Homework 3

In this homework, you will implement two new classes for explicit timestepping. The `timesteppers.py` file has initialization signatures for the new classes, as well as classes for the timestepping schemes discussed in the lecture.

- The first new class is a `Multistage` timestepping class. This will implement Runge-Kutta multistage methods. These timesteppers will solve the equations

$$\partial_t u = f(u) \tag{1}$$

for a general nonlinear function $f(u)$. The user will specify the field `u`, the function `f`, the number of stages, and the a_{ij} and b_j coefficients of the method. Recall that Runge-Kutta methods are specified with a Butcher tableau

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array} \tag{2}$$

where corresponds to the scheme

$$k_1 = f\left(u^n\right) \tag{3}$$
$$k_2 = f\left(u^n + \Delta t k_1/2\right) \tag{4}$$
$$k_3 = f\left(u^n + \Delta t(2k_2 - k_1)\right) \tag{5}$$
$$u^{n+1} = u^n + \frac{\Delta t}{6}\left(k_1 + 4k_2 + k_3\right) \tag{6}$$

For this method, we would have `stages=3`, and `a` and `b` given by

```
a = np.array([[  0, 0, 0],
              [1/2, 0, 0],
              [ -1, 2, 0]])
b = np.array([1, 4, 1])/6
```

You `Multistage` class will need a `_step` function so it can be used to evolve the field `u` forward in time.

- You will also make a new class for multistep methods. In particular, you will derive and implement Adams-Bashforth timestepping methods. An Adams-Bashforth method discretizes the equation

$$\partial_t u = f(u) \tag{7}$$

as

$$u^{n+1} = u^n + \sum_{i=0}^{s-1} a_i f(u^{n-i}). \tag{8}$$

This method has s steps corresponding to f operating on the current data u^n, as well as the previous $s-1$ timesteps, $u^{n-1}, u^{n-2}, \ldots u^{n-s+1}$.

When initializing the Adams-Bashforth timestepper, the user will specify `steps`, the number of steps, as well as the timestep `dt`. Here we assume the timestep is constant. You will derive the coefficients a_i such that the scheme is s-order accurate. This can be done via Taylor expansion, as we've done for deriving spatial derivative.

Your `AdamsBashforth` class will have a `_step` function which will take a timestep, allowing someone to evolve `u` forward in time. *Warning:* An s order Adams-Bashforth scheme requires `u` at the previous $s-1$ timesteps. You will not have this data when you first start a simulation. For the first $s-1$ timesteps, you can use lower-order Adams-Bashforth schemes to approximate the solution (or something more accurate if you want!).