Phase-field Solver for Eutectic Transformation

Tanmay Dutta

November 10, 2021

Contents

This is a phase-field solver pertaining to three phases (α, β and liquid) for OpenFOAM based on finite-volume discretization. The solver can be decomposed into multiple domains for parallel runs. To carry out the simulations, the steps mentioned below can be followed.

1 2D lamellar eutectic growth

The first case is conducted by subjecting the melt to a positive moving thermal gradient ahead of the solid-liquid interface in 2D. The imposed temperature field as a function of thermal gradient in the x direction, G and pulling velocity, v is taken as:

$$T(x,t) = T_{initial} + G(x - vt) \tag{1}$$

1.1 Problem specification

The problem is defined as follows:

Solution domain The domain is considered to be two-dimensional in this problem. The rectangular domain consists of three phases with boundaries as shown in Figure 1 (Umate [(2021)]).

Governing equations The basic idea of the phase-field model consists in the description of the evolution of individual phases using the corresponding set of order parameters (the phase-field variables). For each phase,

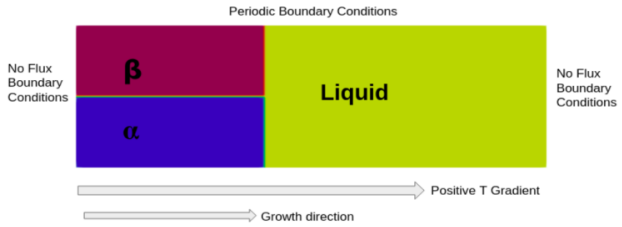

Figure 1: Figure 1: Two-dimensional geometry of the lamellar morphology.

the associated phase-field variable is equal to 1 inside the phase and vanishes to 0 outside the area occupied by the considered phase. The transition of phase fields from 1 to 0 at the phase boundaries is continuous and smooth, i.e.,the phase boundaries are represented by diffuse interfaces with a finite thickness. The three phases, α, β and liquid phase are described by three phase fields ϕ_i, with $i = \alpha$, β, liq, representing the volume fractions. The state of the binary eutectic system is described by the order parameter $\Phi = \{\phi_\alpha, \phi_\beta, \phi_{liq}\}$, the chemical potential μ, and the temperature T. The constraints on individual order parameters are:

$$\phi_i \, \epsilon \, [0,1] \; \forall \; i \sum_i \phi_i = 1 \tag{2}$$

- Phase evolution is determined by the phenomenological minimization of the grand-potential functional (Choudhury and Nestler [(2012)]). The evolution equations for three phases can be written as (Folch and Plapp [(2005)]):

$$\begin{aligned} \tau\epsilon\frac{\partial\phi_i}{\partial t} &= -\frac{2}{3}\left[-\epsilon\left(\nabla\cdot\frac{\partial a(\nabla\Phi)}{\partial\nabla\phi_i}\right)+\frac{1}{\epsilon}\frac{\partial w(\Phi)}{\partial\phi_i}+\frac{\partial\Psi(T,\mu,\Phi)}{\partial\phi_i}\right] \\ &+\frac{1}{3}\left[-\epsilon\left(\nabla\cdot\frac{\partial a(\nabla\Phi)}{\partial\nabla\phi_j}\right)+\frac{1}{\epsilon}\frac{\partial w(\Phi)}{\partial\phi_j}+\frac{\partial\Psi(T,\mu,\Phi)}{\partial\phi_j}\right] \\ &+\frac{1}{3}\left[-\epsilon\left(\nabla\cdot\frac{\partial a(\nabla\Phi)}{\partial\nabla\phi_k}\right)+\frac{1}{\epsilon}\frac{\partial w(\Phi)}{\partial\phi_k}+\frac{\partial\Psi(T,\mu,\Phi)}{\partial\phi_k}\right] \end{aligned} \tag{3}$$

where i, j and k are all different and $\epsilon \, \{\alpha, \beta, liq\}$; τ is the relaxation constant which influences the kinetics of the phase transformation at the interfaces.

- The double well potential $w(\Phi)$ can be written as:

$$w(\Phi) = \sum_i 9\gamma{\phi_i}^2(1-\phi_i)^2 \tag{4}$$

- The gradient energy density, $a(\nabla\Phi)$ has the form:

$$a(\nabla\Phi) = \sum_i \gamma|\nabla\phi_i|^2 \tag{5}$$

where γ controls the interfacial energy of the system and is known as surface energy density.

- The grand potential $\Psi(T, \mu, \phi)$ can be expressed as sum of grand potential of phases present in the system:

$$\Psi(T,\mu,\Phi) = \sum_i \Psi_i(T,\mu)h_i(\Phi) \tag{6}$$

where $\Psi_i(T, \mu)$ for the three phases are the following:

$$\Psi_\alpha(T,\mu) = -A\left(\frac{\mu - B_\alpha}{2A}\right)^2 + D_\alpha(T)\Psi_\beta(T,\mu) = -A\left(\frac{\mu - B_\beta}{2A}\right)^2 + D_\beta(T)\Psi_{liq}(T,\mu) = -A\left(\frac{\mu}{2A}\right)^2 \quad (7)$$

where A, B_i and D_i are derived following the consequence of equilibrium between the different phases described by the common tangent construction with parabolic approximation for free energy. These are given below:

$$B_\alpha(T) = 2A_{liq}c_{liq}^{eut} - 2A_\alpha c_\alpha^{eut} + (T - T^{eut})\left(\frac{2A_{liq}}{m_{liq-\alpha}} - \frac{2A_\alpha}{m_{\alpha-liq}}\right) B_\beta(T) = 2A_{liq}c_{liq}^{eut} - 2A_\beta c_\beta^{eut} + (T - T^{eut})\left(\frac{2A_{liq}}{m_{liq-\beta}} - \frac{2A_\beta}{m_{\beta-liq}}\right) B \quad (8)$$

where c_i^{eut} are the α, β and liquid phase compositions at the eutectic temperature, T^{eut}. The liquidus slopes for $\alpha - liq$ phase equilibrium is denoted by $m_{liq-\alpha}$ while the corresponding solidus slope by $m_{\alpha-liq}$ and similar notation is followed for the $\beta - liq$ phase equilibrium.

- In equation (6), $h_i(\Phi)$ is an interpolation function given as:

$$h_i = \frac{{\phi_i}^2}{4}[15(1-\phi_i)\{1 + \phi_i - (\phi_k - \phi_j)^2\} + \phi_i(9{\phi_i}^2 - 5)] \quad (9)$$

These interpolation functions make sure that at the interface of phases i and j, the phase field for phase k remains zero.

- The phase-field equations for three phases after incorporating above equations can be given as:

$$\begin{aligned} \tau\epsilon\frac{\partial\phi_i}{\partial t} &= -\frac{2}{3}\left[-2\gamma\epsilon\nabla^2\phi_i + \frac{1}{\epsilon}\frac{\partial w(\Phi)}{\partial\phi_i} + \Psi_i(T,\mu)\frac{\partial h_i(\Phi)}{\partial\phi_i}\right] \\ &+ \frac{1}{3}\left[-2\gamma\epsilon\nabla^2\phi_j + \frac{1}{\epsilon}\frac{\partial w(\Phi)}{\partial\phi_j} + \Psi_j(T,\mu)\frac{\partial h_j(\Phi)}{\partial\phi_j}\right] \\ &+ \frac{1}{3}\left[-2\gamma\epsilon\nabla^2\phi_k + \frac{1}{\epsilon}\frac{\partial w(\Phi)}{\partial\phi_k} + \Psi_k(T,\mu)\frac{\partial h_k(\Phi)}{\partial\phi_k}\right] \end{aligned} \quad (10)$$

- These three equations for the phase fields corresponding to the liquid, α and β phases are solved in a coupled manner along with mass balance equation which takes the form:

$$\begin{aligned} &\left[\sum_i \frac{\partial c_i}{\partial\mu}h_i(\Phi)\right]\frac{\partial\mu}{\partial t} + \sum_i c_i\left(\sum_n \frac{\partial h_i}{\partial\phi_n}\frac{\partial\phi_n}{\partial t}\right) \\ &= \nabla\cdot\left(D\phi_{liq}\frac{\partial c_{liq}(T,\mu)}{\partial\mu}\nabla\mu\right) - \nabla\cdot j_{at} \end{aligned} \quad (11)$$

