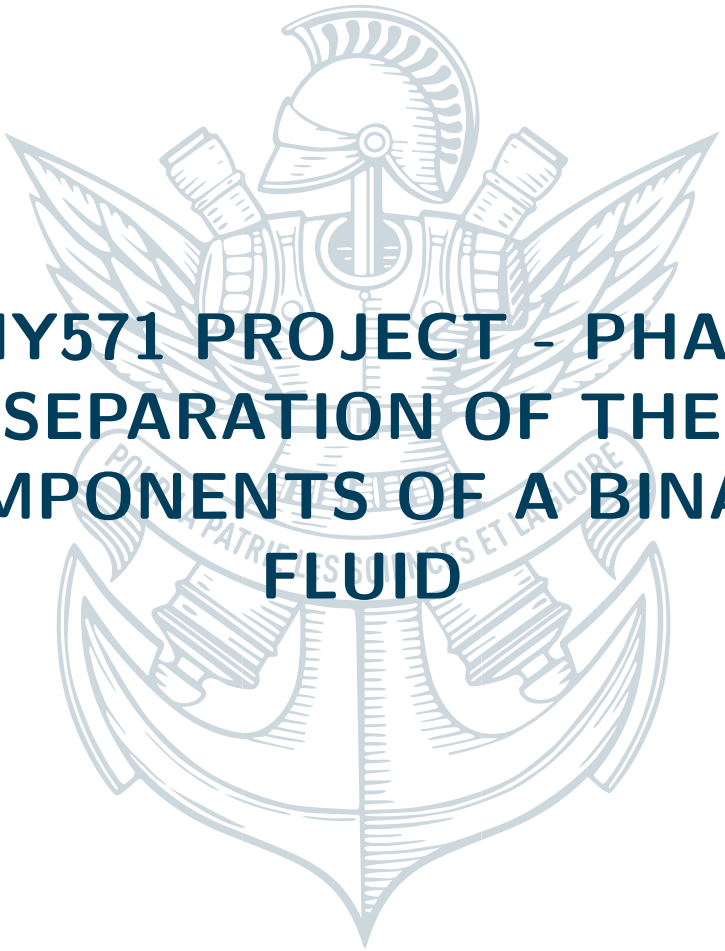

PHY571 PROJECT - PHASE SEPARATION OF THE COMPONENTS OF A BINARY FLUID

Project report

10 December 2021

MACHADO Wesley, DAUMAS Gaspard, MAHJOUB Rayen

Contents

Introduction

A **binary liquid** is a type of chemical combination, which creates a special reaction or feature as a result of mixing two liquid chemicals, that are normally inert or have no function by themselves. A number of chemical products are produced as a result of mixing two chemicals as a binary liquid, such as plastic foams and some explosives. Binary fluid mixtures are examples of complex fluids whose microstructure and flow are strongly coupled. For pairs of simple fluids, the microstructure consists of droplets or bicontinuous demixed domains and the physics is controlled by the interfaces between these domains. At continuum level, the structure is defined by a composition field whose gradients – which are steep near interfaces – drive its diffusive current. These gradients also cause thermodynamic stresses which can drive fluid flow.

The **Cahn-Hillard** equation is useful for retrieving the phase diagram of the fluid, but an explicit resolution of it is too complex. In this report, we will start by describing the problem theoretically. We will then describe a numerical method to tackle it and obtain an approximate solution, as well as the specifics of said method; we will also discuss the physics that surround the problem while incorporating them in our numerical analysis.

Chapter 1

Theory

1.1 Cahn-Hillard Equation

The Cahn-Hilliard equation is a non linear fourth order differential equation which is useful in solving problems such as the phase separation of a binary liquid mixture, tumor growth, the thermal induced phase separation etc. In our attempt to model the phase separation of the components of a binary fluid, the aim is to solve for a concentration variable $c(\mathbf{x}, t)$ that describes the spatial distribution of the two phases. A main advantage of solving the Cahn-Hilliard equation is that the border between the phases is continuous and does not have to be defined precisely.

