Instructor: Dr. S. Hughes

Due date: Sun, March 24 (midnight)

Lab 8 - Partial Differential Equations (PDEs)

Keywords: Partial Differential Equations (PDEs), Heat Equation (1D), Poisson's Equation (2D), Jacobi Iteration, Periodic Boundary Conditions and Exploiting 2D FFTs/IFFTs.

Task: Go through all the questions and coding exercises below and hand in a single modularized Python code where each part runs each part of the question(s). Remember and include your surname at the beginning of the code name.

Code Submissions on onQ: Attach a single (preferable, or no more than several) .py (called LabX_YourLastName.py) and single .py.txt (same code, same name, just a different extension).

Marks: Coding Skills, Efficiency, Correct Results and Clear Graphs and Good Presentation of Results (6); Good use of Comments (2); Good Coding Structure and Readability (2). Total: (10)

Reminder: All codes must run under Python 3.9.x. and the Spyder IDE (check this before submitting). Section off the code by using #%% (and label section in code), and add useful comments, so we can step through it easily.

Acknowledgements: Please comment on any help that you received from your group members or others concerning this Lab assignment.

Some Background (in addition to lecture notes):

PDEs - Wikipedia
Poisson's Equation - Wikipedia
Heat Equation - Wikipedia

Question 1

Consider the 1D heat equation:

$$\frac{\partial u}{dt} - \alpha \frac{\partial^2 u}{\partial x^2} = 0, \tag{1}$$

with the initial (Temperature, units of C):

$$u(x, t = 0) = 20 + 30 \exp[-100(x - 0.5)^2], \tag{2}$$

on a rod of length $L = 1$ m. Also have the hard wall (fixed) boundary condition: $u_{\text{ends}} = 20$ (at all times), as the start and end of the rod.

Write a numerical solution for the 1D heat equation (diffusion equation) and solve on a spatial grid of 100 points, and iterate from $t = 0$ to $t = 61$ seconds (units are seconds), where $dt = 0.1$. Consider a diffusion coefficient of $\alpha = 2.3 \times 10^{-4}$ m^2/s, but make sure (automate) your code to always check the steps sizes ensure numerical stability (namely it should satisfy the Caurant condition, or "Courant–Friedrichs–Lewy" condition, otherwise have a clean excite and write a warning if this is not satisfied (e.g., with an `exit` or `exception`). Use "slicing" with the main algorithm, so no spatial `for` statements are needed (this is especially important if we went to 2D and 3D solutions, so it is good practise to also do this already in 1D) – but you can use one in time to iterate.

Plot your numerical solution graphically, showing the heat profile on the rod at the specific time snapshots: $t = [0, 5, 10, 20, 30, 60]$ (units are in seconds). For reference, see Fig. 1.

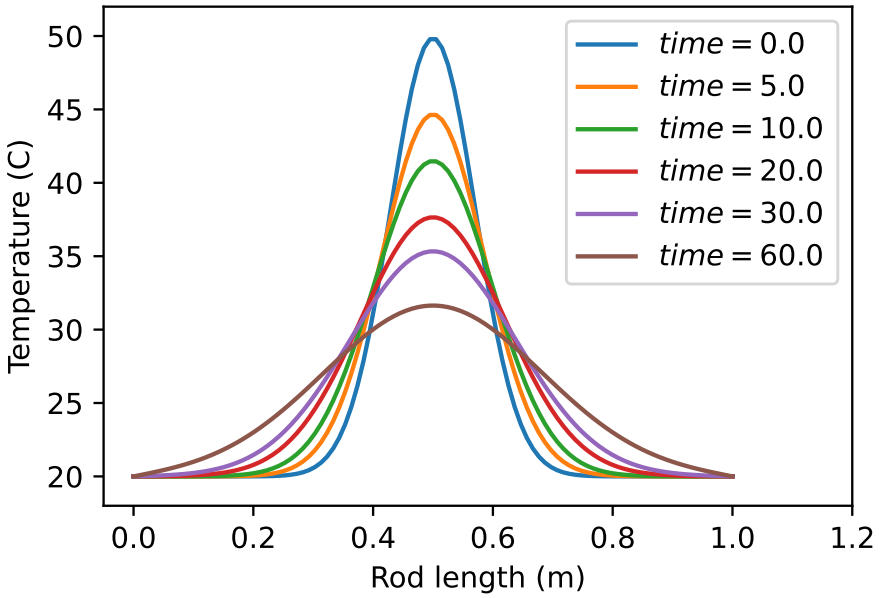

Figure 1: Heat equation solution at various snapshots in time (time units are seconds).

Question 2

(a) Consider Poisson's equation in two dimensions:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = f(x, y), \tag{3}$$

which we will solve on a finite size grid. It solves the potential subject to some charge density (source).

Implement an iterative method by *relaxation*, using a simple Jacobi iteration approach, with the following source:

$$f(x, y) = \cos(10x) - \sin(5y - \pi/4), \tag{4}$$

with the boundary condition on all walls: $\phi_{\text{walls}} = 0$. Use 200 spatial points in x (covering 0 to 2) and 100 spatial points in y (covering 0 to 1), obtain a solution within a tolerance of 10^{-5}, on the relative change in the norm of phi: thus, we iterate until $[norm(\boldsymbol{\phi}^{k+1}) - norm(\boldsymbol{\phi}^{k})]/norm(\boldsymbol{\phi}^{k+1}) < tol$. To compute the norm, e.g., you can use `lg.norm` after the import: `from numpy import linalg as lg`.

