cahn_hilliard

November 8, 2022

1 Cahn-Hilliard equation

1.1 Free energy

Let us consider a box of size $L_x \times L_y \times L_z$ with periodic boundary condition on each side. The volume of the system is $V = L_x L_y L_z$. Let us define $\phi(\mathbf{r}, t)$ to be the rescaled density of the fluid at position $\mathbf{r}$ at time t. For example, $\phi(\mathbf{r}, t) \gtrsim 0$ represents the liquid phase and $\phi(\mathbf{r}, t) \lesssim 0$ represents the vapour phase. The total amount of fluid in the system $\int_V \phi \, dV$ is conserved since there is no mass creation or annihilation in the system. We can now define the global density ϕ_0 to be:

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

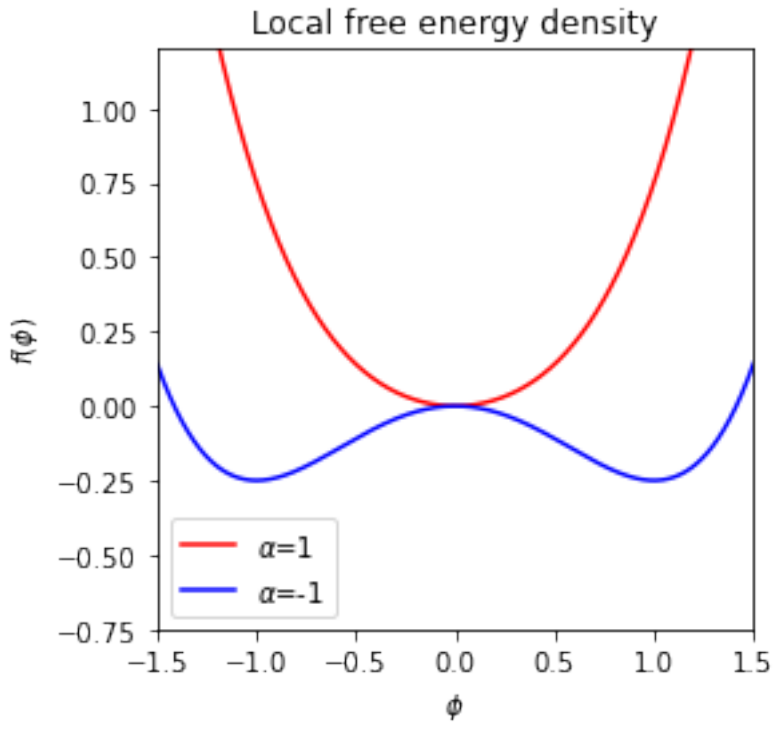
The value of ϕ_0 is fixed from the initial condition. The total free energy of the system can be written as:

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

$f(\phi)$ is a local free energy density 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 $\phi \simeq \sqrt{\frac{-a}{b}}$ and vapour phase $\phi \simeq -\sqrt{\frac{-a}{b}}$. On the other hand if $\alpha > 0$, then the system will remain in a homogenous phase with $\phi = \phi_0$ everywhere.

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

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('Local free 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
alpha = 1.0
f_phi = 0.5*alpha*phi**2 + 0.25*beta*phi**4
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 equilibrium state is given by the minimum of the total free energy $F[\phi]$, *i.e*,

