# Manual for using the FFT-based spectral solver for the KKS formalism implemented using CUDA

Saurav Shenoy, Tushar Jogi and Saswata Bhattacharyya

January 22, 2022

## 1 Introduction

This is a multicomponent multiphase phase-field solver based on a Fourier spectral discretization of a periodic domain (representative volume element) with a forward Euler time marching scheme. The formulation is based on the phase-field model presented in *Physical Review E 60, 7186, 1999*. The solver is parallelized on NVIDIA GPUs using CUDA. Certain basic steps are to be taken before one runs the solver.

## 2 Infile

The input parameters required for the solver are derived from an infile. This contains information about the domain geometry, the thermodynamic functions(free energy), material properties such as the interfacial energies and their anisotropies as well as boundary conditions. It could also contain special flags related to the running of the solver. The following is a basic description of the keys in the infile. **Each key must end with a semicolon**. Additionally, all lines beginning with "#" will be treated as comments in the infile.

```
##Geometrical dimensions of the simulation domain
DIMENSION = 2;
MESH_X = 128;
MESH_Y = 128;
MESH_Z = 1;
##Discretization, space and time
DELTA_X = 1;
DELTA_Y = 1;
DELTA_Z = 1;
DELTA_t = 0.1;
##Number of phases and composition
NUMPHASES = 2;
NUMCOMPONENTS = 2;
#Running and saving information
NTIMESTEPS = 1000;
NSMOOTH = 10;
SAVET = 100;
RESTART = 0;
```

```
STARTTIME = 0;
numworkers = 0;
## Component and Phase names
COMPONENTS = {C1,C2};
PHASES = {alpha,beta};
##Material properties
GAMMA = {0.8};
R = 1.0;
V = 1.0;
DIFFUSIVITY = {1,0,1};
DIFFUSIVITY = {1,1,1};
EIGEN_STRAIN = {0,0.01,0.01,0.0,0.0,0.0,0.0};
EIGEN_STRAIN = {1,0.01,0.01,0.0,0.0,0.0,0.0};
VOIGT_CUBIC = {0,2700,2000,1200};
VOIGT_CUBIC = {1,2700,2000,1200};
##Boundary conditions
BOUNDARY = {phi,3,3,3,3,0,0};
BOUNDARY = {mu,3,3,3,3,0,0};
BOUNDARY = {c,3,3,3,3,0,0};
BOUNDARY = {T,3,3,3,3,0,0};
BOUNDARY_VALUE = {phi,0,0,0,0,0,0};
BOUNDARY_VALUE = {mu,0,0,0,0,0,0};
BOUNDARY_VALUE = {c,0,0,0,0,0,0};
BOUNDARY_VALUE = {T,0,0,0,0,0,0};
##Model-specific parameters: KKS FFT GPU
WRITEFORMAT = ASCII;
TRACK_PROGRESS = 100;
Noise_phasefield = 0;
Amp_Noise_Phase = 0;
relax_coeff = 1.5;
c0 = 0.12;
alpha = 3.0;
lambda = 8.0;
ELAST_INT = 0;
f0 = {0,0.75};
f0 = {1,0.75};
ceq = {0,0,0};
ceq = {0,1,0};
ceq = {1,0,1};
ceq = {1,1,1};
```

## 2.1 Simulation geometry, spatial and temporal discretization

```
##Geometrical dimensions of the simulation domain
DIMENSION = 2;
MESH_X = 100;
MESH_Y = 100;
```

```
MESH_Z = 1;
```

- DIMENSION: This can take values 2,3 for 2D and 3D simulations respectively. This is a required key in the solver and not mentioning this key might lead to unexpected results
- MESH_X,MESH_Y,MESH_Z: These are the number of grid points in the domain(and not the physical lengths) in the respective X, Y, Z directions. When DIMENSION is 2, the value of MESH_Z will redundant and will be taken as 1.

```
##Discretization, space and time
DELTA_X = 2.0;
DELTA_Y = 2.0;
DELTA_Z = 2.0;
DELTA_t = 0.08;
```

- The values DELTA_X, DELTA_Y, DELTA_Z correspond to the grid resolution in the X, Y, Z directions respectively. Similarly, DELTA_t corresponds to the temporal discretization(time-step).

## 2.2 Phases and Components information

```
##Number of phases and composition
NUMPHASES = 2;
NUMCOMPONENTS = 2;
```

- The keys are self-explanatory. NUMPHASES corresponds to the number of phases in the domain, while NUMCOMPONENTS corresponds to the number of components(2, for binary, 3 for ternary etc.)
- These keys are absolutely necessary, please fill carefully for successful running of your codes.

```
COMPONENTS = {Al, Cu};
PHASES = {alpha, beta};
```

- COMPONENTS refers to the tuple that consists of the names of the components. The names are separated by commas and the entire tuple needs to placed within {} followed by a semicolon.
- Similarly, PHASES refers to the names of the phases in the domain.

