Allen-Cahn Phase Field Model

Siwol
code4simulation@gmail.com

This document explains the Allen-Cahn phase-field model and demonstrates its application in both single crystal growth and multi-grain growth through Python codes.

1 Single Crystal Growth Model

1.1 Model Formulation

The Allen-Cahn equation stems from minimizing the following free energy functional (1):

$$F[\phi] = \int \left(\frac{\epsilon^2}{2} |\nabla \phi|^2 + \frac{1}{4}(\phi^2 - 1)^2 \right) d\mathbf{x} \tag{1}$$

- The first term penalizes sharp interfaces (gradient energy).
- The second term is a double-well potential, with minimum values at $\phi = \pm 1$.

The time evolution is given by:

$$\frac{\delta \phi}{\delta t} = -M \frac{\delta F}{\delta \phi} = M \left[\epsilon^2 \Delta \phi - (\phi^3 - \phi) \right] \tag{2}$$

where M is the mobility, ϵ is the gradient coefficient, and Δ is the Laplacian operator.

1.2 Python Implementation

```python
import numpy as np
import matplotlib.pyplot as plt
# Simulation parameters
N = 128                         # The number of grid points
dx = 1.0                        # Size between grid points
dt = 0.01                       # Time step
nsteps = 2500                   # The number of steps
grad_coeff = 0.1                # Gradient coefficient
mobility = 5.0                  # Mobility
radius = 5                      # Seed radius
interval = 100                  # Plot update interval
```

```

# PBC options (Periodic Boundary Conditions)
pbc_x = True
pbc_y = True

# Initialization
phi = np.zeros((N, N))
x = np.arange(N)
y = np.arange(N)
X, Y = np.meshgrid(x, y)
center = N // 2
mask = (X - center)**2 + (Y - center)**2 <= radius**2
phi[mask] = 1.0

def laplacian(phi):
    # X direction calculation using periodic boundary conditions if
     enabled
    if pbc_x:
        phi_xp = np.roll(phi, -1, axis=0)
        phi_xm = np.roll(phi, 1, axis=0)
    else:
        phi_xp = np.zeros_like(phi)
        phi_xm = np.zeros_like(phi)
        phi_xp[:-1, :] = phi[1:, :]
        phi_xm[1:, :] = phi[:-1, :]
    # Y direction calculation using periodic boundary conditions if
     enabled
    if pbc_y:
        phi_yp = np.roll(phi, -1, axis=1)
        phi_ym = np.roll(phi, 1, axis=1)
    else:
        phi_yp = np.zeros_like(phi)
        phi_ym = np.zeros_like(phi)
        phi_yp[:, :-1] = phi[:, 1:]
        phi_ym[:, 1:] = phi[:, :-1]
    return (phi_xp + phi_xm + phi_yp + phi_ym - 4 * phi)/(dx * dx)

# Plotting settings (pcolormesh, colorbar range [-1, 1])
plt.ion()  # Turn on interactive mode
fig, ax = plt.subplots()
c = ax.pcolormesh(phi, cmap='RdBu', vmin=-1, vmax=1)
plt.colorbar(c, ax=ax)

for step in range(nsteps+1):
    lap = laplacian(phi)
    phi += dt * mobility * (grad_coeff * lap - (phi**3 - phi))
    # Update the plot every {interval} steps
    if step % interval == 0:
        ax.clear()
        c = ax.pcolormesh(phi, cmap='RdBu', vmin=-1, vmax=1)
        plt.title(f"Time: {step} steps")
        plt.pause(0.01)

plt.ioff()
plt.show()
```

Listing 1: Single Grain Growth Code

2 Multi-Grain Growth Model

2.1 Model Formulation

For multi-grain growth, the order parameter is vector-valued,

$$\vec{\phi} = (\phi_1, \ldots, \phi_{N_\mathrm{g}}) \tag{3}$$

where each ϕ_i represents the i-th grain and N_g is the number of grains. The free energy functional is extended:

$$F[\vec{\phi}] = \sum_{i=1}^{N_g} \left[\int \left(\frac{\epsilon^2}{2} |\nabla \phi_i|^2 + \frac{1}{4}(\phi_i^2 - 1)^2 \right) d\mathbf{x} + \int \left(\lambda \sum_{i<j} \phi_i^2 \phi_j^2 \right) d\mathbf{x} \right] \tag{4}$$

where ϵ is the gradient coefficient and λ is the penalty coefficient. The Allen-Cahn equations for each grain become:

$$\frac{\partial \phi_i}{\partial t} = M \left[\epsilon^2 \Delta \phi_i - (\phi_i^3 - \phi_i) - 2\lambda \phi_i \sum_{j \neq i} \phi_j^2 \right]$$

- The penalty term ($\lambda \sum_{i<j} \phi_i^2 \phi_j^2$) increases the energy when different grains overlap, thereby reducing field overlap.