$$\frac{\delta F}{\delta \phi} = 0, \tag{3}$$

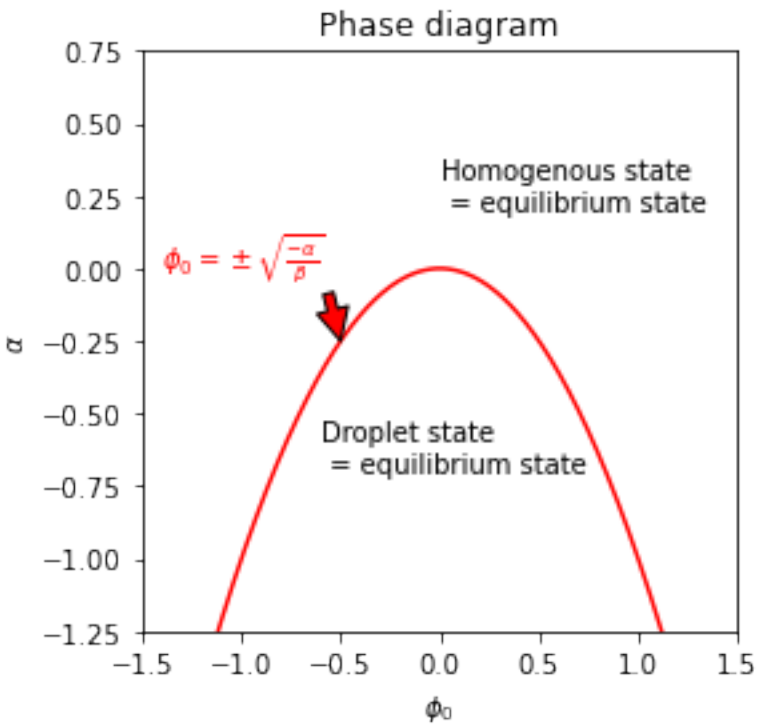
subject to the constraint

$$\phi_0 = \frac{1}{V}\int_V dV\, \phi(\mathbf{r}, t) = \text{constant}. \tag{4}$$

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: $\phi = +\sqrt{\frac{-\alpha}{\beta}}$ and $\phi = -\sqrt{\frac{-\alpha}{\beta}}$