We will here give hints on how to derive the C-H equation from the laws of thermodynamics.

First we consider a system of a mole of a binary solid system of molecules A and B, and we calculate its Helmholtz free energy density $F = E - TS$.

We define c as the proportion of B-type molecules : $c = N_B/(N_A + N_B)$

The mixing of the particules from both phases creates a difference in entropy $\Delta S = kln\frac{W}{W_0}$, where W is the number of ways to order N_A molecules of type A and N_B of type B once they have been mixed together. Basic calculations lead to $\Delta S = -R[(1-c)ln(1-c) + clnc]$, where R is the universal gas constant.

To derive the internal energy change ΔE due to mixing, we now consider the energies of the different types of bonds (A-A, B-B A-B). We have: $E = P_{AA}\epsilon_{AA} + P_{AB}\epsilon_{AB} + P_{BB}\epsilon_{BB}$, with P_{ij} the number of ij bonds and ϵ_{ij} the energy of the bond. Making the hypothesis that every molecule has z neighbors, we conclude that $N_A z = P_{AB} + 2P_{AA}$, and a similar equation for B.

We finally have to find P_{AB}. To do so, an argument is that there are $N_a z/2$ bonds (Avogadros number) in one mole of the solid, and that each bond has a probability $2c(1-c)$ of being an AB bond (each site is occupied by A (B) with probability 1-c (c)), so that the number of bonds AB is finally $N_a zc(1-c)$. All in all, the inernal energy due to the mixing is $\Delta E = \Omega c(1-c)$ (where we don't consider the terms not depending on c), with Ω a constant, here equal to $zN_a(\epsilon_{AB} - 0.5(\epsilon_{AA} + \epsilon_{BB}))$, which can be positive or negative.

In the end, the molar Helmholtz free energy reads: $\Delta F = \Omega c(1-c) + RT[(1-c)ln(1-c) + clnc]$ This function can be plotted for different temperatures:

What we notice is that as long as T is greater than T_{lim}, the long terme solution to the problem will be homogeneous with concentration 0.5 everywhere, as this configuration will lower the energy of the system. But something more interesting happens when we lower the temperature: the homogeneous solution sponteanously evolves towards a binary fluid with two phases of different c.

The next step is to compute the total free energy of the volume of non-uniform concentration: $e(c) = N_V \int_\Omega f d\mathbf{x}$ with N_V the density of molecules and $f(c, \nabla c, \nabla^2 c, ...)$ the free energy of a non

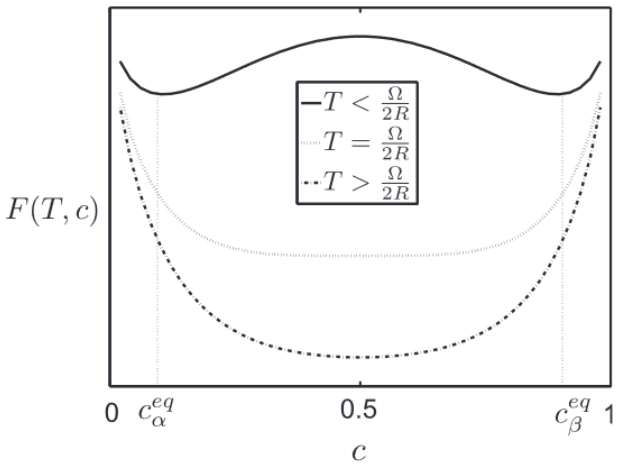

Figure 1.1: Molecular free energy as a function of c for different Temperatures

uniform area. Some considerations of Taylor expansions, symmetry, border conditions and calculus lead to the result: $e(c) = \int_\Omega [F(c) + \frac{\epsilon^2}{2}|\nabla c|^2] d\mathbf{x}$

