UBC CPSC 303, 2018 Winter Term 2

Problem Set 5a, due Wednesday, March 27, 11:00am

Instructions: Produce *well-written* solutions that answer the problems that follow.

- Introduce notation/terminology as required clearly.
- Use the rubric provided on Piazza to guide your solution development. Remember to reference the prescribed learning objectives as well.
- You may work in pairs. Indicate both your CSIDs on your solution write-up and submit through Gradescope. *Do not put your name on work uploaded to Gradescope!*
- Submit your work as a single PDF file; additional code files can be uploaded to Gradescope as well. Include the code and the output from running your programs in your main write-up (i.e., provide some evidence of the codes function that can be assessed prior to executing the code).
- You are not obligated to use MATLAB for programming work; you can use Python (or even another tool language that can run within a Jupyter notebook). However, it is up to you to ensure that the graders can execute your code easily.

About this problem set. Problem Set 5 is the last problem set of the course, and it comes in two parts. This is the first part, hence "Problem Set 5a". In this problem set you will numerically solve a Partial Differential Equation (PDE). However, PDEs will not be covered on the final exam and the primary goal of this problem set is not for you to learn about PDEs (although that is a nice bonus). Rather, the primary goal of this assignment is to review various topics covered in the second half of the course (numerical differentiation, numerical integration, and numerically solving ODEs), and to bring together the skills you have learned in CPSC 303 to tackle a somewhat plausible real-world problem.

Suggested reading. Example 7.13 (stationary convection-diffusion equation), Example 16.17 (heat equation PDE).

The problem. Suppose that there is a kindergarten near two industrial factories. In order to assess health risks to the children, we want to simulate what happens to pollution that is emitted by the factories. We shall suppose that plumes of pollution are released simultaneously by the two factories at the beginning of each day ($t = 0$). The concentration of the pollution as a function of position and time, $u(\mathbf{x}, t)$, is then governed by the convection-diffusion partial differential equation:

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = D\left(\frac{\partial^2 u(\mathbf{x},t)}{\partial x^2} + \frac{\partial^2 u(\mathbf{x},t)}{\partial y^2}\right) - \left(W\cos(\theta)\frac{\partial u(\mathbf{x},t)}{\partial x} + W\sin(\theta)\frac{\partial u(\mathbf{x},t)}{\partial y}\right), \tag{1}$$

where D is the diffusion coefficient and θ and W are the direction and strength of the wind, respectively. We will model the pollution inside the domain $\Omega = [0,1]^2$, for the time interval $t \in [0, t_f]$ with $t_f = 0.25$, and we will use $D = 0.05$ throughout.

The plumes of pollution at the beginning of the day ($t = 0$) is the sum of two scaled Gaussians corresponding to two factories at $(0.25, 0.25)$ and $(0.65, 0.4)$, respectively. This is shown in Figure 1. Mathematically, the initial condition is

$$u(\mathbf{x}, 0) = a_1 \exp\left\{-s_1[(x-0.25)^2 + (y-0.25)^2]\right\} + a_2 \exp\left\{-s_2[(x-0.65)^2 + (y-0.4)^2]\right\}. \tag{2}$$

The parameters a_1 and a_2 represent the intensity of the pollution, while s_1 and s_2 represent the (inverse squared) widths of the initial pollution plumes.

To fully specify the PDE, in addition to the initial values (given above) we also need to impose *boundary conditions*. We will use the boundary condition that u is always zero at the boundaries of the square region we're modelling. In symbols,

$$u(\mathbf{x}, t) = 0\,, \ \mathbf{x} \in \partial\Omega\,, \tag{3}$$

for all $t \in [0, t_f]$, where the symbol $\partial\Omega$ refers to the boundary of the domain Ω. Note: eq. (2) does not apply at the boundary, because eq. (3) overrules it. You do not need to worry about the boundary conditions quite yet; we will return to them in part (d).

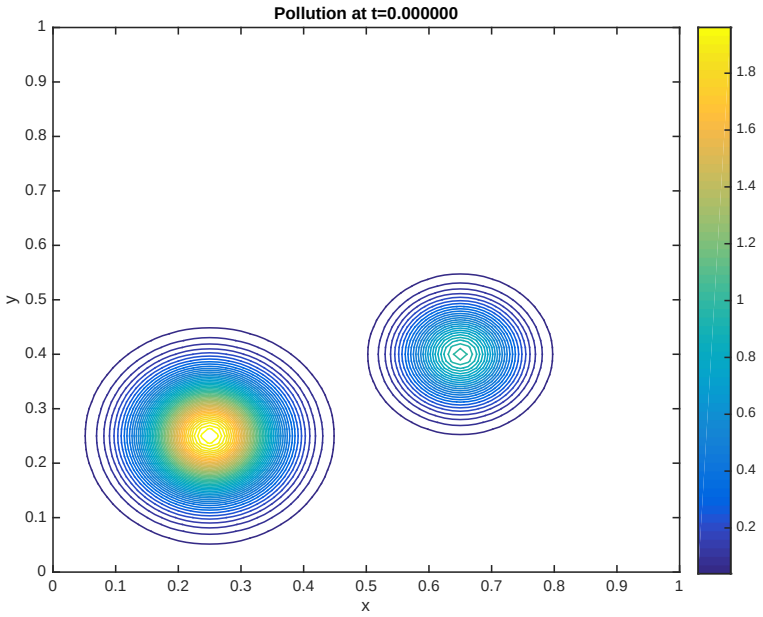

Figure 1: The initial pollution profile corresponding to a larger factory at $(0.25, 0.25)$ and a smaller factory at $(0.65, 0.4)$.