The control parameters 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()
```



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:

$$\mu = \frac{\delta F}{\delta \phi} = \alpha\phi + \beta\phi^3 - \kappa\nabla^2\phi \tag{5}$$

1.2 Dynamics

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

$$\frac{\partial \phi}{\partial t} + \nabla \cdot \mathbf{J} = 0, \tag{6}$$

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

$$\frac{d}{dt}\int_V \phi \, dV + \int_V \nabla \cdot \mathbf{J} \, dV = 0 \tag{7}$$

$$\frac{d}{dt}\int_V \phi \, dV + \underbrace{\oint_S \mathbf{J} \cdot \hat{\mathbf{n}} \, dS}_{=0} = 0, \tag{8}$$

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:

$$\frac{d}{dt}\int_V \phi\, dV = 0 \tag{9}$$

$$\Rightarrow \int_V \phi\, dV = \text{constant}, \tag{10}$$

which implies the total amount of fluid is conserved.

Finally, the current is given by:

$$\mathbf{J} = -M\nabla\mu, \tag{11}$$

i.e., particles diffuse from regions with high chemical potential to regions with low chemical potential. $M > 0$ is called the mobility.

1.3 Linear stability of the homogenous state

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

$$\frac{\partial\phi}{\partial t} = \nabla^2\left[\alpha\phi + \beta\phi^3 - \kappa\nabla^2\phi\right] \tag{12}$$

Suppose we write the density field as follows:

$$\phi(\mathbf{r}, t) = \phi_0 + \delta\phi(\mathbf{r}, t), \tag{13}$$

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:

$$\frac{\partial\delta\phi}{\partial t} = \nabla^2\left[(\alpha + 3\beta\phi_0^2)\delta\phi - \kappa\nabla^2\delta\phi\right]. \tag{14}$$

Now we write $\delta\phi(\mathbf{r}, t)$ as a Fourier series:

$$\delta\phi(\mathbf{r}, t) = \sum_{\mathbf{q}} \delta\tilde{\phi}_{\mathbf{q}}(t) e^{i\mathbf{q}\cdot\mathbf{r}}. \tag{15}$$

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

$$q_x = 0, \pm\frac{2\pi}{L_x}, \pm\frac{4\pi}{L_x}, \dots \tag{16}$$

$$q_y = 0, \pm\frac{2\pi}{L_y}, \pm\frac{4\pi}{L_y}, \dots \tag{17}$$

$$q_z = 0, \pm\frac{2\pi}{L_z}, \pm\frac{4\pi}{L_z}, \dots \tag{18}$$

$$\tag{19}$$

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 $1, e^{\pm i\frac{2\pi}{L}x}, e^{\pm i\frac{4\pi}{L}x}, \dots$, with $\tilde{\phi}_q$ as the amplitude for each standing wave. Substituting the Fourier series into the dynamics, a set of first order ODE:

$$\frac{d\delta\tilde{\phi}_{\mathbf{q}}}{dt} = -\left[(\alpha + 3\beta\phi_0^2)q^2 + \kappa q^4\right]\delta\tilde{\phi}_{\mathbf{q}}, \quad \text{for each } \mathbf{q}. \tag{20}$$

The solution is:

$$\delta\tilde{\phi}_{\mathbf{q}}(t) = A_{\mathbf{q}} e^{r(q)t}, \tag{21}$$

where

$$r(q) = (-\alpha - 3\beta\phi_0^2)q^2 - \kappa q^4 \tag{22}$$

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.

$$r(q) = \underbrace{(-\alpha - 3\beta\phi_0^2)}_{<0} q^2 \underbrace{-\kappa}_{<0} q^4. \tag{23}$$

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.

