Manual for simulating the spinodal decomposition of a binary alloy using a finite difference solver

Nasir Attar

July 4, 2023

1 Introduction

This solver aims to simulate the phase separation of a binary alloy in the spinodal region of the phase diagram. The absence of thermodynamic barrier in spinodal region forces the decomposition to only be driven by diffusion. Finite difference discretization is implemented which is second order accurate in space while forward Euler time marching scheme is used for temporal discretization.

2 Infile

The input file is defined in the executable folder and contains information regarding material properties, spatial and temporal discretization parameters. The infile is mentioned below and the comments determine various parameters used in the simulation.

```
## Determine the number of cells
ncell = 64

## Maximum grid size for domain distribution
maxgrid_size=32

## Real size of the simulation box
Real_dom_low=0.0
Real_dom_high=64.0

## Number of timesteps and timestep size
nsteps=30000
dt=0.01

## Material specific parameters
c0=0.4
mob=1.0
kap=0.5

## Writing output files every 100 timesteps
plot_int=100
```

2.1 Simulation geometry and discretization

```
## Determine the number of cells
ncell = 64

## Maximum grid size for domain distribution
maxgrid_size = 32

## Real size of the simulation box
Real_dom_low = 0.0
Real_dom_high = 64.0
```

- **ncell:** The simulation domain will be divided in 64 cells. This marks the smallest spatial resolution in the simulation domain.
- **maxgrid_size:** Dividing the simulation domain into maximum packets of 32 cells. One can think of maximum grid size as dividing a square in 'p' equal parts where p=(ncell/maxgrid_size)^2. These parts can be distributed to multiple processors for parallel computation.
- **Real_dom_low:** Define the lower limit of the real domain
- **Real_dom_high:** Define the higher limit of the real domain. Here the size of the real domain is 64.0 and the ncell is 64, thus dx = (Real domain size)/(ncell) = 64/64 = 1. The smallest scale of spatial discretization can be further lowered while keeping an eye on the solver stability.

```
## Number of time-steps and time-step size
nsteps = 30000
dt = 0.01
```

- **nsteps:** The number of time-steps to get the evolution of the profile
- **dt:** The time-step size

2.2 Material specific parameters

```
## Material specific parameters
c0 = 0.4
mob = 1.0
kap = 0.5
```

- **c0:** Average alloy composition. Useful to generate an initial profile for the alloy.
- **mob:** Mobility of the diffusion. Basically governs the speed of diffusion.
- **kap:** Gradient energy coefficient.

3 Model formulation

The governing equation for spinodal decomposition is the Cahn-Hilliard equation. We start the formulation by defining the free energy functional in its simplest form,

$$F = \int_V \left[f(c) + \frac{1}{2}\kappa(\nabla c)^2 \right] dv \tag{1}$$

where κ is the gradient energy coefficient and $f(c)$ is the double-well potential/bulk energy. The bulk energy has multiple forms, the one used in this model is described in eq.(2), where A is a positive constant and controls the height of the energy barrier between two phases. The value of the bulk energy in either phases (c=0 and c=1) is zero which is a key aspect while choosing an appropriate expression for double-well potential.

$$f(c) = Ac^2(1 - c^2) \tag{2}$$

The evolution of Cahn-Hilliard equation reads,

$$\frac{c_{ij}^{n+1} - c_{ij}^{n}}{\Delta t} = \nabla^2 M \left(\frac{\delta F_{ij}}{\delta c} \right)^n \tag{3}$$

The term on the right hand side is achieved by taking a functional derivative of eq.(1). Here, "n" denotes the current time-step. Explicit solvers require a small time-step for a smaller spatial resolution (Δx), hence fine grid with an explicit solver is computationally heavy. To get the functional derivative of the free energy functional, the Euler-Langrange equation is used. The Euler-Langrange equation is written as follows,

$$\frac{\delta F_{ij}}{\delta c} = \frac{\partial}{\partial c}\left(f(c_{ij}) + \frac{1}{2}\kappa(\nabla c_{ij})^2 \right) - \nabla \left(\frac{\partial}{\partial \nabla c}\left(f(c_{ij}) + \frac{1}{2}\kappa(\nabla c_{ij})^2 \right) \right) \tag{4}$$

