Homework 5

This homework has two parts. In the first part, you will implement a multistep IMEX scheme based on the backward differentiation formula. In the second part, you will implement two new equation sets.

- **New timestepper:** You will implement a new multistep IMEX timestepper called `BDFExtrapolate`. I've written out the start of an `__init__` method and the call signature for the `_step` method in `timesteppers.py`. This timestepper will solve the equation

$$M \cdot \partial_t X + L \cdot X = F(X) \tag{1}$$

  by approximating

$$\partial_t X^n \approx \sum_{i=0}^{s} a_i X^{n-i}, \tag{2}$$

$$L \cdot X \approx L \cdot X^n, \tag{3}$$

$$F(X)^n \approx \sum_{i=1}^{s} b_i F(X)^{n-i}. \tag{4}$$

  This timestepping scheme calculates the future value of X, denoted by X^n, using the current value, X^{n-1}, and past values, X^{n-2}, X^{n-3}, etc. The coefficients a_i and b_i can be found via Taylor expansion. The approximation of $\partial_t X^n$ is the backwards differentiation formula that you worked out in homework 4. The approximation of $F(X)^n$ is an extrapolation, because we only know $F(X)^{n-1}$, $F(X)^{n-2}$, etc., but the $F(X)^{n-i}$ for $i \geq 1$ can be extrapolated to estimate $F(X)^n$.

  As with the other multistep schemes we've used, you will need to take s timesteps before you can use the full scheme of order s. For the first $s-1$ timesteps, you should use the appropriate lower order schemes for the number of steps you have on hand. Assume the step size will stay constant.

- **Sound waves:** The perturbation equations for an ideal gas in one dimension are

$$\rho_0 \partial_t u' + \partial_x p' = 0, \tag{5}$$

$$\partial_t p' + \gamma p_0 \partial_x u' = 0. \tag{6}$$

  Here u' and p' are the velocity and pressure perturbations, ρ_0 and p_0 are the background density and pressure, and γ is the ratio of specific heats (equal to $5/3$ for a monatomic ideal gas). These equations admit sound waves which travel at the sound speed $c_s^2 = \gamma p_0 / \rho_0$.

You will implement an equation set to solve these equations for sound waves. In `equations.py` I've included the equation sets discussed in class, as well as the initialization call for the `SoundWave` class. This class will be given `numpy array`'s for u and p, as well as the first derivative operator (called `d`). In addition, it will be given ρ_0 and γp_0, which could be numbers, if those quantities are constant, or `numpy array`'s, if they are spatially dependent. Your job is to make a statevector `X`, matrices `L` and `M`, and function `F` for calculating RHS terms explicitly. These will be passed to IMEX timesteppers for time integration. I will be testing cases where ρ_0 and γp_0 are constant, as well as cases where they are spatially varying.

- **Reaction-Diffusion:** We will solve the following one-dimensional reaction-diffusion equation

$$\partial_t c - D\nabla^2 c = c(c_{\text{target}} - c). \tag{7}$$

The left hand side of this equation is the diffusion equation; the right hand side represents an auto-catalytic "reaction" of some material with concentration c. There are two equilibrium configurations: $c = 0$ and $c = c_{\text{target}}$. The first is unstable; the second stable. If you initialize with c zero everywhere, except near x_0, then the system will react to produce c until $c = c_{\text{target}}$. Then the diffusion of c about x_0 will lead to a reaction front which propagates at a velocity $\sim \sqrt{D/c_{\text{target}}}$. In the case of combustion, this reaction front is called a flame.

As for the sound wave problem, you will implement these equations in `equations.py` as a new class `ReactionDiffusion`. I have written the initialization call, which is passed a `numpy array` for `c`, the second derivative operator (`d2`), the target value `c_target` (which could be either constant or a `numpy array` if spatially dependent), and the diffusivity D. As above, your code should specify the statevector `X`, matrices `M` and `L`, and function `F` for calculating terms explicitly. You should test cases where c_{target} is either constant, or spatially varying.