The slope of c–μ curves is expressed according to the parabolic approximation for free energy in the solver source code:

$$\frac{\partial c_i(T,\mu)}{\partial\mu} = \frac{1}{2A} \quad (12)$$

- In this case diffusivity in solid phase is assumed to be negligible. This then becomes the case of one sided diffusion. For the case of one-sided diffusion, it has been shown in various previous works that there exists a thin-interface defect called solute trapping when simulations are performed with interface thicknesses, orders of magnitude larger than those of a real interface. The methodology proposed to correct this effect is the incorporation of an anti-trapping current j_{at} in the evolution equation of the chemical potential. The anti-trapping term is incorporated as an additional flux of solute from the solid to the liquid in the normal direction to the interface (Karma [(2001)]). To make sure that the

anti-trapping current appears in the first-order correction to the chemical potential, the anti-trapping current is formulated into the following form:

$$j_{at} = j_{at}^{\alpha \Longrightarrow liq}\left(-\frac{\nabla\phi_\alpha}{|\nabla\phi_\alpha|}\cdot\frac{\nabla\phi_{liq}}{|\nabla\phi_{liq}|}\right) + j_{at}^{\beta \Longrightarrow liq}\left(-\frac{\nabla\phi_\beta}{|\nabla\phi_\beta|}\cdot\frac{\nabla\phi_{liq}}{|\nabla\phi_{liq}|}\right) \tag{13}$$

where $j_{at}^{i \Longrightarrow liq}$ is:

$$j_{at}^{i \Longrightarrow liq} = -\frac{\epsilon}{2\sqrt{2}}[c_{liq}(\mu) - c_i(\mu)]\frac{\partial\phi_i}{\partial t}\frac{\nabla\phi_i}{|\nabla\phi_i|} \tag{14}$$

All terms in the above equation are used in the nondimensional form, so ϵ is the nondimensional parameter related to the interface width and t is the nondimensional time.

- As the simulation progresses, the solid-liquid interface will choose an undercooling and will traverse with the temperature gradient with the imposed velocity. Since the simulation of solidification begins from an infinite reservoir of liquid, it becomes necessary to implement shifts in all the fields in the domain to ensure that the far field composition of the liquid remains the same as eutectic composition. This is analogous to having a camera frame at the solid liquid interface which is moving along with it into a virtually infinite liquid in the growth direction. This is implemented simply by 'rigidly convecting' all the fields. This requires adding the flux term $v \cdot \nabla\phi$, where v is the imposed velocity and ϕ is phase field parameter of all phases and also the chemical potential field. Since, v is a constant, all these fields will simply be translated as it is in the entire domain. This shifts are required to be performed in the growth direction, which is chosen to be x direction. So, each equation becomes:

$$\begin{aligned}
&\tau\epsilon\frac{\partial\phi_i}{\partial t} - v\frac{\partial\phi_i}{\partial x} = -\frac{2}{3}\left[-2\gamma\epsilon\nabla^2\phi_i + \frac{1}{\epsilon}\frac{\partial w(\Phi)}{\partial\phi_i} + \Psi_i(T,\mu)\frac{\partial h_i(\Phi)}{\partial\phi_i}\right] \\
&+ \frac{1}{3}\left[-2\gamma\epsilon\nabla^2\phi_j + \frac{1}{\epsilon}\frac{\partial w(\Phi)}{\partial\phi_j} + \Psi_j(T,\mu)\frac{\partial h_j(\Phi)}{\partial\phi_j}\right] \\
&+ \frac{1}{3}\left[-2\gamma\epsilon\nabla^2\phi_k + \frac{1}{\epsilon}\frac{\partial w(\Phi)}{\partial\phi_k} + \Psi_k(T,\mu)\frac{\partial h_k(\Phi)}{\partial\phi_k}\right]
\end{aligned} \tag{15}$$

and the chemical diffusion evolution equation becomes:

$$\begin{aligned}
&\left[\sum_i \frac{\partial c_i}{\partial\mu}h_i(\Phi)\right]\frac{\partial\mu}{\partial t} - v\frac{\partial c}{\partial x} + \sum_i c_i\left(\sum_n \frac{\partial h_i}{\partial\phi_n}\frac{\partial\phi_n}{\partial t}\right) \\
&= \nabla\cdot\left(D\phi_{liq}\frac{\partial c_{liq}(T,\mu)}{\partial\mu}\nabla\mu\right) - \nabla\cdot j_{at}
\end{aligned} \tag{16}$$

Since each of the fields is being moved in the direction opposite to growth direction, the temperature gradient is kept stationary and solid-liquid interface is expected to find its equilibrium undercooling.

- Using the relation $\frac{\partial\Psi(T,\mu,\Phi)}{\partial\mu} = -c$ and equations (6–9), ∇c is evaluated.

Boundary conditions

- Cyclic boundary condition is specified at the top and bottom planes, and zero-flux is specified at the remaining planes.

Initial conditions

- $\phi_i = 1.0$ inside the phase i and 0 inside phases j and k (with $i, j, k \in \{\alpha, \beta, liq\}$) (see Figure 1).
- $\mu = 1.0$ in the entire domain.

Physical properties The nondimensionalized physical properties for equal volume fractions of α and β phases are:

- Slope liquidus of $liq - \alpha$ ($m_{liq-\alpha}$) = Slope solidus of $\alpha - liq$ ($m_{\alpha-liq}$) = -0.5
- Slope liquidus of $liq - \beta$ ($m_{liq-\beta}$) = Slope solidus of $\beta - liq$ ($m_{\beta-liq}$) = 0.5
- Eutectic composition of liquid phase (c_{liq}^{eut}) = 0.5
- Eutectic composition of α phase (c_{α}^{eut}) = 0.2
- Eutectic composition of β phase (c_{β}^{eut}) = 0.8
- Thermal gradient (G) = 1.7e-5
- Velocity (v) = 0.001
- Diffusivity in liquid (D) = 1
- Eutectic temperature (T_{eut}) = 1.0
- Constant value from temperature profile ($T_{initial}$) = 0.93
- Relaxation coefficient (τ) = 0.288
- Interface energy parameter (γ) = 1
- Interface width parameter (ϵ) = 8

Solver name

- *eutectic*: An implementation for phase-field method to model eutectic transformation of two solid phases from liquid phase.

Case name

- *lamellar2D*, located in the *$FOAM_RUN/PhaseFieldSolverEutectic* directory.

The problem is solved using *eutectic*, located in the *$FOAM_RUN/PhaseFieldSolverEutectic* directory. The code-guide for this solver is given in the *$FOAM_RUN/PhaseFieldSolverEutectic/codeGuide* directory. Before proceeding further, it may be helpful for the user to become familiar with OpenFOAM documentation.

1.2 Pre-processing

To prepare case files and running the case, the user can change to the case directory:

cd $FOAM_RUN/PhaseFieldSolverEutectic/lamellar2D

1.2.1 Mesh generation

OpenFOAM solves the case in three dimensions by default but can be instructed to solve in two dimensions. Here, the mesh must be 1 cell layer thick. The domain consists of a rectangle of length in the growth direction (x), $3D/v$ and in the normal direction (y), $1.6D/v$. A domain of size 300 in the growth direction, if resolved in 150 grid points, is said to have a dx of 2. The . Large variations in fields can be expected near the interfaces, so the mesh will be graded to be smaller in this region. The graded mesh is achieved by merging three blocks with different refinement. The block structure is shown in Figure 2.

