OPStudio Manual

December 15, 2023

Presented by: OpenPhase Solutions GmbH
Address: Universitätsstraße 136, 44799 Bochum, Germany
Telephone: +49 234 601456 03
Email: info@openphase-solutions.com

Contents

1 Introduction

The software OPStudio is an advanced simulation suite to study microstructure evolution in metallic materials. This documentation highlights the functionality, explains underlying theoretical models and shows real-world examples.

- What is OPStudio?
- Which models are used inside OPStudio, and which scientific papers are the basis?
- How do I use OPStudio?
- What can I simulate with OPtudio? What results can I create, using the presets available in the software?

2 OPStudio

2.1 Installation and Configuration

2.1.1 Remote Licenses

If you have a remote license, the Sentinel Key (USB license dongle) does not need to be connected to machine that runs OPStudio. The following will guide you through the process of setting up a license server. Before you insert the license dongle in the license server, the Sentinel LDK runtime needs to be installed on the machine. You can find the runtime using the following links.
Windows: `https://supportportal.gemalto.com/csm/?id=kb_article_view&sysparm_article=KB0018320`
Linux: `https://supportportal.gemalto.com/csm/?id=kb_article_view&sysparm_article=KB0018315`
After the installation of the runtime, you can insert the license dongle and check it function using the following link:
`http://localhost:1947/_int_/devices.html`

Client In order to use OPStudio you need to give the client machine information about the license server. This is done by placing a file name "hasp_96982.ini" in the following directory

Windows: %LocalAppData%/SafeNet Sentinel/Sentinel LDK/ **Linux:** $HOME/.hasplm/

with these contents:

disable_IPv6 = 1 requestlog = 0 errorlog = 1 getinfo_uncached = 0 serveraddr = "IP address of license server"

The client machine does not need to have the Sentinel LDK runtime install, but for troubleshooting it can make sense to install the runtime. With the runtime installed you can check for Sentinel key on the network using `http://localhost:1947/_int_/devices.html`

3 Models used in OPStudio

OPStudio uses several different models for the simulation of materials processing. The main model is the multi-phase-field method, that allows the tracking of interfaces between different regions of individual properties. It connects with diffusion models, thermodynamic models, elasticity and plasticity models as well as fluid dynamic models in order to simulate the evolution of materials. See also `https://link.springer.com/book/10.1007/978-3-031-21171-3`.

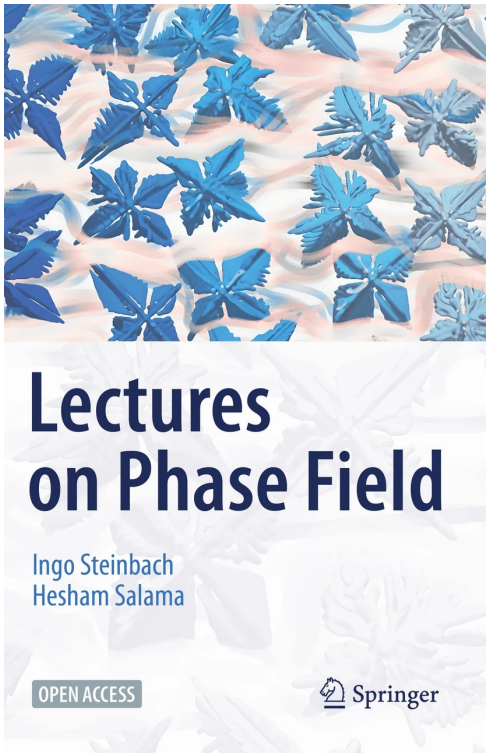


Figure 1: Recommended introduction to the phase field method.
See `https://link.springer.com/book/10.1007/978-3-031-21171-3`

3.1 The multi-phase-field method

The model allows the description of interfaces between sub-domains $\Omega_i \subset \Omega$ in a domain $\Omega \subset \mathbb{R}^d$. In the following these sub-domains will be referred to as phases and should not be confused with sub-domains in the context of domain decomposition methods. The phase-fields $\phi_a : \Omega \times \mathbb{R} \rightarrow [0,1]$ are auxiliary functions that distinguish between different phases and describe their evolution. A function value of 1 indicates the bulk of the phase and a value of 0 indicates the absence of this phase. Phases are connected by a diffuse interface that is described a smooth transition of the phase field functions ϕ_a.

The phase-fields ϕ_α satisfy the condition $\sum_{\alpha=1}^{\infty} \phi_\alpha = 1$ in every point $x \in \Omega$ of the domain Ω. As presented in Tegeler 2017 and Tegeler 2018 the evolution of phase fields ϕ_α in OpenPhase is determined by

$$\dot{\phi}_\alpha = \frac{\partial}{\partial t}\phi_\alpha\,(x,t) = -\sum_{\beta=1}^{N} \frac{\pi^2 M_{\alpha\beta}}{8\eta N}\left[\frac{\delta F}{\delta \phi_\alpha} - \frac{\delta F}{\delta \phi_\beta}\right], \tag{1}$$

with the interface mobility $M_{\alpha\beta}$ between phase fields ϕ_α and ϕ_β and a free energy

$$F = \int_\Omega \{f^{\mathrm{GB}} + \ldots\} dV, \tag{2}$$

and the number of phase fields that contribute

$$\mathrm{N} := N(x) := \sum_{\alpha=1}^{\infty} \mathcal{X}_{T(x)}(\phi_\alpha), \tag{3}$$

with

$$T(x) := \left\{\phi \in C^2(\Omega) | \phi(x) \neq 0 \vee \nabla\phi(x) \neq 0 \vee \nabla^2\phi(x) \neq 0\right\} \tag{4}$$

The interfacial free energy density is determined by

$$f^{\mathrm{GB}} = \sum_{\alpha\neq\beta} \frac{4\sigma_{\alpha\beta}}{\eta}\left[-\frac{\eta^2}{\pi^2}\nabla\phi_\alpha\cdot\nabla\phi_\beta + \phi_\alpha\phi_\beta\right]. \tag{5}$$

With the interface width η and the energy $\sigma_{\alpha\beta}$. Following [12] and [11] with the generalized curvature $I_\alpha := \Delta\phi_\alpha + \frac{\phi^2}{\eta^2}\phi_\alpha$ we get

$$\dot{\phi}_\alpha^{\mathrm{GB}} = \sum_{\beta=1,\ \beta\neq\alpha}^{N} \frac{M_{\alpha\beta}}{N}\left[\sum_{\gamma=1,\gamma\neq\beta}^{N}\left(\sigma_{\beta\gamma}-\sigma_{\alpha\gamma}\right) I_\gamma\right] \tag{6}$$

$$= \frac{1}{N}\sum_{\beta=1,\ \beta\neq\alpha}^{N} M_{\alpha\beta}\left[\sigma_{\alpha\beta}\left[I_\alpha - I_\beta\right] + \sum_{\gamma=1,\alpha\neq\gamma\neq\beta}^{N}\left(\sigma_{\beta\gamma}-\sigma_{\alpha\gamma}\right) I_\gamma\right], \tag{7}$$

$$= \frac{1}{N}\sum_{\beta=1\beta\neq\alpha} \dot{\psi}_{\alpha\beta}^{\mathrm{GB}}, \tag{8}$$

with the interface field $\psi_{\alpha\beta}^{\mathrm{GB}}$ for the contribution of the grain boundary. Additional contributions f^{AD} to the energy density can be considered by expanding the free energy to

$$F = \int_\Omega \{f^{\mathrm{GB}} + f^{\mathrm{AD}}\} dV, \tag{9}$$

Using the variational derivation as above we obtain

$$\dot{\phi}_\alpha^{GB} = \sum_{\beta=1,\ \beta\neq\alpha}^{N} \frac{M_{\alpha\beta}}{N}\left[\sum_{\gamma=1,\gamma\neq\beta}^{N}\left(\sigma_{\beta\gamma}-\sigma_{\alpha\gamma}\right) I_\gamma + \frac{\pi^2}{8\eta}\Delta g_{\alpha\beta}\right], \tag{10}$$

with a driving force

$$\Delta g_{\alpha\beta} = \frac{\partial f^{\mathrm{AD}}}{\partial\phi_\beta} - \frac{\partial f^{\mathrm{AD}}}{\partial\phi_\alpha}. \tag{11}$$