## 2.3 Boundary conditions

```
##0:FREE, 1: Neumann, 2: Dirichlet, 3: Periodic, 4: Complex
##BOUNDARY = {TYPE, X_LEFT, X_RIGHT, Y_FRONT, Y_BACK, Z_TOP,
    ↪ Z_BOTTOM}
BOUNDARY = {phi, 1, 1, 1, 1, 0, 0};
BOUNDARY = {mu, 1, 1, 1, 1, 0, 0};
BOUNDARY = {c, 1, 1, 1, 1, 0, 0};
BOUNDARY = {T, 1, 1, 1, 1, 0, 0};
```

- Any of the solvers in repository will consist of the following scalar default types. Type "phi" will represent the phase-field order parameters whose number is specified by the key, "NUMPHASES". Depending on the solver whether it is the grand-potential based solver, where "mu" will be the independent variable, or if it is the Cahn-Hilliard or Kim-Kim-Suzuki based solvers, where "c" is the independent variable, Type "mu" or "c" will refer to the components in the key "COMPONENTS". Similarly, type "T" will refer to the temperature field. The BOUNDARY key will refer to the boundary condition for the respective Type of field. The following numbers in a given tuple refer to the boundary at the respective (X_LEFT, X_RIGHT, Y_FRONT, Y_BACK, Z_TOP, Z_BOTTOM) boundaries, that refer to the X, Y, Z boundaries in order. The boundary condition is specified by numbers, 0:FREE, 1: Neumann, 2: Dirichlet, 3: Periodic, 4: Complex, where for the DIRICHLET boundary condition the respective boundary can also take in a specified value which can be specified in the following key: BOUNDARY_VALUE. If the key is not present, the default remains the one that is initialized at the start of the simulation and is left unchanged. Further, if for any direction, for eg: X, one of the extremeties is specified as a PERIODIC boundary, then the other X-boundary will also be initialized as PERIODIC irrespective of the entry in the tuple.

```
#BOUNDARY_VALUE = {Type, Value X+, Value X-, Value Y+, Value Y-,
    ↪ Value Z+, Value Z-}
BOUNDARY_VALUE = {phi, 0, 0, 0, 0, 0, 0};
BOUNDARY_VALUE = {mu, 0, 0, 0, 0, 0, 0};
BOUNDARY_VALUE = {c, 0, 0, 0, 0, 0, 0};
BOUNDARY_VALUE = {T, 0, 0, 0, 0, 0, 0};
```

- This key corresponds to a specific value that needs to be specified on a boundary which will be held constant during the duration of the simulation. The definition of this key should follow the BOUNDARY specification and follows the same type of definition, except here the tuple Value X_LEFT, Value X_RIGHT, Value Y_FRONT, Value Y_BACK, Value Z_TOP, Value Z_BOTTOM, refer to values on the respective boundaries. The values in this tuple will only be utilized if the respective boundary condition on that boundary is DIRICHLET. By default, the value will be treated as the same as the one that is initialized at the start of the simulation, which will be utilized if this key is not present.

## 2.4 Number of iterations, smoothing time-steps and writing interval

```
#Running and saving information
NTIMESTEPS = 10000;
NSMOOTH = 10;
SAVET = 1000;
STARTTIME = 10000;
RESTART = 1;
numworkers = 0;
```

- NTIMESTEPS: Total number of iterations that you wish the solver to run. This is not the total time
- NSMOOTH: The number of pre-conditioning steps for smoothening sharp phase-field profiles that are initialised at the start of the simulation
- SAVET: Writing interval, i.e, frequency of writing files of the respective fields
- RESTART : This key tells the solver to restart by reading in the files corresponding to STARTTIME. If either of the keys STARTTIME/RESTART are non-zero, the code will restart after reading in the files corresponding to the value STARTTIME. If STARTTIME = 0 and RESTART = 1, it gives the possibility to start from an already initialized file. This is useful when the filling operations for initialization are time-consuming.
- Number of workers with which the simulation was executed previously. This is required only for the Grand-potential based MPI solvers.

## 2.5 Material parameters

```
##Gas constant and molar volume
R = 1.0;
V = 1.0;
```

- The values "R" and "V" will refer to the gas constant and the molar volume respectively

```
#DIFFUSIVITY={Diagonal:0/1, phase, 11,22,33..(K-1) diagonal
    ↪ elements, 12, 13, 23...(rest of the elements; rowwise)}
DIFFUSIVITY = {1, 0, 1};
DIFFUSIVITY = {1, 1, 1}
```

- The DIFFUSIVITY key refers to the inter-diffusivity matrix which has the tuple in the following form. The first element can taken in values 0/1, "1" refers to as a diagonal matrix and "0" is a full matrix.

- The following element refers to the phase number referring to the phases in the list PHASES. This can take values from 0 to NUMPHASES-1.
- The following elements are the values in the inter-diffusivity matrix. The first elements are the diagonal terms in the matrix, while the following elements correspond rowwise to the off-diagonal terms in the diffusivity matrix.
- If the first element in the tuple is "1", i.e. the matrix is diagonal irrespective of the number of entries in the tuple only the entries corresponding to the diagonal elements in the matrix will be read in.

```
##GAMMA = {12, 13, 14, 23, 24...}
GAMMA = {1.0};
```

- The GAMMA key refers to the interfacial energy $\gamma_{\alpha\beta}$ between the phases $\alpha\beta$. The elements in tuple correspond to all combination of phases forming the interfaces from the list of phases in PHASES numbered from 0 to NUMPHASES-1.
- The elements are numbered in the order $12, 13, 14, 23, 24 \ldots N(N-1)$, $N = NUMPHASES$ where "12" corresponds to the value of the interfacial energy between phase 1 and 2; $\gamma_{12}$.
- In the tuple only combinations $\alpha\beta$ are included where $\alpha < \beta$ as the value of the interfacial energy of the $\alpha\beta$ interface is $\gamma_{\alpha\beta}$ which is the same as $\gamma_{\beta\alpha}$.
- Therefore, the total number of elements in the tuple is $\frac{N(N-1)}{2}$.

```
#EIGEN_STRAIN = {phase, exx, eyy, ezz, eyz, exz, exy};
EIGEN_STRAIN = {0, 0.01, 0.01, 0.0, 0.0, 0.0, 0.0};
EIGEN_STRAIN = {1, 0.01, 0.01, 0.0, 0.0, 0.0, 0.0};
```