The entries in *blockMeshDict* located in the *system* directory for this case are as follows:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
```

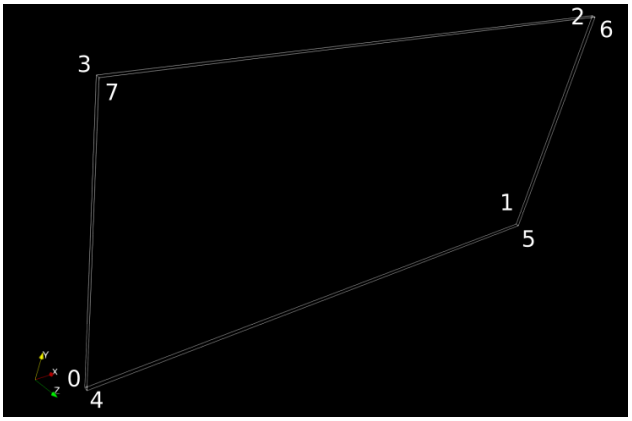

Figure 2: Figure 2: Block structure of the mesh.

```
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
convertToMeters 2;
x_min 0;
x_max0 90;
x_max1 120;
x_max2 150; //3 blocks in direction of growth
y_min 0;
y_max 80;
z_min 0;
z_max 2;

// should all be integers
lx #calc "$x_max0 - $x_min";
ly #calc "$y_max - $y_min";
lz #calc "$z_max - $z_min"; // should all be integers
vertices
(
    ($x_min $y_min $z_min)
    ($x_max0 $y_min $z_min)
    ($x_max0 $y_max $z_min)
    ($x_min $y_max $z_min)
    ($x_min $y_min $z_max)
    ($x_max0 $y_min $z_max)
    ($x_max0 $y_max $z_max)
    ($x_min $y_max $z_max)

    ($x_max0 $y_min $z_min)
    ($x_max1 $y_min $z_min)
    ($x_max1 $y_max $z_min)
    ($x_max0 $y_max $z_min)
    ($x_max0 $y_min $z_max)
```

```
    ($x_max1 $y_min $z_max)
    ($x_max1 $y_max $z_max)
    ($x_max0 $y_max $z_max)

    ($x_max1 $y_min $z_min)
    ($x_max2 $y_min $z_min)
    ($x_max2 $y_max $z_min)
    ($x_max1 $y_max $z_min)
    ($x_max1 $y_min $z_max)
    ($x_max2 $y_min $z_max)
    ($x_max2 $y_max $z_max)
    ($x_max1 $y_max $z_max)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) ($lx 80 1) simpleGrading (1 1 1)
    hex (8 9 10 11 12 13 14 15) (12 40 1) simpleGrading (1 1 1)
    hex (16 17 18 19 20 21 22 23) (6 20 1) simpleGrading (1 1 1)
// for dx = 0.5: simpleGrading (2 2 2) or convertToMeters = 0.5
);

edges
(
);

//================ FOR ZERO-FLUX BOUNDARY CONDITIONS ============
boundary
(
    floor
    {
         type cyclic;
         neighbourPatch ceiling;
        faces
        (
        (0 1 5 4)
        (8 9 13 12)
        (16 17 21 20)
        );
    }

    ceiling
    {
        type cyclic;
        neighbourPatch floor;
        faces
        (
        (2 3 7 6)
        (10 11 15 14)
        (18 19 23 22)
        );
    }
    sideSolid
    {
```

```
        type wall;
        faces
        (
            (0 3 2 1)
            (8 11 10 9)
            (16 19 18 17)
            (0 4 7 3)
            (4 5 6 7)
            (12 13 14 15)
            (20 21 22 23)
            (17 18 22 21)
        );
    }
    wall_merge0
    {
        type wall;
        faces
        (
            (1 2 6 5)
        );
    }
    wall_merge1
    {
        type wall;
        faces
        (
            (8 12 15 11)
        );
    }
    wall_merge2
    {
        type wall;
        faces
        (
            (9 10 14 13)
        );
    }
    wall_merge3
    {
        type wall;
        faces
        (
            (16 20 23 19)
        );
    }

);

mergePatchPairs
(
    (wall_merge0 wall_merge1)
    (wall_merge2 wall_merge3)
);
```

```
// ************************************************************************* //
```

The mesh is generated after accessing blockMesh entry within *$FOAM_RUN/PhaseFieldSolverEutectic/lamellar2D/Allrun*:

```
#!/bin/sh
cd ${0%/*} || exit 1    # Run from this directory

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

application=`getApplication`

runApplication blockMesh

cp 0/phi_alpha.orig 0/phi_alpha

cp 0/phi_beta.orig 0/phi_beta

cp 0/phi_liq.orig 0/phi_liq

cp 0/T.orig 0/T

cp 0/mu.orig 0/mu

runApplication setFields -dict system/setFieldsDict

runApplication decomposePar

#runApplication `getApplication`
runParallel `getApplication`

#runApplication reconstructPar


#------------------------------------------------------------------------------
```

The generated mesh is shown in Figure 3.

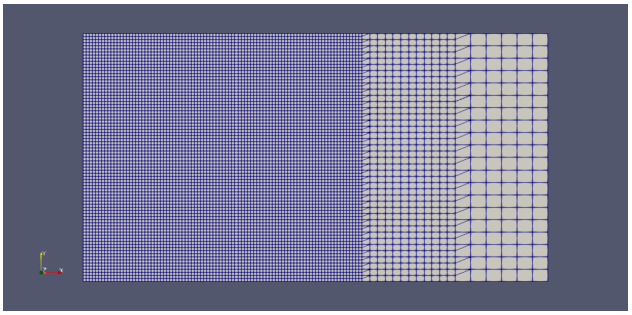

Figure 3: Figure 3: Generated 2D mesh.

The logs are stored into *log.blockMesh*.

1.2.2 Boundary and initial conditions

Once the mesh generation is complete, the user can look at this initial conditions set up for this case. The case is set up to start at time $t = 0$ s, so the initial field data is stored in a *0* sub-directory of the *freeGrowth* directory. The *0* sub-directory contains *phi_alpha.orig*, *phi_beta.orig*, *phi_liq.orig*, and *mu.orig*, one for each of the phase parameter ($\phi_\alpha, \phi_\beta, \phi_{liq}$) and chemical potential (μ) fields whose initial values and boundary conditions must be set. First, the file *phi_alpha.orig* can be examined:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      phi_alpha;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
    {
        type            cyclic;
    }
    ceiling
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
```

For this case, the boundary consists of planes split into patches named: (1) ceiling and floor for cyclic boundary at top and bottom planes; (2) sideSolid patch includes other planes of the 2D case and is given a zeroGradient boundary condition for ϕ_α, meaning "the normal gradient of ϕ_α is zero." The initial fields are set to uniform zero.

The user can similarly examine *0/phi_beta.orig*, *0/phi_liq.orig*, and the chemical potential field in the *0/mu.orig* file. The non-dimensionalized internal field is initialised as uniform zero. The boundary fields require the same boundary condition.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      phi_beta;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
    {
        type            cyclic;
    }
    ceiling
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      phi_liq;
```

```
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
    {
        type            cyclic;
    }
    ceiling
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      mu;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
    {
        type            cyclic;
```

```
    }
    ceiling
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
```

1.2.3 Setting initial field

A non-uniform initial condition is specified for the phase parameter, ϕ_i, where $\phi_i = 1.0$ inside the phase i and 0 inside phases j and k (with $i, j, k \epsilon \{\alpha, \beta, liq\}$). For chemical potential, $\mu = 1.0$ in the entire domain. The solid-liquid interface is set at $1.2D/v$ from the origin.

This is achieved by running the setFields utility. It requires a *setFieldsDict* dictionary, located in the *system* directory, whose entries for this case are shown below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue phi_alpha 0.0
    volScalarFieldValue phi_beta 0.0
    volScalarFieldValue phi_liq 0.0
    volScalarFieldValue T 1.0
    volScalarFieldValue mu 1.0

);

regions
(

    boxToCell
// boxToCell: work on all cells in a rectangular box defined with starting and end point coordinates
    {
    box (-6 -6 -6) (120 80 100);
```

```
// Box can be larger than domain; in general this avoids edge effects
    fieldValues
// NB: no semicolons at the end of the dictionary entries below!
    (
    volScalarFieldValue phi_alpha 1.0
    volScalarFieldValue phi_beta 0.0
    volScalarFieldValue phi_liq 0.0
    volScalarFieldValue T 1.0
    volScalarFieldValue mu 1.0
    );
    }


    boxToCell
// boxToCell: work on all cells in a rectangular box defined with starting and end point coordinates
    {
    box (-6 80 -6) (120 166 100);
// Box can be larger than domain; in general this avoids edge effects
    fieldValues
// NB: no semicolons at the end of the dictionary entries below!
    (
    volScalarFieldValue phi_alpha 0.0
    volScalarFieldValue phi_beta 1.0
    volScalarFieldValue phi_liq 0.0
    volScalarFieldValue T 1.0
    volScalarFieldValue mu 1.0
    );
    }


    boxToCell
// boxToCell: work on all cells in a rectangular box defined with starting and end point coordinates
    {
    box (120 -6 -6) (306 166 100);
// Box can be larger than domain; in general this avoids edge effects
    fieldValues
// NB: no semicolons at the end of the dictionary entries below!
    (
    volScalarFieldValue phi_alpha 0.0
    volScalarFieldValue phi_beta 0.0
    volScalarFieldValue phi_liq 1.0
    volScalarFieldValue T 1.0
    volScalarFieldValue mu 1.0
    );
    }


);


// ************************************************************************* //
```