The main advantage of the phase field model used in OPStudio is the rigorous consideration of the interface properties. One of the key points of the model is the assumption that a steady phase field contour along the interface normal is always maintained during the simulation which also ensures constant width of the interface. Such a steady phase field contour, also called traveling wave solution, is important for the correct consideration of the interface properties and the resulting interface kinetics. Depending on the origin of the driving force Δg and context in which the phase field method is utilized, the driving force Δg might vary across the interface. This can significantly alter the phase field profile across the interface which affects interface kinetics. However, it is important to note, that the correct phase field profile across the interface can only be obtained if the driving force Δg is constant across the interface and varies only along the interface. Therefore, we replace $\Delta g_{\alpha\beta}$ with $h'\ \Delta\bar{g}_{\alpha\beta}$ with an average driving force $\Delta\bar{g}_{\alpha\beta}$, that is constant along the normal direction of the interface. In order to guaranty that the driving force contribution to the phase field evolution is independent of the chosen space discretization the pre-factor $h' = \frac{8}{\pi}\sqrt{\phi_\alpha\phi_\beta}$ is obtained by integrating the traveling wave profile (for more details see [11]). We finally get

$$\dot{\phi}_\alpha = \sum_{\beta=1\beta\neq\alpha}^{N} \frac{M_{\alpha\beta}}{N}\left[\sum_{\gamma=1,\alpha\neq\gamma\neq\beta}^{N}\left(\sigma_{\beta\gamma}-\sigma_{\alpha\gamma}\right) I_\gamma + \frac{\pi}{\eta}\sqrt{\phi_\alpha\phi_\beta}\Delta\bar{g}_{\alpha\beta}\right], \tag{12}$$

and notice that for a I implementation the computational effort would scale with $O\left(n^3\right)$, where n is the number of phase fields. This would create an unsuitable amount of effort in simulation with a high number of different phase fields. [15] introduced a technique known as active parameter tracking, that reduced the computational effort significantly by discarding contributions of phase fields that are

determined to be insignificant. In [15] the double well potential is used for which the profile of the diffuse interface is determined by a tanh-function, so that phase field values are close to, but not really 0 or 1. Therefore, in this case active parameter tracking affects the results of the simulation as phase field values close to 0 or 1 are cut off. Using the double obstacle potential, which has an sin -profile, we notice: If both $\phi_k(x) = 0$ and $\nabla^2\phi_k(x) = 0$ then phase k does not make a contribution to $\dot{\psi}_{ij}^{\mathrm{GB}}$. Therefore, a simulation using the double obstacle potential can safely skip the computation of contributions by phase fields with $\phi_k(x) = 0$ and $\nabla^2\phi_k(x) = 0$. However, $\dot{\phi}_i^n(x) > 0$ is possible even if both $\phi_k(x) = 0$ and $\nabla^2\phi_k(x) = 0$. This means a nucleation event is triggered that would create a new phase, which is handled separately.

3.1.1 Driving Force Averaging

As mentioned above we need to ensure the correct phase field profile across the interface an averaging procedure for the driving force is implemented in the OpenPhase. During this procedure, the driving force is averaged twice in each grid point over a set of points

$$S_{\alpha\beta}^R(x_h) \;=\; \{y_h \;\in \Omega_h \mid (R - |y_h - x_h|_2)\phi_\alpha(y_h)\phi_\beta(y_h) \;>\; 0\;\}, \tag{13}$$

that is the intersection of a sphere with radius $R = \frac{2}{3}\eta$ and the interface between phases α and β. In the first step the average

$$\Delta\bar{g}_{\alpha\beta}^0(x_h) = \frac{\sum_{y_h \in S_{\alpha\beta}^R(x_h)} \omega_1(y_h)\,\Delta g_{\alpha\beta}(y_h)}{\sum_{y_h \in S_{\alpha\beta}^R(x_h)} \omega_1(y_h)} \tag{14}$$

is weighted with $\omega_1(y_h) := \phi_\alpha(y_h)^2\phi_\beta(y_h)^2$, which emphasizes values near the center of the interface. In the second step the average

$$\Delta\bar{g}_{\alpha\beta}(x_h) = \frac{\sum_{y_h \in S_{\alpha\beta}^R(x_h)} \omega_2(y_h)\,\Delta\bar{g}_{\alpha\beta}^0(y_h)}{\sum_{y_h \in S_{\alpha\beta}^R(x_h)} \omega_2(y_h)} \tag{15}$$

is calculated using the weight $\omega_2(y_h) := \Theta(\frac{\phi_\alpha}{\phi_\beta} - 0.5)\Theta(2 - \frac{\phi_\alpha}{\phi_\beta})$, in which $\Theta(.)$ Is the Heaviside function. This average only uses the values near the centre of the interface.

3.1.2 Sparse Storages

As mentioned above the computational effort needed compute all combinations of phase fields can be extremely large as the amount of work scales cubically with the number of phase fields. We can, however, significantly reduce the computational load by skipping calculations that do not make a contribution. In a grid point $x_h \in \Omega_h$ with $\phi_\alpha(x_h) = 0$ and $\nabla^2\phi_\alpha(x_h) = 0$ we do not need to consider the contribution of the phase field ϕ_α to any other phase field or changes to the phase field ϕ_α itself. That means we only need to compute updates in regions, in which $\phi_\alpha(x) \neq 0$ for at least two phases or $\Delta\phi_\alpha(x) \neq 0$ for at least one phase. This region is henceforth called the interface between phases. Also, in this calculation we only need to regard the contribution of phases with $\phi_\alpha(x) \neq 0$ or $\nabla^2\phi_\alpha(x) \neq 0$. In addition, the utilization of sparse storages allows a significantly reduction of the memory requirement, when handling large numbers of phase fields. In order to store the phase fields ϕ_α and the interactions $\psi_{\alpha\beta}$ OPStudio uses vectors of a container, that stores up to two indices and up to two values. In each grid point this vector only stores nodes with non-zero values. Values of zero are implied if no node for a phase field or interaction exits. Generally, in the majority of grid points the vector consists of only one node, which means we are in the bulk of a phase. Vectors with two nodes will be quite frequent as well, however, grid points in which five or more nodes are needed are very rare. Due to the small data sets a linear search algorithm is used as it is the most efficient in this case. Using these sparse storages OPStudio can handle an arbitrarily large number of different phase fields. Depending on the specific data this storage can be interpreted as symmetric or antisymmetric, which further reduces the amount of data by half the number of possible combinations. Using these sparse storages however makes the use of implicit time stepping schemes for the phase field equation impossible.

3.1.3 Interface energy, stiffness and mobility

In the Interface energy module, you can set the interface energies $\sigma_{\alpha\beta}$ between thermodynamic phases in J/m^2. Additionally, you can set an anisotropy type, either "Cubic" or "Hexagonal" and the intensity of the anisotropy between two thermodynamic phases. Additional anisotropy types and the option to select individual anisotropy types for each phase pair will be added in a future version of OPStudio. The anisotropy of the interface is implemented by the interface stiffness $\sigma^* = \sigma + \sigma''$ In order to calculate the anisotropic interface stiffness and mobility the phase field normal $n = \frac{\nabla\phi}{|\nabla\phi|}$ is multiplied with the rotation matrix R determined by the crystallographic orientation

$$\begin{pmatrix} \bar{n}_x \\ \bar{n}_y \\ \bar{n}_z \end{pmatrix} = \bar{n} = Rn = R\frac{\nabla\phi}{|\nabla\phi|}. \tag{16}$$

In the case of a cubic symmetry the interface stiffness is then given by

$$\sigma^*_{cubic} = \sigma_0\left(1 + \epsilon\left(1.5 - 2.5\left(\bar{n}_x^4 + \bar{n}_y^4 + \bar{n}_z^4\right)\right)\right), \tag{17}$$

With the reference interface energy σ_0 and the interface anisotropy parameter ϵ. In the case of a hexagonal symmetry you have multiple options. First the anisotropy function given by [3]

$$\sigma^*_{Boettger} = \sigma_0(1 - \epsilon(\bar{n}_x^6 - \bar{n}_y^6 + \bar{n}_z^6 + 5(\bar{n}_z^4 - \bar{n}_z^2) + 15(\bar{n}_y^4\bar{n}_x^2 - \bar{n}_x^4\bar{n}_y^2)), \tag{18}$$

The function given by [13]

$$\sigma^*_{Sun} = \sigma_0(1 + \epsilon_1\frac{1}{4}\sqrt{\frac{5}{\pi}}\left[3\bar{n}_z^2 - 1\right] + \epsilon_2\frac{3}{16}\sqrt{\frac{1}{\pi}}\left[35\bar{n}_z^4 - 30\bar{n}_z^2 + 3\right] \tag{19}$$

$$+ \epsilon_3\frac{1}{32}\sqrt{\frac{13}{\pi}}\left[231\bar{n}_z^6 - 315\bar{n}_z^4 + 105\bar{n}_z^2 - 5\right] + \frac{1}{64}\sqrt{\frac{6006}{\pi}}\left[\bar{n}_x^6 - 15\bar{n}_x^4\bar{n}_y^2 + 15\bar{n}_x^2\bar{n}_y^4 - \bar{n}_y^6\right]), \tag{20}$$

