3. Partial differential equations

3.1. Initial value problems: the Cahn-Hilliard equation

The Cahn-Hilliard equation describes phase separation in a physical system such as a water-and-oil emulsion. The equation is written in terms of a compositional order parameter, $\phi(\mathbf{x}, t)$, which depends on position $\mathbf{x}$ and time t which is, for instance, positive if locally there is more water than oil, and negative otherwise. The overall integral of this order parameter is conserved, as water cannot turn into oil (or vice versa). The order parmeter ϕ obeys the following (Cahn-Hilliard) equation,

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} = M\nabla^2 \mu(\mathbf{x}, t) \tag{1}$$

where M is a positive constant, and the chemical potential $\mu(\mathbf{x}, t)$ is given by,

$$\mu(\mathbf{x}, t) = -a\phi(\mathbf{x}, t) + a\phi(\mathbf{x}, t)^3 - \kappa\nabla^2\phi(\mathbf{x}, t), \tag{2}$$

with a, κ positive constants with the appropriate dimensions.

1. A numerical solution of Eq. 1 involves the following explicit (Euler) algorithm,

$$\begin{aligned}\phi(i, j; n+1) &= \phi(i, j; n) + \frac{M\delta t}{\delta x^2}\Big[\mu(i+1, j; n) + \mu(i-1, j; n) \qquad (3)\\ &+ \mu(i, j+1; n) + \mu(i, j-1; n) - 4\mu(i, j; n)\Big],\end{aligned}$$

   where δx and δt denote spatial and temporal discretisation step, while i and j label position in $2D$ (along x and y respectively). Write down a Python code to simulate the Cahn-Hilliard equation based on this algorithm (you'll need to discretise the chemical potential $\mu(\mathbf{x}, t)$ as $\mu(i, j)$!). As simulation domain, you should use a square lattice, with periodic boundary conditions, ideally with dimension 100×100 (or, if this proves too slow, you can use 50×50).

2. Experiment with values of parameters, and with δx and δt, to see when the algorithm converges. An appropriate choice of parameters is $a = M = 0.1$, $\kappa = 0.1$; as an initial condition you should choose $\phi(\mathbf{x}, t) = \phi_0$ plus some (small) random noise.

3. What is the behaviour of the system for $\phi_0 = 0$ and for $\phi_0 = \pm 0.5$? Use your code to find the qualitative behaviour.

4. The free energy density, f, associated with the order parameter ϕ can be defined as follows,

$$f = -\frac{a}{2}\phi(\mathbf{x}, t)^2 + \frac{a}{4}\phi(\mathbf{x}, t)^4 + \frac{\kappa}{2}\left(\nabla\phi(\mathbf{x}, t)\right)^2. \tag{4}$$

Over time, we expect the system should evolve so as to minimise the free energy; compute the free energy density over time to show this is the case in practice in your algorithm.

3.2 Boundary value problems: Poisson's equation

The electrostatic potential $\varphi(\mathbf{r})$ due to a (scalar) field of electric charges $\rho(\mathbf{r})$ is the solution of Poisson's equation

$$\nabla^2\varphi = -\frac{\rho}{\epsilon_0}, \tag{5}$$

where ϵ_0 is the dielectric constant of the vacuum, and where one should imagine appropriate boundary conditions are imposed. The electric field is obtained from the potential via $\mathbf{E} = -\nabla\varphi$.

1. In free space, the appropriate boundary condition on φ is that it should vanish at infinity. On a computer, we are necessarily restricted to a finite system. You should therefore set $\varphi = 0$ on the boundary of your simulation domain. [Which boundary condition does this correspond to: Dirichlet or Neumann?]

2. A numerical solution of Poisson's equation involves discretising space, so that the values of $\varphi(\mathbf{r})$ and $\rho(\mathbf{r})$ are defined only on a lattice points $\mathbf{r}_{ijk}$ where i, j, k are integers and δx is the lattice spacing. In practice, we can choose $\delta x = \epsilon_0 = 1$ (in simulation units). [You should convince yourself that this is possible without loss of generality.]

np.gradient

3. Write down an approximate expression for the electric field $\mathbf{E}$ as a function of i, j and k given the values of ρ on this lattice. This will be useful for your code later on.