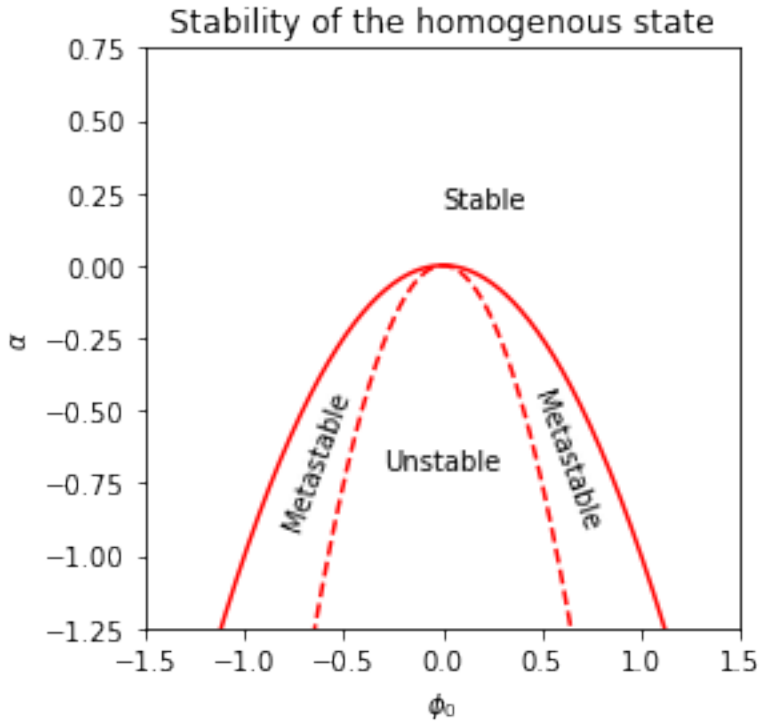
$$r(q) = \underbrace{(-\alpha - 3\beta\phi_0^2)}_{>0} q^2 \underbrace{-\kappa}_{<0} q^4. \tag{24}$$

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 $\lambda^* = \frac{2\pi}{q^*}$.

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 note that from the previous section, if $\sqrt{\frac{-\alpha}{3\beta}} < \phi_0 < \sqrt{\frac{-\alpha}{\beta}}$ and $-\sqrt{\frac{-\alpha}{\beta}} < \phi_0 < \sqrt{\frac{-\alpha}{3\beta}}$ then the actual equilibrium state (lowest energy state) of the system is 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.

[ ]:
```
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()
```



1.4 Numerical simulation

Let us consider a two-dimensional system. In computer simulation the space $\mathbf{r}$ 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:

$$x \to i\Delta x, \quad \text{where } i = 0, 1, 2, \ldots, N_x - 1 \tag{25}$$
$$y \to j\Delta y, \quad \text{where } j = 0, 1, 2, \ldots, N_y - 1. \tag{26}$$

$N_x \in \mathbb{N}$ 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:

$$t \to n\Delta t, \quad \text{where } n = 0, 1, 2, \ldots, N_t - 1 \tag{27}$$

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

$$\phi(\mathbf{r}, t) \to \phi^n_{ij}. \tag{28}$$

Now we want to solve:

$$\frac{\partial\phi}{\partial t} = \nabla^2\mu, \quad \text{where} \quad \mu = \alpha\phi + \beta\phi^3 - \kappa\nabla^2\phi. \tag{29}$$

First we can write the time derivative as:

$$\frac{\partial\phi}{\partial t} \simeq \frac{\phi^{n+1}_{ij} - \phi^n_{ij}}{\Delta t} + \mathcal{O}(\Delta t) \tag{30}$$

So the dynamics becomes:

$$\phi^{n+1}_{ij} = \phi^n_{ij} + \Delta t\nabla^2\mu^n_{ij}. \tag{31}$$

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.*:

$$\frac{\partial\phi}{\partial x} \simeq \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} + \mathcal{O}(\Delta x^2) \tag{32}$$
$$\frac{\partial\phi}{\partial y} \simeq \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} + \mathcal{O}(\Delta y^2) \tag{33}$$
$$\frac{\partial^2\phi}{\partial x^2} \simeq \frac{\phi_{i+1,j} - 2\phi_{ij} + \phi_{i-1,j}}{\Delta x^2} + \mathcal{O}(\Delta x) \tag{34}$$
$$\frac{\partial^2\phi}{\partial y^2} \simeq \frac{\phi_{i,j-1} - 2\phi_{ij} + \phi_{i,j-1}}{\Delta y^2} + \mathcal{O}(\Delta y) \tag{35}$$
$$\tag{36}$$

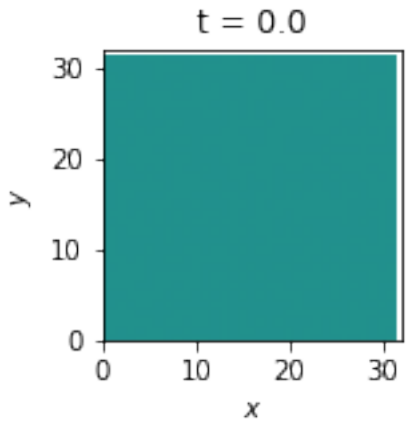
The Laplacian is then given by:

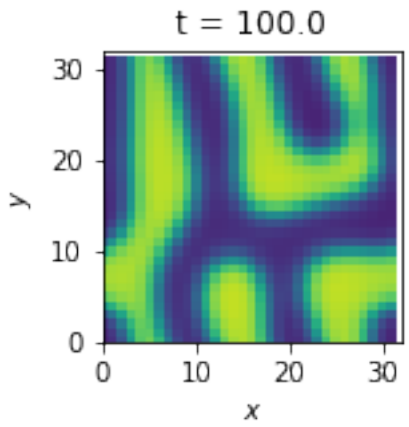
$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = \frac{\phi_{i+1,j} - 2\phi_{ij} + \phi_{i-1,j}}{\Delta x^2} + \frac{\phi_{i,j-1} - 2\phi_{ij} + \phi_{i,j-1}}{\Delta y^2} + \mathcal{O}(\Delta x) + \mathcal{O}(\Delta y). \tag{37}$$

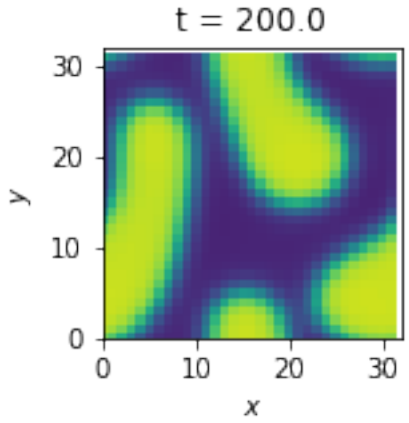
[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

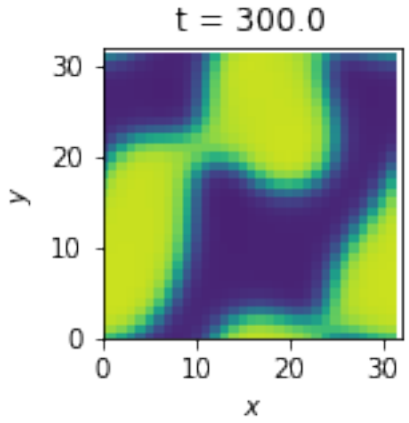
```
dx = 1.0  # normally dx = dy
dt = 0.01