Uses four anisotropy parameters ϵ_1,ϵ_2, ϵ_3 and ϵ_3, that default to $\epsilon_1 = -0.026$, $\epsilon_2 = \epsilon_3 = 0$ and $\epsilon_4 = 0.003$. [17] gives the anisotropy function

$$\sigma^*_{Yang} = \sigma_0\left(1 + \epsilon_1\left[3\bar{n}_z^2 - 1\right]^2 + \epsilon_2\left[\bar{n}_x^3 - 3\bar{n}_x\bar{n}_y^2\right]^2\left[9\bar{n}_z^2 - 1 + \epsilon_3\right]^2\right), \tag{21}$$

With three anisotropy parameters ϵ_1,ϵ_2, and ϵ_1, which default to $\epsilon_1 = -0.02$ and $\epsilon_2 = \epsilon_3 = 0.15$. For the mobility the anisotropy functions look similar but have the opposite signs in the anisotropic terms, here we have

$$M_{cubic} = M_0\left(1 - \epsilon^M\left(1.5 - 2.5\left(\bar{n}_x^4 + \bar{n}_y^4 + \bar{n}_z^4\right)\right)\right), \tag{22}$$

$$M_{Boettger} = M_0(1 + \epsilon^M(\bar{n}_x^6 - n_y^6 + n_z^6 + 5(n_z^4 - n_z^2) + 15(n_y^4n_x^2 - n_x^4n_y^2)), \tag{23}$$

$$M_{Sun} = M_0(1 - \epsilon_1^M\frac{1}{4}\sqrt{\frac{5}{\pi}}\left[3\bar{n}_z^2 - 1\right] - \epsilon_2^M\frac{3}{16}\sqrt{\frac{1}{\pi}}\left[35\bar{n}_z^4 - 30\bar{n}_z^2 + 3\right] \tag{24}$$

$$- \epsilon_3^M\frac{1}{32}\sqrt{\frac{13}{\pi}}\left[231\bar{n}_z^6 - 315\bar{n}_z^4 + 105\bar{n}_z^2 - 5\right] - \epsilon_4^M\frac{1}{64}\sqrt{\frac{6006}{\pi}}\left[\bar{n}_x^6 - 15\bar{n}_x^4\bar{n}_y^2 + 15\bar{n}_x^2\bar{n}_y^4 - \bar{n}_y^6\right]), \tag{25}$$

$$M_{Yang} = M_0\left(1 - \epsilon_1^M\left[3\bar{n}_z^2 - 1\right]^2 - \epsilon_2^M\left[\bar{n}_x^3 - 3\bar{n}_x\bar{n}_y^2\right]^2\left[9\bar{n}_z^2 - 1 + \epsilon_3^M\right]^2\right), \tag{26}$$

With anisotropy parameters ϵ_i^M that are not necessarily the same as the anisotropy parameters ϵ_i for the interface stiffness.

3.1.4 Temperature

In most phase field simulations temperature plays an important role. OPStudio allows for different ways to handle the temperature evolution. One option is to provide a temperature curve over time by and

"*.opit" file, that defines a global temperature at different points in time. "*.opit" files are ASCII-files with the following information: In between two defined states the temperature is linearly interpolated. In this case the temperature is always constant in space. Another possibility is setting a starting temperature and a reference point with that starting temperature and giving the three components of a temperature gradient. Then a constant cooling or heating rate can be applied. Additionally, latent heat production can be applied globally under the assumption, that heat diffusion is fast compared to solute diffusion and phase field motion. In this case the latent heat must be set to "global". The amount of latent heat created by the phase transformation between two thermodynamic phases can be set in the latent heat matrix in J/mol. If the solute diffusion or phase field motion is very fast, e.g. additive manufacturing, it makes sense to resolve the diffusion of heat energy. In this case the latent heat must be set to "local" and the Heat Diffusion model must be activated. Optionally, local heat sources or heat sinks can be added by "New Local Temperature", which allows the definition of a rectangular region, which heat is inserted or extracted by a given rate in K/s. In the Heat Diffusion module, the heat conductivity k in each thermodynamic phase needs be given in J/msK. The Heat Diffusion module solves the heat equation

$$\frac{\partial\, T}{\partial\, t} = \frac{k}{\rho\, c_p}\nabla^2 T. \tag{27}$$

Using the implicit Jacobi method. As the termination condition for the iterative solver you need to set a desired tolerance for the maximum local residual. You can also define the number of iteration steps after which a warning will be displayed on the console. This can be useful when the heat equation solver does not converge. The Heat Diffusion module also allows you to enable one-dimensional extension of the domain on any side of the simulation domain. The extensions are activated by setting their lengths in grid points to a value greater than zero. The extensions use the average thermal conductivity, density and heat capacity of the outermost two-dimensional layer at the specific side of the simulation domain. You can also set different boundary conditions for the heat diffusion that are independent of the boundary conditions given in the boundary conditions module.

3.1.5 Driving Force

In order to preserve the sine-profile of the phase field and avoid spreading or contracting of the interface the driving force is limited according

$$\Delta g = \Delta g^{limit} \tanh(\frac{\Delta g}{\Delta g^{max}}), \tag{28}$$

with a maximum driving force

$$\Delta g^{limit} = C\frac{2\pi}{\eta N}\sigma_m^* in \tag{29}$$

and a cut-off parameter C, that can be set in the driving force module. A value $C < 1$ guaranties a stable phase field but can be too severe of a cut-off in cases of nucleation, if the driving force is limited below the critical driving force for nucleus growth. If the driving force is limited in this way a warning will appear on the output console, that looks like this: n gives the number of grid points in which the driving force has been significantly limited, that is the points in which

$$\Delta g > 0.4\Delta g^{limit}. \tag{30}$$

Max value shows the maximum driving force value Δg^{max} and max overshoot the ratio

$$\frac{\Delta g}{\max \Delta g^{limit}}. \tag{31}$$

In many simulations the driving force is limited in the beginning as the initial conditions are not relaxed and far from the equilibrium. As the driving force is reduced during the simulation, the driving overshoot should disappear. In the case of continuous cooling however, it might occur that the driving force is increasing as the phase transformation lacks behind due to the driving force limiting, in which case the overshoot increases. A remedy to this is the Stabilization method that will be described later or decreasing the grid spacing dx, which decreases the interface width η and increases the maximum driving force Δg^{max}.

3.1.6 User Driving Force

The UserDrivingForce can be used to apply a simple driving force using the following equation:

$$-\frac{T - T_{eq}}{T_{eq}} \left(L_1 - L_2\right), \tag{32}$$

With temperature T, equilibrium temperature T_{eq} and latent heats L_i of phase i.

3.1.7 Nucleation

The nucleation module allows the addition of new phase fields during the simulation if they are energetically favourable. You can specify which phase can nucleate in which matrix phase. "Bulk" enables nucleation only in bulk points, that is $\varphi_{Matrix} = 1$. "GB" only allows nucleation in the interface, that is $0 < \varphi_{Matrix} < 1$. "Bottom" is designed for directional solidification. With this option, nucleation happens only at the bottom of the domain, that is $z - coordinate = 0$. Only if a nucleation event is allowed, the remaining parameters for the phase pair need to be specified.

The Nucleation density defines the number of test sites, in which the driving force for the new phase is calculated. If the driving force is larger than the critical driving force, the nucleus is planted. New sites are tested in each nucleation step, until the nucleation density is reached, or no new potential nucleation sites can be found. This can happen due to disappearance of the matrix phase or shielding between different nucleation sites. This value is given in $1/m^3$.

You can define a temperature interval in which nucleation sites are tested. This allows you to avoid unnecessary calculations in temperature regions in which the nucleation phase is not stable.

The Grain orientation mode allows you to specify the rule for determining the orientation of new nuclei. Random, RandomInPlane: for 2D simulations, prevents grains from turning out of plane, Parent: inherits orientation of the parent grain. Reference does not rotate the nucleus and uses the orientation of the reference frame.

MobilityReduction is an option that might help with numerical issue, as nuclei do not use the complete interface stiffness contribution, their interface can become unstable and spread. Reducing the mobility of nuclei with the specified factor helps to avoid interface spreading.