To conclude, we introduce the chemical potential $\mu = \frac{\delta e}{\delta c}$. The net flux of components B $I = -M\nabla\mu$ satisfies the continuity equation $\frac{\partial c}{\partial t} = -\nabla I$, and finally gives us the Cahn-Hilliard equation:

$$\frac{\partial c}{\partial t} = \Delta(F'(c) - \epsilon^2 \Delta c)$$

1.2 Treating the problem: Numerical methods and error

Solving the CH equation explicitly is not possible; thus, we will resort to numerical simulations which are essential in understanding long-term behaviours of the solutions of the equation. There are several methods applicable for solving this problem, and they all rely on the first step of discretizing the time variable in steps of dt; some are more naive than others, but they all eventually yield acceptable results, provided the discretization is narrow enough.

We will now consider the C-H equation: $\frac{\partial \phi}{\partial t} = \Delta(F'(\phi) - \epsilon^2 \Delta\phi)$, with $F(\phi) = 0.25(\phi^2 - 1)^2$. Several schemes are possible to recursively solve for $\phi(kdt) = \phi^k$, and we will be especially interested in two of them.

Some preliminary hypotheses are:

- ϕ is obviously space-dependent as well, and we will be treating the problem in 2D.
- The Fourier transform of ϕ will be noted $\tilde{\phi}$.
- We will discretize space as well (and there will be a corresponding Fourier space discretization associated to it), but this will not bring further error to the calculations as we have transformed our problem into a simple equation resolution in Fourier space.

1.2.1 Semi implicit Euler scheme

The fourth order term will be treated implicitly and the others explicitly. Taking the Fourier transform of the equation and applying this scheme gives:

$$(1 + dt\epsilon^2|\mathbf{k}|^4)\tilde{\phi}^{n+1}(\mathbf{k}) = \tilde{\phi}^n(\mathbf{k}) - dt\mathbf{k}^2 F'(\tilde{\phi^n})(\mathbf{k})$$

Since $\frac{\phi^{n+1}-\phi^n}{dt} = \phi'(n.dt) + \frac{dt}{2}\phi''(n.dt) + O(dt^2)$ (the derivation is done with respect to time here), an error of $\frac{dt}{2}\phi''(n.dt)$ propagates from the nth term to the (n+1)th term. This means that, as long as the second derivative in time of ϕ is bounded, the error will not grow out of proportion. We can verify through experiments that, since a good approximation of ϕ''^n is $\frac{\phi^{n+1}+\phi^{n-1}-2\phi^n}{dt^2}$, ϕ'' remains within an acceptable threshold.

Of course, one could argue that there is an error on ϕ'' depending on ϕ''', and in fact for a large enough value of dt the process does not converge, but the experiments show that ϕ' and ϕ'''s value remain consistently low (for a low enough time of execution). This may be because we are looking at a continuous function in a compact set [0, N_t dt] x [0,N_x dx] x [0,N_y dy].

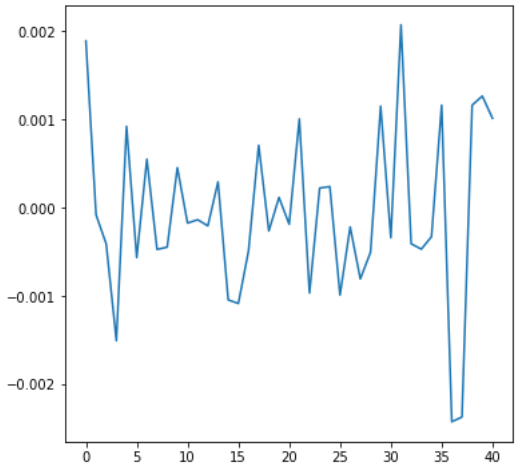

Figure 1.2: Calculating $\sum_{space} \phi''(x, y)$ throughout the experiment

Finally, the **discretization error** is bounded by $\frac{\phi''_\infty}{2}$dt. There is also the advantage that this is a **semi-implicit scheme**, in the sense that we gain in the method's stability by not using a fully explicit scheme, as well as in simplicity by not using a fully implicit one.