The user can execute setFields through *$FOAM_RUN/PhaseFieldSolverEutectic/lamellar2D/Allrun*. The logs are stored into *log.setFields*.

1.2.4 Physical properties

The physical properties for the case are specified in *constant/transportProperties* dictionary. The entries of *transportProperties* dictionary are shown below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimx        dimx [0 1 0 0 0 0 0] 1; //Dimension of position
dimt        dimt [0 0 1 0 0 0 0] 1; //Dimension of time
ms_alpha    ms_alpha [0 0 0 0 0 0 0] -0.5; //Slope liq-alpha and alpha-liq
ms_beta     ms_beta [0 0 0 0 0 0 0] 0.5; //Slope liq-beta and beta-liq
c_eq_liq    c_eq_liq [0 0 0 0 0 0 0] 0.5; //Eutectic composition of liquid phase
c_eq_alpha  c_eq_alpha [0 0 0 0 0 0 0] 0.2; //Eutectic composition of alpha phase
c_eq_beta   c_eq_beta [0 0 0 0 0 0 0] 0.8; //Eutectic composition of beta phase
G           G [0 0 0 0 0 0 0] 1.0e-3; //Thermal gradient
v           v [0 0 0 0 0 0 0] 0.001; //Velocity
A           A [0 0 0 0 0 0 0] 1.0;
D           D [0 0 0 0 0 0 0] 1.0; //Diffusivity in liquid
T_eut       T_eut [0 0 0 0 0 0 0] 1.0; //Eutectic temperature
initial     initial [0 0 0 0 0 0 0] 0.89; //Constant value from temperature profile
tau         tau [0 0 0 0 0 0 0] 0.288; //Relaxation coefficient
gamma       gamma [0 0 0 0 0 0 0] 1.0; //Interface energy parameter
epsilon     epsilon [0 0 0 0 0 0 0] 8.0; //Interface width parameter
// ************************************************************************* //
```

The physical properties are read by *readTransportProperties.H* while running the case.

1.2.5 Time step control

Input data relating to the control of time, reading and writing of the solution data are read from the *controlDict* dictionary located in the *system* directory.

The run is started at time $t = 0$ to read field data from a directory named *0*. Therefore, the startFrom keyword is set to startTime and then the startTime keyword to be 0.

The end time can be considered to be the time taken for steady-state diffusion of each phase, which is found to be 1000 for this case. To specify this end time, the stopAt keyword is set to endTime and then the endTime keyword to 1000.

Next, the time step must be fixed which is represented by the keyword deltaT. To achieve temporal accuracy and numerical stability while reducing computational effort, deltaT is set to 0.1.

As the simulation progresses, results written at certain intervals of time can later be viewed with a post-processing package. The writeControl keyword can be set to the runTime option, where the time interval for writing is specified under the writeInterval keyword. The writeInterval can be set to 100. OpenFOAM creates a new directory named after the current time, e.g. 200, on each occasion that it writes a set of data containing the results for each field, ϕ_i and μ, into the time directories. For this case, the entries in the *controlDict* are shown below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

application     eutectic;

startFrom       startTime; //latestTime

startTime       0; // the actual time i.e. runtime.value() or no. of iterations*deltaT

stopAt          endTime;

endTime         1000.0; //actual time = total_iterations*deltaT

deltaT          0.2;

writeControl    runTime;

writeInterval   50.0;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;
```

```
// ************************************************************************* //
```

1.2.6 Discretisation schemes

In this case, the term $\nabla \cdot \left(D\phi_{liq} \frac{\partial c_{liq}(T,\mu)}{\partial \mu} \nabla \mu \right)$ in the chemical potential equation includes laplacian(D*phi_liq,mu). The laplacian is solved by fixing laplacianSchemes to Gauss linear corrected. The gradient and divergence are solved by fixing gradSchemes and divSchemes to Gauss linear.

The other discretised terms use commonly employed schemes so that the *fvSchemes* dictionary entries should therefore be:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(phi)         Gauss linear;
}

divSchemes
{
    default         Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}
```

```
snGradSchemes
{
    default         corrected;
}


// ************************************************************************* //
```

1.2.7 Solver settings

In the *fvSolution*, the solver tolerance should be set to 10^{-6} for this case. The solver relative tolerance, denoted by relTol, is set to 0.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


solvers
{
    mu
    {
        solver                   smoothSolver;
        preconditioner            DIC;
        smoother                symGaussSeidel;
        tolerance                1e-6;
        relTol                    0.0;
    }

    phi_alpha
    {
    solver                    smoothSolver;
        preconditioner            DIC;
        smoother                symGaussSeidel;
        tolerance                1e-6;
        relTol                    0.0;
    }

    phi_beta
    {
    solver                    smoothSolver;
        preconditioner            DIC;
```

```
        smoother                symGaussSeidel;
        tolerance                1e-6;
        relTol                   0.0;
    }

    phi_liq
    {
    solver                    smoothSolver;
        preconditioner            DIC;
        smoother                symGaussSeidel;
        tolerance                1e-6;
        relTol                   0.0;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
}

Tol_is_defined_here
{
    Tol 1e-6; //get_tol_from_this
}

// ************************************************************************* //
```

The *fvSolution* dictionary contains sub-dictionaries: SIMPLE that contains a control parameter nNonOrthogonalCorrectors set to 0, and Tol_is_defined_here that contains a control parameter Tol set to 1e-6 (which is read by *createTol.H*). The description of other options can be found in the OpenFOAM userguide.

1.3 Running the application

The first step to run a parallel case is to decompose the domain using the decomposePar utility for assigning to different processors. The dictionary associated with decomposePar, *decomposeParDict* is located in the *system* directory. The first entry is numberOfSubdomains corresponding to the number of processors available for the case. The method of decomposition can be simple. The domain is split into subdomains, in the x, y and z directions, the number of subdomains in each direction being given by the vector n. As this geometry is two dimensional, the 3rd direction, z, cannot be split, hence n_z must be 1. The n_x and n_y components of n split the domain in the x and y directions following $n_x n_y$ = numberOfSubdomains. To minimize the communication time between the processors, the number of cell faces adjoining the subdomains are kept to a minimum. For growth direction parallel to x, the domain is split in the y direction to have optimum load balance. The delta keyword is set to 0.001.

For this case, numberOfSubdomains = 4 and n = (1, 4, 1). When executing *Allrun*, decomposePar is run. The logs are stored into *log.decomposePar*.