You can further set a shielding radius, that determines the minimum distance between nuclei.

The nucleation density can be set relative to the domain volume or the matrix phase volume, by checking the appropriate checkbox.

By default, the nucleation module tries to generate new nucleation sites at every global time step. The frequency of this can be decreased by setting the interval to a higher value.

3.2 Fluid Dynamics

In order to simulate fluids OPStudio utilizes the Lattice Boltzmann method, a simple and powerful tool for the modelling of complex fluid systems, see [8], [4] and [14]. The method calculates the evolution of the particle velocity distributions f_i in two steps. First a collision step

$$f_i^c\left(x,t\right) = f_i\left(x,t\right) - \frac{1}{\tau}\left[f_i\left(x,t\right) - f_i^{eq}\left(x,t\right)\right], \tag{33}$$

and then a streaming

$$f_i\left(x + e_i\Delta t, t + \Delta t\right) = f_i^c\left(x,t\right). \tag{34}$$

The Lattice-Boltzmann method is a numerical scheme that is derived by simplifying the Boltzmann equation

$$\left(\partial_t + v\cdot\nabla_x + \frac{1}{m}F\cdot\nabla_v\right) f\left(x,v,t\right) = \Omega\left(f\right), \tag{35}$$

for a particle density function $f\left(x,v,t\right)$ with position x, velocity v, time t, mass m and an external force F and collision integral $\Omega\left(f\right)$. Intermediate steps are the lattice gas automata that discretize space, time and the particle velocities, see also [16]. They use a regular lattice of nodes x_h and Boolean variables

$n_i\,(x_h, t_n)$ for $i = 1, \ldots$,m that indicates whether a node contains a particle with the velocity e_i. Ignoring any external force F the evolution equation becomes

$$n_i(x_h + e_i, t_n + 1) \;=\; n_i(x_h, t_n) \;+\; \Omega_i(n(x_h, t_n) \tag{36}$$

with the collision operator $\Omega_i(n(x_h, t_n)$, that describes the interaction of particles at a node according to a scattering rule, see [4]. For the Lattice-Boltzmann method the Boolean variables $n_i\,(x_h, t_n)$ are replaced by particle distribution function $f_i = \langle n_i \rangle$, that allow us to define the density ρ of a fluid by

$$\rho = \sum_{i=1}^{m} f_i, \tag{37}$$

and the momentum ρv by

$$\rho\; v = \sum_{i=1}^{m} f_i e_i \tag{38}$$

with the velocity $v \in R^d$ and the particle velocity $e_i \in R^d$. The kinetic equation is given by

$$f_i(x + e_i \Delta x,\; t + \;\Delta t) \;=\; f_i(x, t) \;+\; \Omega_i(f(x, t), \tag{39}$$

with the collision operator Ω_i that is required to satisfy

$$\sum_{i=1}^{m} \Omega_i = 0, \text{ and } \sum_{i=1}^{m} \Omega_i e_i = 0, \tag{40}$$

to conserve mass and momentum. Taylor expansion in time and space gives

$$\epsilon \left[\partial_t f_i + \partial_{e_i} f_i + \epsilon \left(\frac{1}{2} \partial_{e_i} \partial_{e_i} f_i + \partial_{e_i} \partial_t f_i + \frac{1}{2} \partial_t^2 f_i \right) \right] = \Omega_i. \tag{41}$$

Expansion of the particle distribution function f_i about the local equilibrium distribution gives

$$f_i = f_i^{\text{eq}} + \epsilon f_i^{(1)} + \epsilon^2 f_i^{(2)} + O\left(\epsilon^3\right), \tag{42}$$

which are subject to the constraints

$$\rho = \sum_{i=1}^{m} f_i^{\text{eq}}, \text{ and } \rho v = \sum_{i=1}^{m} f_i^{\text{eq}} e_i, \tag{43}$$

as well as

$$\sum_{i=1}^{m} f_i^{(k)} = 0, \text{ and } \sum_{i=1}^{m} f_i^{(k)} e_i = 0. \tag{44}$$

With non-equilibrium part

$$f_i^{\text{neq}} = f_i^{(1)} + \epsilon f_i^{(2)} + O\left(\epsilon^2\right) \tag{45}$$

the collision operator Ω_i can be expanded to

$$\Omega_i\,(f) = \Omega_i\,(f^{\text{eq}}) + \epsilon \sum_{j=0}^{M} M_{ij} f_j^{(1)} \tag{46}$$

$$+ \epsilon^2 \left(\sum_{j=0}^{M} M_{ij} f_j^{(2)} + \sum_{k=0}^{M} \sum_{j=0}^{M} \partial_{f_k} M_{ij} f_j^{(1)} f_k^{(1)} \right) + O\left(\epsilon^3\right) \tag{47}$$

$$= \Omega_i\,(f^{\text{eq}}) + \epsilon \sum_{j=0}^{M} M_{ij} \left(f_j^{(1)} + \epsilon f_j^{(2)} + O\left(\epsilon^2\right) \right) + O\left(\epsilon^3\right), \tag{48}$$

with the collision matrix $M_{ij} := \partial_j \Omega_i(f_{eq})$. Using $\Omega_i\left(f^{\mathrm{eq}}\right) = 0$, we get

$$\frac{\Omega_i\left(f\right)}{\epsilon} = \frac{M_{ij}}{\epsilon}\left(f_j - f_j^{\mathrm{eq}}\right). \tag{49}$$

Using $M_{ij} = -\frac{1}{\tau}\delta_{ij}$ we obtain the LBGK equation

$$f_i\left(x + e_i\Delta x, t + \Delta t\right) = f_i\left(x,t\right) - \frac{\left(f_j - f_j^{\mathrm{eq}}\right)}{\tau}. \tag{50}$$

The relaxations time

$$\tau = 3\nu\frac{\Delta t}{\left(\Delta x\right)^2} + \frac{1}{2}, \tag{51}$$

depends on the kinematic viscosity ν and the time and length scales Δt and Δx, see also [14].

3.2.1 Two-Phase Flows

A method for the simulation of fluid flows with multiple phases was introduced by [9]. The special case of a liquid-gas two phase system was covered by [10]. The behaviour of non-ideal gases is modelled by the body-force term

$$F\left(x\right) = -G\psi\left(\rho\left(x\right)\right)\sum_{i=1}^{m}\psi\left(\rho\left(x + e_i\right)\right)e_i \tag{52}$$

with an interaction parameter G and a potential

$$\psi\left(\rho\right) = 1 - e^{\frac{-\rho}{\rho_0}}, \tag{53}$$

that depends on the fluid density ρ and reference density ρ_0. In the following $\rho_0 = 1$ is used. Considering the non-ideal effects, the equation of state becomes

$$p = c_s^2\rho + \frac{1}{2}c_s^2\psi\left(\rho\right)^2, \tag{54}$$

with the speed of sound c_s.

3.2.2 The Exact Difference Method

Different body forces such as gravity, buoyancy, drag or the Shan-Chen Force for liquid-gas two phase system can act on the fluid. All forces acting on the fluid are combined in a total local body force $F = F_1 + F_2 + \ldots$, where F_1 and F_2 are different force contributions. Following [5] the evolution equation is expanded to

$$f_i\left(x + e_i\Delta x, t + \Delta t\right) = f_i\left(x,t\right) - \frac{\left(f_j - f_j^{\mathrm{eq}}\right)}{\tau} + \Delta f_i, \tag{55}$$

with the difference of the equilibrium distributions

$$\Delta f_i = f_i^{\mathrm{eq}}\left(\rho, u + \frac{F\Delta t}{\rho}\right) - f_i^{\mathrm{eq}}\left(\rho, u\right). \tag{56}$$

3.2.3 Fluid-Solid-Interaction

Two kinds of fluid-solid interaction must be considered; one is the momentum exchange between the fluid and solid bodies submerged in the fluid. Another is the incorporation of wetting properties of the solid in multiphase settings. Regarding the momentum exchange a modification of the streaming step is used following [1]. Grid points x_h are marked as obstacles if in this grid point no fluid is present, in conjunction with the phase field method this means

$$\phi_i\left(x_h\right) = 0 \text{ for all } i \in I_{Fluid} \tag{57}$$

with the index set I_{Fluid} that contains all fluid phases. This means grid points in the interface between solid and fluid are not marked as obstacles. The update to particle populations $f_i(x_h, t)$ at grid point x_h is modified to include the bounce back at obstacle grid points $x_h + e_i$, denoting with $e_{\bar{i}}$ the opposite direction to e_i we have

$$f_{\bar{i}}(x_h, t+1) = f_i(x_h, t) + 2w_i e_{\bar{i}} v_s(x_h + e_i), \tag{58}$$

with the local solid velocity $v_s(x_h + e_i)$, that includes the translational and angular velocities. The fluid also exerts a force on the solid body given by

$$F = 2(f_i + w_i e_i v_s(x_h + e_i))e_i. \tag{59}$$

3.3 The Grand Potential method

We define the grand potential functional

$$\Omega[\phi_\alpha, \mu_i] = \int_V \left(\omega^{\text{int}}(\phi_\alpha, \nabla\phi_\alpha) + \sum_\alpha \phi_\alpha \omega_\alpha^{\text{bulk}}(\mu_i)\right) \mathrm{d}^3x \tag{60}$$

