PHASETREE

April 15, 2025

**Assignment | Phase-Field Code Implementation**

Your task is to develop a computational framework for simulating the evolution of BCC/B2 microstructures in high-entropy alloys using the Cahn-Hilliard phase-field model. The model is based on the research by Li et al. (2020) and involves solving coupled equations that account for both chemical and elastic energy contributions.

We expect you will spend around one or two full days of work on this task, depending on your skill level, experience, and familiarity with the topic. The assignment will be assessed based on practical time constraints. If you should make some simplifying assumptions due to the time constraint, please explicitly mention which assumptions are made.

**Task**

Your task is to develop a phase-field simulation tool capable of reproducing the work reported in the aforementioned paper. As the focus is not on reproducing the exact results of the paper, you are encouraged to use smaller grids (e.g., $200 \times 200$ or less) to reduce the computational resources required. Although the paper showcases the results for three materials compositions, you are asked to demonstrate your code using only the No.3-Al17 case. Please use the parameters provided in Tables 1 and 2 of the paper. Although you are encouraged to use open-source libraries and make simplifying assumptions to complete the task, you must solve the physical problems presented in the paper without omitting and simplifying the governing equations. When assessing the assignment, emphasis is given to

1. Implementation of the Phase-Field Model:
    - Implement a Phase-Field model (using both Python and C++) to solve the coupled equations described in the paper.
    - Incorporate the chemical free energy (double-well potential) and elastic strain energy (using Khachaturyan's microelasticity theory for elastic equilibrium, $\nabla \cdot \sigma = 0$).
    - Simulate the evolution of the order parameter (composition) and displacement fields.
2. Numerical Scheme:
    - Implement a stable and efficient numerical scheme. You can choose either implicit or implicit-explicit numerical methods for stability and efficiency. Avoid fully explicit methods.
3. Use of External Libraries:
    - Utilize open-source libraries such as PETSc, NumPy, SciPy, and other BSD/MIT-licensed software. Commercial software usage (e.g., COMSOL, Ansys, Abaqus) is prohibited.
4. Parameter Handling:
    - Adopt parameters such as elastic constants, misfit strains, and gradient energy coefficients directly from the paper.
    - Design your code to easily modify these parameters through configuration files (e.g., JSON, YAML, or Python dictionaries).

- Hardcoding tensors and physical properties are acceptable if clearly documented.

5. Output and Visualization:
   - Provide clear visualizations or save intermediate stage distributions of the order parameter as .h5, .npy, or image files (e.g., using Matplotlib or VTK).

**Deliverables**

1. Source Code: A well-structured, modular, and documented codebase in Python and C++. The code must be submitted via a Git repository (GitHub or GitLab). Include clear commit messages and a README file with instructions on how to run the code. Please make a public repository and provide the link. You can make the repository private afterward.
2. Configuration Files**:** Easily modifiable files for simulation parameters.
3. Visualization Outputs: Sample images or data files demonstrating the simulation results. The simulation output should include snapshots at $t = 0,\ 1000,\ 5000,\ 10000,$
4. Documentation:
   a. Detailed docstrings and comments within the code.
   b. A user guide covering:
      i. Code compilation and execution instructions.
      ii. Parameter descriptions.
      iii. Environment setup (e.g., Docker or Conda).
      iv. Reference to relevant equations from the paper.
5. Unit Tests: Implement unit tests to verify the functionality of key code components.

**Assessment Criteria**

- Code Structure and Modularity: Design a clean, modular, and well-organized codebase
- Scalability and Reusability: Ensure the code can be easily extended to 3D simulations (initial implementation is in 2D).
- Library Integration: Document how external libraries are used to solve PDEs.
- Documentation Quality: Provide comprehensive and clear documentation.
- Verification and Reliability: Demonstrate the correctness of the code through unit tests.

**Deadline**

Please submit all the deliverables by **Sunday, 27 April 2025, 23:59** by sending the link to the git repository to jin@phasetree.ai.

**Additional Notes**

- Document any assumptions or deviations from the original paper.
- Focus on writing readable and maintainable code.
- Complex parsers or sophisticated file formats are not required; simple CSV, numpy binary, or image formats are enough.
- Initial composition fields can be hardcoded directly.