After simplification, the term looks like,

$$\left(\frac{\delta F_{ij}}{\delta c} \right)^n = 2A\left(c_{ij}^n \left(1 - c_{ij}^n\right)^2 - \left(c_{ij}^n\right)^2 \left(1 - c_{ij}^n\right) \right) - \kappa \nabla^2 c_{ij}^n \tag{5}$$

Equation(3) and (5) are the final equations that are solved in the solver. Five point stencil is used for the Laplacian in eq.(3) and (5).

3.1 Spatial discretization

The five point stencil used for the Laplacian reads as,

$$\nabla^2 c_{ij} = \frac{(c_{i+1,j} + c_{i-1,j} + c_{i,j-1} + c_{i,j+1} - 4c_{i,j})}{\Delta x * \Delta y} \tag{6}$$

The above formula is applicable when $dx = dy$. For accurate solution it is recommended to at least have a second order accurate spatial discretization

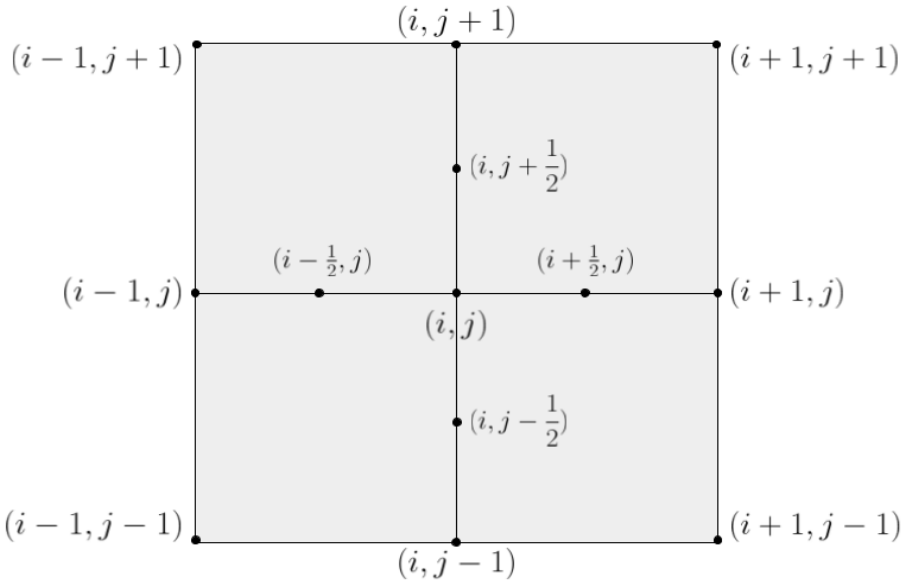

Figure 1: Computation grid (lowest resolution is Δx and Δy)

scheme. The derivation for obtaining the Laplacian is given below,
The Laplacian is written as,

$$\nabla^2 c_{ij} = \left(\frac{\partial^2 c}{\partial x^2}\right)_{ij} + \left(\frac{\partial^2 c}{\partial y^2}\right)_{ij} \tag{7}$$

The second order accurate derivative in the x-direction is computed as follows,

$$\left(\frac{\partial c}{\partial x}\right)_{ij} = \frac{c_{i+\frac{1}{2},j} - c_{i-\frac{1}{2},j}}{\Delta x} + O(\Delta x)^2 \tag{8}$$

This form helps us stay within the square stencil and keeps the errors confined. The separate derivatives are computed as,

$$\left(\frac{\partial^2 c}{\partial x^2}\right)_{ij} = \frac{\partial}{\partial x}\left(\frac{\partial c}{\partial x}\right)_{ij} = \frac{c_{i+1,j} + c_{i-1,j} - 2c_{i,j}}{(\Delta x)^2} \tag{9}$$

$$\left(\frac{\partial^2 c}{\partial y^2}\right)_{ij} = \frac{\partial}{\partial y}\left(\frac{\partial c}{\partial y}\right)_{ij} = \frac{c_{i,j+1} + c_{i,j-1} - 2c_{i,j}}{(\Delta y)^2} \tag{10}$$

Adding eq.(9) and eq.(10) and keeping $\Delta x = \Delta y$, we land at eq.(6).