Analogous as a above, we use the variational derivatie of the grand potential function to obtain the phase-field equation

$$\dot{\phi_\alpha} = \sum_{\alpha\beta} M_{\alpha\beta}\left(\frac{\delta\Omega}{\delta\phi_\beta} - \frac{\delta\Omega}{\delta\phi_\alpha}\right) \tag{61}$$

the driving force force term is replaced by a difference in the Grand Potential density

$$\dot{\phi} = \sum_{\beta=1,\ \beta\neq\alpha}^{N} \frac{M_{\alpha\beta}}{N}\left[\sum_{\gamma=1,\gamma\neq\beta}^{N}(\sigma_{\beta\gamma} - \sigma_{\alpha\gamma})I_\gamma + \frac{\pi^2}{8\eta}(\omega_\alpha - \omega_\beta)\right], \tag{62}$$

Note that grand potential density, ω, is the negative hydrostatice pressure in absense of shear stress because $\Omega = -pV$.

The temporal evolution of the chemical potential can be obtained with the definition

$$\rho_i = \sum_\alpha \phi_\alpha \rho_{\alpha i} = \sum_\alpha \phi_\alpha \frac{\partial \omega^{\text{bulk}}}{\partial \mu} \tag{63}$$

so that

$$\dot{\rho_i} = \sum_\alpha \left(\dot{\phi_\alpha}\frac{\partial\omega^{\text{bulk}}}{\partial\mu_i} + \phi_\alpha\dot{\mu_i}\frac{\partial^2\omega^{\text{bulk}}}{\partial\mu_i^2}\right) \tag{64}$$

with the conservation of mass

$$\dot{\rho_i} = \nabla\mathbf{M}_i\nabla\mu_i \tag{65}$$

where $\mathbf{M}_i$ is a mobility tensor so that

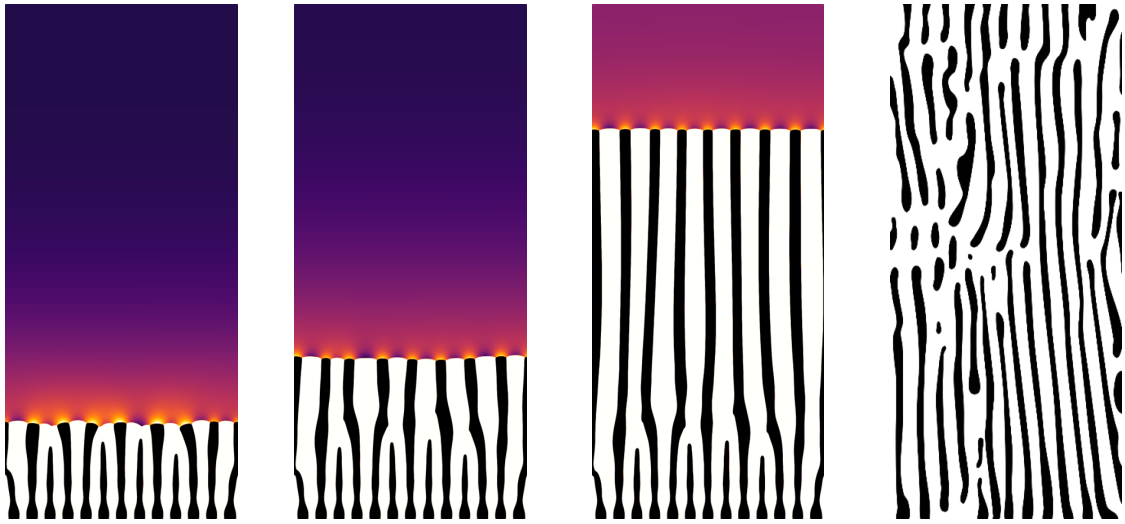
$$\dot{\mu_i} = \left(\sum_\alpha \phi_\alpha \frac{\partial^2\omega_\alpha}{\partial\mu^2}\right)^{-1}\left(\nabla(\mathbf{M}_i\nabla\mu_i) + \sum_\alpha \dot{\phi_\alpha}\frac{\partial\omega_\alpha}{\partial\mu}\right) \tag{66}$$

3.3.1 Parabolic

$$f_\alpha^{\text{bulk}} = \sum_i \epsilon_{\alpha i}(c_{\alpha i} - c_{0\alpha i})^2 \tag{67}$$

legendre transform

$$\omega_\alpha^{\text{bulk}} = \sum_i \frac{\mu_i}{2\epsilon_{\alpha i}} - \mu_i c_{0\alpha i} \tag{68}$$



(a) after 30 ms (b) after 50 ms (c) after 120 ms (d) experimental

Figure 2: Eutectic solidification - simulation and experiment

4 Simulations with OPStudio

Both in the trial-version and in the full-version, several short examples are provided to show the capability of OPStudio in quick-running simulations.

4.1 Eutectic solidification of Copper/Silver

A eutectic system features an equilibrium point, where two solid phases and the liquid phase are stable, at a lower temperature than the melting points of the pure elements. During casting of such a material, depending on the composition of the melt, the remaining melt will solidify into the two solid phases with a characteristic microstructure. With fine tuned solidification parameters, the mi-crostructure of the eutectic regions can be adjusted to the desired properties in texture and compo-sition. This eutectic microstructure is formed by cooperative growth. This occurs, if a single thermody-namic phase decomposes into two different thermodynamic phases and the composition has to be divided between the two new phases. Depending on the diffusion kinetics and the size of the micro-structure, this can lead to complex structures in the microstructure. Often a stable spacing of the growing phases can be found, where the two phases alternate and can only grow together, and the composition diffuses away from/or towards the thermodynamic phases. To show an example of this cooperative and diffusion limited growth, a eutectic solidification in a copper-silver system is chosen.

Here the eutectic system consists of liquid phase, a copper-rich phase and a silver-rich phase with a wide two-phase region between them, as can be seen from the phase-diagram in Fig. 2. Undercooling controls the driving force and diffusion kinetics, and therefore also controls the final microstruc-ture, as both parameters are vital to the evolution of the different phases. Undercooling is chosen to be 30°C in an alloy with a Cu-composition of $39at\%$, similar to the conditions in [18]. For a short simulation duration, the simulation domain is reduced to the dimensions of 0.35x0.8 µm (red rectangle in Fig. 2d. At the given parameters, such a small sample solidifies in 1.5ms. From the phase-diagram of the system it is visible, that the silver-phase (blue) and the copper-phase (red) nucleate in the melt (grey) below 780°C. The composition of the liquid phase ($39at\%$ Cu) has to be distributed between the two new phases. Diffusion processes have to move the Cu-composition from the silver-phase towards the copper-phase, as can be seen from the equilibrium phase-diagram:

The simulation is set up with an undercooling of 30°C into the two-phase region. The silver-rich phase will grow with the preferable composition of Cu, its equilibrium composition with respect to the liquid-phase. This value can be taken from the linearized equilibrium in Fig. 2 at around $16at\%$ Cu. The melt will enrich with alloying composition to maximum of $46at\%$. As soon as the growing silver-rich phase cannot maintain a maximum Cu-composition of $16at\%$, the driving force against liquid will decrease to