- The EIGEN_STRAIN key refers to the eigen-strain tensor in a given phase. The information about the elements of the eigen-strain are derived from the elements in the tuple.
- The first element refers to the phase number in the list PHASES and can take in values from 0 to NUMPHASES-1.
- The following elements in tuple are mapped to the eigen-strain matrix in the following order $exx, eyy, ezz, eyz, exz, exy$.
- This is an important key required for problems where there are coherent interfaces and the phase transformation is coupled with the stress distribution arising due to coherency strains at the interface.

```
#VOIGT_ISOTROPIC = {phase, c11, c12, c44};
VOIGT_ISOTROPIC = {0, 270, 187.5, 125.0};
VOIGT_ISOTROPIC = {1, 270, 187.5, 125.0};
```

- VOIGT_ISOTROPIC is the key that refers to the elastic stiffness properties in the Voigt notation for an isotropic material.
- The first element in the tuple refers to the phase which will be a number from 0 to NUMPHASES-1.
- The following elements are the values $C11, C12, C44$ in order.

```
#VOIGT_CUBIC = {phase, c11, c12, c44};
VOIGT_CUBIC = {0, 270, 187.5, 125.0};
VOIGT_CUBIC = {1, 270, 187.5, 125.0};
```

- VOIGT_CUBIC is the key that refers to the elastic stiffness properties in the Voigt notation for a cubic material.
- The first element in the tuple refers to the phase which will be a number from 0 to NUMPHASES-1
- The following elements are the values $C11, C12, C44$ in order

```
#VOIGT_TETRAGONAL = {phase, c11, c12, c13, c33, c44, c66};
```

- VOIGT_TETRAGONAL is the key that refers to the elastic stiffness properties in the Voigt notation for a tetragonal material.
- The first element in the tuple refers to the phase which will be a number from 0 to NUMPHASES-1.
- The following elements are the values $C11, C12, C13, C33, C44, C66$ in order.

```
##FILEWRITING and OUTPUTTING TO SCREEN
## WRITEFORMAT ASCII/BINARY
##TRACK_PROGRESS: interval of writing out the progress of the
    ↪ simulation to stdout.
WRITEFORMAT = ASCII;
TRACK_PROGRESS = 10;
```

- The key WRITEFORMAT mentions the type of the output files to be written. The possible values are ASCII or BINARY that are self-explanatory.
- For the case when the type chosen is BINARY, the format is BIG-ENDIAN such that it matches the BINARY type required for PARAVIEW.
- TRACK_PROGRESS is a key that will inform the solver about the frequency with which the progress of the simulation will be written to stdout.
- The number can be anything other than 0.

## 2.6 Model specific parameters

The following parameters correspond to multicomponent, multiphase model of precipitate growth and coarsening in the solid state. These are as follows:

- ELAST_INT: Flag to enable/disable elastic stress contribution in the Allen-Cahn equation that governs the evolution of the phase-field variables. If ELAST_INT = 0, it will be disabled. A value of 1 will enable the elastic effects.
- `relax_coeff` or $L_\phi$: Relaxation coefficient used in Allen-Cahn equations governing evolution of phase-field variables $\phi_\alpha$;
- $c_0$: supersaturation in the matrix phase;
- $f_0^\alpha$: bulk free energy coefficients for each phase $\alpha = 1, \ldots, N$. If we assume simple parabolic free energy density for each phase $\alpha$, $f_0^\alpha$ denotes the curvature of the parabola. At any given temperature, the curvatures of these parabolas $f_0^\alpha$ can be obtained from CALPHAD to represent bulk free energy densities of real alloy systems.
- $\bar{\alpha}$: a constant in the KKS model associated with equilibrium phase-field profile. The constant is used to obtain gradient energy coefficient $\kappa_{\alpha\beta}$ and potential barrier height $\omega_{\alpha\beta}$ from a given interfacial energy and interfacial width according to
- $\lambda$: interfacial width

  Interfacial energy $\gamma_{\alpha\beta}$ and interfacial width $\lambda_{\alpha\beta}$ are inputs to the model. From these inputs we obtain gradient energy coefficient $\kappa_{\alpha\beta}$ and potential barrier height as

$$\omega_{\alpha\beta} = 6\alpha\frac{\gamma_{\alpha\beta}}{2\lambda_{\alpha\beta}}, \tag{1}$$

$$\kappa_{\alpha\beta} = \frac{3}{2\bar{\alpha}}\gamma_{\alpha\beta}(2\lambda_{\alpha\beta}). \tag{2}$$

- In these simulations of growth and coarsening of precipitate phases in the matrix phase, we use a scaled composition $c$ instead of actual composition $x$. For example, if we consider two phases $\alpha$ and $\beta$ with equilibrium compositions $x_{eq}^\alpha$ and $x_{eq}^\beta$ at a given temperature, the relation between $c$ and $x$ is given as

$$c = \frac{x - x_{eq}^\alpha}{x_{eq}^\beta - x_{eq}^\alpha}.$$

  Thus, the values of scaled composition $c$ can assume values smaller than zero or greater than unity, while the true composition $x$ lies between $[0, 1]$.

The order parameter fields $\phi_\alpha$ $\alpha = 1, \ldots, N$ are used to describe the distribution of N phases in the system and follow the constraint

$$\phi_\alpha \in [0, 1]; \ \sum_{\alpha=1}^{N} \phi_\alpha = 1$$

The total free energy $\mathcal{F}$ of the system is given as a sum of chemical and elastic contributions:

$$\mathcal{F} = \mathcal{F}_{\mathrm{c}} + \mathcal{F}_{\mathrm{e}}, \tag{3}$$

where $\mathcal{F}_{\mathrm{c}}$ represents the chemical free energy, and $\mathcal{F}_{\mathrm{e}}$ denotes the elastic free energy. We describe each energy contribution below.