4 Compilation package

The solver package contains of two folders (1)Exec (2)Source. The "Exec" folder contains 'GNUmakefile' and an input file. The "Source" folder contains the source and header files and a 'Make.package'. The details of all the files necessary for compilation is mentioned below.

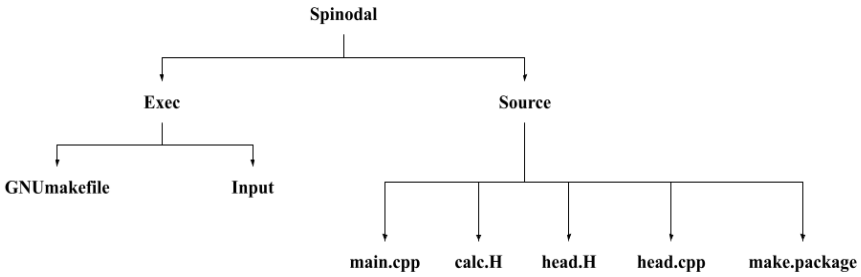

Figure 2: Path of the files in the Spinodal folder

4.1 GNUmakefile

The GNUmakefile is necessary for AMReX to locate the present working directory of the solver and provides vital information for compiling. One must make sure the AMREX_HOME is located properly every time a GNUmakefle is written. GNUmakefile gives freedom to use a debugger, MPI for parallel processing, choose a compiler, set the dimensionality of the problem we are solving. Last six lines are required to link the make package from the source folder of our solver with the universal make file of AMReX. This step is necessary to link the header files with the rest of the code.

```
AMREX_HOME ?= ../../../../../amrex

DEBUG = FALSE
USE_MPI = FALSE
MPI_THREAD_MULTIPLE = FALSE # amrex.async_out=1 with more than 64
    ↪ processes requires MPI_THREAD_MULTIPLE
USE_OMP = FALSE
COMP = gcc
DIM = 2

USE_CUDA = FALSE
USE_HIP = FALSE
USE_DPCPP = FALSE

include $(AMREX_HOME)/Tools/GNUMake/Make.defs

include ../Source/Make.package
VPATH_LOCATIONS += ../Source
INCLUDE_LOCATIONS += ../Source

include $(AMREX_HOME)/Src/Base/Make.package

include $(AMREX_HOME)/Tools/GNUMake/Make.rules
```

4.2 Make.package

The Make.package file contains links to the source as well as header files in the "Source" folder.

```
CEXE_sources += main.cpp head.cpp
CEXE_headers += head.H calc.H
```

5 Compiling, running and discussion

- The solver can be compiled using the command "make" in the command line in a terminal that is opened in the "Exec" folder.
- An executable is created after the make command is completed.
- To run the program on a single processor, type the command written in double quotes "./<name of the executable file> <name of the input file>". For eg. ./main2d.gnu.ex input
- To run the program on multiple processors, make USE_MPI = TRUE in the GNUmakefile. This will generate an MPI executable. Type the command written in double quotes in the terminal "mpiexec -n < number of processors > ./<name of the executable file> <name of the input file>". For eg. mpiexec -n 2 ./main2d.gnu.MPI.ex input for 2 processors.
- The plot files will be generated after every 100 time-steps. These files can be viewed in a post processing software. Currently we are using Paraview to see the evolution.
- As time progresses the diffusion leads to phase separation. The bigger precipitate wants to expand itself by taking in smaller precipitates , thus the number of smaller precipitates decrease over time this phenomenon is known as Ostwald ripening.