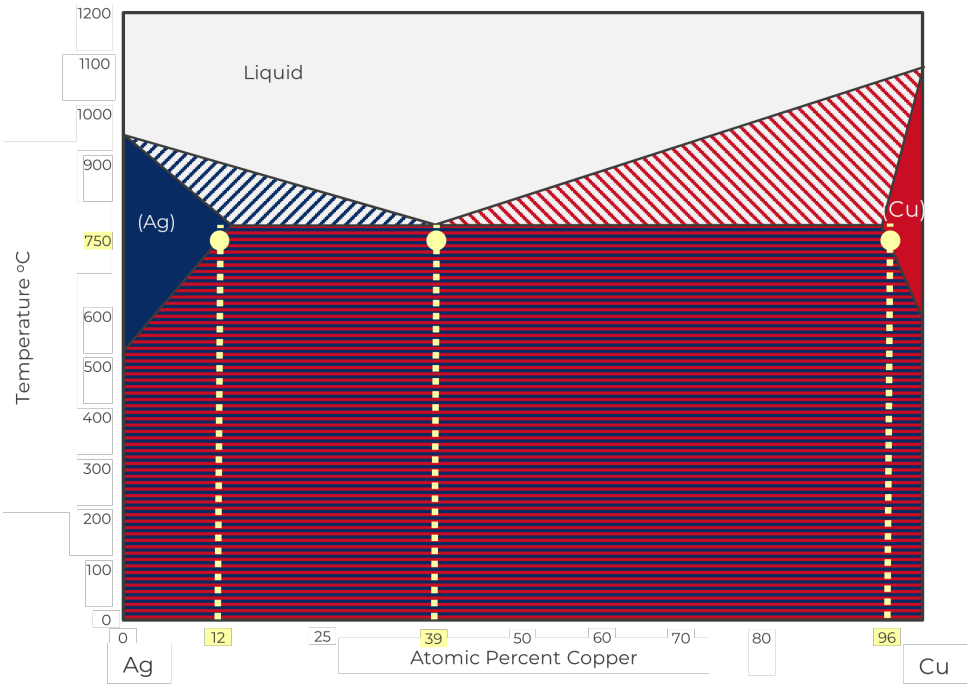


Figure 3: Phase-diagram of Ag-Cu

zero or might even go negative, in case of a phase-growth which overshot the equilibrium. This is the reason, why this eutectic system cannot solidify at this temperature with only a single thermodynamic phase. In front of the copper-rich phase, the Cu-composition in the melt decreases to a value of $33at\%$ Cu (not highlighted in the phase-diagram). Steady diffusion flux from Cu-rich liquid regions in front of the silver-rich phase to Cu-poor liquid regions in front of the copper-rich phase – however – enables a steady solidification front and a full solidification. This cooperative growth of a phase saturating the liquid and a phase desaturating the liquid with alloying composition, will be the core of this example. Thermodynamic data is provided by simple linearized phase-diagram data, supplied in the *.opid format. It describes the dual-phase region be-tween all phase-pairs with two linear lines. The data for the liquid/Cu-rich-phase is described as:

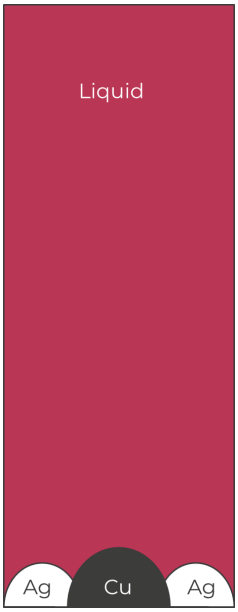
```
Key Description Value
$EC_0_1_CU  Equilibrium composition for liquid in equilibrium with Cu-rich phase.   0.39
$EC_1_0_CU  Equilibrium composition for Cu-rich phase in equilibrium with liquid.   0.95
$ES_0_1_CU  Slope of the line separating the liquid-phase and the dual-phase region.    510.0
$ES_1_0_CU  Slope of the line separating the copper-phase and the dual-phase region.    6055.0
$ET_0_1 Temperature at which the equilibrium-compositions are taken from.   1053.0
$ED_0_1 Entropy difference between both phases. Multiplier of the driving force.    1.0
```

Every equilibrium line is fixed with a point (EC,ET) and a slope (ES). Two lines (0_1 and 1_0) fully de-scribe the two-phase region. It is important to note, that the linearization only needs to be valid in the temperature and composition range of the simulation. Therefore, multiple instances of linear-ized data can exist for the same thermodynamic system. The entropy difference only serves as a multiplier, with which the driving force for this specific phase-pair is multiplied. In this setting its value can be safely regarded as 1.0. The data for the liquid/Ag-rich-phase is described as:

```
Key Description Value
$EC_0_2_CU  Equilibrium composition for liquid in equilibrium with Ag-rich phase.   0.39
$EC_2_0_CU  Equilibrium composition for Ag-rich phase in equilibrium with liquid.   0.14
$ES_0_2_CU  Slope of the line separating the liquid-phase and the dual-phase region.    -460.0
$ES_2_0_CU  Slope of the line separating the silver-phase and the dual-phase region.    -1333.0
$ET_0_2 Temperature at which the equilibrium-compositions are taken from.   1053.0
$ED_0_2 Entropy difference between both phases. Multiplier of the driving force.    1.0
```

Parameter	Value
Nx	35
Ny	0
Nz	80
dx	1E-8
iwidth	4.5
dt	1E-5
nSteps	150
T	1023 K
Sigma	0,25
Mu	1E-10 (1E-12)

(a) Simulation parameters



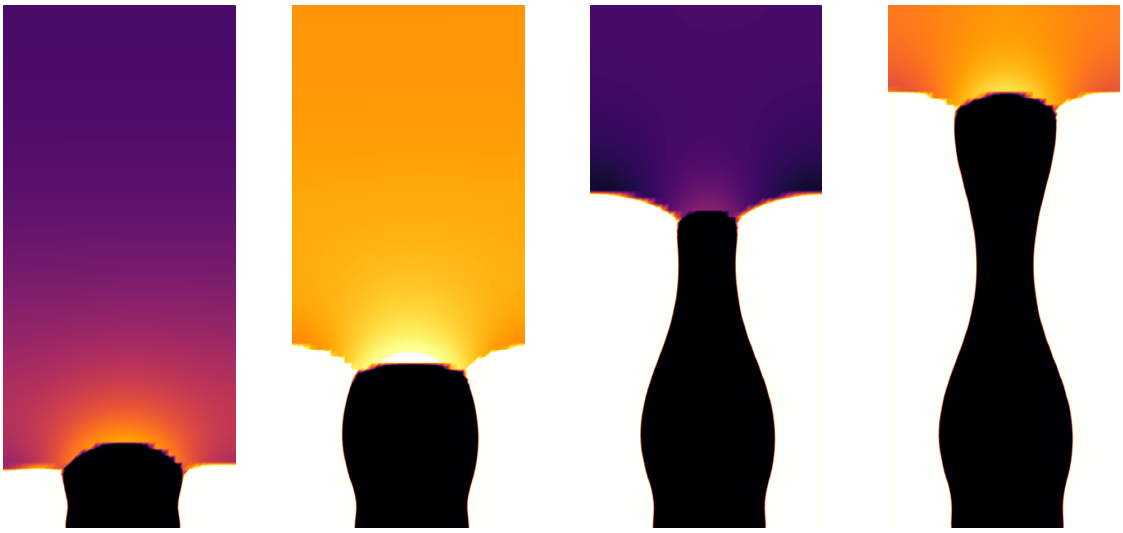
(b) Simulation set-up

Figure 4: Simulation parameters and set-up for the eutectic solidition of CuAg.

For solving the diffusion fluxes in the simulation, especially in the liquid-phase in front of the solidification front, chemical diffusion coefficients need to be supplied as well. The input data is provided in the same *.opid file and described as:

```
Key Description Value
$DC_0_CU_CU Chemical diffusion coefficient for CU in liquid-phase.  1E-9
$DC_1_CU_CU Chemical diffusion coefficient for CU in copper-rich phase. 1E-12
$DC_2_CU_CU Chemical diffusion coefficient for CU in silver-rich phase. 1E-12
```

With this notation, a full diffusion matrix can be used to describe the kinetics. Here only the diffusion coefficient of copper against the reference element silver is supplied, with identical values in the solid phases and an increased value for the liquid phase. The simulation domain is initialized with a single phase (liquid), with three spheres of solid phases at the bottom.