Nx = 32  # normally the total number of lattice = 2^(some integer)
Ny = 32  # this is required for Fourier transform later
Nt = 100000

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

# create a phi-matrix of size Nx by Ny (similarly for mu-matrix)
# the ith row and jth column element of the matrix represents the value␣
 ↪\phi_{ij}
# note that in Python the y-axis is inverted
phi = np.zeros((Nx, Ny))
mu = np.zeros((Nx, Ny))

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

    return laplacianphi

# method to plot phi
def plot(phi, n):
    fig, ax = plt.subplots(figsize=(2,2))

    # set title
    ax.set_title(f't = {n*dt}')

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

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

    # set tick interval
```

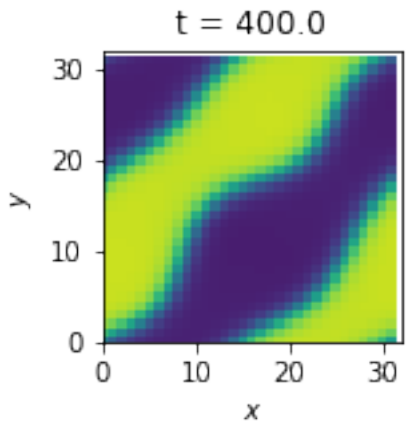
```
    ax.tick_params(axis="both")
    plt.xticks(np.arange(0, Nx, 10)*dx)
    plt.yticks(np.arange(0, Ny, 10)*dx)

    # cartesian coordinates
    x = np.arange(0, Nx)*dx
    y = np.arange(0, Ny)*dx
    x, y = np.meshgrid(x, y)

    ax.pcolormesh(x, y, phi, shading='auto', vmin=-1.2, vmax=1.2)

    plt.show()

# initial condition with noise mean = 0 and standard deviation = 0.001
phi = np.ones((Nx, Ny))*phi0 + np.random.normal(0.0, 0.001, (Nx, Ny))

