coarsening

March 28, 2023

1 Coarsening process in Cahn-Hilliard equation

1.1 The continuum hypothesis

Let us consider a box of fluid with size $L_x \times L_y \times L_z$ and periodic boundary condition on each side of the box. The volume of the system is $V = L_x L_y L_z$.

Continuum hypothesis. We assume that we can divide the system into many many small volume elements dV's, such that each volume element dV contains a large number of molecules $\sim 10^{23}$, but dV is still small compared to the overall system volume $V = L_x L_y L_z$. We can then define the hydrodynamic variable $\rho(\mathbf{r}, t)$, which is the mass density of the fluid. Physically, $\rho(\mathbf{r}, t)$ is defined such that:

$$\rho(\mathbf{r}, t)\, dV = \text{mass of the fluid inside the volume element } dV\text{, located at } \mathbf{r} \text{ and at time } t. \tag{1}$$

There are other hydrodynamic variables of interest such as pressure $p(\mathbf{r}, t)$ and fluid velocity $\mathbf{u}(\mathbf{r}, t)$, but we will not worry about these in this chapter.

We consider the following problems, involving phase separation: 1. Two-phase, one-component systems, *e.g.*, liquid water and water vapour phase separation 2. One-phase, two-component systems, *e.g.*, liquid water and liquid oil phase separation.

For both cases, the phase-separation dynamics can be described by a single scalar order parameter $\phi(\mathbf{r}, t)$ (which is sometimes also called the rescaled density).

For Case 1, $\phi(\mathbf{r}, t)$ is defined to be the rescaled density, relative to the critical density ρ_c:

$$\phi(\mathbf{r}, t) = \frac{\rho(\mathbf{r}, t) - \rho_c}{\rho_c}. \tag{2}$$

In the above equation, $\rho(\mathbf{r}, t)$ is the mass density of the fluid (which can be gas/liquid/both) at position $\mathbf{r}$ and at time t, and ρ_c is the density of the fluid at the critical point (we will come back to critical point later). Thus from the definition above, $\phi(\mathbf{r}, t) > 0$ corresponds to the liquid phase and $\phi(\mathbf{r}, t) < 0$ corresponds to the gas phase.

For Case 2, $\phi(\mathbf{r}, t)$ is defined to be the relative densities between the two molecules, say A and B:

$$\phi(\mathbf{r}, t) = \frac{2\rho(\mathbf{r}, t) - \rho_A - \rho_B}{\rho_A - \rho_B}. \tag{3}$$

In the above equation ρ_A is the density of pure A molecules and ρ_B is the density of the pure B molecules. Thus from the definition above, $\phi(\mathbf{r}, t) \simeq 1$ indicates that the fluid contains pure A molecules and $\phi(\mathbf{r}, t) \simeq -1$ indicates that the fluid contains pure B molecules.

By conservation of mass and assuming that there is no chemical reaction $A \leftrightarrow B$, it follows that the integral:

$$\phi_0 = \frac{1}{V} \underbrace{\int_V dV\, \phi(\mathbf{r}, t)}_{\text{total amount of fluid}} = \text{constant}. \tag{4}$$

We call the constant ϕ_0 to be the global density, which is fixed by the initial condition.

Local equilibrium hypothesis. We assume each fluid element dV to be in a state of thermodynamic equilibrium *locally* with a given temperature, entropy, internal energy, pressure, chemical potential, *etc.* Note that this does not mean that the whole system is in a state of *global* thermodynamic equilibrium because a fluid element dV at $\mathbf{r}$ might have a different pressure or chemical potential compared to the fluid element dV at $\mathbf{r}'$. However over time, if we leave the system to sit long enough, energy and mass can be exchanged between different fluid elements dV's until a global thermodynamic equilibrium is reached.

Since each fluid element dV is in thermodynamic equilibrium locally, we can define the free energy of this fluid element to be:

$$g(\phi, \nabla\phi)\, dV = \text{free energy of the fluid element } dV\text{, located at } \mathbf{r} \text{ and at time } t. \tag{5}$$

We call $g(\phi, \nabla\phi)$ to be the free energy density. In general g depends on the order parameter ϕ and its gradient $\nabla\phi$. Landau and Ginzburg's idea is to Taylor expand g around the critical point. At the critical point $\phi = 0$ and thus we can expand g for small ϕ:

$$g(\phi, \nabla\phi) = \underbrace{\frac{\alpha}{2}\phi^2 + \frac{\beta}{4}\phi^4}_{\text{local}} + \underbrace{\frac{\kappa}{2}|\nabla\phi|^2}_{\text{semi-local}}. \tag{6}$$

Note that, we have ignored the cubic term ϕ^3, because we can redefine ϕ to eliminate this cubic term, making it unnecessary. We have also ignored the linear term ϕ because this does not contribute the dynamics, as we shall see below. Finally, $\sim |\nabla\phi|^2$ is the smallest scalar term we can form using ∇ and ϕ. Note that the first two terms above are purely local as they only depend on ϕ. The last term in the equation above is semi-local, because it depends on the gradient $\nabla\phi$. This means to calculate the last term in the equation above, we need some information about the neighbouring fluid elements dV's.

Coarse graining. When we did the Landau expansion, *i.e.* Taylor expansion around $\phi = 0$ above, we get various phenomenological parameters such as α and β, which are unknown. In this section, we will investigate how to derive these parameters from microscopic models through *coarse-graining.* Let us consider some fluid, as shown in the figure on the left below. Yellow indicates the liquid phase ($\phi \simeq 1$) while dark blue indicates the gas phase ($\phi \simeq -1$).

As before, we can divide the system into many many volume elements dV's. Let us just consider one such volume element dV, which is indicated by dark square in the figure below. Inside this volume element, we have a large number of molecules, which are indicated by the blue discs in the figure on the right below. Let us denote N_p to be the number of particles/fluid molecules, *e.g.* H_2O, inside the volume element dV. (In the figure we have $N_p = 36$ particles for illustration purposes, but in reality $N_p \sim 10^{23}$.) Let us suppose that we can further divide the volume element dV into N lattice cells, as shown in the figure on the right below. (In the figure, $N = 8 \times 8 = 64$, but in reality, $N \sim 10^{23}$.) Each cell has a volume a^3, which is even smaller than dV, and a is the typical size of the molecule. Now we assume that each fluid molecule has to fit inside one of the lattice cells and cannot be on the fence. Furthermore, each lattice cell cannot contain more than one fluid molecule due to hard-core repulsion.

Now let us compute the free energy of this volume element dV. The free energy is given by:

$$F = U - TS, \quad \text{where } U \text{ is the potential energy and } S \text{ is the entropy of this lattice system.} \tag{7}$$

First let us calculate U. Let us denote the interaction energy between two neighbouring molecules to be $-\varepsilon$, where $\varepsilon > 0$. The interaction energy is negative because the molecules tend to attract each other. Now let us denote z to be the number of nearest neighbour cells. In two-dimension, $z = 4$, as shown by the red lines in the figure on the right below. Thus each fluid molecule has, on average, $z\frac{N_p}{N}$ other neighbouring molecules. The interaction energy between the fluid molecules is then:

$$U = -\frac{\varepsilon z}{2} N_p \frac{N_p}{N} \quad \Rightarrow \quad \frac{U}{N} = -\frac{\varepsilon z}{2}\frac{N_p}{N}\frac{N_p}{N}. \tag{8}$$

Note that we have added a factor of $1/2$ to avoid double counting the bonds between two neighbouring molecules. Now to calculate the entropy, we use the Boltzmann formula:

$$S = k_B \ln \Omega, \quad \text{where } \Omega \text{ is the number of microstates.} \tag{9}$$