(a) after 0.05 ms (b) after 0.35 ms (c) after 0.8 ms (d) after 1.1 ms

Figure 5: Eutectic solidification - Simulation results of composition in the liquid

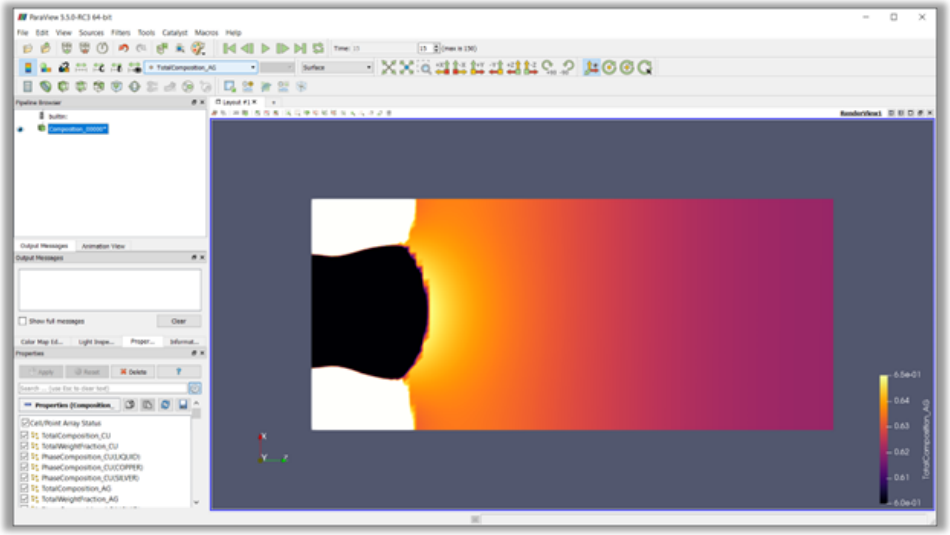


Figure 6: Visualization with ParaView

5 Graphical User Interface

5.1 Magnesium-Aluminum solidification

As an example application of OPStudio we take a look at the dendritic solidifaction of magnesium-aluminum alloys. These alloys are used in lightweight structural applications in automitive and aerospace engineering as well as high-end consumer electronics due to their low density. Their low nobility, however, can cause problems with corrosion. The Mg-Al cast alloys consist primary HCP dendrites, which are then enclosed by an interdendritic eutectic. A way to reduce corrosion is to ensure the enclosure of the primary dendrites by the eutectic and to prevent a network of primary dendrites in direct contact. Further information can be found in [6] and [7]. Here we take a look at how to setup such a solidification simulation in OPStudio. First we need an OPID file for the Mg-Al system, to create such a file please take a look at the OPID file creation tutorial that can be found under Help in the OPStudio GUI. For phases we need LIQUID, HCP and AL12MG17. For this tutorial the AM50.opid is used that can be found in the Presets folder. After starting the GUI you should see the window shown in figure 8, select Custom Project and then check Thermodynamics and click continue. Now select the OPID file under **Linearized phase diagram** and click **Load database** and then continue. Now the screen should look similar to 9, for the LIQUID phase set the aggregate state to liquid and click **Apply and continue**. For Boundary Conditions everything should be set to **Periodic**. In Interface Properties select **HEXBOETTGER** for the LIQUID-HCP interface for the stiffness model; set all interace energies to 0.1, the interface mobility should be set to 5e-6 for LIQUID-LIQUID, LIQUID-HCP and HCP-HCP, the other values should be set to 5e-10. For LIQUID-LIQUID, LIQUID-HCP and HCP-HCP set the activation energy to 6e4, this activates the Arrhenius-type temperature dependency as described in section 6.5. Also set the interface stiffness anisotropy to 0.35. In Driving Force, set driving force averaging to active; the cut off values can stay at 0.95.
In Runtime Control you can name your simulation and set the number of time steps, the time step and the output frequency. Internally the OPStudio will reduce the time step to ensure stability, but not necessarily accuracy. For this simulation set **Number of time steps** to 150, **Output to disk every** and **Output to screen every** to 1. The **time step** should be set to 5e-1. Choose a number of openmp threads appropriate to your processor in your machine. The remaining options can be left at the default value. In Settings set the grid spacing (dx) to 1e-6 and choose a system size; try **System size in X direction** and **System size in Y direction** 100 and 0 **System size in Z direction**. Also select consider nucleus volume and set the aggregate state of LIQUID to liquid (Note: In version 1.1.1.0 it is necessary to define this in two places, this will be fixed in future releases). In Composition we can set the initial composition for each phase, as only LIQUID is present in the beginning technically only the initial composition for LIQUID needs to set, but if any other phases are added to the initialization later on, we need the initial values for these phases as well. In this simulation we use LIQUID-AL = 0.15, LIQUID-MG = 0.85, HCP-AL = 0.01, HCP-MG = 0.99, AL12MG17-AL = 0.4 and AL12MG17-MG = 0.6. Otherwise leave everything else as default. DiffusionProperties, ThermodynamicFunctions and ThermodynamicInterface can be left unchanged. In Temperature set the Initial System Temperature to 902.3 K; and set the Latent Heat Mode to Global and the latent heats LIQUID-HCP and LIQUID-AL12MG17 to -15737. In Nucleation we can set the nucleation density and temperature ranges as well as size distributions. For this simulation we allow nucleation of HCP in LIQUID and AL12MG17 in LIQUID by selecting YES in the dropdown menu. As Input mode we use sites for the HCP-LIQUID pair and density for the AL12MG17-LIQUID pair. For the HCP-LIQUID pair we use 5 nucleation sites and for AL12MG17-LIQUID we use a density of 1e16. For AL12MG17-LIQUID the maximum temperature should be set to 700 K, the mimimum distance between HCP nuclei should be 10 grid points and between AL12MG17 should be 5 grid points. The other settings could be left on the default values. In Termination set AverageTemperature, Termination Value to 300 and Comparison to smaller, to stop the simulation once it reaches 300 K. Click **Apply and continue** and in the microstructure tab nothing needs to changed as the simulation should start with only LIQUID; and you can click **Apply and continue** again. Then click on **Start Simulation**, this will ask you to save the opi file and start the simulation. During the simulation you can click on Update data to show volumes and temperature evolution.

Figure 13 shows the temperature evolution during the simulation as well as the release of latent heat. The evolution of the phase fractions is shown in figure 14.

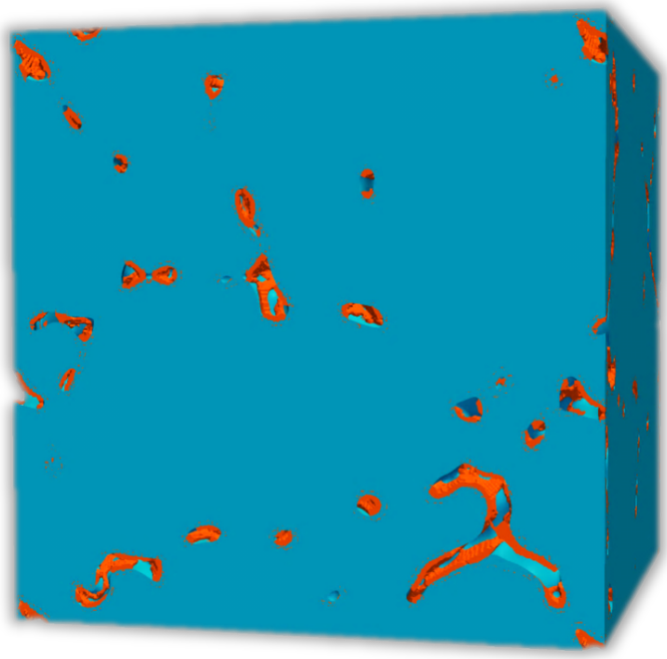


Figure 7: Microstructure of a simulation with 5% AL.

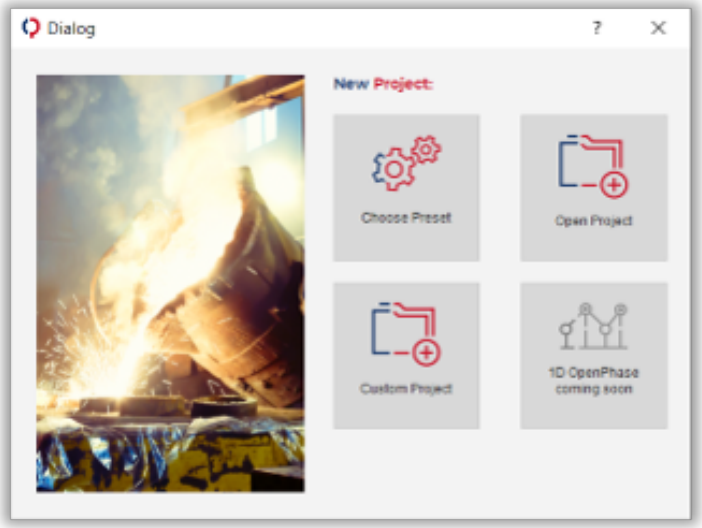


Figure 8: Starting screen of GUI.

