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,

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

   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 code to simulate the Cahn-Hilliard equation based on this algorithm (you'll need to discretise the chemical potential $\mu(\mathbf{x}, t)$ as $\mu(i, j; n)$!). As simulation domain, you should use a square lattice, with periodic boundary conditions.

2. Experiment with values of parameters, and with δx and δt, to see when the algorithm converges. An appropriate choice of parameters is $a = M = 0.1$, $\kappa = 0.01$; 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 $\phi_0 = \pm 0.5$? Explain your results physically.

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 Poissons 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. What physical set-up does putting φ on the boundary of some box correspond to?

2. A numerical solution of Poissons 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 pracice, we can choose $\delta x = \epsilon_0 = 1$ (in simulation units); can you see why we can do this?

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.

4. Let $\varphi(i,j,k;n)$ be an estimate of the potential after n steps of some iterative algorithm. The Jabobi 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. Why does this converge to the same solution?

6. Write a Java 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.

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? Why (not)?

8. A magnetic field can be expressed as the curl of a vector potential $\mathbf{A}$. If one chooses a gauge so that $\nabla \cdot \mathbf{A} = 0$, Maxwells 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 Java program to solve for the magnetic potential due to a single wires 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)?