# AMATH 481/581 - Autumn 2022

# Homework #3

Submissions accepted until 11:59 PM (PT) Wednesday, November 16, 2022

To submit this assignment, upload your main homework file (.m, .py, or .ipynb) to Gradescope. Additionally you may upload a .pdf you create to demonstrate mastery of one or both presentation skills.

**Warning:** In this homework we are working with really large arrays. You should be using sparse storage otherwise it may slow down your computer and the autograder. However when you submit your solutions, you need to submit them in full form (`full` in MATLAB and `A.todense()` in python).

1. The advection equation models the flow of a specific quantity, $u = u(x,t)$ as it is advected by a velocity field, $c$. The velocity field may be nonconstant. For instance, the velocity field may come from a periodic source such as waves reaching the beach. In this problem we will consider the 1D advection equation,

   $$u_t + c(x,t)u_x = 0, \qquad -\infty < x < \infty,\ t > 0,$$

   with initial condition

   $$u(x,0) = e^{-(x-5)^2}.$$

   We will truncate the problem and solve it for $-L \leq x \leq L$ for $L = 10$, using 200 equally spaced $x$ points and periodic boundary conditions,

   $$u(-L,t) = u(L,t).$$

   We will solve for $0 \leq t \leq 10$ with a step size of 0.5.

   (a) Setup the matrix $A \approx \partial_x$ using second-order central differences and sparse storage.

   (b) Solve the advection equation with $c(x,t) = -0.5$, this corresponds to a constant velocity field. In other words, solve $u_t = 0.5Au$.

   (c) **This problem is required for AMATH 581 students but optional for AMATH 481 students. You will still need to save an answer for**

**A3 in your solutions, but you can just set A3 = 0 and still receive mastery on the homework.**

Solve the advection equation with

$$c(x,t) = -\left(1 + 2\sin(5t) - H(x-4)\right),$$

where

$$H(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$$

is the Heaviside function (`heaviside` in MATLAB and `np.heaviside` in python). You will be asked to think about what this velocity field means physically. (Hint: You can apply the heaviside function on an array. Think about how each equation (row) in the matrix $A$ is affected by $c(x,t)$.

**Deliverables:** Save the matrix $A$ to the variable `A1`. **To save the matrix $A$, you need to convert it to a full matrix. Use `full` in matlab and `A.todense()` in python**.

Save the solution $u$ with $c(x,t) = -0.5$ to the variable `A2`, which should have 200 rows and 21 columns (200 $x$ values, 21 $t$ values). Save the solution $u$ from part (c) to the variable `A3`. This should also be a $200 \times 21$ matrix.

**Presentation Skills:** To earn mastery in discussing problems from a physical perspective, discuss what $c(x,t)$ means and how it impacts the solution. Discuss the difference in the solution from $c(x,t) = -0.5$ and $c(x,t) = -\left(1 + 2\sin(5t) - H(x-4)\right)$. What does this non-constant speed correspond to? How does it affect the solution? Does this agree with your intuition? You should create and use the figures from (b) and (c) to backup your arguments.

(Hint: The velocity field has two components: one temporal (time) and one spatial. How does each component factor in?)

2. The time evolution of the vorticity $\omega(x,y,t)$ and the streamfunction $\psi(x,y,t)$ are given by the governing equations

$$\omega_t + [\psi, \omega] = \nu \nabla^2 \omega, \tag{1}$$

where $[\psi, \omega] = \psi_x \omega_y - \psi_y \omega_x$ and $\nabla^2 = \partial_x^2 + \partial_y^2$. The streamfunction satisfies

$$\nabla^2 \psi = \omega. \tag{2}$$

In most applications, the diffusion is a small parameter. Here we will let $\nu = 0.001$, $(x, y) \in [-10, 10]^2$, $t \in [0, 4]$, and let the initial vorticity equal to a *Gaussian bump*, stretched in the $y$ direction,

$$\omega(x, y, 0) = \exp\left(-2x^2 - \left(\frac{y^2}{20}\right)\right).$$

We will use periodic boundary conditions,

$$\omega(-10, y, t) = \omega(10, y, t), \qquad \omega(x, -10, t) = \omega(x, 10, t),$$
$$\psi(-10, y, t) = \psi(10, y, t), \qquad \psi(x, -10, t) = \psi(x, 10, t),$$

as encountered in class. You should use 64 points in the $x$ and $y$ directions and use a step size of $\Delta t = 0.5$ in time.