### 2.6.1 Chemical free energy

The chemical free energy of the N-phase, N-component system is expressed as:

$$\mathcal{F}_{\mathrm{c}} = \int_V \Big[ f\left(c(\boldsymbol{r}), \phi(\boldsymbol{r})\right)) + \boldsymbol{\kappa} : (\boldsymbol{\nabla}\phi \otimes \boldsymbol{\nabla}\phi) \Big] \mathrm{d}V, \tag{4}$$

where $f\left(c(\boldsymbol{r}), \phi(\boldsymbol{r})\right)$ is the bulk chemical free energy density, $\boldsymbol{\kappa}$ is the second-rank gradient energy coefficient tensor. For cubic crystals, $\boldsymbol{\kappa} = \kappa \boldsymbol{I}$, where $\kappa$ is the scalar gradient energy coefficient. The bulk chemical free energy density, $f\left(c_i(\mathbf{r}), \phi_\alpha(\mathbf{r})\right))$, is represented as:

$$f\left(c_i(\boldsymbol{r}), \phi_\alpha(\boldsymbol{r})\right)) = \sum_{\alpha=1}^{N} h(\phi_\alpha) f^\alpha(c_i^\alpha) + \sum_{\alpha=1}^{N} f_{\mathrm{dw}}(\phi_\alpha(\boldsymbol{r})), \tag{5}$$

where $h(\phi_\alpha) = \phi_\alpha^3(6\phi_\alpha^2 - 15\phi_\alpha + 10)$ is a pairwise interpolation function between phases $\alpha = 1, \ldots, N$ and $\beta \neq \alpha$ such that $h(\phi_\alpha = 0) = 0$, $h(\phi_\alpha = 1) = 1$, $h'(0) = h'(1) = 0$ and $0 < h(\phi_\alpha) < 1$ for $0 < \phi_\alpha < 1$, $f^\alpha(c_i^\alpha)$ $\alpha = 1, \ldots, N$ represents chemical free energies of the designated phases, $f_{\mathrm{dw}}(\phi_\alpha(\boldsymbol{r}))$ is given by a set of double-well potential functions setting the potential barrier between $N$ phases.

Bulk free energies $f^\alpha$ $\alpha = 1, \ldots, N$ can be described using CALPHAD free energies. Here, we describe the chemical free energy density of each phase as simple parabolic free energies. The curvatures of these parabolas $f_0^\alpha$ $\alpha = 1, \ldots, N$ can be obtained from CALPHAD to represent real alloy systems.

Let us take the example of two coexisting phases $\alpha$ and $\beta$, The bulk free energy densities of $\alpha$ and $\beta$ phases are expressed as

$$f^\alpha\Big(c_\alpha(\boldsymbol{r})\Big) = \frac{f_0^\alpha}{V_{\mathrm{m}}} \Big[ c_\alpha(\boldsymbol{r}) - c_\alpha^{\mathrm{e}} \Big]^2, \tag{6}$$

$$f^\beta\Big(c_\beta(\boldsymbol{r})\Big) = \frac{f_0^\beta}{V_{\mathrm{m}}} \Big[ c_\beta(\boldsymbol{r}) - c_\beta^{\mathrm{e}} \Big]^2, \tag{7}$$

where $f_0^\alpha / V_{\mathrm{m}}$ and $f_0^\beta / V_{\mathrm{m}}$ correspond to curvatures of bulk free energy densities of the designated phases, $V_{\mathrm{m}}$ is the molar volume, $c_\alpha^{\mathrm{e}}$ and $c_\beta^{\mathrm{e}}$ represent the equilibrium compositions of matrix ($\alpha$) and precipitate ($\beta$) phases, respectively. The double-well potential $f_{\mathrm{dw}}(\phi(\boldsymbol{r}))$ is then expressed as

$$f_{\mathrm{dw}}(\phi(\boldsymbol{r})) = \omega\, \phi^2(\boldsymbol{r}) \left(1 - \phi(\boldsymbol{r})\right)^2, \tag{8}$$

where $\omega$ represents the barrier height of the potential.

### 2.6.2 Elastic free energy

The elastic free energy of an elastically inhomogeneous, coherent two-phase alloy system is given as follows:

$$\mathcal{F}_{\mathrm{e}} = \frac{1}{2}\int_V \Big[\boldsymbol{\sigma}(\boldsymbol{r}) : \boldsymbol{\varepsilon}^{\mathrm{e}}(\boldsymbol{r})\Big]\,\mathrm{d}V, \tag{9}$$

where $\boldsymbol{\sigma}(\boldsymbol{r})$ denotes the stress field and $\boldsymbol{\varepsilon}^{\mathrm{e}}(\boldsymbol{r})$ denotes the elastic strain field defined at a point $\boldsymbol{r}$ in the system. Assuming Hookean description of elasticity, $\boldsymbol{\sigma}(\boldsymbol{r})$ can be written as

$$\boldsymbol{\sigma}(\boldsymbol{r}) = \mathbb{C}\,(\boldsymbol{r}) : \boldsymbol{\varepsilon}^{\mathrm{e}}(\boldsymbol{r}), \tag{10}$$

where $\mathbb{C}(\boldsymbol{r})$ denotes the position-dependent elastic stiffness tensor. $\mathbb{C}(\boldsymbol{r})$ can be expressed as a sum of homogeneous contribution, $\mathbb{C}^0$, and position-dependent contribution, $\mathbb{C}'(\boldsymbol{r})$:

$$\mathbb{C}(\boldsymbol{r}) = \mathbb{C}^0 + \mathbb{C}'(\boldsymbol{r}) = \mathbb{C}^0 + \Delta\mathbb{C}\,b(\phi), \tag{11}$$

where $b(\phi) = 2h(\phi) - 1$ is a scalar function of $\phi(\boldsymbol{r})$, $\Delta\mathbb{C} = \dfrac{1}{2}(\mathbb{C}^\beta - \mathbb{C}^\alpha)$. $\mathbb{C}^\alpha$ and $\mathbb{C}^\beta$ are the elastic constants of the designated phases.

