (RBGS) method. Perform a fixed number of iterations $c$ (which is given on the command line) and measure the runtime (use Timer.h from the previous assignment). Afterwards compute the discrete L2-norm of the residual:

$$\left\|\vec{b} - \mathbf{A}\vec{x}\right\|_2 = \|\vec{r}\|_2 = \sqrt{\frac{1}{|\Omega_i|}\sum_{z\in\Omega_i} r_z^2} \tag{4}$$

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 `gnuplot`[1].

2. Also answer the following theoretical part:

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

   (c) Provide a PDF file with *well-explained* performance graphs illustrating the speedup 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.

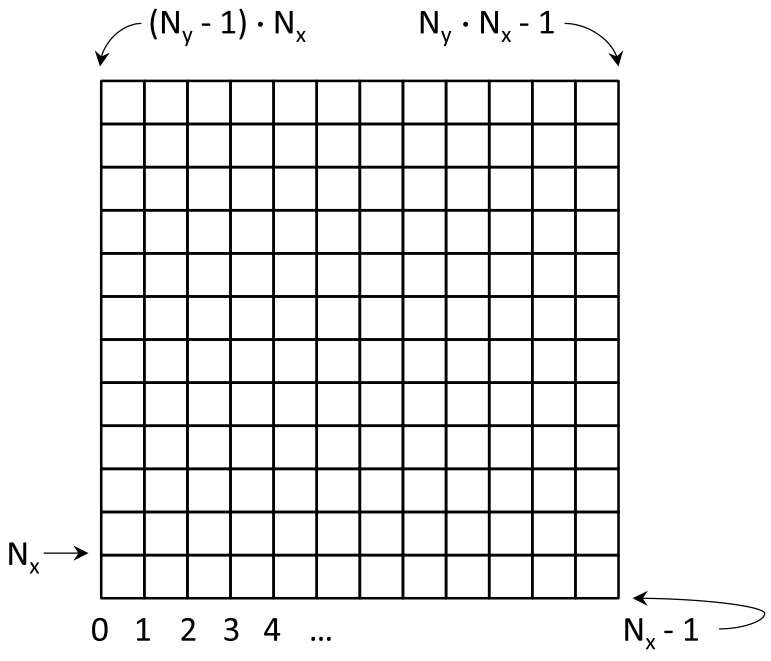



Figure 1: Grid point numbering.

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

   `./rbgs` $n_x$ $n_y$ $c$

4. 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
   ```

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

   ```
   qsub -I -l nodes=1:ppn=32 -q siwir -l walltime=01:00:00
   ```

   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 don't 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.

5. Use double precision for your computations.

6. Please hand in your solution to this exercise until Monday, November 30, 2015, 8:00 am. Make sure the following requirements are met:

   (a) The program must be compilable with a Makefile. Your program must compile successfully when calling "make".

   (b) The program must be callable as specified above.

   (c) The program must compile without errors or warnings on `LSS cluster` nodes with the following g++ compiler flags (you can make use of C++11 features):

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

   If you want to, you can add additional compiler flags.
   The reference compiler is GCC version 4.9.2 on the `LSS cluster`. You can check the version by typing "g++ –version". You can load GCC 4.9.2 by typing "module load gcc/4.9.2".

   (d) The solution should contain well commented source files, a fitting Makefile that satisfies all the conditions specified above, instructions how to use your program and a PDF file with the required theoretical part. When submitting the solution, remove all temporary files from your project directory and pack it using the following command:

   ```
   tar -cjf ex02_groupXX.tar.bz2 ex02_groupXX/
   ```

   where `XX` stands for your group number and `ex02_groupXX/` contains your solution. Then upload your solution in StudOn.

### Performance Challenge

We will test your code and measure the performance. Every code is compiled with the g++ compiler version 4.9.2. The program is run several times on the `LSS cluster`. We will use 32 threads and the parameters $n_x = n_y = 2049$ when measuring the performance. The winning team will be awarded an extra credit point.

### Beat Your Tutors!

The ultimate performance challenge: If you are the winning team of the performance challenge and on top of that can beat the time of "/software/sles/siwir/rbgs", you will get another reward that will be disclosed once your program was demonstrated to beat our implementation.

### Credits

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

1. You are guaranteed to receive at least one point 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 `LSS cluster` act as reference environments.
2. Two points are awarded if you correctly answer the questions in Task 2.

## References

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

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