(a) Use sparse-matrix storage (`spdiags`) to generate the three matrices $A = \nabla^2 = \partial_x^2 + \partial_y^2$, $B = \partial_x$, and $C = \partial_y$ which take derivative in two dimensions. Use the discretization of the domain and its transformation to a vector as in class. Use the second-order central difference formula for each matrix. To make sure that $A$ is nonsingular, set the first element of $A$ (first row, first column) equal to 2 instead of the usual $-4$ (we will see why this is important on November 7).

(b) Now we want to integrate the equations numerically in time. Start by discretizing (2), using the matrices above, so that it is a linear problem, $A\vec{\psi} = \vec{\omega}$. Then discretize (1) using the matrices created above and use the built-in RK45 algorithm to step forward in time (your time step will include both solving $A\vec{\psi} = \vec{\omega}$ and the ODE (1)). In each step you will need to solve the linear system $A\vec{x} = \vec{b}$. Do this in two separate ways:

   i. using Gaussian Elimination (`A\b` in MATLAB or `scipy.sparse.linalg.spsolve` in python); and

   ii. using LU decomposition. When using LU decomposition, make sure to create the matrices $L$, $U$, and $P$ **outside** of the function and pass the

decomposition into the function for a fast solve. If using python, use `scipy.sparse.linalg.splu`.

We would like to compare the speed of each of the methods using `tic` and `toc` in MATLAB, or `time.time` in python.

**Deliverables:** Save the matrix $A$ to the variable `A4` (make sure you set the first element to 2 instead of the usual $-4$), the matrix $B$ to the variable `A5`, and the matrix $C$ to the variable `A6`. **To save the matrices, you need to convert them to full matrices. Use `full` in matlab and `A.todense()`, `B.todense()`, and `C.todense()` in python**. Save the solution ($\omega$) using Gaussian elimination to the variable `A7` and the solution ($\omega$) using LU decomposition to the variable `A8`. `A7` and `A8` should be $9 \times 4096$ matrices, corresponding to 9 time values and $64 \times 64 = 4096$ locations ($x$ and $y$ values). Finally, reshape the $9 \times 4096$ solution array from LU decomposition (`A8`) to one of shape $9 \times 64 \times 64$ corresponding to the $64 \times 64$ solution at the 9 time values and save it to the variable `A9`.

**Hint 1:** You will need to reshape the arrays several times, starting with putting the initial condition into a $4096 \times 1$ column vector (or array in python). You might want to first use a meshgrid (like we do in 3D plotting) and evaluate $\omega(x, y, 0)$ on that grid. You will then need to reshape it to a vector. Be careful about how you reshape and make sure that it matches the syntax we have used in class. If you are not sure about how the reshape commands work, try it with an example, e.g., $z = x + \frac{1}{2}y$ with $x = [1, 2, 3],\ y = [4, 5, 6]$. You may need to transpose the result.

**Hint 2:** You need to solve $A\vec{\psi} = \vec{\omega}$ for each time step. Make sure to include that step in your definition of the ODE.

**Hint 3:** Check your work! Create some contour plots using `contourf` or another 2D image creator. Your solution should look similar to Figure 14 of the Kutz lecture notes (Page 50).

**Presentation Skills:** There are two opportunities to earn presentation mastery here.

(a) To earn mastery in *animation*, create an animation showing the vortex dynamics over the interval $0 \leq t \leq 4$. This animation should be made using

`contourf`, `pcolor`, `tricontour`, `imshow`, etc. It should be the animation of a 2D plot (like if you animated Figure 14 of the notes). In order to make this animation smooth, you may need to decrease $\delta t$ from 0.5. Your result should not be choppy. This means you will need to do multiple solves of system, so you should use the fastest solver here. There will be some examples of how to create animations in class the week of November 7. You will turn this in on Canvas, not Gradescope.

(b) To earn mastery in *discussing problems from a mathematical perspective*, compute how long it takes to solve this PDE using Gaussian Elimination and LU decomposition. Do this for the specified grid size here and with 128 points in the $x$ and $y$ directions. Discuss the times that you find. Does it surprise you is it as you expect? Explain why it is as you expect or not.