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
```

```
    version     2.0;
    format      ascii;
    class       dictionary;
    note        "mesh decomposition control dictionary";
    location    "system";
    object      decomposeParDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains  4;

//- Keep owner and neighbour on same processor for faces in zones:
// preserveFaceZones (heater solid1 solid3);

//method          scotch;
method          simple;

simpleCoeffs
{
    n           (1 4 1);
    delta       0.001;
}
```

After compiling the solver (see *codeGuide*), it is executed in parallel using runParallel within *Allrun*. The progress of the job is stored into *log.eutectic*. It includes the current time, initial and final residuals for all fields.

1.4 Post-processing

Once the case has completed running, the decomposed fields and mesh from processor directories can be reconstructed for post-processing. Using reconstructPar, the decomposed mesh with fields can be reassembled. The results reconstructed to time directories can be viewed using ParaView by creating and opening *lamellar2D.foam* case module:

cd $FOAM_RUN/PhaseFieldSolverEutectic/lamellar2D

touch lamellar2D.foam

paraview lamellar2D.foam

The phase-fields at steady-state for equal volume fractions of α and β phases are shown in Figure 4–6. These are corresponding to the time, when the temperature becomes constant at a particular value despite being different during initialization.

An interesting result is observed, when the simulation is run by changing the equilibrium compositions to $c_\alpha^{eut} = 0.41$ and $c_\beta^{eut} = 0.71$ for 70% α and 30% β phase. At steady-state, the volume fractions become 70% α and 30% β despite starting off with equal volume fractions of α and β, which is shown in Figure 7.

2 3D lamellar eutectic growth

This case corresponds to 3D lamellar morphology with equal volume fractions of α and β phases initialized randomly in the plane perpendicular to the growth direction.

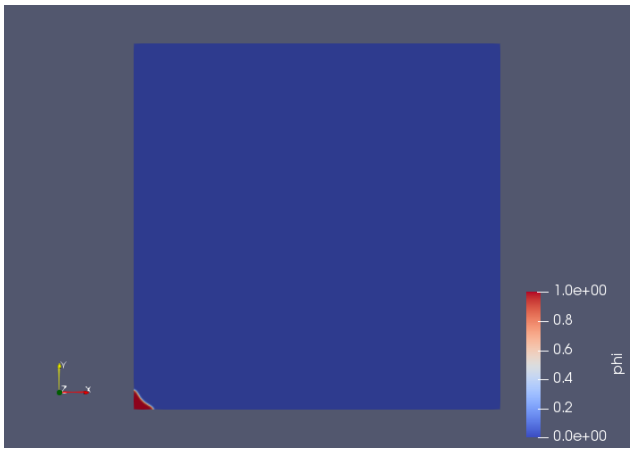

Figure 4: Figure 4: Variation of alpha phase-field at steady-state.

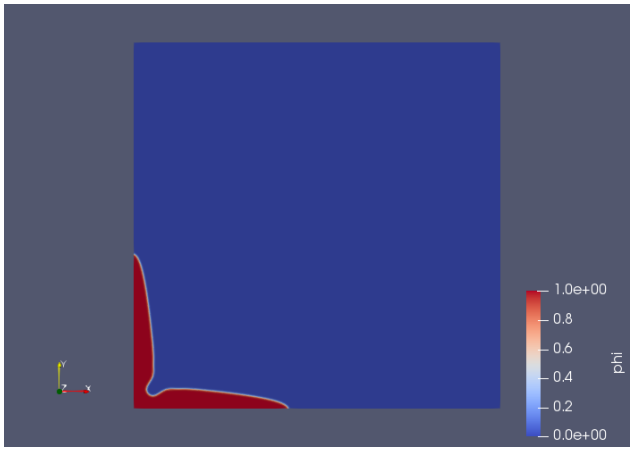

Figure 5: Figure 5: Variation of beta phase-field at steady-state.

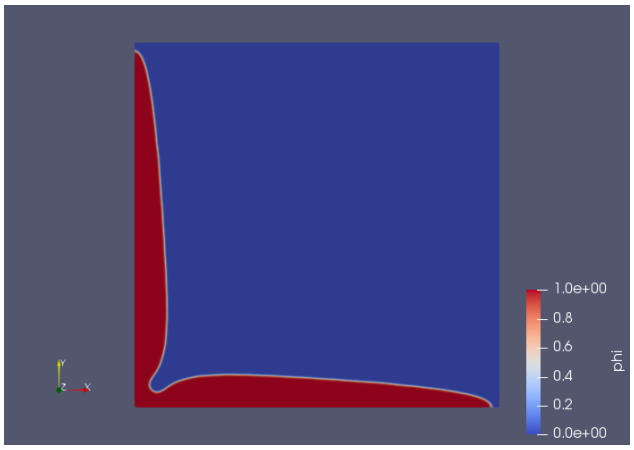

Figure 6: Figure 6: Variation of liquid phase-field at steady-state.

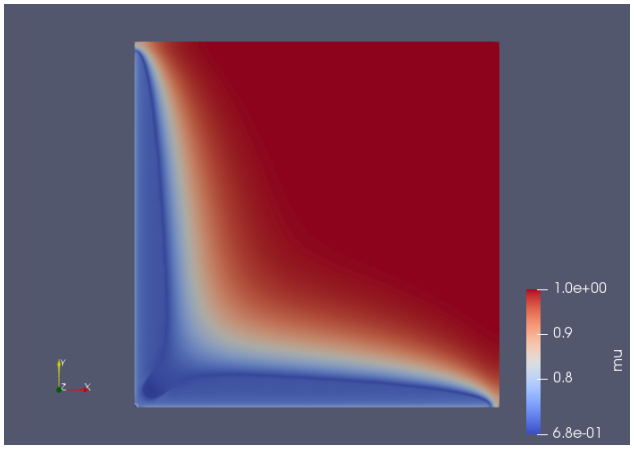

Figure 7: Figure 7: Variation of alpha phase-field at steady-state for 70-30 equilibrium composition.

2.1 Problem specification

Solution domain The domain is considered to be three-dimensional in this problem. The cubic domain consists of three phases with random α and β phase distribution perpendicular to the growth direction as shown in Figure 8.

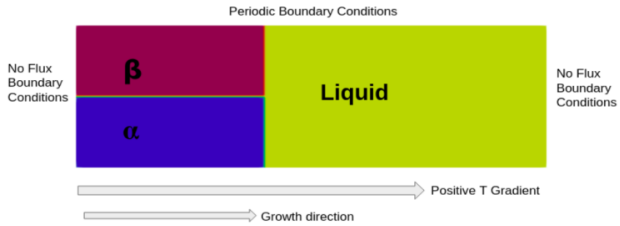

Figure 8: Figure 8: Three-dimensional geometry of the lamellar morphology.

Governing equations The governing equations used are same as in section 1.1.

Boundary conditions Cyclic boundary condition is specified at the top, bottom, front and back planes, and zero-flux is specified at the remaining planes.

Initial conditions The initial conditions are same as in section 1.1.

Physical properties The physical properties are same as in section 1.1.

Solver name *eutectic*.

Case name *lamellar3D*, located in the *$FOAM_RUN/PhaseFieldSolverEutectic* directory.

2.2 Pre-processing

To prepare case files and running the case, the user can change to the case directory:

cd $FOAM_RUN/PhaseFieldSolverEutectic/lamellar3D

2.2.1 Mesh generation

A domain of size 160 × 160 × 160 with dx of 2 is considered. Similar to section 1.2.1, the graded mesh is obtained by merging three blocks with different refinement.

The entries in *blockMeshDict* located in the *system* directory for this case are as follows:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
convertToMeters 2;
x_min 0;
x_max0 48;
x_max1 64;
x_max2 80; //3 blocks in direction of growth
y_min 0;
y_max 80;
z_min 0;
z_max 80;

// should all be integers
lx #calc "$x_max0 - $x_min";
ly #calc "$y_max - $y_min";
lz #calc "$z_max - $z_min"; // should all be integers
vertices
(
    ($x_min $y_min $z_min)
    ($x_max0 $y_min $z_min)
    ($x_max0 $y_max $z_min)
    ($x_min $y_max $z_min)
    ($x_min $y_min $z_max)
    ($x_max0 $y_min $z_max)
    ($x_max0 $y_max $z_max)
    ($x_min $y_max $z_max)

    ($x_max0 $y_min $z_min)
    ($x_max1 $y_min $z_min)
    ($x_max1 $y_max $z_min)
    ($x_max0 $y_max $z_min)
    ($x_max0 $y_min $z_max)
    ($x_max1 $y_min $z_max)
    ($x_max1 $y_max $z_max)
    ($x_max0 $y_max $z_max)

    ($x_max1 $y_min $z_min)
    ($x_max2 $y_min $z_min)
    ($x_max2 $y_max $z_min)
    ($x_max1 $y_max $z_min)
    ($x_max1 $y_min $z_max)
```

```
    ($x_max2 $y_min $z_max)
    ($x_max2 $y_max $z_max)
    ($x_max1 $y_max $z_max)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) ($lx 80 80) simpleGrading (1 1 1)
    hex (8 9 10 11 12 13 14 15) (8 40 40) simpleGrading (1 1 1)
    hex (16 17 18 19 20 21 22 23) (4 20 20) simpleGrading (1 1 1)
// for dx = 0.5: simpleGrading (2 2 2) or convertToMeters = 0.5
);

edges
(
);

//================ FOR ZERO-FLUX BOUNDARY CONDITIONS ============
boundary
(
    floor
    {
         type cyclic;
         neighbourPatch ceiling;
        faces
        (
        (0 1 5 4)
        (8 9 13 12)
        (16 17 21 20)
        );
    }

    ceiling
    {
        type cyclic;
        neighbourPatch floor;
        faces
        (
        (2 3 7 6)
        (10 11 15 14)
        (18 19 23 22)
        );
    }

    front
    {
         type cyclic;
         neighbourPatch back;
        faces
        (
        (4 5 6 7)
        (12 13 14 15)
        (20 21 22 23)
        );
```

```
}

back
{
    type cyclic;
    neighbourPatch front;
    faces
    (
    (0 3 2 1)
    (8 11 10 9)
    (16 19 18 17)
    );
}

sideSolid
{
    type wall;
    faces
    (
    (0 4 7 3)
    (17 18 22 21)
    );
}

wall_merge0
{
    type wall;
    faces
    (
        (1 2 6 5)
    );
}

wall_merge1
{
    type wall;
    faces
    (
        (8 12 15 11)
    );
}

wall_merge2
{
    type wall;
    faces
    (
        (9 10 14 13)
    );
}

wall_merge3
{
    type wall;
```

```
        faces
        (
            (16 20 23 19)
        );
    }

);

mergePatchPairs
(
    (wall_merge0 wall_merge1)
    (wall_merge2 wall_merge3)
);

// ************************************************************************* //
```

