Prof. Dr. Christoph Pflaum
Florian Schornbaum
Dominik Bartuschat

Winter Term
2014/2015

Simulation and Scientific Computing
Assignment 4

Exercise 4 (*The Parabolic Heat Equation*)

Tasks

1. In this assignment you are supposed to calculate an approximate solution $u(x, y, z)$ for $0 \leq t \leq t_{end}$ to the (parabolic) heat equation[1]:

$$\frac{\partial u}{\partial t} = \kappa \Delta u, \tag{1}$$

with the thermal diffusivity κ. The domain is defined as $[0, 1] \times [0, 1]$, and the boundary conditions for this problem are homogeneous Dirichlet boundary conditions. The initial condition for $t = 0$ is

$$u(x, y, 0) := u_0(x, y) := sin(\pi x) sin(\pi y). \tag{2}$$

There are different possibilities for the time discretization which are differently stable. Use a parameter α (read in from the command line) that switches between implicit $(\alpha = 1)$, Crank-Nicolson $(\alpha = 0.5)$, and explicit $(\alpha = 0)$ scheme in the following way:

$$\frac{u^{new} - u^{old}}{\tau} = \alpha \kappa \Delta u^{new} + (1 - \alpha) \kappa \Delta u^{old} \tag{3}$$

2. You are now supposed to solve the LSE arising from the implicit and Crank-Nicolson scheme with a MPI parallel Conjugate Gradient method. You can use your solver from the previous assignment.

3. It should be possible to start your program in the following form:

   mpirun -np N ./heat nx ny c eps timesteps τ κ α *vtk_spacing*,

   where *heat* is the name of your executable, running on *N* processes in parallel.

   nx and *ny* are the number of grid intervals in x-direction and y-direction, respectively. Time discretization parameters are *timesteps* for the number of time steps and $\tau > 0$ specifying the time step length. Furthermore, κ is the thermal diffusivity and $\alpha \in [0, 1]$ the parameter determining the time discretization scheme.

   The primary stopping criterion for the CG method is the residual threshold specified by *eps*, where the maximum number of iterations (in one time step) is limited by *c*.

   vtk_spacing is the number of time steps that lie in-between two time steps that generate VTK output (*vtk_spacing* $= 0$: VTK output is generated at the end of each time step, *vtk_spacing* $= 1$: VTK output is generated only every second time step, ...).

4. In every time step, your program should write as standard output in one line the number of the CG iteration steps and the final residual norm.

5. Visualization:
   Your program must generate VTK output files (more precisely: VTK files in the parallel XML file format[2]) according to the parameter *vtk_spacing* - at the end of the first time step, the first VTK output is generated. The VTK output should look like as follows: All VTK files are supposed to be saved in a subfolder called *vtk* (your program may assume that this subfolder already exists in your working directory when your program is executed). For every time step that is saved, each process generates one

   time_step_$(CURRENT_TIME_STEP)_$(PROCESS_RANK).vtu

   file which contains the simulation data that corresponds to the part of the domain that was assigned to this process. Additionally, only the master process generates one file called

   time_step_$(CURRENT_TIME_STEP).pvtu

   which refers to all vtu files of this time step. At the end of your simulation, the master process generates on file called *solution.pvd* which refers to all pvtu files.
   An example VTK output can be found on the website of this class[4]. VTK files can be visualized with Paraview[3]. An introduction to VTK and a detailed description of the vtu/pvtu/pvd XML file format will be given in the board exercise class related to this assignment.

6. Automated Makefile generation with CMake[5]:
   Instead of writing a Makefile, you are supposed to provide a CMakeLists.txt file that can be used to generate a Makefile automatically. Your CMakeLists.txt file must meet the following requirements:

   (a) Your executable should be called *heat*.

   (b) If the detected compiler is a gnu cc compatible compiler, the following flags should be added to the compilation process: *-Wall -Wextra -Wshadow -std=c++0x*

   (c) You should add a CMake option called *ARCHITECTURE_OPTIMIZATION* which is ON by default (but could be switched to OFF before generating the Makefile). If this option is ON, the flag *-march=native* must be added to the compiler if the detected compiler is a gnu cc compatible compiler.

   (d) You should add another CMake option called *ENABLE_WERROR* which is also ON by default. If this option is ON, the flag *-Werror* must be added to the compiler if the detected compiler is a gnu cc compatible compiler.

   An example program that uses CMake for generating its Makefile can be found on the website of this class[4]. A detailed introduction to CMake will be given in the board exercise class related to this assignment.

7. Please hand in your solution to this exercise until Friday, January 30th, 2015. Make sure the following requirements are met:

   (a) Use double precision for your computations.

   (b) The program should contain a valid CMakeLists.txt file for the compilation.

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

   (d) The program must generate VTK output files as specified above.

   (e) Please include a specialized `pbs` script.

   (f) The solution should contain well commented source files and instructions how to use your program. When submitting the solution, remove all temporary files from your project directory with the name `ex04_groupXX/` and pack it using the following command:

   ```
   tar -cjf ex04_groupXX.tar.bz2 ex04_groupXX/
   ```

   where `XX` stands for your group number and `ex04_groupXX/` contains your solution. Then send the archive to the course mailing list:

   `siwir@i10.informatik.uni-erlangen.de`

Credits

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

1. You are guaranteed to receive at least a yellow point if your program correctly performs the above tasks and fulfills all of the above requirements. Submissions with compile errors will lead to a red point! The LSS cluster acts as reference environment.

2. A green point is awarded if, additionally, you implement an option to choose a second initial condition that does not lead to a starting configuration that looks like a 'dot' (produced by Equation 2) but more like a donut.

References

[1] `www.f.kth.se/~jjalap/numme/FDheat.pdf`

[2] `www.vtk.org/VTK/img/file-formats.pdf`

[3] `http://www.paraview.org/`

[4] `https://www10.informatik.uni-erlangen.de/Teaching/Courses/WS2014/SiWiR/exercises.shtml`

[5] `http://www.cmake.org/cmake/help/cmake_tutorial.html`