In our case Ω is the number of arranging N_p indistinguishable particles into N cells:

$$\Omega = C^N_{N_p} = \frac{N!}{(N - N_p)! N_p!}. \tag{10}$$

Thus the entropy is:

$$S = k_B \left[\ln N! - \ln(N - N_p)! - \ln N_p!\right]. \tag{11}$$

Next we can use the Stirling's approximation $\ln N! \simeq N \ln N - N$ to get:

$$\frac{S}{N} = -k_B \left[\left(1 - \frac{N_p}{N}\right)\ln\left(1 - \frac{N_p}{N}\right) + \frac{N_p}{N}\ln\left(\frac{N_p}{N}\right)\right] \tag{12}$$

Thus the free energy (per unit cell) of this volume element is:

$$\begin{aligned}\frac{F}{N} &= \frac{U}{N} - T\frac{S}{N} && (13)\\ &= -\frac{\varepsilon z}{2}\frac{N_p}{N}\frac{N_p}{N} + k_B T\left[\left(1 - \frac{N_p}{N}\right)\ln\left(1 - \frac{N_p}{N}\right) + \frac{N_p}{N}\ln\left(\frac{N_p}{N}\right)\right]. && (14)\end{aligned}$$

Note that the free energy per unit cell above is an intensive quantity, as expected. From the picture below, obviously if $N_p = N$ we get a pure liquid phase and if $N_p = 0$ we get a pure gas phase. Thus at critical point, we expect $N_p = N/2$. This motivates us to expand F/N around the critical point by writing:

$$\frac{N_p}{N} = \frac{1}{2} + \phi, \tag{15}$$

where ϕ is small. The free energy per unit cell is then:

$$\frac{F}{N} = -\frac{\varepsilon z}{2}\left(\frac{1}{2} + \phi\right)\left(\frac{1}{2} + \phi\right) + k_B T\left[\left(\frac{1}{2} - \phi\right)\ln\left(\frac{1}{2} - \phi\right) + \left(\frac{1}{2} + \phi\right)\ln\left(\frac{1}{2} + \phi\right)\right] \tag{16}$$

Next we expand the logarithm as power series up to order ϕ^4:

$$\ln\left(\frac{1}{2} \pm \phi\right) = -\ln 2 \pm 2\phi - 2\phi^2 \pm \frac{8}{3}\phi^3 - 4\phi^4 \ldots. \tag{17}$$

We can then expand F/N up to order ϕ^4:

$$\frac{F}{N} = -\frac{z\varepsilon}{2}\phi + \left(2k_BT - \frac{z\varepsilon}{2}\right)\phi^2 + \frac{4k_BT}{3}\phi^4 + \mathcal{O}(\phi^6). \tag{18}$$

Note that we have ignored the constant terms in the equation above. Furthermore, we can also ignore the linear term $\propto \phi$ as this does not affect the $\dot{\phi}$-dynamics as we shall see below. To find the free energy density, we can then divide F/N by the volume of the lattice cell a^3 to get:

$$g(\phi) = \frac{F}{Na^2} = \underbrace{\left(\frac{2k_BT}{a^3} - \frac{z\varepsilon}{2a^3}\right)}_{\alpha}\phi^2 + \underbrace{\frac{4k_BT}{3a^3}}_{\beta}\phi^4 + \mathcal{O}(\phi^6). \tag{19}$$

If we compare with the Landau-Ginzburg free energy density $g(\phi, \nabla\phi)$, we identify the ϕ^2 and ϕ^4 coefficients to be:

$$\alpha = \frac{2k_BT}{a^3} - \frac{z\varepsilon}{2a^3} \quad \text{and} \quad \beta = \frac{4k_BT}{3a^3}, \tag{20}$$

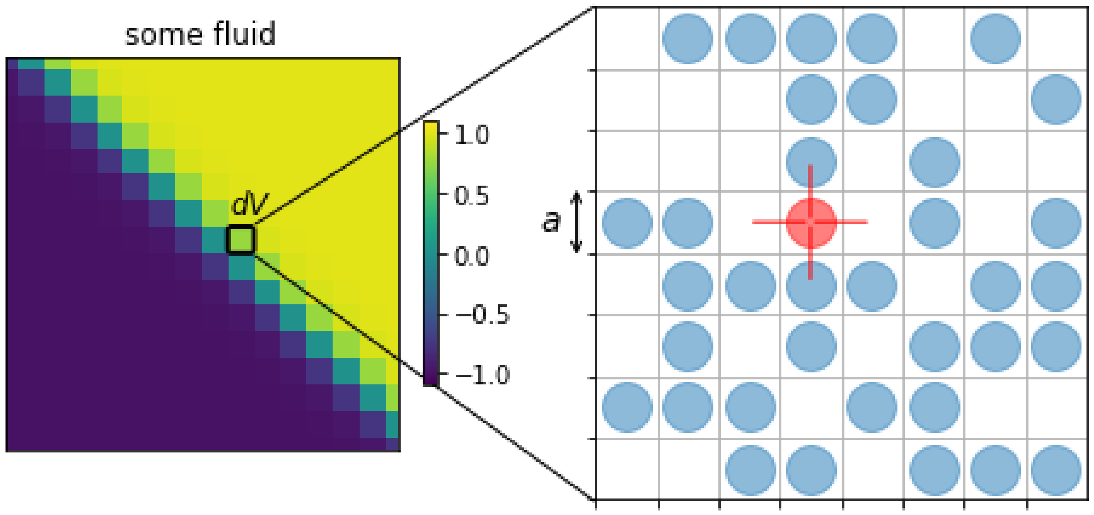
respectively. Note that β is always positive, whereas α becomes negative when:

$$T < \frac{z\varepsilon}{4k_B} = T_c. \tag{21}$$

We call the right hand side the critical temperature T_c, below which, phase separation can occur (for a given range of ϕ_0). If $T > T_c$, the coefficient of ϕ^2 in the free energy is always positive, and the system remains homogenously mixed, and there is no phase separation into liquid or gas. Finally, we should also note that we did not the get the gradient term $\sim |\nabla\phi|^2$ through coarse-graining. This term has to be added phenomenologically with some unknown coefficient $\kappa > 0$.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

# array of cartesian coordinates (needed for plotting)
dx, dy = 1.0, 1.0
Nx, Ny = 16, 16
x = np.arange(0, Nx)*dx
y = np.arange(0, Ny)*dx
y, x = np.meshgrid(y, x)

# some random phi field
phi = np.tanh(x - 0.5*Nx*dx*np.ones(len(x)) + y - 0.5*Ny*dy*np.ones(len(x)))

# particles
Np = 35
np.random.seed(0)
X = np.zeros(Np)
Y = np.zeros(Np)
for p in range(0, Np, 1):
    repeated = True
    while repeated:
        X[p] = np.random.randint(0, 8)
        Y[p] = np.random.randint(0, 8)
        repeated = False
        for q in range(0, p, 1):
            if X[p] == X[q] and Y[p] == Y[q]:
                repeated = True