The generated 3D mesh is shown in Figure 9.

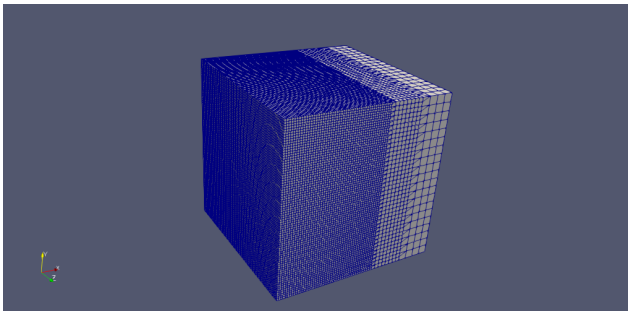

Figure 9: Figure 9: Generated 3D mesh for lamellar morphology case.

2.2.2 Boundary and initial conditions

Similar to section 1.2.2, the *0* sub-directory contains *phi_alpha.orig*, *phi_beta.orig*, *phi_liq.orig*, and *mu.orig*. For this case, the boundary consists of planes split into patches named: (1) ceiling, floor, front and back for cyclic boundary at top, bottom, front and back planes, respectively; (2) sideSolid patch includes other planes of the 3D case and is given a zeroGradient boundary condition.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      phi_alpha;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
    {
        type            cyclic;
    }
    ceiling
    {
        type            cyclic;
    }
    front
    {
        type            cyclic;
    }
    back
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      phi_beta;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//
```

```
boundaryField
{
    floor
    {
        type            cyclic;
    }
    ceiling
    {
        type            cyclic;
    }
    front
    {
        type            cyclic;
    }
    back
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      phi_liq;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
    {
        type            cyclic;
    }
```

```
    ceiling
    {
        type            cyclic;
    }
    front
    {
        type            cyclic;
    }
    back
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      mu;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
    {
        type            cyclic;
    }
    ceiling
    {
        type            cyclic;
    }
    front
    {
```

```
        type            cyclic;
    }
    back
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
```

2.2.3 Setting initial field

Similar to section 1.2.3, a non-uniform initial condition is specified for the phase parameters. The solid-liquid interface is set at 64 units in the growth direction from the origin. The α and β phases are distributed randomly in the plane perpendicular to the growth direction.

The *setFieldsDict* dictionary for this case is shown below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue phi_alpha 0.0
    volScalarFieldValue phi_beta 0.0
    volScalarFieldValue phi_liq 0.0
    volScalarFieldValue T 0.93
    volScalarFieldValue mu 1.0

);

regions
(

    boxToCell
// boxToCell: work on all cells in a rectangular box defined with starting and end point coordinates
    {
    box (-6 -6 -6) (64 166 166);
```

```
// Box can be larger than domain; in general this avoids edge effects
    fieldValues
// NB: no semicolons at the end of the dictionary entries below!
    (
    volScalarFieldValue phi_alpha 1.0
    volScalarFieldValue phi_beta 0.0
    volScalarFieldValue phi_liq 0.0
    volScalarFieldValue T 0.93
    volScalarFieldValue mu 1.0
    );
    }


    boxToCell
// boxToCell: work on all cells in a rectangular box defined with starting and end point coordinates
    {
    box (-6 75 0) (64 150 100);
// Box can be larger than domain; in general this avoids edge effects
    fieldValues
// NB: no semicolons at the end of the dictionary entries below!
    (
    volScalarFieldValue phi_alpha 0.0
    volScalarFieldValue phi_beta 1.0
    volScalarFieldValue phi_liq 0.0
    volScalarFieldValue T 0.93
    volScalarFieldValue mu 1.0
    );
    }


    boxToCell
// boxToCell: work on all cells in a rectangular box defined with starting and end point coordinates
    {
    box (64 -6 -6) (166 166 166);
// Box can be larger than domain; in general this avoids edge effects
    fieldValues
// NB: no semicolons at the end of the dictionary entries below!
    (
    volScalarFieldValue phi_alpha 0.0
    volScalarFieldValue phi_beta 0.0
    volScalarFieldValue phi_liq 1.0
    volScalarFieldValue T 0.93
    volScalarFieldValue mu 1.0
    );
    }


);


// ************************************************************************* //
```

2.2.4 Physical properties

The entries of *transportProperties* dictionary for this case are kept same as section 1.2.4.

2.2.5 Time step control

Similar to section 1.2.5, the startTime and deltaT are specified. The endTime keyword is set to 6000000 to observe the growth of secondary dendrites from the primary dendrite, growing normal to the direction of imposed thermal gradient. The writeInterval can be set to 200000. For this case, the entries in the *controlDict* are shown below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

application     eutectic;

startFrom       startTime;

startTime       0; // the actual time i.e. runtime.value() or no. of iterations*deltaT

stopAt          endTime;

endTime         500.00; //actual time = total_iterations*deltaT

deltaT          0.05;

writeControl    runTime;

writeInterval   20.00;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;
```

```
runTimeModifiable true;


// ************************************************************************* //
```

2.2.6 Discretisation schemes and solver settings

The *fvSchemes* and *fvSolution* dictionary entries are kept same as section 1.2.6 and 1.2.7.

2.3 Running the case

For growth direction parallel to x, the domain is divided in the y and z directions to have optimum load balance. Hence, numberOfSubdomains = 4 and n = (1, 2, 2) are set.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    note        "mesh decomposition control dictionary";
    location    "system";
    object      decomposeParDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains  4;

//- Keep owner and neighbour on same processor for faces in zones:
// preserveFaceZones (heater solid1 solid3);

//method          scotch;
method          simple;

simpleCoeffs
{
    n           (1 4 1);
    delta       0.001;
}
```

Similar to section 1.3, the solver for this case is run in parallel using runParallel within *Allrun*.