The elastic strain tensor field is given as

$$\boldsymbol{\varepsilon}^{\mathrm{e}}(\boldsymbol{r}) = \boldsymbol{\varepsilon}(\boldsymbol{r}) - \boldsymbol{\varepsilon}^0(\boldsymbol{r}), \tag{12}$$

where $\boldsymbol{\varepsilon}(\boldsymbol{r})$ is the total strain tensor field, and $\boldsymbol{\varepsilon}^0(\boldsymbol{r})$ represents the eigenstrain tensor field. The dilatational eigenstrain field, $\boldsymbol{\varepsilon}^{\mathbf{0}}(\boldsymbol{r})$, is written as

$$\boldsymbol{\varepsilon}^{\mathbf{0}}(\boldsymbol{r}) = \epsilon^*\mathbf{I}a(\phi) \tag{13}$$

where $a(\phi) = \phi \mathrm{or} h(\phi)$, $\epsilon^*$ is the strength of lattice mismatch (misfit strain), and $\mathbf{I}$ is the second-rank identity tensor.

Note that this code uses homogeneous modulus approximation, i.e., $\mathbb{C}(\boldsymbol{r}) = \mathbb{C}^0$.

## 2.7 Kinetics

Allen-Cahn equation and diffusion equation govern the spatiotemporal evolution of field variables $\phi(\boldsymbol{r}, t)$ and $c(\boldsymbol{r}, t)$. The elastic fields are obtained at each step of evolution by solving the mechanical equilibrium equation $\boldsymbol{\nabla}\cdot\boldsymbol{\sigma}(\boldsymbol{r}) = \mathbf{0}$, where $\boldsymbol{\sigma}(\boldsymbol{r})$ is expressed using Khachaturyan's microelasticity theory.

### 2.7.1 Allen-Cahn equation

The temporal evolution of order parameter field $\phi(\boldsymbol{r}, t)$ is given as

$$\frac{\partial\phi(\boldsymbol{r}, t)}{\partial t} = -L_\phi\frac{\delta\mathcal{F}}{\delta\phi} = -L_\phi\left[\frac{\delta\mathcal{F}_{\mathrm{c}}}{\delta\phi} + \frac{\delta\mathcal{F}_{\mathrm{e}}}{\delta\phi}\right], \tag{14}$$

where $L_\phi$ is the relaxation coefficient. The variational derivative $\frac{\delta \mathcal{F}_c}{\delta \phi}$ is given as

$$\frac{\delta \mathcal{F}_{\mathrm{c}}}{\delta \phi} = f'_{\mathrm{dw}}(\phi(\boldsymbol{r})) - 2\epsilon^2 \nabla^2 \phi(\boldsymbol{r}) + 2\Gamma_{\mathrm{I}} \nabla^4 \phi(\boldsymbol{r}) + 2\Gamma_{\mathrm{A}} \sum_{i=1}^{3} \nabla_i^4 \phi(\boldsymbol{r}) - h'(\phi) \left[ f^{\beta}\left(c_\beta(\boldsymbol{r})\right) - f^{\alpha}\left(c_\alpha(\boldsymbol{r})\right) - \left(c_\beta(\boldsymbol{r}) - c_\alpha(\boldsymbol{r})\right) \frac{\partial f^{\alpha}(c_\alpha(\boldsymbol{r}))}{\partial c_\alpha} \right]. \quad (15)$$

The variation of $\mathcal{F}_{\mathrm{e}}$ with respect to $\phi$ is given as

$$\frac{\delta \mathcal{F}_{\mathrm{e}}}{\delta \phi} = \frac{1}{2} h'(\phi) \left\{ \boldsymbol{\varepsilon}(\boldsymbol{r}) - \boldsymbol{\varepsilon}^0(\boldsymbol{r}) \right\} : \Delta \mathbb{C} : \left\{ \boldsymbol{\varepsilon}(\boldsymbol{r}) - \boldsymbol{\varepsilon}^0(\boldsymbol{r}) \right\} - \left\{ \boldsymbol{\varepsilon}(\boldsymbol{r}) - \boldsymbol{\varepsilon}^0(\boldsymbol{r}) \right\} : \mathbb{C}(\boldsymbol{r}) : \boldsymbol{\varepsilon}^{0'}(\boldsymbol{r}). \quad (16)$$

### 2.7.2 Diffusion equation

The gradient of the effective chemical potential (diffusion potential) $\widetilde{\mu}$ governs the net flux of solute atoms $\boldsymbol{J}$:

$$\boldsymbol{J} = -M \boldsymbol{\nabla} \widetilde{\mu}, \quad (17)$$

where $M$ is the mobility. Invoking the law of conservation of mass, we obtain the temporal evolution of local composition field $c(\mathbf{r})$ as:

$$\frac{\partial c(\mathbf{r}, t)}{\partial t} = -\boldsymbol{\nabla} \cdot \boldsymbol{J} = \boldsymbol{\nabla} \cdot M \boldsymbol{\nabla} \widetilde{\mu}, \quad (18)$$

where the local composition field $c(\boldsymbol{r}, t)$ is written as:

$$c(\boldsymbol{r}, t) = h(\phi) c_\beta(\boldsymbol{r}) + (1 - h(\phi)) c_\alpha(\boldsymbol{r}). \quad (19)$$