To solve a PDE, we need to discretize in space as well, not just time. We will approximate this convection-diffusion equation using a finite difference method, with a backward Euler discretization in time, and second-order accurate centred differences for the spatial derivatives. Throughout this assignment, we will use a uniform grid with $n_x = n_y = 81$ points in each spatial direction in Ω, such that the spacing between adjacent grid points is the same in x and y, namely $\Delta_x = \Delta_y = 1/80 = 0.0125$.

(a) Use the centred difference formulas for the first and second derivatives to spatially discretize eq. (1). The resulting equation should still involve a derivative with respect to time but should no longer involve the spatial derivatives. Replace all occurrences of $u(\mathbf{x}, t)$ with discretizations of the form $u_{i,j}(t)$, where the indices i and j run from 1 to 81 and represent the grid points. Make sure that i is the vertical index such that $i \in \{1, 2, \ldots, n_y\}$ and j is the horizontal index such that $j \in \{1, 2, \ldots, n_x\}$.

(b) We now have a system of $n_x n_y = 6561$ coupled ODEs and we are, more or less, back in familiar territory. Write down the backward Euler update rule for your ODE system from part (a). The resulting equation should no longer involve any derivatives or continuous variables; occurrences of $u_{i,j}(t)$ should be replaced with $u_{i,j,\tau}$, where τ is a time index running from 1 to the number of time steps, which is $1 + t_f/\Delta t$ (we called this time step h in class, but we'll call it Δt here to be explicit about space vs. time).

(c) In part (d) we will rewrite our discretized system in matrix notation. But, before we get to that, we need to do some bookkeeping. We are currently representing the state of our system at a given time τ as a grid with indices i and j; i.e., as $u_{i,j,\tau}$. In order to write the evolution over time as a linear system, we need to give a single label to each point on our grid, so that we can represent the state of our system at a given time as a vector (you can think of this as "flattening" the 2-D grid). The total number of points on the grid is $n_x n_y = 81 \times 81 = 6561$. We will use the *column-major order*, which is the convention used by

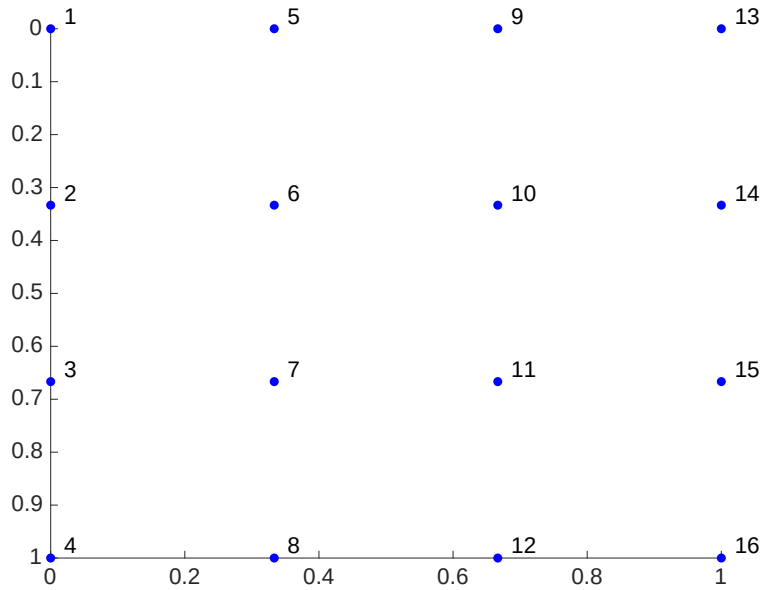

Figure 2: Linear indices for a 4×4 grid, using the column-major order.

MATLAB; see Figure 2 for clarification. Consider the function $G(i, j)$ that takes in the grid indices i and j, and returns the linear index based on the column-major order, for a grid of size $n_x \times n_y$. Mathematically, the form of G is

$$G(i, j) = i + (j - 1)n_y \, .$$

Try this out on a few cases in the figure above, to convince yourself that it's correct.

Let us now refer to the state of the system at time τ as $\mathbf{u}_\tau$, where

$$\mathbf{u}_\tau = \begin{bmatrix} u_{1,1,\tau} \\ u_{2,1,\tau} \\ \vdots \\ u_{n_y,1,\tau} \\ u_{1,2,\tau} \\ \vdots \\ u_{n_y,n_x,\tau} \end{bmatrix} ,$$

In other words, $\mathbf{u}_\tau$ is the "flattened" representation of the grid as a vector. What is the length of this vector?

(d) Starting from your discretization from part (b), write the update rule as a linear system of the form,

$$A\mathbf{u}_{\tau+1} = \mathbf{u}_\tau \, ,$$

and describe the system matrix A. Because of the index mapping issue, and because it's a pretty big matrix, A is a bit unwieldy to write down on a piece of paper. Instead, describe all the nonzero entries of A using the function G as needed. For example (and this is a major hint!), the diagonal entries of A are given by

$$A_{G(i,j),G(i,j)} = 1 + 2D\Delta t \left(\frac{1}{\Delta_x^2} + \frac{1}{\Delta_y^2} \right) , \tag{4}$$

for all (i, j) not on the boundary.

We also need to deal with the boundary condition given by eq. (3). In order to make sure the boundary condition is enforced at all time steps, design the matrix A so that all rows corresponding to boundary points are just rows of the identity matrix. This way, the u-value of these points will stay the same at each time step, and so if the boundary condition is satisfied at $t = 0$, it will continue to be satisfied throughout.

Also: what is the size of this matrix?

(e) How many nonzero entries does the A matrix have? To answer this, keep in mind that each row of A corresponds to a point on our grid. Thus, you must first figure out how many boundary points (and non-boundary points) you have on your grid. Then, multiply these by the number of nonzero elements per boundary and non-boundary row, respectively. As a hint, you should get a number between 10^4 and 10^5.