X = X + 0.5
Y = Y + 0.5
Xtag, Ytag = 3.5, 4.5

# create instance of pyplot
fig, (ax1, ax2) = plt.subplots(1,2,figsize=(8,4))

ax1.set_xlim(0, (Nx-1)*dx)
ax1.set_ylim(0, (Ny-1)*dy)
ax1.set_xticks([])
ax1.set_yticks([])
ax1.set_aspect('equal')
ax1.set_title('some fluid')
colormap = ax1.pcolormesh(x, y, phi, shading='auto', vmin=-1.1, vmax=1.1)

ax1.annotate('', xy=(9.6,7.5), xytext=(8.4,7.5),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=2))
ax1.annotate('', xy=(9.6,8.5), xytext=(8.4,8.5),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=2))
ax1.annotate('', xy=(8.5,8.6), xytext=(8.5,7.4),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=2))
ax1.annotate('', xy=(9.5,8.6), xytext=(9.5,7.4),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=2))

fig.colorbar(colormap, ax=ax1, shrink=0.5)

ax1.annotate('', xy=(22.6,-2.04), xytext=(9.4,7.5),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=1),␣
 ↪annotation_clip=False)
ax1.annotate('', xy=(22.6,16.98), xytext=(9.4,8.5),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=1),␣
 ↪annotation_clip=False)
ax1.annotate('$dV$', fontsize=12, xy=(8.5,9))

ax2.set_xlim(0, 8)
ax2.set_ylim(0, 8)
ax2.set_xticks(np.arange(0,8,1))
ax2.set_yticks(np.arange(0,8,1))
ax2.set_xticklabels([])
ax2.set_yticklabels([])
ax2.scatter(X, Y, alpha=0.5, s=400)
```

```
ax2.scatter(Xtag, Ytag, alpha=0.5, c='red', s=400)
ax2.grid()
ax2.set_aspect('equal')

ax2.annotate('', xy=(4.5,4.5), xytext=(3.5,4.5),␣
 ↪arrowprops=dict(edgecolor='red', linestyle='-', arrowstyle='-', linewidth=1))
ax2.annotate('', xy=(2.5,4.5), xytext=(3.5,4.5),␣
 ↪arrowprops=dict(edgecolor='red', linestyle='-', arrowstyle='-', linewidth=1))
ax2.annotate('', xy=(3.5,5.5), xytext=(3.5,4.5),␣
 ↪arrowprops=dict(edgecolor='red', linestyle='-', arrowstyle='-', linewidth=1))
ax2.annotate('', xy=(3.5,3.5), xytext=(3.5,4.5),␣
 ↪arrowprops=dict(edgecolor='red', linestyle='-', arrowstyle='-', linewidth=1))
ax2.annotate('', xy=(-0.3,5.1), xytext=(-0.3,3.9),␣
 ↪arrowprops=dict(edgecolor='black', linestyle='-', arrowstyle='<->',␣
 ↪linewidth=1), annotation_clip=False)
ax2.annotate('$a$', fontsize=14, xy=(-0.9,4.4), annotation_clip=False)

plt.show()
```



1.2 Coarse-grained free energy

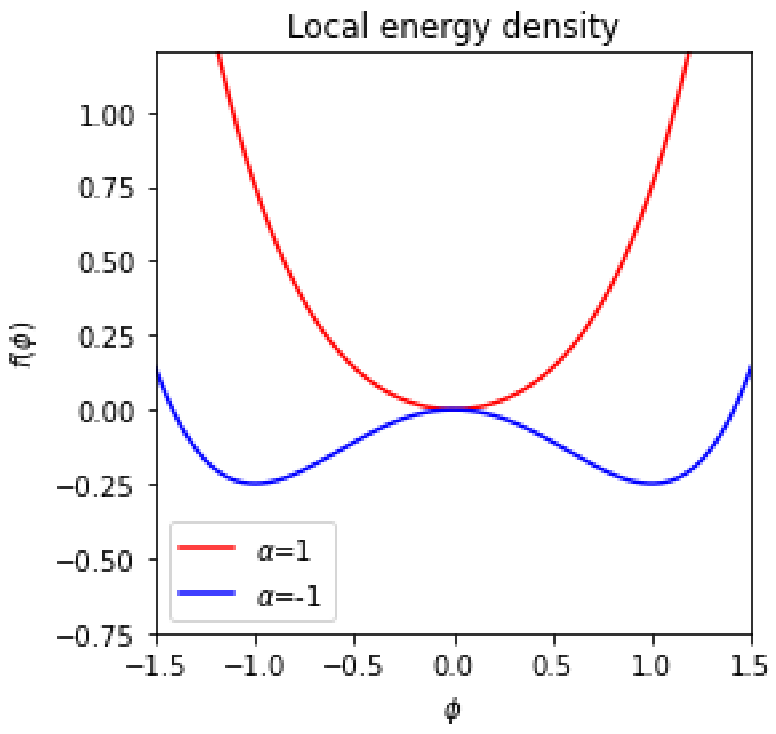
The total coarse-grained free energy of the system can then be written as:

$$\mathcal{F}[\phi] = \int_V dV \left[\underbrace{\frac{\alpha}{2}\phi^2 + \frac{\beta}{4}\phi^4}_{f(\phi)} + \frac{\kappa}{2}|\nabla\phi|^2 \right], \tag{22}$$

where $f(\phi)$ is the local free energy density term and $\frac{\kappa}{2}|\nabla\phi|^2$ is a semi-local free energy density term. β and κ are positive constants. α can be positive or negative. If $\alpha < 0$, then the system will favour phase separation into the liquid $\phi \simeq \sqrt{\frac{-\alpha}{\beta}}$ and vapour phase $\phi \simeq -\sqrt{\frac{-\alpha}{\beta}}$. On the other hand if

$\alpha > 0$, the system will remain in a homogenous phase with $\phi = \phi_0$ everywhere. The reason for this has something to do with the shape of the local (or bulk) free energy density $f(\phi)$, as you can see below. For $\alpha < 0$, the local (or bulk) free energy density $f(\phi)$ has two minima at $\phi = \pm\sqrt{\frac{-\alpha}{\beta}}$.

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt

     alpha, beta, kappa = 1.0, 1.0, 1.0

     phi = np.arange(-2.0, 2.0, 0.001)
     f_phi = 0.5*alpha*phi**2 + 0.25*beta*phi**4

     fig, ax = plt.subplots(figsize=(4,4))

     # set title
     ax.set_title('Local energy density')

     # set x label and y label
     ax.set_ylabel('$f(\phi)$')
     ax.set_xlabel('$\phi$')

     # to change x range and y range
     ax.set_xlim([-1.5, 1.5])
     ax.set_ylim([-0.75, 1.2])

     # add ticks
     ax.tick_params(axis='both')

     # plot f(phi) for alpha = 1
     ax.plot(phi, f_phi, c='red', label='$\\alpha$=1')

     # plot f(phi) for alpha = -1
     alpha = -1.0
     f_phi = 0.5*alpha*phi**2 + 0.25*beta*phi**4
     ax.plot(phi, f_phi, c='blue', label='$\\alpha$=-1')

     # legend location legend font
     ax.legend(loc='lower left')

     plt.show()
```



The *global* equilibrium state is given by the minimum of the total free energy $\mathcal{F}[\phi]$, *i.e*,

$$\frac{\delta \mathcal{F}}{\delta \phi} = 0, \tag{23}$$

subject to the constraint

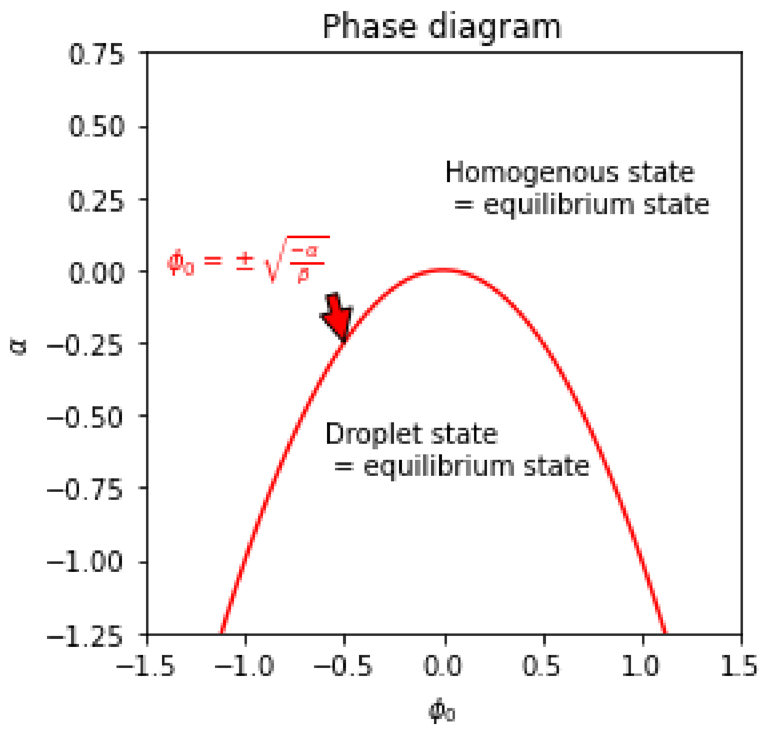
$$\phi_0 = \frac{1}{V}\int_V dV\, \phi(\mathbf{r},t) = \text{constant.} \tag{24}$$