# update the \phi_{ij} for all n
for n in range(0, Nt, 1):
    # plot phi
    if (n % 10000 == 0):
        plot(phi, n)

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

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

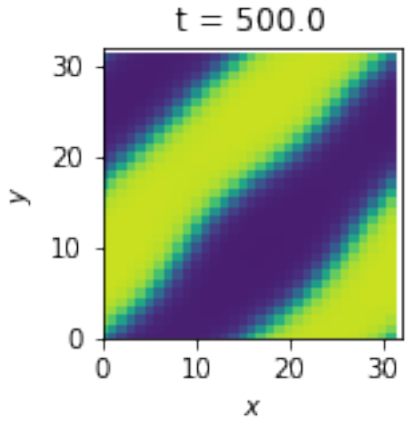


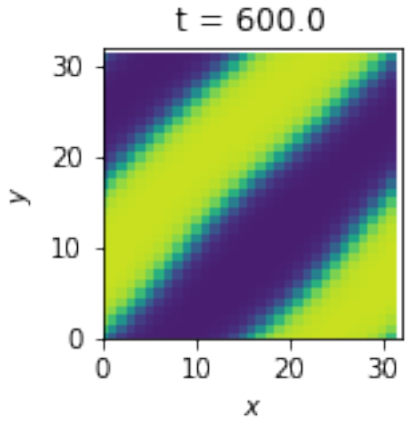

t = 100.0
30
20
10
0
y
0
10
20
30
x

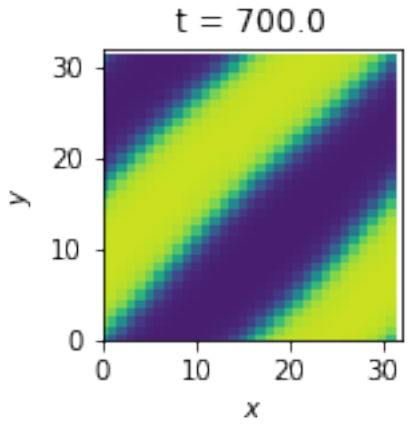

t = 200.0
30
20
10
0
y
0
10
20
30
x

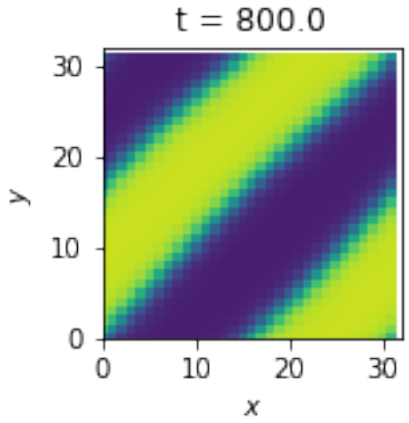

t = 300.0
30
20
10
0
y
0
10
20
30
x

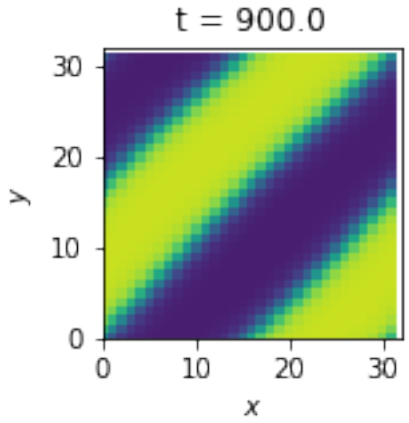

t = 400.0
30
20
10
0
y
0
10
20
30
x


t = 500.0
30
20
10
0
y
0
10
20
30
x

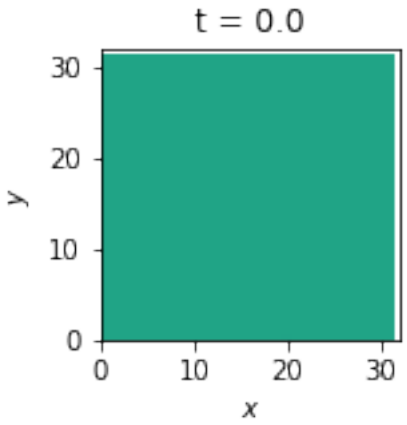

t = 600.0
30
20
10
0
y
0
10
20
30
x

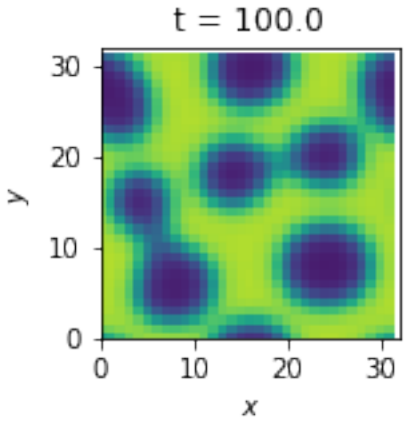

t = 700.0
30
20
10
0
y
0
10
20
30
x

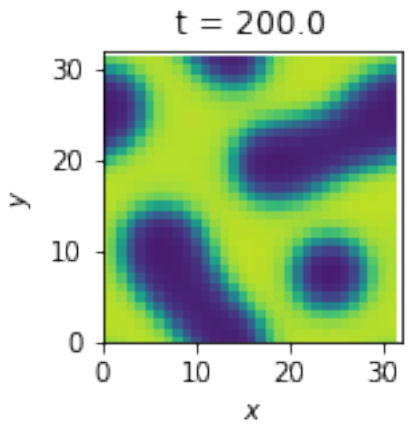

t = 800.0
30
20
10
0
y
0
10
20
30
x

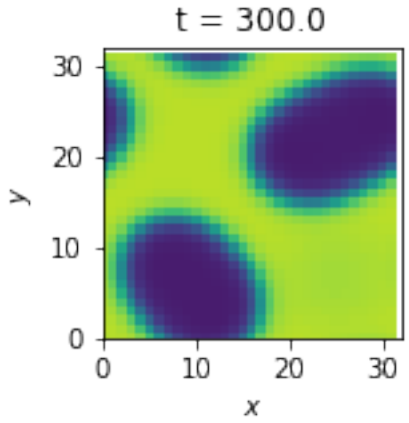

t = 900.0
30
20
10
0
y
0
10
20
30
x

