User Guide – Kobayashi Phase-Field Simulation (Isotropic)

Tushar Jogi

July 1, 2025

Overview

This guide walks you through compiling and executing the code, setting up the environment, understanding parameters, and how libraries like PETSc are integrated to solve the coupled PDE system based on Kobayashi's 1993 model. Currently the package only runs simulation in 2D. The package can be easily extended to 3D. However, to ensure efficient execution of 3D simulation parallelization will be needed.

1. Code Compilation and Execution

Python Version

Run the simulation using:

```
./run.sh python
```

This will execute `main.py` and store `.h5` and `.png` outputs in `python/data/`.

C++ Version

Run the C++ implementation with:

```
./run.sh cpp
```

Output will be saved to `cpp/data/`.
Ensure the script is executable:

```
chmod +x run.sh
```

2. Parameter Description

Simulation parameters are specified in `config/params.yaml`.

Parameter	**Type**	**Description**
`epsilon`	float	Gradient energy coefficient (interface width)
`tau`	float	Phase-field relaxation time
`K`	float	Dimensionless latent heat
`alpha`	float	Controls steepness of the $m(T)$ function
`gamma`	float	Controls tanh width in $m(T)$
`dt`	float	Time step size
`dx`, `dy`	float	Grid spacing

`Nx`, `Ny`	int	Number of grid points in X and Y
`a`	float	Strength of noise added to the phase field
`steps`	int	Total number of simulation steps
`output_interval`	int	Output frequency

3. Environment Setup (Conda)

We recommend using Conda with packages from `conda-forge`:

```
conda env create -f environment/env.yml
conda activate kobayashi
```

Key packages:

- numpy
- matplotlib
- h5py
- pyyaml
- petsc4py

4. Governing Equations and Reference

This implementation is based on:
R. Kobayashi, *Modeling and numerical simulations of dendritic crystal growth*, Physica D 63 (1993), pp. 410–423.

Phase-Field Equation (Eq. 3)

$$\tau \frac{\partial p}{\partial t} = \nabla \cdot \left(\epsilon^2 \nabla p\right) + p(1-p)\left(p - \frac{1}{2} + m(T)\right) \tag{1}$$

Heat Equation with Latent Heat Coupling (Eq. 5)

$$\frac{\partial T}{\partial t} = \nabla^2 T + K \frac{\partial p}{\partial t} \tag{2}$$

where

$$m(T) = \frac{\alpha}{\pi} \tan^{-1}(\gamma(T_e - T)) + \xi$$

$$\xi = a\chi,$$

where ξ is random noise and χ is uniformly distributed on the interval $[\frac{-1}{2}, \frac{1}{2}]$

Phase-field equation is solved using IMEX scheme and heat equation is solved using implicit backward Euler scheme.

Boundary conditions:

- Dirichlet (cooling) on the left wall for T
- Neumann on remaining walls for T and on all walls for p

The discretized form of phase-field and heat equation are as follows:

$$\tau \frac{p^{n+1} - p^n}{\Delta t} = \epsilon^2 \nabla^2 p^{n+1} + f(p^n, T^n) \tag{3}$$

$$\left(\tau I - \Delta t\, \epsilon^2 L\right) p^{n+1} = \tau p^n + \Delta t f(p^n, T^n)$$

$$\frac{T^{n+1} - T^n}{\Delta t} = \nabla^2 T^{n+1} + K \frac{p^{n+1} - p^n}{\Delta t} \tag{4}$$

$$\left(I - \Delta t \nabla^2\right) T^{n+1} = T^n + K \left(p^{n+1} - p^n\right)$$

5. Library Integration and PDE Solving

This package uses following libraries:

PETSc (petsc4py)

- Solves the phase-field equation via implicit-explicit (IMEX) time stepping
- Solves the heat equation via backward Euler method
- Sparse matrices constructed using `PETSc.Mat().createAIJ`
- Solvers configured with PETSc KSP interface

NumPy

Used for grid setup, arithmetic operations, and array-based logic.

HDF5 + Matplotlib

- Field data saved as `.h5` files via h5py
- PNG visualizations rendered using matplotlib

Output

The simulation generates field data files and image files: Every `output_xxxxx.h5` file contains

- `p` : Phase field
- `T` : Temperature field

Visualization images (`visualization_xxxxx.png`) provide quick snapshots of field evolution