There are two equilibrium states: 1. Homogenous state, where the density is constant everywhere $\phi(\mathbf{r},t) = \phi_0 = \text{constant}$. 2. Droplet state, where the system is separated into two domains with density $\phi \simeq +\sqrt{\frac{-\alpha}{\beta}}$ and $\phi \simeq -\sqrt{\frac{-\alpha}{\beta}}$ in each domain.

The control parameters in our system are α and ϕ_0. Depending on the values of α and ϕ_0, the equilibrium state of the system can either be the homogenous state or the droplet state.

[ ]:
```
beta = 1.0
kappa = 1.0

phi = np.arange(-2.0, 2.0, 0.001)

fig, ax = plt.subplots(figsize=(4,4))

# set title
ax.set_title('Phase diagram')
```

```
# set x label and y label
ax.set_ylabel('$\\alpha$')
ax.set_xlabel('$\phi_0$')

# to change x range and y range
ax.set_xlim([-1.5, 1.5])
ax.set_ylim([-1.25, 0.75])

ax.tick_params(axis='both')

alpha = -beta*phi**2
ax.plot(phi, alpha, c='red')

# add annotation
ax.annotate('Homogenous state \n = equilibrium state', xy=(0.0,0.2))
ax.annotate('Droplet state \n = equilibrium state', xy=(-0.6,-0.7))
ax.annotate('$\phi_0=\pm\sqrt{\\frac{-\\alpha}{\\beta}}$',
            c='red',
            xy=(-0.5,-0.25),  # location of the arrow tip
            xytext=(-1.4,0.0),  # location of the text
            arrowprops=dict(facecolor='red'))  # the arrow points from the text␣
 ↪to the arrowtip

plt.show()
```



1.3 Functional derivative

Let us consider some functional $\mathcal{F}[\phi(\mathbf{r})]$, which in general, can be written as:

$$\mathcal{F}[\phi] = \int_V g(\phi, \partial_\alpha \phi)\, dV. \tag{25}$$

In the above equation, $g(\phi, \partial_\alpha \phi)$ is some function of ϕ and its derivative $\partial_\alpha \phi$. Here the index $\alpha = x, y, z$ represents the Cartesian coordinates. We also use the following notation for partial derivatives $\partial_x = \frac{\partial}{\partial x}$, $\partial_y = \frac{\partial}{\partial y}$, and $\partial_z = \frac{\partial}{\partial z}$. In the case of our energy, g is given by:

$$g(\phi, \partial_\alpha \phi) = \frac{\alpha}{2}\phi^2 + \frac{\beta}{4}\phi^4 + \frac{\kappa}{2}(\partial_\alpha \phi)(\partial_\alpha \phi). \tag{26}$$

Here, we have used the repeated index notation, *i.e* summation over that index is implied so the last term in the above equation is implied to be:

$$(\partial_\alpha \phi)(\partial_\alpha \phi) = (\partial_x \phi)^2 + (\partial_y \phi)^2 + (\partial_z \phi)^2 = |\nabla \phi|^2 \tag{27}$$

Now let us consider some density field $\phi(\mathbf{r})$. Let's suppose that we add some perturbation to the density field such that $\phi(\mathbf{r}) \to \phi(\mathbf{r}) + \delta\phi(\mathbf{r})$, where $\delta\phi$ is a small perturbation. And we also suppose that $\delta\phi$ vanishes at the boundary (*i.e.* the surface of the box). We want to calculate the change in the total energy due to this perturbation:

$$\begin{aligned}
\delta\mathcal{F} &= \mathcal{F}[\phi + \delta\phi] - \mathcal{F}[\phi] && (28)\\
&= \int_V g(\phi + \delta\phi, \partial_\alpha \phi + \partial_\alpha \delta\phi)\, dV - \int_V g(\phi, \partial_\alpha \phi)\, dV && (29)
\end{aligned}$$

Next we take Taylor series for the first term to get:

$$\begin{aligned}
\delta\mathcal{F} &= \int_V g(\phi + \delta\phi, \partial_\alpha \phi + \partial_\alpha \delta\phi)\, dV - \int_V g(\phi, \partial_\alpha \phi)\, dV && (30)\\
&= \int_V \left\{ g(\phi, \partial_\alpha \phi) + \frac{\partial g}{\partial \phi}\delta\phi + \frac{\partial g}{\partial(\partial_\alpha \phi)}\partial_\alpha(\delta\phi) \right\} dV - \int_V g(\phi, \partial_\alpha \phi)\, dV && (31)\\
&= \int_V \left\{ \frac{\partial g}{\partial \phi}\delta\phi + \frac{\partial g}{\partial(\partial_\alpha \phi)}\partial_\alpha(\delta\phi) \right\} dV && (32)
\end{aligned}$$

Using product rule (or integration by parts), we can write:

$$\frac{\partial g}{\partial(\partial_\alpha \phi)}\partial_\alpha(\delta\phi) = \partial_\alpha \left(\frac{\partial g}{\partial(\partial_\alpha \phi)}\delta\phi \right) - \partial_\alpha \left(\frac{\partial g}{\partial(\partial_\alpha \phi)} \right) \delta\phi \tag{33}$$

Thus, $\delta\mathcal{F}$ becomes:

$$\delta\mathcal{F} = \int_V \left\{ \frac{\partial g}{\partial \phi}\delta\phi \right\} dV + \int_V \partial_\alpha \left(\frac{\partial g}{\partial(\partial_\alpha \phi)}\delta\phi \right) dV - \int_V \left\{ \partial_\alpha \left(\frac{\partial g}{\partial(\partial_\alpha \phi)} \right) \delta\phi \right\} dV \tag{34}$$

For the second term, we can use divergence theorem:

$$\int_V \partial_\alpha J_\alpha\, dV = \oint_S J_\alpha\, dS_\alpha, \tag{35}$$

where S is the surface bounding the volume V. So we end up with:

$$\delta\mathcal{F} = \int_V \left\{ \frac{\partial g}{\partial \phi}\delta\phi \right\} dV + \underbrace{\oint_S \left(\frac{\partial g}{\partial(\partial_\alpha \phi)}\delta\phi \right) dS_\alpha}_{=0} - \int_V \left\{ \partial_\alpha \left(\frac{\partial g}{\partial(\partial_\alpha \phi)} \right) \delta\phi \right\} dV \tag{36}$$

$$= \int_V \Bigg\{ \underbrace{\frac{\partial g}{\partial \phi} - \partial_\alpha \left(\frac{\partial g}{\partial(\partial_\alpha \phi)} \right)}_{\delta\mathcal{F}/\delta\phi(\mathbf{r})} \Bigg\} \delta\phi \, dV \tag{37}$$