2.4 Post-processing

Similar to section 1.4, the results reconstructed to time directories can be viewed by creating and opening the *lamellar3D.foam* case module using ParaView. The α-β surface area becomes minimized while maintaining equal volume fractions. Further, it establishes the lamellar morphology. The steady-state phase-field profile is shown in Figure 10.

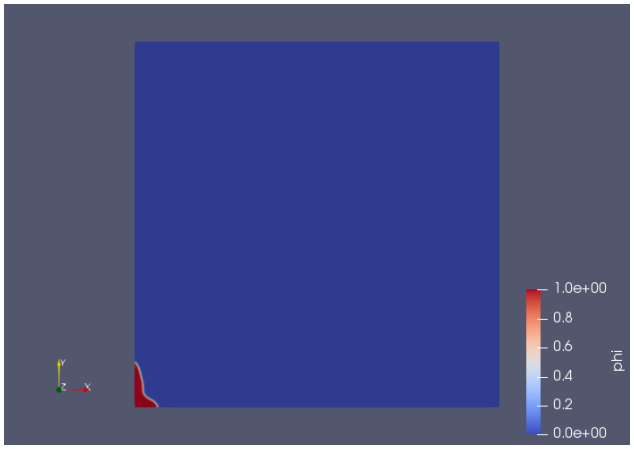

Figure 10: Figure 10: Lamellar morphology at solid-liquid interface at steady-state.

3 3D rod eutectic growth

This case pertains to 3D rod morphology with volume fractions of 70% α and 30% β phase in the plane perpendicular to the growth direction.

3.1 Problem specification

Solution domain The domain is considered to be three-dimensional in this problem. The cuboid shaped domain consists of three phases with rod shaped β phase parallel to the growth direction embedded within α phase as shown in Figure 11.

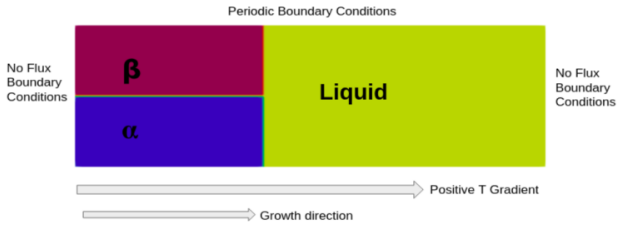

Figure 11: Figure 11: Three-dimensional geometry of the rod morphology.

Governing equations The governing equations used are same as in section 1.1.

Boundary conditions Cyclic boundary condition is specified at the top, bottom, front and back planes, and zero-flux is specified at the remaining planes same as in section 2.1.

Initial conditions The initial conditions are same as in section 2.1.

Physical properties The physical properties are same as in section 1.1.

Solver name *eutectic*.

Case name *rod3D*, located in the *$FOAM_RUN/PhaseFieldSolverEutectic* directory.

3.2 Pre-processing

To prepare case files and running the case, the user can change to the case directory:

cd $FOAM_RUN/PhaseFieldSolverEutectic/rod3D

3.2.1 Mesh generation

A domain of size 400 × 184 × 184 with dx of 2 is considered. Similar to section 1.2.1, the graded mesh is obtained by merging three blocks with different refinement.

The entries in *blockMeshDict* located in the *system* directory for this case are as follows:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
convertToMeters 2;
x_min 0;
x_max0 120;
x_max1 160;
x_max2 200; //3 blocks in direction of growth
y_min 0;
y_max 92;
z_min 0;
z_max 92;

// should all be integers
lx #calc "$x_max0 - $x_min";
ly #calc "$y_max - $y_min";
lz #calc "$z_max - $z_min"; // should all be integers
vertices
(
    ($x_min $y_min $z_min)
    ($x_max0 $y_min $z_min)
    ($x_max0 $y_max $z_min)
    ($x_min $y_max $z_min)
    ($x_min $y_min $z_max)
    ($x_max0 $y_min $z_max)
    ($x_max0 $y_max $z_max)
    ($x_min $y_max $z_max)

    ($x_max0 $y_min $z_min)
    ($x_max1 $y_min $z_min)
    ($x_max1 $y_max $z_min)
    ($x_max0 $y_max $z_min)
    ($x_max0 $y_min $z_max)
    ($x_max1 $y_min $z_max)
    ($x_max1 $y_max $z_max)
    ($x_max0 $y_max $z_max)
```

```
    ($x_max1 $y_min $z_min)
    ($x_max2 $y_min $z_min)
    ($x_max2 $y_max $z_min)
    ($x_max1 $y_max $z_min)
    ($x_max1 $y_min $z_max)
    ($x_max2 $y_min $z_max)
    ($x_max2 $y_max $z_max)
    ($x_max1 $y_max $z_max)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) ($lx 92 92) simpleGrading (1 1 1)
    hex (8 9 10 11 12 13 14 15) (20 46 46) simpleGrading (1 1 1)
    hex (16 17 18 19 20 21 22 23) (10 23 23) simpleGrading (1 1 1)
// for dx = 0.5: simpleGrading (2 2 2) or convertToMeters = 0.5
);

edges
(
);

//================ FOR ZERO-FLUX BOUNDARY CONDITIONS ============
boundary
(
    floor
    {
         type cyclic;
         neighbourPatch ceiling;
        faces
        (
        (0 1 5 4)
        (8 9 13 12)
        (16 17 21 20)
        );
    }

    ceiling
    {
        type cyclic;
        neighbourPatch floor;
        faces
        (
        (2 3 7 6)
        (10 11 15 14)
        (18 19 23 22)
        );
    }

    front
    {
         type cyclic;
         neighbourPatch back;
```

```
    faces
    (
    (4 5 6 7)
    (12 13 14 15)
    (20 21 22 23)
    );
}

back
{
    type cyclic;
    neighbourPatch front;
    faces
    (
    (0 3 2 1)
    (8 11 10 9)
    (16 19 18 17)
    );
}

sideSolid
{
    type wall;
    faces
    (
    (0 4 7 3)
    (17 18 22 21)
    );
}

wall_merge0
{
    type wall;
    faces
    (
        (1 2 6 5)
    );
}

wall_merge1
{
    type wall;
    faces
    (
        (8 12 15 11)
    );
}

wall_merge2
{
    type wall;
    faces
    (
        (9 10 14 13)
```

```
        );
    }

    wall_merge3
    {
        type wall;
        faces
        (
            (16 20 23 19)
        );
    }

);

mergePatchPairs
(
    (wall_merge0 wall_merge1)
    (wall_merge2 wall_merge3)
);

// ************************************************************************* //
```

The generated 3D mesh is shown in Figure 12.

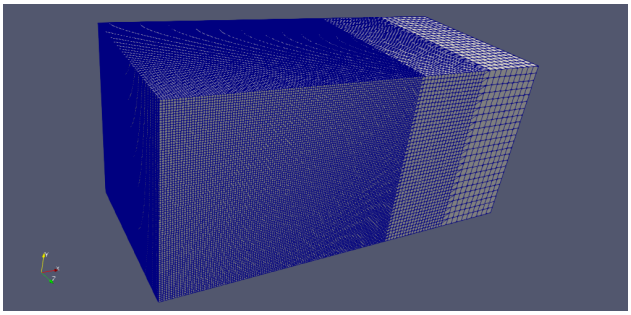

Figure 12: Figure 12: Generated 3D mesh for rod morphology case.

3.2.2 Boundary and initial conditions

Similar to section 2.2.2, the *0* sub-directory contains *phi_alpha.orig*, *phi_beta.orig*, *phi_liq.orig*, and *mu.orig*.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      phi_alpha;
```

```
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
    {
        type            cyclic;
    }
    ceiling
    {
        type            cyclic;
    }
    front
    {
        type            cyclic;
    }
    back
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      phi_beta;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;
```

```
//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
    {
        type            cyclic;
    }
    ceiling
    {
        type            cyclic;
    }
    front
    {
        type            cyclic;
    }
    back
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      phi_liq;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
```

```
    {
        type            cyclic;
    }
    ceiling
    {
        type            cyclic;
    }
    front
    {
        type            cyclic;
    }
    back
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.x                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      mu;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

//============== NO-FLUX BOUNDARY CONDITIONS ====================//

boundaryField
{
    floor
    {
        type            cyclic;
    }
    ceiling
    {
        type            cyclic;
```

```
    }
    front
    {
        type            cyclic;
    }
    back
    {
        type            cyclic;
    }
    sideSolid
    {
        type            zeroGradient;
    }

}
```

3.2.3 Setting initial field

Similar to section 1.2.3, a non-uniform initial condition is specified for the phase parameters. The solid-liquid interface is set at 160 units in the growth direction from the origin. The rod shaped β phase parallel to the growth direction is embedded within α phase.

