AMCS/CS 312: High Performance Computing II

Problem Set #4, due 27 November 2011

This problem set uses the deal.II library which you should have already installed by downloading from `http://www.dealii.org/` and following the installation procedures given on the ReadMe page. If you have trouble, let me know!

The task involves modifying and running the step-22 tutorial program of that library that contains a solver for the Stokes equations which you have in a different form already seen in the `ex50` example of PETSc for homework #3. The goal is to determine some of the characteristics of the algorithms behind this program and how they relate to the applications that motivate it.

Note 1: To work with step-22 you need to have deal.II configured with the UMFPACK sparse direct solver. To this end, you will need to call the configuration command again, this time as `./configure --with-umfpack`, and then do `make all` again; if you forget about this, you will get an error message when running step-22.

Note 2: This problem set will be graded by Prof. Bangerth. You will have to produce a write-up that contains the numerical and graphical results indicated in the problems below. In case of questions please email `bangerth@math.tamu.edu`.

Problem 1 (Getting acquainted). Run step-22 and understand the screen output. When the program finishes, it should have produced files `solution-00.vtk` through `solution-06.vtk` that contain a graphical representation of the solution in VTK format on the initial mesh and five successively adapted meshes. Visualize the content of these files, for example using the Visit or Paraview programs. Alternatively, you can use a different program such as Avizo but you may have to find a different file format that step-22 can write (take a look at the `output_results())` function; other formats are listed in the `DataOutBase` class). Present these visualizations as part of your write-up and explain what you see. **(20 points)**

Problem 2 (Modifying the program). The `ex50` program solved a problem on a square domain. On the other hand, step-22 solves a problem on a rectangle, and the boundary conditions are different as well. Go through the program and identify where (i) the shape of the domain is described, and (ii) where the boundary conditions are set and what *kind* of boundary conditions it uses (boundary conditions describe what the velocity or pressure should be at the boundary of the domain).

Modify the program so that it solves the following problem: (i) The domain is the square $[0,1]^2$; (ii) the boundary conditions are $\mathbf{u} = (1,0)^T$ at the top, $\mathbf{u} = (0,0)^T$ at the left and right side of the domain, and do-nothing at the bottom. This is an approximation of the problem solved in `ex50`.

Explain in your write-up what parts of the program you had to change and how to solve this problem. Show a visualization of the flow with these changes. **(20 points)**

Problem 3 (Understanding complexity). The goal of much of High Performance Computing is to understand (and if possible) improve the complexity of algorithms, i.e., how their CPU time, memory consumption and communication needs depend on the size of the problem. step-22 solves a problem on a sequence of meshes that become finer and finer and so serves as a good testcase to evaluate the complexity of its algorithms as the mesh becomes finer and the number of unknowns becomes larger.

step-22 primarily consists of the following components, each in their own function: setup_dofs, assemble_system, solve, output_results, refine_mesh. For each of these functions, find out whether they contribute significantly to the overall run time of the program. If they do, determine how their run time depends on the total number N of degrees of freedom the program prints for each mesh refinement step. To this end, produce a table or graphic that shows this dependence. Can you identify whether the behavior is linear or quadratic with N, or any other behavior?

(Hints: To determine CPU times, you may wish to look at the `Timer` or `TimerOutput` classes, which you can find through the "Classes" view of the deal.II manual. Time the run time of the entire function. In order to determine linear/quadratic behavior you may have to run the program for more than just the 6 refinement cycles executed in the `run` function since it otherwise terminates too quickly: on my laptop, it runs in less than 10 seconds. Choose the run time somewhere in the range of a few minutes to get a good feel for the complexity of each function. To determine the complexity of an algorithm, it is most useful to plot run time against N in a log-log plot. As a side note: If you're curious about making the program run faster, open the Makefile, set debug-mode to off, and repeat your observation.) **(30 points)**

Problem 4 (Understanding accuracy). Being able to make a program run fast (or getting a high giga/tera/petaflop rate) is not really the goal in scientific computing. Rather, the goal is to obtain an *accurate* answer fast. To improve a program can therefore mean to make an existing algorithm faster, or to replace an existing algorithm with a different one – which may not be optimized and may yield a low gigaflop rate but provides better accuracy anyway.

Remember that the problem you solve with your modified program describes the flow of a fluid in a box where some force at the top provides a velocity to the right, the fluid extends below the bottom, and the velocity is zero at the left and right. To investigate the accuracy you get here, evaluate the flow velocity at the point $(0.5, 0.5)^T$ using the `VectorTools::point_value` function (the version that returns its results as a vector through the fourth argument). Do these values converge? Provide your best guess of what this value might be if the mesh were infinitely fine and from there determine the percentage error on each of the meshes on which the solution is computed. Produce a table or plot (e.g. a log-log plot) that shows the percentage error against the number N of degrees of freedom and against the total run time. Interpret the results. **(30 points)**

Bonus problem (A better solver). The results section of the web site of step-22 has a suggestion for a better solver. Repeat the computations from Problem 3 with the better solver presented there. Produce graphics that contrast the two solvers and their complexity. The best graphics will be put into this section of the tutorial program and you will be credited by name! **(5 bonus points)**