1.2.2 Linearly stabilized splitting scheme

In this scheme the $-\phi$ in F' is split in -3ϕ that are treated explicitly and 2ϕ that are treated implicitly. The rest doesn't differ from the semi implicit scheme. This gives:

$$(1 + dt(\epsilon^2|\mathbf{k}|^4 - 2\mathbf{k}^2))\tilde{\phi}^{n+1}(\mathbf{k}) = \tilde{\phi}^n(\mathbf{k}) + dt(\mathbf{k}^2\tilde{\phi^n}(\mathbf{k})^3 - 3\mathbf{k}^2\tilde{\phi^n}(\mathbf{k}))$$

Chapter 2

Algorithm and Code Specifics

2.1 Spectral solver

This section details the code **implementation**. For its actual documentation use the `help()` method in a Python interpreter. `help()` is designed to be used interactively and has a simple usage. In Jupyter pressing `Shift+Tab` shows a popup with the documentation. You can also directly read the docstrings and comments in the code itself.

The code design is very simple. A class representing the problem is implemented (`FFTspectral` or `DCTspectral`), with attributes, or fields, storing the problem's information `(Lx, Nx, epsilon)`. None of the attributes are left constant, so everything can be played with mid-run. This obviously may cause bugs, but the freedom of intervention was preferred in our implementation. Basically only two fields are really meant to be updated on-the-fly, not left static as an initial condition, the ones representing our desired order parameter $\phi(\mathbf{x}, t)$, and its spectral decomposition $\phi(\hat{\mathbf{k}}, t)$ a function of space, which we aim to determine over time. It is stored in the field `u` of the object. The `t` field is used to store and keep track of the elapsed time since the object's initialization. In the end we can see the data on $\phi(\mathbf{x}, t)$ and $\phi(\hat{\mathbf{k}}, t)$ as a snapshot on a given instant in time containing only their spatial/spectral dependency. The data object can be evolved in time with methods detailed further on in this section.

In the initialization we also create two arrays with a discretized representation of the two axis of the space. We call it the spatial grid, with `Nx, Ny` equally spaced values in each axis. The same values `Nx, Ny` are used in its dual, the spectral domain. The numerical representation of the spectral domain is detailed later. Instead of leaving these representations as actual grids we separated them in their two component axes, `x` and `y` for example. Implementing a grid would probably make it clearer, but separating it in two axes was easier for debugging and also for initial development.

The contructor or initializer of the class simply takes all the system's information as input and stores it in the fields of a new object of the class. A good way to visualize it is that each time we initialize an object of this class a new problem system is created.

The temporal evolution of the system is thus done by updating the data object (its `u` and `u_hat` fields, mainly) dynamically. The spectral classes provide an evolution method to evolve the data object in time through time steps. As such, time stepping methods are at the heart of the evolution method. The stepping method is also set as a field of the object, the `step` field. It can be reassigned even mid-run. This is useful for using different stepping methods on different scales of time, since their performance varias between them.

Now for the temporal stepping method implementation. It is our main 'evolution problem' solver. It operates on $\hat{\phi}$, the Fourier Transform of ϕ. The complicated part, which cost us time, was the understanding of the numerical representation of the discretized wavelength domain, which is different for the FFT and DCT algorithms. Before we enter in further details on this discussion, remember: we are originally working in a spatial domain, but we use a Fourier Transform to take