2.2 Python Implementation

```python
import numpy as np
import matplotlib.pyplot as plt
# Simulation parameters
N = 128                         # The number of grid points
num_grains = 5                  # The number of grains
dx = 1.0                        # Size between grid points
dt = 0.01                       # Time step
nsteps = 2500                   # The number of steps
grad_coeff = 0.1                # Gradient coefficient
p_coeff = 2.0                   # Penalty term coefficient
mobility = 5.0                  # Mobility
radius = 5                      # Seed radius
interval = 100                  # Update interval for plotting

# PBC options (Periodic Boundary Conditions)
pbc_x = True
pbc_y = True

# phi (N x N x num_grains)
phi = np.zeros((N, N, num_grains))
X, Y = np.meshgrid(np.arange(N), np.arange(N))

# Initialization: set the initial seed for each grain
for i in range(num_grains):
    x0 = np.random.randint(0, N)
    y0 = np.random.randint(0, N)
    mask = (X - x0)**2 + (Y - y0)**2 <= radius**2
    phi[mask, i] = 1.0

def laplacian(phi_2d):
    # X direction calculation using periodic boundary conditions if
     enabled
    if pbc_x:
        phi_xp = np.roll(phi_2d, -1, axis=0)
        phi_xm = np.roll(phi_2d, 1, axis=0)
    else:
        phi_xp = np.zeros_like(phi_2d)
        phi_xm = np.zeros_like(phi_2d)
        phi_xp[:-1, :] = phi_2d[1:, :]
        phi_xm[1:, :] = phi_2d[:-1, :]
    # Y direction calculation using periodic boundary conditions if
     enabled
    if pbc_y:
        phi_yp = np.roll(phi_2d, -1, axis=1)
        phi_ym = np.roll(phi_2d, 1, axis=1)
    else:
        phi_yp = np.zeros_like(phi_2d)
        phi_ym = np.zeros_like(phi_2d)
        phi_yp[:, :-1] = phi_2d[:, 1:]
        phi_ym[:, 1:] = phi_2d[:, :-1]
    return (phi_xp + phi_xm + phi_yp + phi_ym - 4 * phi_2d) / (dx *
     dx)

# Plotting settings (pcolormesh, colorbar range [-1, 1])
# (2x3 subplots: 5 grains plots + combined view)
plt.ion()
fig, axs = plt.subplots(2, 3, figsize=(15, 8))
cmap_combined = plt.get_cmap('tab10', num_grains)  # Grain coloring

for i in range(num_grains):
    ax = axs.flatten()[i]
    c = ax.pcolormesh(phi[:, :, i], cmap='RdBu', vmin=-1, vmax=1)
    ax.set_title(f'Grain {i+1}')
    plt.colorbar(c, ax=ax)

# Initializing combined view
ax_combined = axs[1, 2]
cmap = plt.get_cmap('tab10', num_grains)
grain_colors = [np.array(cmap(i)[:3]) for i in range(num_grains)]
combined_rgb = np.ones((N, N, 3))  # Initialize with white
    background (RGB 3-channel, float)
for i in range(num_grains):
    combined_rgb += np.expand_dims(phi[:, :, i], axis=2) *
    grain_colors[i]
im_combined = ax_combined.imshow(combined_rgb,
                                 interpolation='nearest',
                                 extent=[0, N, 0, N])
ax_combined.set_title('Combined Grains')

for step in range(nsteps):
    lap = np.zeros_like(phi)
    for i in range(num_grains):
        lap[:, :, i] = laplacian(phi[:, :, i])
```

```python

    # Update each grain
    total_phi_sq = np.sum(phi**2, axis=2)
    for i in range(num_grains):
        penalty = 2 * p_coeff * phi[:, :, i] * (total_phi_sq - phi[:, :, i]**2)
        phi[:, :, i] += dt * mobility * (grad_coeff * lap[:, :, i] - (phi[:, :, i]**3 - phi[:, :, i]) - penalty)

    # Update every {interval}
    if step % interval == 0:
        # Update each grain plot
        for i in range(num_grains):
            ax = axs.flatten()[i]
            ax.clear()
            c = ax.pcolormesh(phi[:, :, i], cmap='RdBu', vmin=-1, vmax=1)
            ax.set_title(f'Grain {i+1} (Step {step})')

        # Update combined view
        #(Combine each grain's field weighted by its respective color
        combined_rgb = np.ones((N, N, 3))
        for i in range(num_grains):
            combined_rgb -= np.expand_dims(phi[:, :, i], axis=2) * grain_colors[i]
        combined_rgb = np.clip(combined_rgb, 0, 1)

        im_combined.set_data(combined_rgb)
        ax_combined.set_title(f'Combined Grains (Step {step})')
        plt.pause(0.01)

plt.ioff()
plt.show()
```

Listing 2: Multi Grain Growth Code