The surface integral above vanishes since $\delta\phi$ vanishes at the boundary. The functional derivative of $\mathcal{F}$ with respect to $\phi(\mathbf{r})$ is defined to be the terms inside the curly brackets:

$$\frac{\delta\mathcal{F}}{\delta\phi(\mathbf{r})} = \frac{\partial g}{\partial \phi} - \partial_\alpha \left(\frac{\partial g}{\partial(\partial_\alpha \phi)} \right). \tag{38}$$

The functional derivative is like a generalization of the regular derivative. When the total energy is minimum (or maximum), the functional derivative of that energy must be equal to zero:

$$\frac{\delta\mathcal{F}}{\delta\phi} = 0. \tag{39}$$

1.4 Dynamics

The chemical potential $\mu(\mathbf{r}, t)$ is defined to be the energy cost of adding a particle locally at $\mathbf{r}$. Mathematically, it can be written as:

$$\mu = \frac{\delta\mathcal{F}}{\delta\phi}. \tag{40}$$

Using the formula for functional derivative above, we can calculate:

$$\mu = \frac{\partial g}{\partial \phi} - \partial_\alpha \left(\frac{\partial g}{\partial(\partial_\alpha \phi)} \right) = \alpha\phi + \beta\phi^3 - \kappa \underbrace{\partial_\alpha \partial_\alpha \phi}_{\nabla^2 \phi} \tag{41}$$

Note that:

$$\partial_\alpha \partial_\alpha = \partial_x^2 + \partial_y^2 + \partial_z^2 = \nabla^2 \tag{42}$$

Since the total amount of fluid in the system is conserved, the dynamics of $\phi(\mathbf{r}, t)$ must follow the continuity equation:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot \mathbf{J} = 0, \tag{43}$$

where $\mathbf{J}$ is the current. To see this, we can integrate the continuity equation over the whole box to get:

$$\frac{d}{dt} \int_V \phi \, dV + \int_V \nabla \cdot \mathbf{J} \, dV = 0 \tag{44}$$

$$\frac{d}{dt} \int_V \phi \, dV + \underbrace{\oint_S \mathbf{J} \cdot \hat{\mathbf{n}} \, dS}_{=0} = 0, \tag{45}$$

where we have used the divergence theorem on the second term. S is the surface covering the boundary of the box. Now since we have periodic boundary condition on each side of the box, the second term in the equation above vanishes so we end up with:

$$\frac{d}{dt}\int_V \phi\, dV = 0 \quad \Rightarrow \quad \int_V \phi\, dV = \text{constant}, \tag{46}$$

which implies the total amount of fluid is conserved.

Finally, the current is given by:

$$\mathbf{J} = -M\nabla\mu, \tag{47}$$

i.e., particles diffuse from regions with high chemical potential to regions with low chemical potential. $M > 0$ is called the mobility. Thus the dynamics can also be written as:

$$\frac{\partial\phi}{\partial t} = M\nabla^2\mu. \tag{48}$$

Now let us consider the rate of change of the total free energy $\mathcal{F}$:

$$\frac{d\mathcal{F}}{dt} = \frac{d}{dt}\int_V g(\phi, \partial_\alpha\phi)\, dV = \int_V \frac{\partial}{\partial t} g(\phi, \partial_\alpha\phi)\, dV. \tag{49}$$

Next using chain rule, we get:

$$\begin{aligned}\frac{d\mathcal{F}}{dt} &= \int_V \left\{ \frac{\partial g}{\partial\phi}\frac{\partial\phi}{\partial t} + \frac{\partial g}{\partial(\partial_\alpha\phi)}\partial_\alpha\left(\frac{\partial\phi}{\partial t}\right)\right\} dV &(50)\\ &= \int_V \underbrace{\left\{ \frac{\partial g}{\partial\phi} - \partial_\alpha\left(\frac{\partial g}{\partial(\partial_\alpha\phi)}\right)\right\}}_{=\mu} \underbrace{\frac{\partial\phi}{\partial t}}_{=M\nabla^2\mu} dV, &(51)\end{aligned}$$

where we have used integration by parts again and the surface integral term vanishes (similar to previous section). Finally, we substitute the dynamics to get:

$$\frac{d\mathcal{F}}{dt} = M\int_V \mu\nabla^2\mu\, dV = -M\int_V \nabla\mu\cdot\nabla\mu\, dV \leq 0, \tag{52}$$

where we have used the integration by parts again. Thus we have shown that $\frac{d\mathcal{F}}{dt}$ is always negative as long as the dynamics follows $\dot{\phi} = \nabla^2\mu$ and $M \geq 0$. In other words the dynamics $\dot{\phi} = \nabla^2\mu$ will guarantee that the total energy always decreases with time until the minimum (equilibrium state) is reached.

1.5 Linear stability of the homogenous state

The dynamics for $\phi(\mathbf{r}, t)$ can be now written as:

$$\frac{\partial\phi}{\partial t} = M\nabla^2\left[\alpha\phi + \beta\phi^3 - \kappa\nabla^2\phi\right] \tag{53}$$

Suppose we write the density field as follows:

$$\phi(\mathbf{r}, t) = \phi_0 + \delta\phi(\mathbf{r}, t), \tag{54}$$

where $\delta\phi(\mathbf{r}, t)$ is a small fluctuation around the homogenous state $\phi = \phi_0$. We are interested in whether the fluctuations will grow or decay as the time t increases. To do this, we substitute $\phi = \phi_0 + \delta\phi$ into the dynamics to get:

$$\frac{\partial \delta\phi}{\partial t} = M\nabla^2 \left[(\alpha + 3\beta\phi_0^2)\delta\phi - \kappa\nabla^2\delta\phi\right]. \tag{55}$$

Note that we have neglected small terms of order $\sim \delta\phi^2$ and higher. Now we write $\delta\phi(\mathbf{r}, t)$ as a Fourier series:

$$\delta\phi(\mathbf{r}, t) = \sum_{\mathbf{q}} \delta\tilde{\phi}_{\mathbf{q}}(t)e^{i\mathbf{q}\cdot\mathbf{r}}. \tag{56}$$

$\mathbf{q} = (q_x, q_y, q_z)^T$ is the wavevector and each component of $\mathbf{q}$ is discrete:

$$q_x = 0, \pm\frac{2\pi}{L_x}, \pm\frac{4\pi}{L_x}, \ldots \tag{57}$$

$$q_y = 0, \pm\frac{2\pi}{L_y}, \pm\frac{4\pi}{L_y}, \ldots \tag{58}$$

$$q_z = 0, \pm\frac{2\pi}{L_z}, \pm\frac{4\pi}{L_z}, \ldots \tag{59}$$

$$\tag{60}$$

The summation in the Fourier series indicates summing over all discrete values of $\mathbf{q}$'s. The Fourier series can be thought as decomposing a function $\phi(x)$, which is defined on $x \in [0, L]$, into standing waves $1, e^{\pm i\frac{2\pi}{L}x}, e^{\pm i\frac{4\pi}{L}x}, \ldots$, with $\tilde{\phi}_q$ as the amplitude for each standing wave. Substituting the Fourier series into the dynamics, a set of first order ODE:

$$\frac{d\delta\tilde{\phi}_{\mathbf{q}}}{dt} = -M\left[(\alpha + 3\beta\phi_0^2)q^2 + \kappa q^4\right]\delta\tilde{\phi}_{\mathbf{q}}, \quad \text{for each } \mathbf{q}. \tag{61}$$

The solution is:

$$\delta\tilde{\phi}_{\mathbf{q}}(t) = A_{\mathbf{q}}e^{r(q)t}, \tag{62}$$

where

$$r(q) = M[(-\alpha - 3\beta\phi_0^2)q^2 - \kappa q^4] \tag{63}$$

is the growth rate constant (here $q = |\mathbf{q}|$). If $r(q) > 0$ then the fluctuation $\delta\tilde{\phi}_{\mathbf{q}}$ will grow exponentially. If $r(q) < 0$ then the fluctuation $\delta\tilde{\phi}_{\mathbf{q}}$ will decay exponentially to zero. Now let's consider case by case.

