Homework 3

Time Integration

The goal of this worksheet is to look at a non-linear PDE in 2D, and how we can solve it with finite difference methods/Operator splitting. Please feel free to work on it together with your group or individually. However, everyone should come to our meeting with their own write-up.

The Allen-Cahn equation is very nice model used in fluid systems with two inmiscable fluids or phases (e.g oil and water, water and air, liquid copper and iron). If the fluids (say oil and water) begin well mixed, they will slowly separate over time into two regions - one region of all water, another of all oil. We model the 'phase' of the fluids with a single function

$$\phi(x, y, t)$$

where $\phi = 1$ corresponds to the 'water' phase and $\phi = -1$ to 'oil', while values in between are sort of a 'fuzzy' transition/interface region. The Allen-Cahn equation models the dynamics of ϕ as

$$\phi_t = \epsilon^2 \nabla^2 \phi - F'(\phi), \quad x, y \in [0,1] \times [0,1] \tag{1}$$

$$F(\phi) = \frac{\left(\phi^2 - 1\right)^2}{4}. \tag{2}$$

$$\phi_x(x = 0, 1, y, t) = \phi_y(x, y = 0, 1, t) = 0, \tag{3}$$

where ϵ is a small number related to the width of the 'interface' region, and $F'(\phi) = \phi^3 - \phi$ denotes a derivative of F with respect to ϕ.

1) Basic Operator Splitting In a language of your choice, implement a finite difference discretization of the Allen-Cahn equation. Use a grid size of your choice, but keep $\Delta x = \Delta y$ and take $\epsilon = \Delta x$. Use the following random initial condition

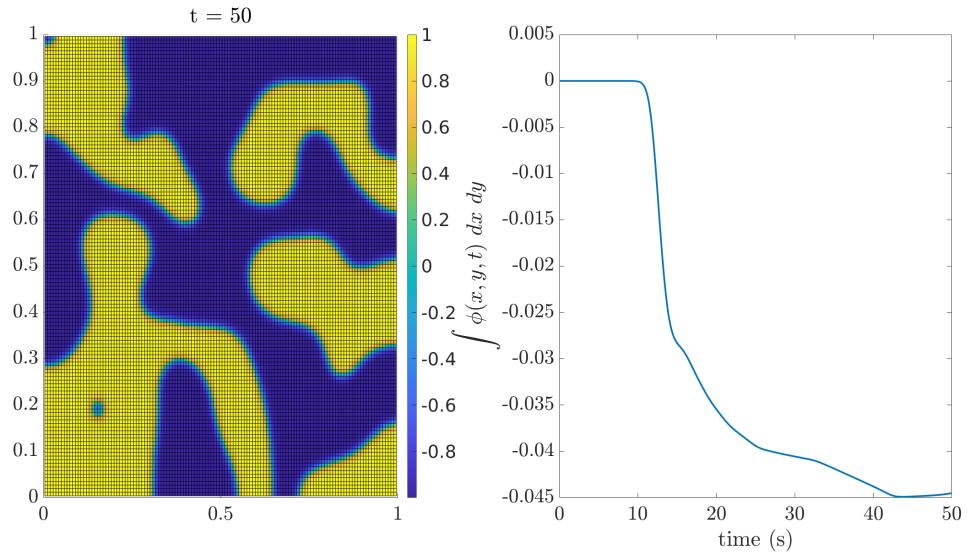
```
rng(0); % set a seed to make randomness reproducible
U = epsilon*randn(N*N,1); % random initial data with variance epsilon^2
U = U - mean(U); % make the initial data have mean = 0;
```

and simulate up to $t_{\text{final}} = 50s$. To integrate in time, use the following operator splitting scheme:

1. Solve: $\phi_t = \epsilon^2 \nabla^2 \phi$ from $t = n\ \Delta t$ to $t = (n+1)\ \Delta t$
2. Based on the solution to 1) solve: $\phi_t = F'(\phi)$ from $t = n\Delta t$ to $t = (n+1)\Delta t$.

Nottice that step 2) is an **ODE** for ϕ which can be solved **analytically** for some arbitrary initial data ϕ_0. Call the solution to step 2) $\phi^{\text{ODE}}[t_f, \phi_0]$ and note that it depends only on the initial data ϕ_0 and the final time t_f, then we can write our operator splitting scheme as

1. Solve:
$$\frac{\phi^* - \phi^n}{\Delta t} = \epsilon^2 \mathbf{L} \phi^*$$
(where **L** is the finite difference laplacian)



2. Evaluate

$$\phi^{n+1} = \phi^{\text{ODE}}\left[t_f = \Delta t,\ \phi_0 = \phi^*\right]$$

[**Hint: the sign of ϕ^{n+1} should have the same sign as ϕ^***]

Note that you will have to experiment a bit to find a stable time step Δt (why aren't we guaranteed stability?) but a good place to start is with $\Delta t \propto \Delta x$.