The *setFieldsDict* dictionary for this case is shown below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue phi_alpha 0.0
    volScalarFieldValue phi_beta 0.0
    volScalarFieldValue phi_liq 0.0
    volScalarFieldValue T 0.93
    volScalarFieldValue mu 1.0

);

regions
(

    boxToCell
```

```
// boxToCell: work on all cells in a rectangular box defined with starting and end point coordinates
    {
    box (-6 -6 -6) (160 190 190);
// Box can be larger than domain; in general this avoids edge effects
    fieldValues
// NB: no semicolons at the end of the dictionary entries below!
    (
    volScalarFieldValue phi_alpha 1.0
    volScalarFieldValue phi_beta 0.0
    volScalarFieldValue phi_liq 0.0
    volScalarFieldValue T 0.93
    volScalarFieldValue mu 1.0
    );
    }

    cylinderToCell
    {
    p1 (-6 0 0) ;
    p2 (160 0 0);
    radius 92;

    fieldValues
    (
    volScalarFieldValue phi_alpha 0.0
    volScalarFieldValue phi_beta 1.0
    volScalarFieldValue phi_liq 0.0
    volScalarFieldValue T 0.93
    volScalarFieldValue mu 1.0
    );
    }

    cylinderToCell
    {
    p1 (-6 184 184) ;
    p2 (160 184 184);
    radius 92;

    fieldValues
    (
    volScalarFieldValue phi_alpha 0.0
    volScalarFieldValue phi_beta 1.0
    volScalarFieldValue phi_liq 0.0
    volScalarFieldValue T 0.93
    volScalarFieldValue mu 1.0
    );
    }

    boxToCell
// boxToCell: work on all cells in a rectangular box defined with starting and end point coordinates
    {
    box (160 -6 -6) (406 190 190);
// Box can be larger than domain; in general this avoids edge effects
    fieldValues
// NB: no semicolons at the end of the dictionary entries below!
```

```
    (
    volScalarFieldValue phi_alpha 0.0
    volScalarFieldValue phi_beta 0.0
    volScalarFieldValue phi_liq 1.0
    volScalarFieldValue T 0.93
    volScalarFieldValue mu 1.0
    );
    }


);


// ************************************************************************* //
```

3.2.4 Physical properties

The entries of *transportProperties* dictionary for this case are kept same as section 1.2.4 except $c_{\alpha}^{eut} = 0.41$ and $c_{\beta}^{eut} = 0.71$ for 70% α and 30% β phase. The entries of *transportProperties* dictionary are shown below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*-------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimx        dimx [0 1 0 0 0 0 0] 1; //Dimension of position
dimt        dimt [0 0 1 0 0 0 0] 1; //Dimension of time
ms_alpha    ms_alpha [0 0 0 0 0 0 0] -0.5; //Slope liq-alpha and alpha-liq
ms_beta     ms_beta [0 0 0 0 0 0 0] 0.5; //Slope liq-beta and beta-liq
c_eq_liq    c_eq_liq [0 0 0 0 0 0 0] 0.5; //Eutectic composition of liquid phase
c_eq_alpha  c_eq_alpha [0 0 0 0 0 0 0] 0.41; //Eutectic composition of alpha phase
c_eq_beta   c_eq_beta [0 0 0 0 0 0 0] 0.71; //Eutectic composition of beta phase
G           G [0 0 0 0 0 0 0] 1.0e-3; //Thermal gradient
v           v [0 0 0 0 0 0 0] 0.001; //Velocity
A           A [0 0 0 0 0 0 0] 1.0;
D           D [0 0 0 0 0 0 0] 1.0; //Diffusivity in liquid
T_eut       T_eut [0 0 0 0 0 0 0] 1.0; //Eutectic temperature
initial     initial [0 0 0 0 0 0 0] 0.89; //Constant value from temperature profile
tau         tau [0 0 0 0 0 0 0] 0.288; //Relaxation coefficient
gamma       gamma [0 0 0 0 0 0 0] 1.0; //Interface energy parameter
epsilon     epsilon [0 0 0 0 0 0 0] 8.0; //Interface width parameter
// ************************************************************************* //
```

3.2.5 Time step control

Similar to section 1.2.5, the startTime and deltaT are specified. The endTime keyword is set to 6000000 to observe the growth of secondary dendrites from the primary dendrite, growing normal to the direction of imposed thermal gradient. The writeInterval can be set to 200000. For this case, the entries in the *controlDict* are shown below:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

application     eutectic;

startFrom       startTime;

startTime       0; // the actual time i.e. runtime.value() or no. of iterations*deltaT

stopAt          endTime;

endTime         500.00; //actual time = total_iterations*deltaT

deltaT          0.05;

writeControl    runTime;

writeInterval   20.00;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;
```

```
// ************************************************************************* //
```

3.2.6 Discretisation schemes and solver settings

The *fvSchemes* and *fvSolution* dictionary entries are kept same as section 1.2.6 and 1.2.7.

3.3 Running the case

Similar to section 2.3, the domain is divided in the y and z directions to have optimum load balance. Hence, numberOfSubdomains = 4 and n = (1, 2, 2) are set.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    note        "mesh decomposition control dictionary";
    location    "system";
    object      decomposeParDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains  4;

//- Keep owner and neighbour on same processor for faces in zones:
// preserveFaceZones (heater solid1 solid3);

//method          scotch;
method          simple;

simpleCoeffs
{
    n           (1 2 2);
    delta       0.001;
}
```

3.4 Post-processing

Similar to section 1.4, the results reconstructed to time directories can be viewed by creating and opening the *rod3D.foam* case module using ParaView. The α-β surface area becomes minimized while maintaining equal volume fractions. The phase-field profile of rod morphology at steady-state is shown in Figure 13.

References

[Choudhury and Nestler(2012)] A Choudhury and B Nestler. Grand-potential formulation for multicomponent phase transformations combined with thin-interface asymptotics of the double-obstacle potential. *Physical review. E, Statistical, nonlinear, and soft matter physics.* 85:021602, 2012.

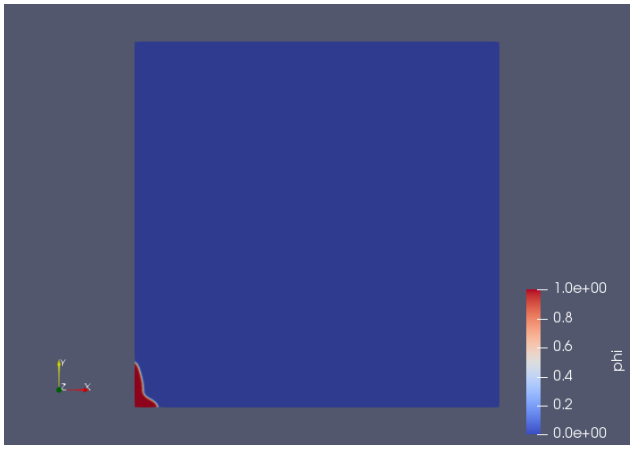

Figure 13: Figure 13: Rod morphology at solid-liquid interface at steady-state.

[Folch and Plapp(2005)] R Folch and M Plapp. Quantitative phase-field modeling of two-phase growth. *Phys. Rev. E.* 72:011602, 2005.

[Karma(2001)] A Karma. Phase-field formulation for quantitative modeling of alloy solidification. *Phys. Rev. Lett.* 87:115701, 2001.

[Umate(2021)] K Umate. Development of framework for eutectic solidification in OpenFOAM. *IISc Thesis.*