Case I: $\alpha > 0$ In this case, the coefficients of q^2 and q^4 in $r(q)$ are always negative.

$$r(q) = M[\underbrace{(-\alpha - 3\beta\phi_0^2)}_{<0}q^2 \underbrace{-\kappa}_{<0} q^4]. \tag{64}$$

Therefore the growth rate is always negative So the fluctuations decay to zero exponentially for all wavevector $\mathbf{q}$.

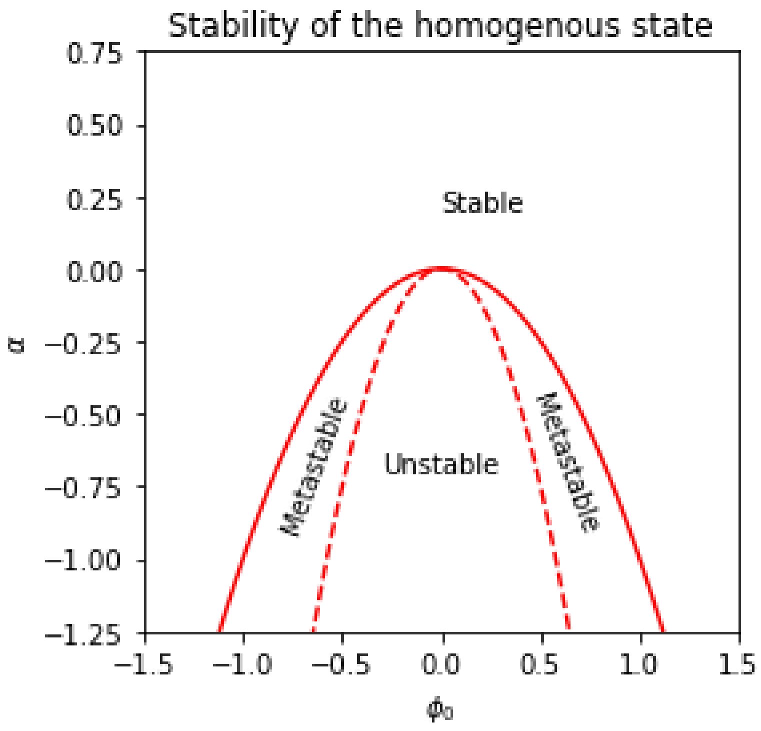
Case IIa: $\alpha > 0$ **and** $-\sqrt{\frac{-\alpha}{3\beta}} < \phi_0 < \sqrt{\frac{-\alpha}{3\beta}}$ In this case, the coefficient of q^2 is positive whereas the coefficient of q^4 is negative.

$$r(q) = M[\underbrace{(-\alpha - 3\beta\phi_0^2)}_{>0} q^2 \underbrace{-\kappa}_{<0} q^4]. \tag{65}$$

We can plot the growth rate $r(q)$ as a function of $q = |\mathbf{q}|$. The growth rate is positive for some range of q. This indicates the fluctuations will grow exponentially with time for some values of $\mathbf{q}$. We can also define a characteristic wavevector q^* such that the growth rate is maximum. This correponds to the initial growth lengthscale $\lambda^* = \frac{2\pi}{q^*}$. This instability is illustrated in the numerical simulation at the bottom of this notebook.

Case IIb: $\alpha > 0$ **and** $|\phi_0| > \sqrt{\frac{-\alpha}{3\beta}}$ In this case, both coefficients of q^2 and q^4 are negative in the growth rate. Thus all fluctuations decay to zero for all $\mathbf{q}$, similar to case I. However from the previous section, we also learnt that if $\sqrt{\frac{-\alpha}{3\beta}} < \phi_0 < \sqrt{\frac{-\alpha}{\beta}}$ or $-\sqrt{\frac{-\alpha}{\beta}} < \phi_0 < \sqrt{\frac{-\alpha}{3\beta}}$, the actual equilibrium state (or lowest energy state) of the system is actually a droplet state. However, if we initialize the system from a homogenous state $\phi(\mathbf{r}, t = 0) = \phi_0 +$ small noise, the system will actually remain in the homogenous state, even though the homogenous state is not the lowest energy state. We call the homogenous state to be a metastable state in this regime. In order for the system to jump into the actual equilibrium state, we need to add a rather large perturbation in the initial condition. For instance we may introduce a nucleus (*i.e.* a tiny droplet) into the system at the initial time. If the size of the nucleus is larger than the critical size, the nucleus will then grow until the equilibrium state of a macroscopic droplet is reached. This process is called nucleation process, which is quite distinct from Case IIa.

[ ]:
```
beta = 1.0
kappa = 1.0

phi = np.arange(-2.0, 2.0, 0.001)

fig, ax = plt.subplots(figsize=(4,4))

# set title
ax.set_title('Stability of the homogenous state')

# set x label and y label
ax.set_ylabel('$\\alpha$')
ax.set_xlabel('$\phi_0$')

# set x range and y range
ax.set_xlim([-1.5, 1.5])
ax.set_ylim([-1.25, 0.75])

ax.tick_params(axis='both')

alpha1 = -beta*phi**2
alpha2 = -3.0*beta*phi**2
ax.plot(phi, alpha1, c='red')
ax.plot(phi, alpha2, '--', c='red')

# add annotation
ax.annotate('Stable', xy=(0.0,0.2))
ax.annotate('Unstable', xy=(-0.3,-0.7))
ax.annotate('Metastable', xy=(-0.82,-0.9), rotation=70)
ax.annotate('Metastable', xy=(0.42,-0.9), rotation=-70)

plt.show()
```



1.6 Numerical simulation

Let us consider a two-dimensional system. In computer simulation the space **r** is discretized into lattice with step size Δx along the x-axis and Δy along the y-axis. (Ideally Δx and Δy have to be small.) The coordinates x and y then become:

$$x \to i\Delta x, \quad \text{where } i = 0, 1, 2, \ldots, N_x - 1 \tag{66}$$

$$y \to j\Delta y, \quad \text{where } j = 0, 1, 2, \ldots, N_y - 1. \tag{67}$$

$N_x \in \mathbb{N}$ and $N_y \in \mathbb{N}$ are the number of lattice points along x and along y respectively. The system size is now $L_x \times L_y$, where $L_x = N_x\Delta x$ and $L_y = N_y\Delta y$. Similarly the time t is also discretized into:

$$t \to n\Delta t, \quad \text{where } n = 0, 1, 2, \ldots, N_t - 1 \tag{68}$$

where $N_t\Delta t$ is total length of time that we run the simulation for. The density field then becomes:

$$\phi(\mathbf{r}, t) \to \phi^n_{ij}. \tag{69}$$

Now we want to solve:

$$\frac{\partial \phi}{\partial t} = M\nabla^2\mu, \quad \text{where} \quad \mu = \alpha\phi + \beta\phi^3 - \kappa\nabla^2\phi. \tag{70}$$

First we can write the time derivative as:

$$\frac{\partial \phi}{\partial t} \simeq \frac{\phi^{n+1}_{ij} - \phi^n_{ij}}{\Delta t} + \mathcal{O}(\Delta t) \tag{71}$$