The chemical potential $\widetilde{\mu}$ can be expressed as:

$$\widetilde{\mu} = \frac{\partial f\left(\phi(\boldsymbol{r}), c(\boldsymbol{r})\right)}{\partial c(\boldsymbol{r})}. \quad (20)$$

We represent $f_{cc} = \frac{\partial \widetilde{\mu}}{\partial c(\boldsymbol{r})} = \frac{\partial^2 f\left(\phi(\boldsymbol{r}), c(\boldsymbol{r})\right)}{\partial c^2(\boldsymbol{r})}$, $f_{c\phi} = \frac{\partial \widetilde{\mu}}{\partial \phi(\boldsymbol{r})} = \frac{\partial^2 f\left(\phi(\boldsymbol{r}), c(\boldsymbol{r})\right)}{\partial c(\boldsymbol{r})\, \partial \phi(\boldsymbol{r})}$, and $\frac{f_{c\phi}}{f_{cc}} = h'(\phi)(c_\alpha(\boldsymbol{r}) - c_\beta(\boldsymbol{r}))$. From Eqns. (17) and (18), we yield

$$\begin{aligned}
\frac{\partial c(\boldsymbol{r}, t)}{\partial t} &= \boldsymbol{\nabla} \cdot \left[ M \boldsymbol{\nabla} \left( \frac{\partial f\left(\phi(\boldsymbol{r}), c(\boldsymbol{r})\right)}{\partial c(\boldsymbol{r})} \right) \right], && (21)\\
&= \boldsymbol{\nabla} \cdot \left[ \frac{D}{f_{cc}} \boldsymbol{\nabla} \left( \frac{\partial f\left(\phi(\boldsymbol{r}), c(\boldsymbol{r})\right)}{\partial c(\boldsymbol{r})} \right) \right], && (22)\\
&= \boldsymbol{\nabla} \cdot \left[ \frac{D}{f_{cc}} \Big( f_{cc} \boldsymbol{\nabla} c(\mathbf{r}, t) + f_{c\phi} \boldsymbol{\nabla} \phi(\mathbf{r}, t) \Big) \right], && (23)\\
&= \boldsymbol{\nabla} \cdot \Big[ D \boldsymbol{\nabla} c(\boldsymbol{r}, t) \Big] + \boldsymbol{\nabla} \cdot \left[ D \frac{f_{c\phi}}{f_{cc}} \boldsymbol{\nabla} \phi(\boldsymbol{r}, t) \right], && (24)\\
&= \boldsymbol{\nabla} \cdot \Big[ D \boldsymbol{\nabla} c(\boldsymbol{r}, t) \Big] + \boldsymbol{\nabla} \cdot \Big[ D\, h'(\phi)(c_\alpha - c_\beta) \boldsymbol{\nabla} \phi(\boldsymbol{r}, t) \Big], && (25)\\
&= D \nabla^2 c(\boldsymbol{r}, t) + D\, \boldsymbol{\nabla} \cdot \Big[ h'(\phi)(c_\alpha(\boldsymbol{r}) - c_\beta(\mathbf{r})) \boldsymbol{\nabla} \phi(\boldsymbol{r}, t) \Big]. && (26)
\end{aligned}$$

We use Eqn. (26) to evolve the local composition field $c(\boldsymbol{r}, t)$.

### 2.7.3 Equality of diffusion potential

In KKS formalism, the contribution of bulk free energies to the interfacial energy is nullified by equating the diffusion potentials of two phases. The equality of diffusion potential reads as

$$\frac{\partial f^{\alpha}\left(c_{\alpha}(\boldsymbol{r})\right)}{\partial c_{\alpha}(\boldsymbol{r})} = \frac{\partial f^{\beta}\left(c_{\beta}(\boldsymbol{r})\right)}{\partial c_{\beta}(\boldsymbol{r})}. \tag{27}$$

Since we describe the bulk free energy densities as parabolic functions of $c^{\alpha}$ and $c^{\beta}$, the solution to the Eqn. (27) is a simple algebraic relation given as

$$c_{\alpha}(\boldsymbol{r}) = \frac{f_0^{\beta}}{f_0^{\alpha}}\left(c_{\beta}(\boldsymbol{r}) - c_{\beta}^{\mathrm{e}}\right) + c_{\alpha}^{\mathrm{e}}, \tag{28}$$

$$c_{\beta}(\boldsymbol{r}) = \frac{f_0^{\alpha}}{f_0^{\beta}}\left(c_{\alpha}(\boldsymbol{r}) - c_{\alpha}^{\mathrm{e}}\right) + c_{\beta}^{\mathrm{e}}. \tag{29}$$

Using Eqn. (19), Eqns. (28) and (29) can be expressed as

$$c_{\alpha}(\boldsymbol{r}) = \frac{f_0^{\beta}c(\boldsymbol{r}) - \left[f_0^{\beta}c_{\beta}^{\mathrm{e}} - f_0^{\alpha}c_{\alpha}^{\mathrm{e}}\right]h(\phi)}{f_0^{\alpha}h(\phi) + f_0^{\beta}\left(1-h(\phi)\right)}, \tag{30}$$

$$c_{\beta}(\boldsymbol{r}) = \frac{f_0^{\alpha}c(\boldsymbol{r}) - \left[f_0^{\beta}c_{\beta}^{\mathrm{e}} - f_0^{\alpha}c_{\alpha}^{\mathrm{e}}\right]\left(1-h(\phi)\right)}{f_0^{\alpha}h(\phi) + f_0^{\beta}\left(1-h(\phi)\right)}. \tag{31}$$

Eqns. (30) and (31) can be used to obtain phase composition fields $c_{\alpha}(\boldsymbol{r})$ and $c_{\beta}(\boldsymbol{r})$ as functions of order parameter $\phi(\boldsymbol{r})$ and local composition $c(\boldsymbol{r})$.

