Assignment 5, PX390, 2021/2022: triggered release.

January 12, 2022

An experiment is being conducted to understand the spread of a chemical in the presence of active agents. A room is filled with 'agents' that start releasing a chemical compound once subject to a sufficiently large concentration of that chemical.

We consider this in a continuous form in two spatial dimensions, the local concentration of the chemical is denoted u. The time evolution of this quantity is governed by

$$\frac{\partial u}{\partial t} = \nabla^2 u + f(u) \tag{1}$$

with $f(u) = u^2/(1+u^2)$. Here, we are working on the Euclidian (x, y) plane, and $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$.

Ideally, your time-evolution code should implement an implicit solver, to allow long timesteps given a fine grid: it is only necessary to implicitly solve the Laplacian term (second spatial derivative). You may assume that the nonlinear term f does not impact the stability condition (the diagnostic timestep t_d will be set short enough for this to normally not be an issue).

The spatial domain is a rectangle on the (x, y) plane, with $x \in [0, L_x]$ and $y \in [0, L_y]$.

Boundary conditions are either Dirichlet ($u = 0$) or Neumann ($\nabla u.\hat{\mathbf{n}} = 0$, where $\hat{\mathbf{n}}$ is normal to the boundary) depending on input switches.

The code reads in the initial condition (at time $t = 0$) for u.

1 Grids

The grid for input and output have N_x and N_y points in the x and y direction respectively. They are evenly spaced with $x_i = L_x i/(N_x - 1)$ and $y_j = L_y j/(N_y - 1)$. You may assume $N_x, N_y \geq 2$.

2 Input

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

- N_x: number of x grid points.
- N_y: number of y grid points.
- L_x: Length of x domain.
- L_y: Length of y domain.
- t_f: final time for time evolution mode.
- t_D: diagnostic timestep.
- s_{x0} switch for $x = 0$ boundary condition.
- s_{x1} switch for $x = L_x$ boundary condition.
- s_{y0} switch for $y = 0$ boundary condition.
- s_{y1} switch for $y = L_y$ boundary condition.

The meaning of the boundary condition switch parameters s is that for $s = 0$ the related boundary must satisfy Dirichlet boundary conditions ($u = 0$) and for $s = 1$ it is must satisfy Neumann conditions ($\nabla u.\hat{n} = 0$, where $\hat{\mathbf{n}}$ is normal to the boundary).

The code should input the initial value of the quantity $u(x, y)$ from a file called 'coefficients.txt' in single column format (one value per line, which is the u value at that grid point) for all the grid points (x_i, y_j). The order is first all the x grid points in ascending order at $y = y_0$, then all the x grid points for $y = y_1$, and so on.

3 Compiling

You can (and probably should) use the LAPACK library to invert matrices, which is on nenneke. Some instructions are given for installing it on a linux system on the course website; if you aren't used to compiling/installing libraries on your own system, it might be quicker just to log in to nenneke. Compile instructions are the same as assignment 4. The code should generate no warnings when compiled with -wall.

Please call the LAPACKE C interface and not LAPACK directly (because these are FORTRAN functions and the interface is not guaranteed to work the same way when compiled on different machines).

4 Test cases and suggestions.

Useful test cases include the trivial cases where diffusion dominates, you may wish to test modified versions of your code with simplified functions f. Since the number of grid points may be small, this allows effectively 1D test cases to be run (for certain boundary condition choices).

It might not be obvious what to do in the corners of the grid where 'two boundary conditions apply'. Actually the Neumann boundary conditions are not well defined in the corners, but are well-defined arbitrarily close to the corner point. You should be able to satisfy yourself that, if you expand the function as a Taylor series (we are asuming our function is locally well-behaved, so this should be possible in some ball around the corner point) to first order $u(x, y) = a + bx + cy$ around the corner point), each combination of boundary conditions leads to constraints on the coefficients of this series. Your numerical implementation of the boundary conditions should be satisfied for linear functions (represented on a grid) that satisfy those constraints.

5 Output

The code should output the quantity $u(x, y)$ to a file 'output.txt' in four column format (ie four values per line, the time point t, the x grid point, the y grid point, then the u value) for all the grid points (x_i, y_j), at time points $t = 0, t_D, 2t_D, ...$ etc. The order is first all the x grid points at $y = y_0$, then all the x grid points for $y = y_1$, and so on.