So the dynamics becomes:

$$\phi^{n+1}_{ij} = \phi^n_{ij} + \Delta t M \nabla^2 \mu^n_{ij}. \tag{72}$$

So for a given initial condition ϕ^0_{ij} we can find ϕ for subsequent timesteps: $\phi^1_{ij}, \phi^2_{ij}, \ldots$. Next we need to calculate the spatial derivatives of ϕ and μ in the lattice, *i.e.*:

$$\frac{\partial \phi}{\partial x} \simeq \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} + \mathcal{O}(\Delta x^2) \tag{73}$$

$$\frac{\partial \phi}{\partial y} \simeq \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} + \mathcal{O}(\Delta y^2) \tag{74}$$

$$\frac{\partial^2 \phi}{\partial x^2} \simeq \frac{\phi_{i+1,j} - 2\phi_{ij} + \phi_{i-1,j}}{\Delta x^2} + \mathcal{O}(\Delta x) \tag{75}$$

$$\frac{\partial^2 \phi}{\partial y^2} \simeq \frac{\phi_{i,j+1} - 2\phi_{ij} + \phi_{i,j-1}}{\Delta y^2} + \mathcal{O}(\Delta y) \tag{76}$$

$$\tag{77}$$

The Laplacian is then given by:

$$\nabla^2\phi = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = \frac{\phi_{i+1,j} - 2\phi_{ij} + \phi_{i-1,j}}{\Delta x^2} + \frac{\phi_{i,j+1} - 2\phi_{ij} + \phi_{i,j-1}}{\Delta y^2} + \mathcal{O}(\Delta x) + \mathcal{O}(\Delta y). \tag{78}$$

In Python ϕ_{ij} is represented as a matrix:

$$\phi = \begin{pmatrix} \phi_{00} & \phi_{01} & \cdots & \phi_{0,N_y-1} \\ \phi_{10} & \phi_{11} & & \phi_{1,N_y-1} \\ \vdots & & \ddots & \vdots \\ \phi_{N_x-1,0} & \phi_{N_x-1,1} & \cdots & \phi_{N_x-1,N_y-1} \end{pmatrix} \downarrow x\text{-direction} \tag{79}$$

$$\longrightarrow y\text{-direction} \tag{80}$$

Note that the x- and y-axis are transposed.

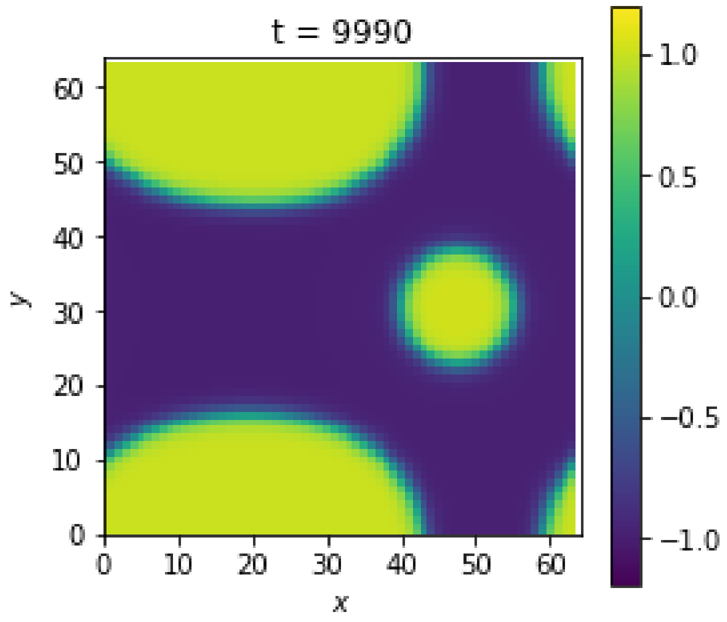
[ ]:
```python
# everything here are global variables
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

dx = 1.0  # normally dx = dy
```

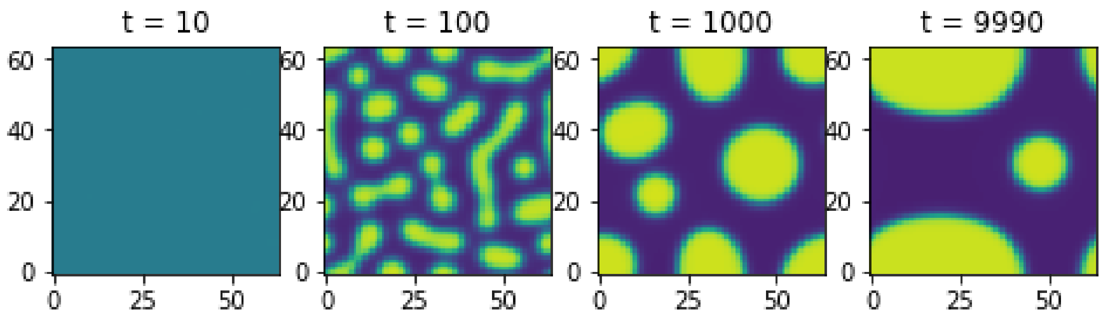
```
dt = 0.01

Nx = 64  # normally the total number of lattice = 2^(some integer)
Ny = 64  # this is required for Fourier transform later
Nt = 1000000
T = int(Nt*dt)  # total length of time

alpha, beta, kappa = -1.0, 1.0, 1.0
M = 1.0
phi0 = 0.0

# array of cartesian coordinates (needed for plotting)
x = np.arange(0, Nx)*dx
y = np.arange(0, Ny)*dx
y, x = np.meshgrid(y, x)

# create a phi-matrix of size Nx by Ny (similarly for mu-matrix)
phi = np.zeros((Nx, Ny))  # phi at instantaneous time
phi_t = np.zeros((Nx, Ny, T))  # phi as a function of time t
mu = np.zeros((Nx, Ny))
```

[ ]:
```
# method to calculate the laplacian
def laplacian(phi):
    # axis=0 --> roll along x-direction
    # axis=1 --> roll along y-direction
    laplacianphi = (np.roll(phi,+1,axis=0) - 2.0*phi + np.roll(phi,-1,axis=0))/
 ↪(dx*dx) \
                 + (np.roll(phi,+1,axis=1) - 2.0*phi + np.roll(phi,-1,axis=1))/
 ↪(dx*dx)

    return laplacianphi

# create animation
def animate(phi_t):
    # initialize figure and movie objects
    fig, ax = plt.subplots(figsize=(4,4))

    # set label
    ax.set_xlabel("$x$")
    ax.set_ylabel("$y$")

    # set x range and y range
    ax.set_xlim([0, Nx*dx])
    ax.set_ylim([0, Ny*dx])

    # set tick interval
    ax.tick_params(axis="both")
    plt.xticks(np.arange(0, Nx, 10)*dx)
    plt.yticks(np.arange(0, Ny, 10)*dx)

    # set aspect ratio
    ax.set_aspect('equal')

    # create colormap of phi
    colormap = ax.pcolormesh(x, y, phi_t[:,:,0], shading='auto', vmin=-1.2,␣
 ↪vmax=1.2)
    plt.colorbar(colormap)

    def animate(t):
        # set title
        ax.set_title(f't = {t}')
        colormap.set_array(phi_t[:,:,t].flatten())  # update data

    # interval = time between frames in miliseconds
    ani = animation.FuncAnimation(fig, animate, interval=10,␣
 ↪frames=range(0,T,10))
    ani.save("movie.mp4")

# plot phi at 4 different times
def plot(phi_t, t1, t2, t3, t4):
    # initialize figure and movie objects
    fig, (ax1, ax2, ax3, ax4) = plt.subplots(1,4,figsize=(8,2))

    ax1.set_title(f't = {t1}')
    ax1.set_aspect('equal')
    colormap = ax1.pcolormesh(x, y, phi_t[:,:,t1], \
                              shading='auto', vmin=-1.2, vmax=1.2)

    ax2.set_title(f't = {t2}')
    ax2.set_aspect('equal')
    colormap = ax2.pcolormesh(x, y, phi_t[:,:,t2], \
                              shading='auto', vmin=-1.2, vmax=1.2)

    ax3.set_title(f't = {t3}')
    ax3.set_aspect('equal')
    colormap = ax3.pcolormesh(x, y, phi_t[:,:,t3], \
                              shading='auto', vmin=-1.2, vmax=1.2)

    ax4.set_title(f't = {t4}')
    ax4.set_aspect('equal')
    colormap = ax4.pcolormesh(x, y, phi_t[:,:,t4], \
                              shading='auto', vmin=-1.2, vmax=1.2)

    plt.show()
```