## 2.8 Numerical implementation

We implement the semi-implicit Fourier spectral method to solve Eqns. (14) and (26). We solve the mechanical equilibrium in Fourier space.

### 2.8.1 Numerical implementation of Allen-Cahn equation

Spatial Fourier transform of both sides of Eqn. (14) yield

$$\frac{\partial\widetilde{\phi}(\boldsymbol{k},t)}{\partial t} = -L_{\phi}\widetilde{g}(\boldsymbol{k},t) - L_{\phi}\zeta\widetilde{\phi}(\boldsymbol{k},t) - L_{\phi}\widetilde{\mu}_{\mathrm{el}}(\boldsymbol{k},t), \tag{32}$$

where $g(\boldsymbol{r},t) = f'_{\mathrm{dw}}(\phi) - h'(\phi)\left[f^{\beta}\left(c_{\beta}(\boldsymbol{r})\right) - f^{\alpha}\left(c_{\alpha}(\boldsymbol{r})\right) - \left(c_{\beta}(\boldsymbol{r}) - c_{\alpha}(\boldsymbol{r})\right)\frac{\partial f^{\alpha}(c_{\alpha}(\boldsymbol{r}))}{\partial c_{\alpha}}\right]$, $\mu_{\mathrm{el}}(\boldsymbol{r},t) = \left[\frac{\delta\mathcal{F}_{\mathrm{e}}}{\delta\phi}\right]$, $\zeta = 2\epsilon^2|\boldsymbol{k}|^2 + 2\Gamma_{\mathrm{I}}|\boldsymbol{k}|^4 + 2\Gamma_{\mathrm{A}}(k_x^4 + k_y^4 + k_z^4)$. $\widetilde{\phi}(\boldsymbol{k},t)$, $\widetilde{\mu}_{\mathrm{el}}(\boldsymbol{k},t)$, and $\widetilde{g}(\boldsymbol{k},t)$ are the Fourier transforms of $\phi(\boldsymbol{r},t)$, $\mu_{\mathrm{el}}(\boldsymbol{r},t)$ and $g(\boldsymbol{r},t)$, respectively. The discrete form after the semi-implicit scheme is

$$\frac{\widetilde{\phi}(\boldsymbol{k},t+\Delta t) - \widetilde{\phi}(\mathbf{k},t)}{\Delta t} = -L_{\phi}\widetilde{g}(\boldsymbol{k},t) - L_{\phi}\,\zeta\,\widetilde{\phi}(\boldsymbol{k},t+\Delta t) - L_{\phi}\widetilde{\mu}_{\mathrm{el}}(\boldsymbol{k},t). \tag{33}$$

Rearranging terms in above equation leads to

$$\widetilde{\phi}(\boldsymbol{k},t+\Delta t) = \frac{\widetilde{\phi}(\boldsymbol{k},t) - L_{\phi}\Delta t\,\left[\widetilde{g}(\boldsymbol{k},t) + \widetilde{\mu}_{\mathrm{el}}(\boldsymbol{k},t)\right]}{1 + L_{\phi}\,\Delta t\,\zeta}. \tag{34}$$

### 2.8.2 Numerical implementation of diffusion equation

Taking spatial Fourier transforms of Eqn. (26) on both sides, we get

$$\frac{\partial \widetilde{c}(\boldsymbol{k},t)}{\partial t} = -|\boldsymbol{k}|^2 D\widetilde{c}(\boldsymbol{k},t) + JD\,\boldsymbol{k}\cdot\widetilde{\boldsymbol{p}}(\boldsymbol{k},t), \tag{35}$$

where $\boldsymbol{p}(\boldsymbol{r},t) = h'(\phi)(c_\alpha(\boldsymbol{r}) - c_\beta(\boldsymbol{r}))\boldsymbol{\nabla}\phi(\boldsymbol{r},t)$ and $J = \sqrt{-1}$. $\widetilde{c}(\boldsymbol{k},t)$ and $\widetilde{\mathbf{p}}(\boldsymbol{k},t)$ are Fourier transforms of $c(\boldsymbol{r},t)$ and $\boldsymbol{p}(\boldsymbol{r},t)$, respectively. Following semi-implicit discretization, Eqn. (35) reads as

$$\frac{\widetilde{c}(\boldsymbol{k},t+\Delta t) - \widetilde{c}(\boldsymbol{k},t)}{\Delta t} = -|\boldsymbol{k}|^2 D\widetilde{c}(\boldsymbol{k},t+\Delta t) + JD\,\boldsymbol{k}\cdot\widetilde{\boldsymbol{p}}(\boldsymbol{k},t). \tag{36}$$

Rearranging the terms of above equation, we obtain

$$\widetilde{c}(\boldsymbol{k},t+\Delta t) = \frac{\widetilde{c}(\boldsymbol{k},t) + JD\Delta t\,\boldsymbol{k}\cdot\widetilde{\boldsymbol{p}}(\boldsymbol{k},t)}{1+|\boldsymbol{k}|^2 D\Delta t}. \tag{37}$$