2) Conservative Allen-Cahn (don't tread on my integral) If we plot the total mass in the system vs time

$$\int_{[0,1]\times[0,1]} \phi(x,y,t)\ dxdy$$

we can see that it changes by a somewhat significant amount. This is a **bad** kind of error since it's very unphysical (mass is beiing created/destroyed). Take the following steps to fix it

1. Show that if

$$\beta(t) = \frac{\displaystyle\int_{[0,1]\times[0,1]} F'(\phi)\, dxdy}{\displaystyle\int_{[0,1]\times[0,1]} \sqrt{F(\phi)}\ dxdy},$$

the the following **modified** Allen-Cahn equation

$$\phi_t = \epsilon^2\nabla^2\phi - F'(\phi) + \beta(t)\sqrt{F(\phi)} \tag{4}$$

(with the same BCs and ICs as before) will conserve the total mass. [Hint use $\frac{d}{dt}\int \phi\, dxdy = \int \phi_t\ dxdy$ and then try to apply the divergence theorem]

2. Modify the operator splitting scheme from part 1) to solve equation (4) as follows

(a) Solve:

$$\frac{\phi^* - \phi^n}{\Delta t} = \epsilon^2 \mathbf{L}\phi^*$$

(where **L** is the finite difference laplacian)

(b) Evaluate

$$\phi^{**} = \phi^{\text{ODE}}\left[t_f = \Delta t,\ \phi_0 = \phi^*\right]$$

(c) Solve:

$$\frac{\phi^{n+1} - \phi^{**}}{\Delta t} = \beta^{n+1}\sqrt{F\left(\phi^{**}\right)}$$

Step c) requires you to find β^{n+1} so that the **discrete mass** (sum rather than integral) is exactly conserved. By summing both sides of step c) and arguing that the mass at step $n+1$ must not change from the mass at step n, show that

$$\beta^{n+1} = \frac{1}{\Delta t}\frac{\sum_{i,j}\phi_{ij}^0 - \phi_{ij}^{**}}{\sum_{i,j}\sqrt{F\left(\phi_{ij}^{**}\right)}}$$

3. Show numerically (eg with a plot) that your numerical solution to the modified Allen-Cahn exactly conserves the sum $\sum_{i,j}\phi_{ij}^n$

3) Charlie's rule: play around with your code! The following are just some ideas that are **not** sorted by difficulty. Some might make good course projects and some are just small modifications on what you've already done. But the only purpose of this question is to get you to do something fun with your code and show it off.

1. You'll notice that your simulation spends the first 10 seconds or so rapidly 'coarsening' into two distinct phases, and after this point the dynamics become **very** slow. So why are you still using a very small time step? Try implementing either a) a **fully** implicit method which will require using Newton's method at each step rather that operator splitting in order to account for the nonlinear term]. b) implement some for of adaptive time stepping to take larger steps after the system phase separates.
2. Try using different initial conditions and see what kinds of patterns you can generate after the system has some time to evolve
3. You'll notice that you also have to use a fine grid *in space* to resolve the interface region - but after the system phase separates, most of your solution is a constant ($\phi = 1$ or $\phi = -1$) so you shouldn't need that much resolution. Try to implement some form of **adaptive mesh refinement** so that your spatial grid is only resolved near the interface. [**note that this is very cool, but would be on the harder side**]

4. Look into other reaction diffusion systems. Some equation that admit very cool solutions (like spiral waves) are: FitzHugh–Nagumo, Belousov–Zhabotinsky reaction, Swift–Hohenberg,...
5. You can use *contour* in MATLAB to get the $\phi = 0$ contour (the interface between the phases). Compute and plot both the mean and maximum curvature of the interface over time. What do you notice?
6. Implement some form of the immersed-boundary method to solve the heat equation (or even Allen-Cahn) in a funny shaped domain.
7. Try to beat backslash. Look into fast methods to solve the linear system involved in the heat equation. One can show that a multi-grid method is optimally fast for systems related to the finite difference Laplacian. Try to implement a multi-grid v-cycle in 2 or 3D and time it against MATLAB's backslash operator (or python's builtin sparse solvers). [note that you're more likely to beat MATLAB in 3D, but this will be slightly harder to code]