it to the wavelength domain. The `freq` terms in the various Python FFT algorithms allude to frequencies, but our input data is not temporal. It is spatial. If you look into it, the help page on the fftfreq method, on both the scipy and numpy packages, provides no information on why it is arranged like it is (Positive values first, negatives later). As for the DCT's range of used frequencies, the scipy.fft.dctn help page is quite easy to understand. For the type-2 DCT on n-dimensional space the frequency domain is represented as an array of shape `s`, a simple grid of equally spaced points starting from 0. The shape `s` defaults to that of the original domain. We have the option to discretize the spectral domain differently. The negative values in the frequency grid for the FFT are left latter for a simple reason: they are seldom used. This is due to redundancy in the transformation of real-valued functions. The negatives represent complex conjugate terms $(e^{-2\pi\lambda t})^* = e^{-2\pi(-\lambda)t}$. Since our ϕ, or `u` field, is real-valued they are needless. So it actually makes sense to leave these values in the latter half. For most use cases, only the first half is essential.

Here are some results we've obtained:

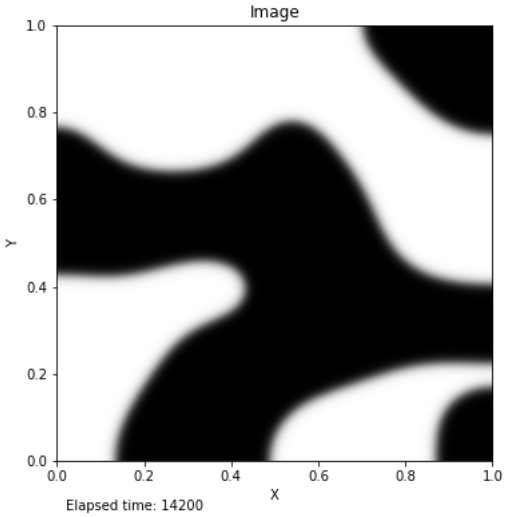

Figure 2.1: Binary fluid phase diagram after 14200 iterations (dt=100).

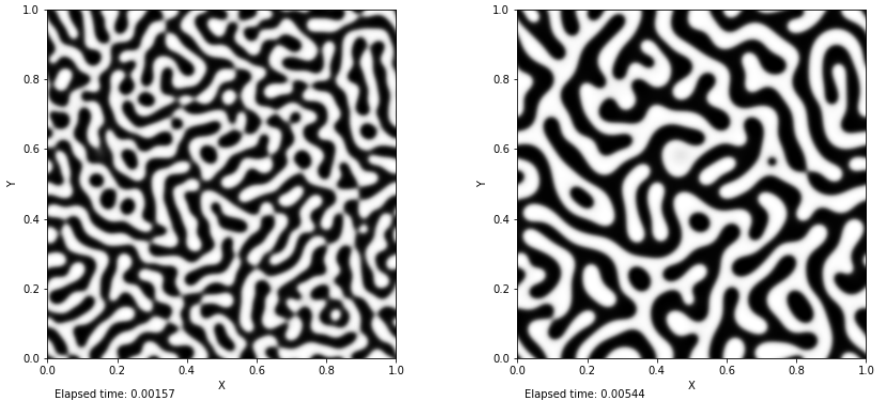

Figure 2.2: Same experiment with a shorter elapsed time (dt = 10^{-5}). Left: 157 steps, right: 544 steps.

2.2 Numerical analysis

1. Conservation of the total relative concentration Φ

We define Φ as the integral of our order parameter over all the available space Ω, that is,

$$\Phi(t) = \int_{\Omega} \phi(\mathbf{x}, t) dS$$

Just like the total number of particles, Φ is conserved. To demonstrate so, we start by reminding ourselves that $\phi(\mathbf{x}, t) = c_A(\mathbf{x}, t) - c_B(\mathbf{x}, t)$. Whence

$$\Phi(t) = N_A - N_B$$

Φ is thus conserved over time, since N_A and N_B also are. As a side-note:

$$N = N_A + N_B = \int_{\Omega} dS = \mu(\Omega),$$

which is the measure of the area of Ω, which has 0 error for a fixed-size grid.