4. Let $\varphi(i,j,k;n)$ be an estimate of the potential after n steps of some iterative algorithm. The Jacobi algorithm improves on this estimate via the update

$$\begin{aligned}\varphi(i,j,k;n+1) \quad = \quad & \frac{1}{6}\Big[\varphi(i+1,j,k;n) + \varphi(i-1,j,k;n) \\ & +\varphi(i,j+1,k;n) + \varphi(i,j-1,k;n) + \\ & \varphi(i,j,k+1;n) + \varphi(i,j,k-1;n) + \rho(i,j,k)\Big]\end{aligned} \tag{6}$$

where we have taken $\epsilon_0 = 1$. Eventually this algorithm will converge, that is, $\varphi(i,j,k;n+1) = \varphi(i,j,k;n)$ for sufficiently large n. Construct a measure of the distance between two successive estimates of φ, and use this to devise a condition for terminating the iterative algorithm.

5. An alternative to the Jacobi algorithm is the Gauss-Seidel algorithm, which is the same except that the same array of values is used for the

previous estimate (on the rhs of the previous equation) and the next estimate (on the lhs). That is, the array is updated in-place, as opposed to creating a new array with values based on the old one. You should convince yourself that this algorithm converges to the same solution.

6. Write a Python program to solve the Poisson equation for an arbitrary charge distribution contained within cubic box in three dimensions and with fixed boundary conditions $\varphi = 0$ at the edge of the box. Your program should allow the user to choose the system size and control the accuracy of the final solution. It should display a representation of the potential and resulting electric field once the calculation is finished (or it should write the data to a file which you can then visualise with gnuplot).

7. Use your code to calculate the potential and field due to a single charge at the centre of the box. Does the result agree quantitatively with what you would expect from Gauss' law?

8. A magnetic field can be expressed as the curl of a vector potential $\mathbf{A}$. If one chooses a gauge so that $\nabla \cdot \mathbf{A} = 0$, Maxwell's equations imply that the magnetic potential arising from a current field $\mathbf{j}$ is $\nabla^2 \mathbf{A} = -\mu_0 \mathbf{j}$. Suppose that a system comprises wires that all point in the z direction. Show that the magnetic potential has $A_x = A_y = 0$ and that A_z depends only on x and y. What are appropriate boundary conditions to use to solve the equation $\nabla^2 A_z = -\mu_0 j_z(x, y)$ numerically?

9. Modify your program to solve for the magnetic potential due to a single wire aligned with the z axis, and running through the origin. Your code should also compute the resulting magnetic field. (Hint: relative to your existing code, this amounts only to a change of the charge field, and the relationship between the potential and resulting vector field; it should be possible to reuse your existing Poisson equation solver more-or-less intact).

10. A trick to speed up the convergence of the Gauss-Seidel algorithm is to 'over-relax'. That is, if the difference between the current and next estimate of the potential at a point is $\delta\varphi$, then over-relaxation involves adding on an amount larger than this to the current value to modify the next estimate: $\varphi' = \varphi + \omega\delta\varphi$ where $\omega > 1$. Can you find the value of ω that minimises the number of iterations required to reach convergence (you can select one case and study convergence as a function of ω)?

3.3 Figure checklist for checkpoint

For convenience, the basic plots and datafiles which you should have for the quantitative analysis for this checkpoint are given below (these correspond to the graphs and datafiles mentioned in the marksheet).

1. For Cahn-Hilliard, you should have a plot of the free energy versus time, ideally for both the case of $\phi_0 = 0$ and $\phi_0 = 0.5$. You should ideally

✓

consider a sufficient number of timesteps in the plot that the free energy converges to an equilibrium value. If this is too costly computationally, you should at least show the behaviour for 50000 iterations.

2. For the monopole (or Gaussian charge) graphs, you should have a contour plot of the potential strength in space (a cut through the midplane), a vector plot of the electric field, and a datafile with the potential and electric field in 3D, or on a cut through the midplane. We will be checking the behaviour of potential and electric field strength as a function of the distance to the charge, so you can if you wish already plot and fit these two curves. ✓

3. For the magnetic field problem, you should have a contour plot of the vector potential, a vector plot of the magnetic field, and datafiles for the vector potential and magnetic field. We will be checking the behaviour of vector potential and magnetic field strength as a function of the distance to the wire, so you can if you wish already plot and fit these two curves. ✓

4. For the SOR part, you should have a plot of the number of iterations to solve one of the two problems above (electric or magnetic) as a function of ω. ✓