To code this efficiently, use vectorization for the finite-difference calculations (no `for` loops are needed in x or y), with a central differencing scheme. Also set up the 2d function as a subroutine and call it with a 2d meshgrid of points.

Plot your 2D solution as a 2D image/contour graph. For reference, see Fig. 2 (which is a very basic example, so feel free to improve the graphical presentation), where I use the routine `imshow`, but you do not need to use this specific routine (e.g., `contourf` is also a good option). Make sure your 2D plot also shows the correct aspect ratio, and add a colorbar with an informative label.

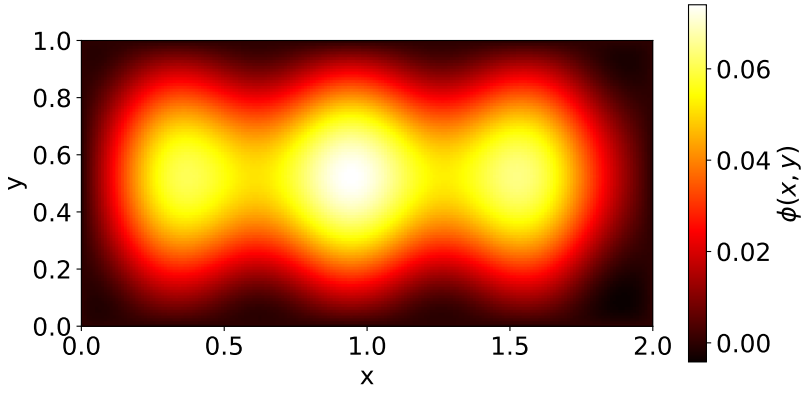

Figure 2: Solution to Poisson's equation on a finite size grid, with fixed boundary conditions at the walls and a specific source term.

(b) Repeat with a different boundary condition on the walls:

$$\phi(x, 0) = 0.2, \quad \phi(x, y_{\max}) = 0.1, \quad \phi(0, y) = 0.3, \quad \phi(x_{\max}, 0) = 0.4.$$

When you plot the solution, make sure the origin is in the right place and the wall values are as expected.

Bonus Question [☺ 3 extra points are possible here ☺]

Consider the 2D Poisson equation again, but this time on a *periodic* spatial grid:

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = f(x,y), \quad f(x,y) = \cos(3x+4y) - \cos(5x-2y),$$
$$\phi(x,0) = \phi(x,2\pi), \quad \phi(0,y) = \phi(2\pi,y). \tag{5}$$

For a periodic grid, we can solve this equation in Fourier space as (see lecture notes):

$$\tilde{\phi}_{kl} = \frac{1}{2}\frac{h^2\tilde{f}_{kl}}{\cos(2\pi k/n) + \cos(2\pi l/n) - 2}\ , \tag{6}$$

where the tilde represents the 2D Fourier transform, and $h = dx = dy$. An inverse Fourier transform thus returns the solution (which is periodic) in real space. Implement an FFT/IFFT routune to solve this Poisson problem (you can use Numpy routines for this, namely `np.fft.fft2` and `np.fft.ifft2`), using equal x and y grids, spanning 0 to 2π with 800 spatial points. Be careful of any points that may divide by zero at the 0,0 point, which you can exclude in the formula above, but work out from the periodicity, and print out the run time of your routine (it should be way below 1 second – I obtain 0.05-0.06s on my laptop computer.) As always, vectorize as much as possible (and use `meshgrid`).

As in Q2, plot your 2D solution as an 2D contour image graph. For reference, see Fig. 3, where I again use the routine `imshow`. Once more, make sure your 2D plot also shows the correct aspect ratio (square in this example), and add a colorbar with a label. Feel free to also experiment with the colors – here I am using the basic "hot".

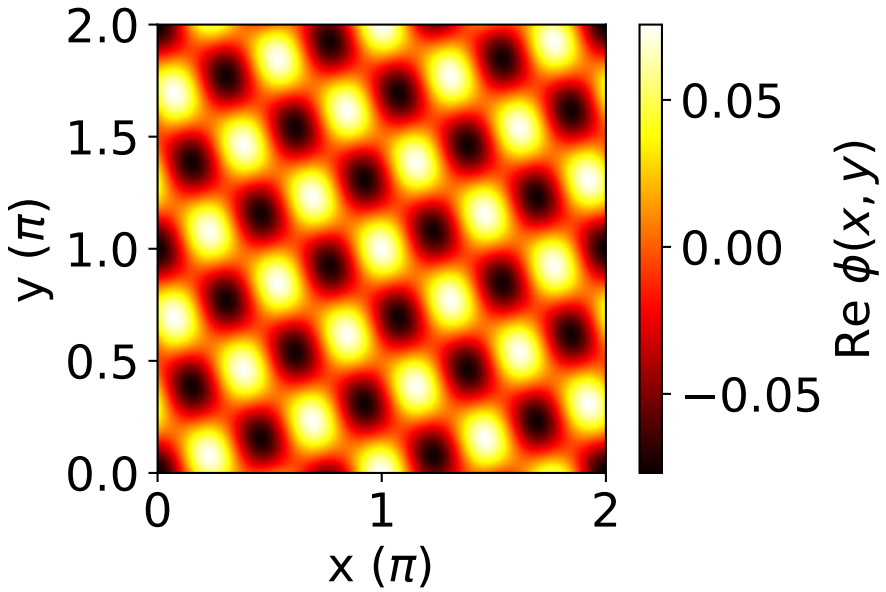

Figure 3: Solution to Poisson's equation on a periodic grid, with periodic conditions at the walls and solved in Fourier space.