Since $2N_A = \Phi + 1$, Φ is conserved if and only if N_A and N_B also are.

Finally, we define as an error metric the change in this quantity. Numerically, we just calculate the sum of the relative concentration over the whole spatial grid. We define

$$\mathcal{E}(t) = |\Phi(t) - \Phi_0|$$

We take the logarithm since the scales are very small and note that for our tests $log_2\mathcal{E}(t) \approx -58$. Which is on the order of the precision of the `float64` type we used (2^{-53}).

2.3 Understanding the free energy density

Here we make a short discussion about the function $f(\phi) = F(\phi) + \frac{\epsilon^2}{2}|\nabla\phi|^2$ (density of free energy)

The second term on $f(\phi)$ describes the "gradient energy", or how the local variability in ϕ increases the free energy. So as to decrease the free energy, the system normally increases the entropy through mixing, this is accounted for in the first term $F(\phi)$. But the second term is responsible for local regularity - locally, it favors regions of less variant concentration. That is, it accounts for the decrease in the free energy due to the homogeneity in the system's composition, modelling spinodal decomposition in a simple way. Higher homogeneity implies smaller local variation in composition. Which implies smaller magnitude of concentration gradient. Which implies lower $\frac{\epsilon^2}{2}|\nabla\phi|^2$. Which implies lower free energy.

So ϵ could be visualized as being responsible for overall homogeneity. For higher ϵ values, to minimize the Helmholtz free energy, the system tends to make the gradient smaller overall, creating less but larger "islands of constancy" pure in each phase. This can also be seen as less total "transition regions" or "surfaces of variation" between the 2 main phases. So after long time steps we would expect spherical/circular boundaries between the 2 main phases to form (minimizing the transition surface). Relatively high ϵ values give rise to very weak phase separation, resulting in a miscible, homogenous, monophasic fluid. This was observed numerically as ϕ converging to its average value in all points of space, thus becoming constant over (Ω, T).

Chapter 3

Conclusion

In this report we have presented some numerical methods to solve the Cahn-Hillard equation and the phase separation problem in a binary fluid. We have been able to derive some results for different time discretizations and different initial spreads of the fluid components' relative concentrations, namely the fact that if the fluid's composition is fairly imbalanced towards one of its parts, we obtain a phase diagram that looks like bubbles of said part dispersed among the rest of it. Our main conclusion is the "homogenizing" effect of the added term on the free energy density function. Normally, when $\epsilon = 0$, we would expect a random distribution of concentrations -1 and 1, pure black and pure white, which would maximize entropy and thus minimize our objective, the free energy. In our case we clearly observed we still had these almost pure black and white regions, but with a miscibility effect in their boundaries, which is a direct consequence of a tendency for the minimization of the fluid's spatial variability in constitution, $|\nabla\phi(\mathbf{x}, t)|^2$.

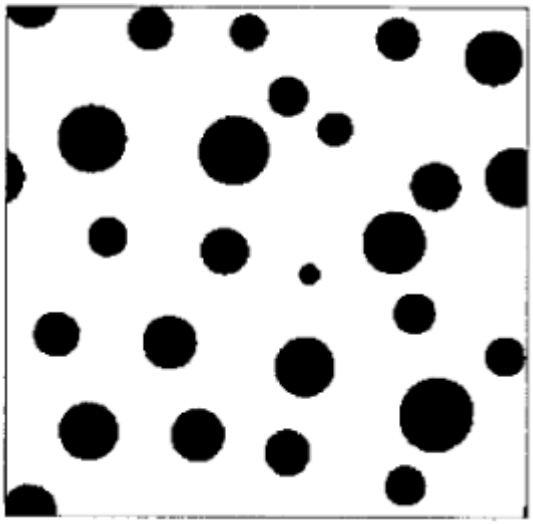

Figure 3.1: Binary fluid phase diagram for 21% of the black part initially present.