Assignment 5, PX390, 2022/2023: Modelling heat transfer around the toroidal shell of a tokamak.

January 10, 2023

You have been given the task of understanding heat flow around the first wall of a tokamak reactor, upon which plasma deposits energy gained from the fusion process. The reactor shell is topologically toroidal. The shell is not homogeneous, that is, it has low and high conductivity regions and is externally cooled. In addition, the plasma does not deposit heat evenly around the shell. The model equation to be solved is

$$\begin{aligned}\frac{\partial T}{\partial t} = {} & \frac{\partial}{\partial \theta}\left(Q^{11}\frac{\partial T}{\partial \theta}\right) + \frac{\partial}{\partial \zeta}\left(Q^{22}\frac{\partial T}{\partial \zeta}\right) + \frac{\partial}{\partial \theta}\left(Q^{12}\frac{\partial T}{\partial \zeta}\right) + \frac{\partial}{\partial \zeta}\left(Q^{21}\frac{\partial T}{\partial \theta}\right) \\ & - R(\theta,\zeta)T + S(\theta,\zeta)\end{aligned}$$

For a fixed toroidal shell, angles θ and ζ are toroidal coordinates periodic on the interval $[0, 2\pi]$. The coefficients $Q^{11}(\theta,\zeta)$, $Q^{22}(\theta,\zeta)$, $Q^{12}(\theta,\zeta)$ and $Q^{21}(\theta,\zeta)$ represent a combination of geometric quantities (related to the metric tensor) that capture the shape of the shell, multiplied by the thermal conductivity. These coefficients must satisfy the following relations: $Q^{11} > 0$, $Q^{22} > 0$, $Q^{12} = Q^{21}$ and $Q^{11}Q^{22} - Q^{12}Q^{21} > 0$. $R(\theta,\zeta)$ represents the connection of the shell to external heat sinks. The temperature T here is actually representing the difference between the actual local temperature of the shell and the temperature of the external heat sink. You may assume $R \geq 0$ everywhere, and there will be at least one point where $R > 0$. $S(\theta,\zeta)$ is the (normalised) energy source due to the fusion plasma heating the walls.

Your code needs to produce time-evolution of the temperature up to a certain time with the initial condition for the temperature $T(t = 0) = 0$. Your time-evolution code should implement an implicit solver, to allow large time steps. Note, however, that there will be a minimum number of time steps required for evolving the temperature to a final time.

1 Grids

The grid for input and output have N_θ and N_ζ points in the θ and ζ direction respectively. They are evenly spaced with $\theta_i = 2\pi i/N_\theta$ and $\zeta_j = 2\pi j/N_\zeta$. You may assume N_θ, $N_\zeta \geq 1$.

2 Input

The input file, 'input.txt' will contain the following values, one on each line:

- N_θ: number of θ grid points.
- N_ζ: number of ζ grid points.
- t_f: final time for time evolution mode.
- I_{min}: minimum number of timesteps (of equal length).

The coefficients file, 'coefficients.txt' will have $N_\theta \times N_\zeta$ lines containing values of the equation coefficients. Each line contains the values Q^{11}, Q^{22}, Q^{12}, S and R (in that order), for a particular grid point on the θ, ζ grid. The grid points for input and output are indexed in the order $L = i \cdot N_\zeta + j$, where i is the θ grid point index running from $[0, N_\theta - 1]$, and j is the ζ grid point index running from $[0, N_\zeta - 1]$. That is, the first line in the coefficients file contains values corresponding to grid point $L = 0$, and subsequent lines are in order of increasing values of L.

3 Compiling

You are required to use the LAPACK library to invert matrices, so the compiler flags are different this time. This library is installed on the lab computers and on vonneumann. We will not provide any support for installing these libraries on your own computers, it might be quicker just to log in to vonneumann to test your program. In order to compile your program (here, named prog5.c) on vonneumann you need to execute the following commands:

1. module purge
2. module load intel impi imkl
3. gcc prog5.c -lmkl -liomp5 -lm

You also need to add

```
#include <mkl_lapacke.h>
```

in your source code. As before, you should test your code with simple test cases. We will, again, be marking it mostly based on how well it reproduces certain tests.

4 Test cases

Useful test cases include the trivial case with the uniform conductivity. It would be useful to consider cases where the shell is much longer in one direction than the other. Since the number of grid points may be 1 in either direction this allows 1D test cases to be run. For large final times, the numerical solution should converge to a time stationary solution. You may find it useful to find time stationary numerical solution and verify that your code evolves the temperature towards it, for long runs.

5 Output

The code should output the solution T at a given time t_i and at all the grid points as four columns in a text file 'output.txt' with values t_i, θ_i, ζ_j, $T(\theta_i, \zeta_j)$. These should be output in order such that data associated with point (θ_i, ζ_j) is output on line $L = i \cdot N_\zeta + j$, that is, following the same order as the coefficients input file.