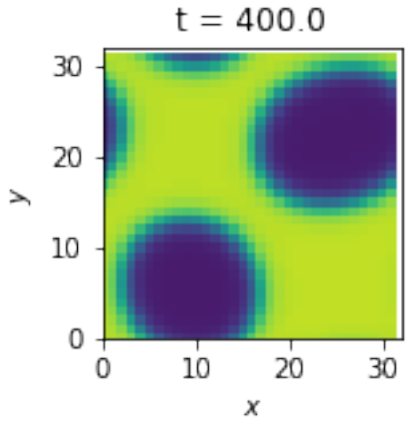
[ ]:
```
phi0 = 0.2

# initial condition with noise mean = 0 and standard deviation = 0.001
phi = np.ones((Nx, Ny))*phi0 + np.random.normal(0.0, 0.001, (Nx, Ny))

# update the \phi_{ij} for all n
for n in range(0, Nt, 1):
    # plot phi
    if (n % 10000 == 0):
        plot(phi, n)

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

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

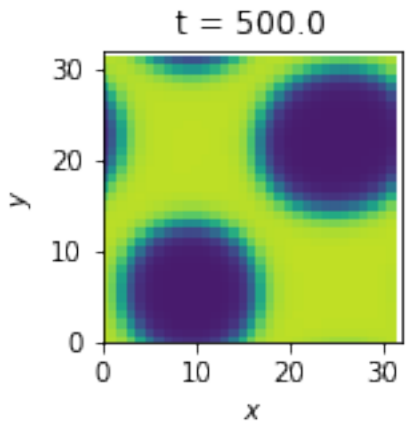


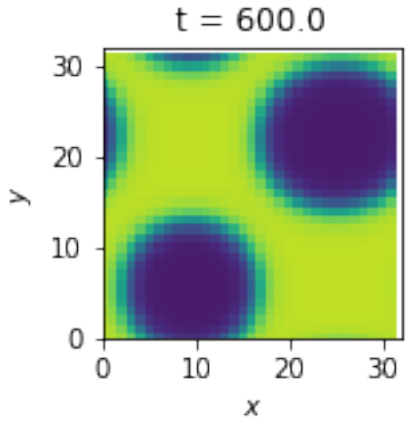


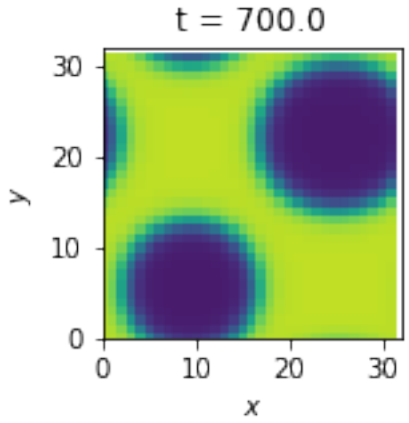

t = 200.0
30
20
10
0
y
0
10
20
30
x

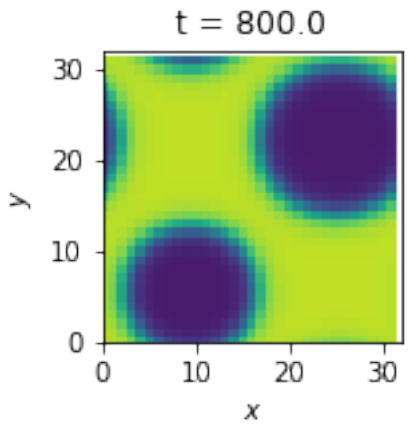

t = 300.0
30
20
10
0
y
0
10
20
30
x

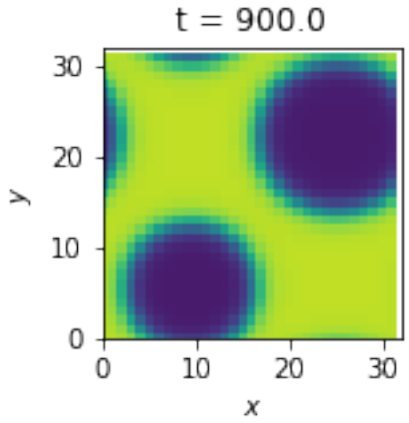

t = 400.0
30
20
10
0
y
0
10
20
30
x


t = 500.0
30
20
10
0
y
0
10
20
30
x


t = 600.0
30
20
10
0
y
0
10
20
30
x


t = 700.0
30
20
10
0
y
0
10
20
30
x


t = 800.0
30
20
10
0
y
0
10
20
30
x


t = 900.0
30
20
10
0
y
0
10
20
30
x

[ ]: