Homework 7

This homework has two parts. First you will implement derivative operators for non-periodic functions. Second you will implement two different sets of equations with boundary conditions.

- **Derivatives:** You will construct derivative operators which can take derivatives of different orders. This will involve modifying the class `DifferenceUniformGrid` in `finite.py`. This operator takes derivatives by applying its `matrix` to data, so you must modify `self.matrix`, which is constructed in the `_build_matrices` method. The current matrix is mostly correct—you only need to modify its values which act on the function near the boundaries. I gave an example of what this should look like in the lecture. Do not assume any boundary conditions when constructing the matrix. Modify the matrix such that each row has the same number of non-zero entries. Note: This means some derivatives will not converge at order `convergence_order`. Why is that?

- **Diffusion Equation:** You will solve the equation

$$\partial_t c - D\nabla^2 c = 0, \tag{1}$$

  in two spatial dimensions, x and y. Assume c is periodic in y. In the x direction, c satisfies the boundary conditions

$$c|_{x=x_l} = 0, \tag{2}$$
$$\partial_x c|_{x=x_r} = 0, \tag{3}$$

  where x_l and x_r are the left and right endpoints of the x interval. Because we do not want to be limited by the CFL stability constraint, you should timestep this problem implicitly. The timestepping algorithm should be 2nd order accurate (use Crank-Nicolson). This will be implemented in a class `DiffusionBC` in the file `equations.py`, which will be passed an array for c, a diffusivity D, the desired spatial-order accuracy of the scheme, and the `domain` the equation is solved on. In addition to setting up the problem, this class should also keep track of the current time `t` and iteration `iter`. It should also have a `step` method which will take a timestep of size dt, and increment the `t` and `iter` attributes appropriately.

- **Wave Equation:** You will solve the equation

$$\partial_t \boldsymbol{u} + \boldsymbol{\nabla} p = 0, \tag{4}$$
$$\partial_t p + \boldsymbol{\nabla} \cdot \boldsymbol{u} = 0. \tag{5}$$

  This is the equation for linear sound waves that we solved in a previous homework, but with $\rho_0 = \gamma p_0 = 1$. You will solve this equation in two spatial

dimensions, x and y. All quantities are periodic in the y direction, and the boundary conditions in the x direction are

$$\boldsymbol{e}_x \cdot \boldsymbol{u}|_{x=x_l} = 0, \tag{6}$$

$$\boldsymbol{e}_x \cdot \boldsymbol{u}|_{x=x_r} = 0, \tag{7}$$

where x_l and x_r are the left and right endpoints of the x interval and $\boldsymbol{e}_x$ is the unit vector in the x direction. The velocity vector $\boldsymbol{u}$ can be decomposed into x and y components, $\boldsymbol{u} = (u, v)$. You should implement this in a class `Wave2DBC` in the file `equations.py`. The problem will be timestep explicitly, so you need to make an `F` function and a `BC` function and store them in `self.F` and `self.BC`. The class is called as

```
wave2DBC = equations.Wave2DBC(u, v, p, spatial_order, domain)
```

where `u`, `v`, `p` are all 2D arrays containing the initial conditions, `spatial_order` is the desired spatial-order accuracy of the scheme, and `domain` is the Domain object that the equation is solved on.