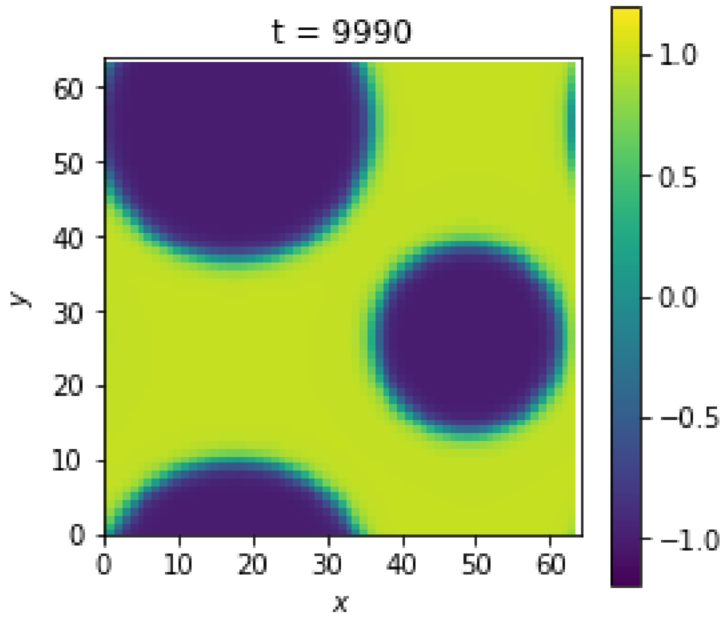
```
# apply wall boundary condition
def apply_walls(phi):
    phi[:, 0] = phi[:, 2]
    phi[:, 1] = phi[:, 2]
    phi[:, Ny-1] = phi[:, Ny-3]
    phi[:, Ny-2] = phi[:, Ny-3]

    return phi

# update phi
def update(phi):
    # calculate mu
    mu = alpha*phi + beta*phi*phi*phi - kappa*laplacian(phi)

    # update phi
    phi = phi + dt*M*laplacian(mu)

    return phi

# method to run the simulation from a uniform phi + a little bit of noise
def run():
    # initialize the system with uniform state + a little bit of noise
    phi = np.ones((Nx, Ny))*phi0 + np.random.normal(0.0, 0.001, (Nx, Ny))

    # update the \phi_{ij}^n for all n
    for n in range(0, Nt, 1):

        # save current phi
        if (n % int(1/dt) == 0):
            phi_t[:,:,int(n*dt)] = phi

        phi = update(phi)
```

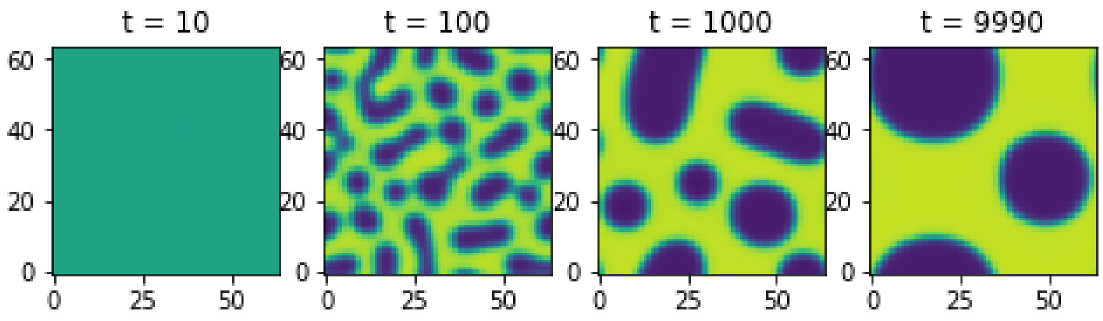
[ ]:
```
# run simulation for phi0 = -0.2
phi0 = -0.2
run()
#animate(phi_t)
plot(phi_t, 10, 100, 1000, 9990)
```



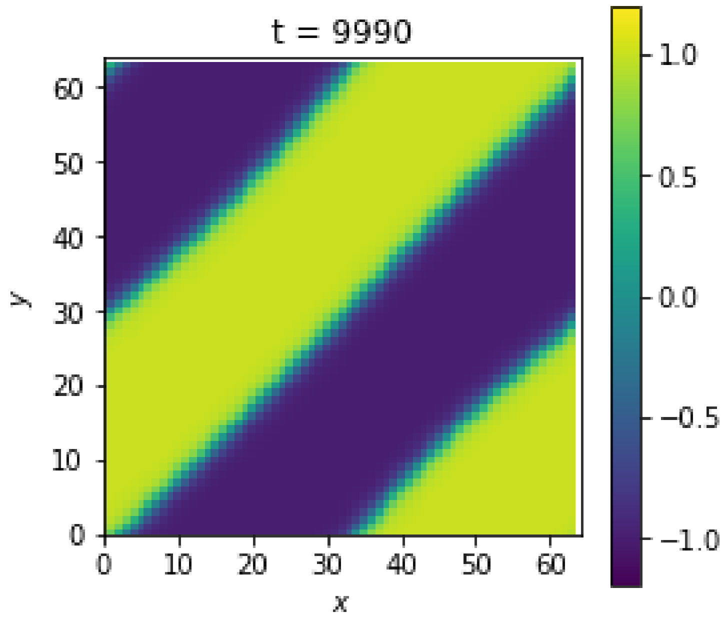


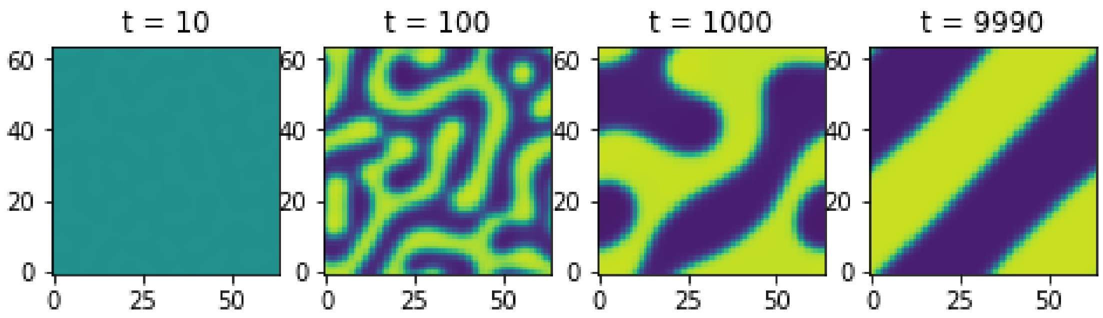
```
[ ]: # run simulation for phi0 = 0.2
     phi0 = 0.2
     run()
     #animate(phi_t)
     plot(phi_t, 10, 100, 1000, 9990)
```





```
[ ]: # run simulation for phi0 = 0.0
     phi0 = 0.0
     run()
     #animate(phi_t)
     plot(phi_t, 10, 100, 1000, 9990)
```


t = 9990
y
x
0
10
20
30
40
50
60
1.0
0.5
0.0
−0.5
−1.0


t = 10
t = 100
t = 1000
t = 9990
0
20
40
60
25
50