### 2.8.3 Spectral method for mechanical equilibrium

### 2.8.4 Zeroth-order solution

Assuming homogeneous modulus approximation, we solve the mechanical equilibrium equation as follows:

$$\boldsymbol{\nabla}\cdot\left(\mathbb{C}^0 : \boldsymbol{\nabla}\boldsymbol{u}(\boldsymbol{r})\right) = \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}^0(\boldsymbol{r}). \tag{38}$$

Taking Fourier transform on both sides of the equation above, we obtain

$$\widetilde{\boldsymbol{u}}(\boldsymbol{k}) = -J\mathbf{G}(\boldsymbol{\xi})\cdot\boldsymbol{z}^{\mathbf{0}}(\boldsymbol{k}), \tag{39}$$

where $J = \sqrt{-1}$, $\boldsymbol{k}$ is the position vector in reciprocal (Fourier) space, $\boldsymbol{\xi} = \dfrac{\boldsymbol{k}}{|\boldsymbol{k}|}$, $\mathbf{G}(\boldsymbol{\xi})$ is the Green tensor defined as $\mathbf{G}(\boldsymbol{\xi}) = \left[\mathbb{C}^0 : (\boldsymbol{\xi}\otimes\boldsymbol{\xi})\right]^{-1}$, $\boldsymbol{z}^{\mathbf{0}}(\boldsymbol{k}) = \widetilde{\boldsymbol{\sigma}}^0(\boldsymbol{k})\cdot\boldsymbol{k}$, $\widetilde{\boldsymbol{u}}(\boldsymbol{k})$ and $\widetilde{\boldsymbol{\sigma}}^0(\boldsymbol{k})$ represent the Fourier transforms of $\boldsymbol{u}(\boldsymbol{r})$ and $\boldsymbol{\sigma}^{\mathbf{0}}(\boldsymbol{r})$, respectively.

# 3 GPU implementation

We perform all our simulations using Nvidia GPUs, which provides a speedup for compute-intensive solvers. Using a single GPU, workers reported a speedup as large as 50 times compared to a single CPU. Here, we use CUDA-enabled Nvidia Tesla P100 and V100 GPUs. We utilize the *cuFFT* library from Nvidia CUDA toolkit to compute spatial Fast Fourier Transforms of phase field and composition fields.

A general CUDA program involves kernels to exploit thread-based parallelism where a grid of thread blocks is input to the kernel. Here, we employ a three-dimensional decomposition scheme where data decomposition at the grid level and block level is enforced. Fig. 1 (cyan color box) shows the three-dimensional decomposition of the simulation domain into a grid of thread blocks, known as coarse-grained decomposition. On the top of coarse-grained decomposition, each thread block comprises a three-dimensional array of threads (see violet color box in Fig. 1), known as fine-grained decomposition.

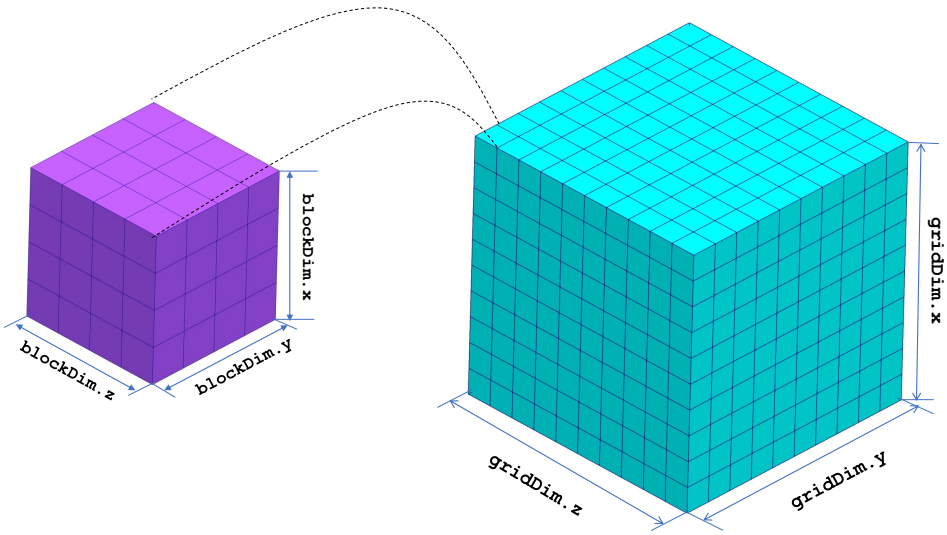


Figure 1: Schematic representation of three dimensional thread block and grid of blocks. The data block is partitioned into grid of blocks and each block consists of threads arranged in three dimensions. `gridDim.x`, `gridDim.y`, and `gridDim.z` are the number of thread blocks along $x$, $y$, and $z$ directions, respectively. `blockDim.x`, `blockDim.y`, and `blockDim.z` represent the number of threads in each block along $x$, $y$, and $z$ directions, respectively.

Every thread can be accessed using an index (`i`,`j`,`k`) which is given as

```
i = blockIdx.x * blockDim.x + threadIdx.x,
j = blockIdx.y * blockDim.y + threadIdx.y,
k = blockIdx.z * blockDim.z + threadIdx.z,
```

where (`blockIdx.x`,`blockIdx.y`,`blockIdx.z`) represent the index of a block in the grid and (`threadIdx.x`,`threadIdx.y`,`threadIdx.z`) denote the index of a thread in the thread block. Depending upon the GPU, the maximum number of threads in a block varies (e.g., Tesla V100 can have 1024 threads per block).