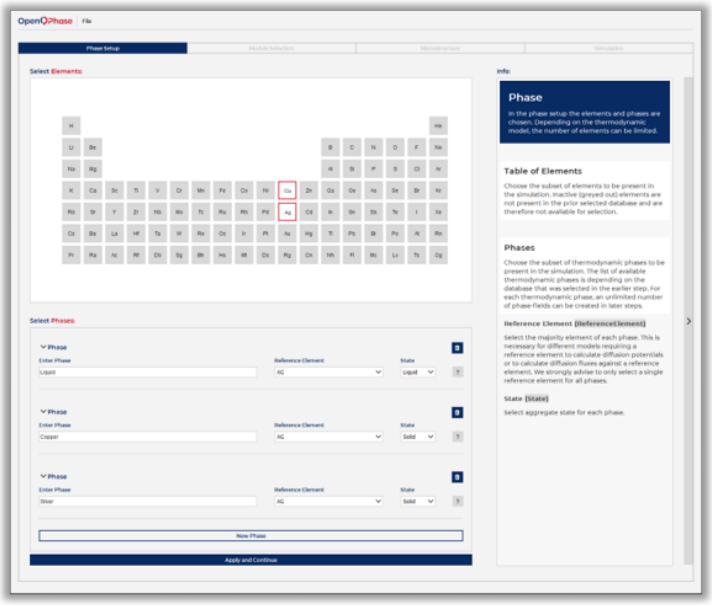


Figure 9: Element and Phase selection.

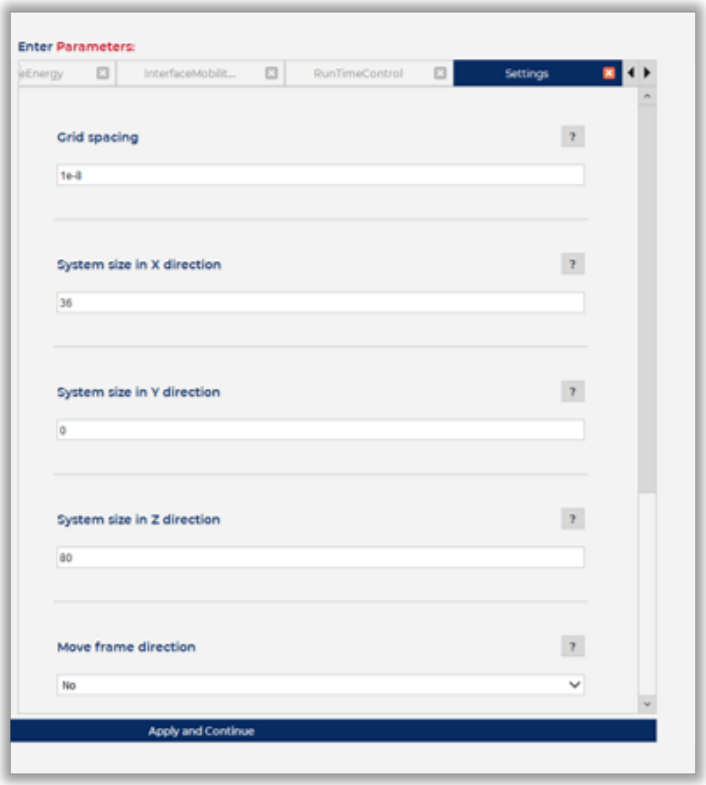


Figure 10: Settings of grid properties

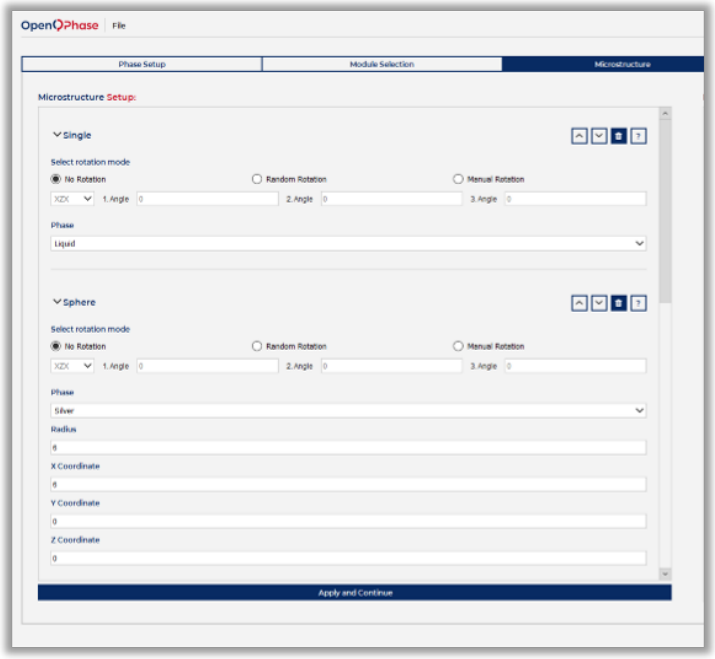


Figure 11: Initial microstructure selection

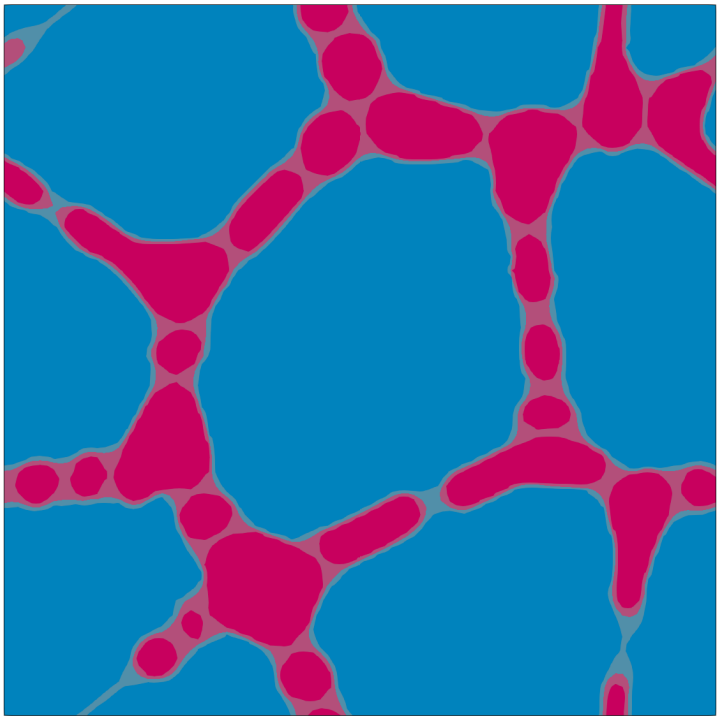


Figure 12: Microstructure evolution during simulation

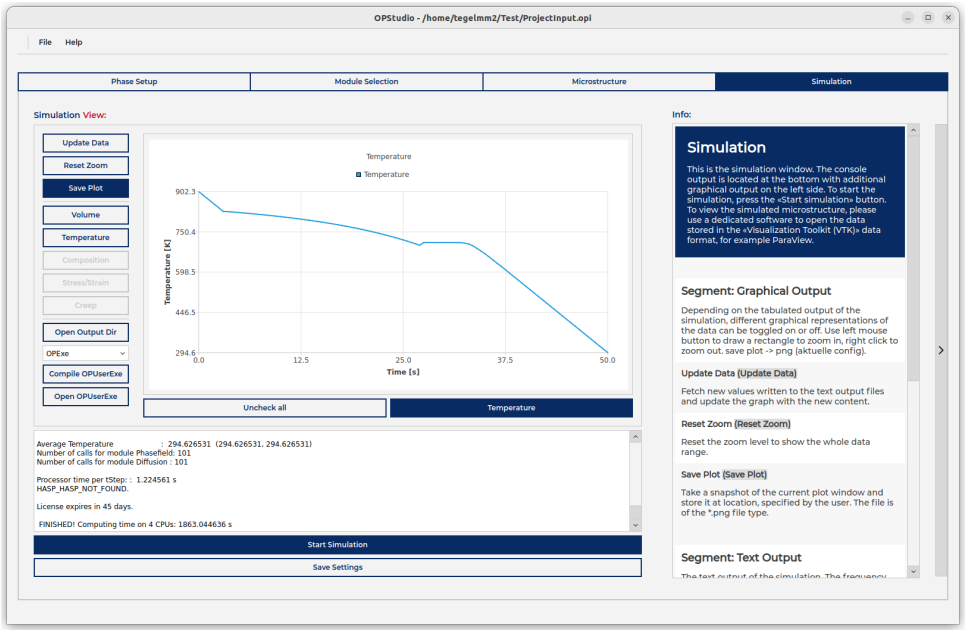


Figure 13: Temperature

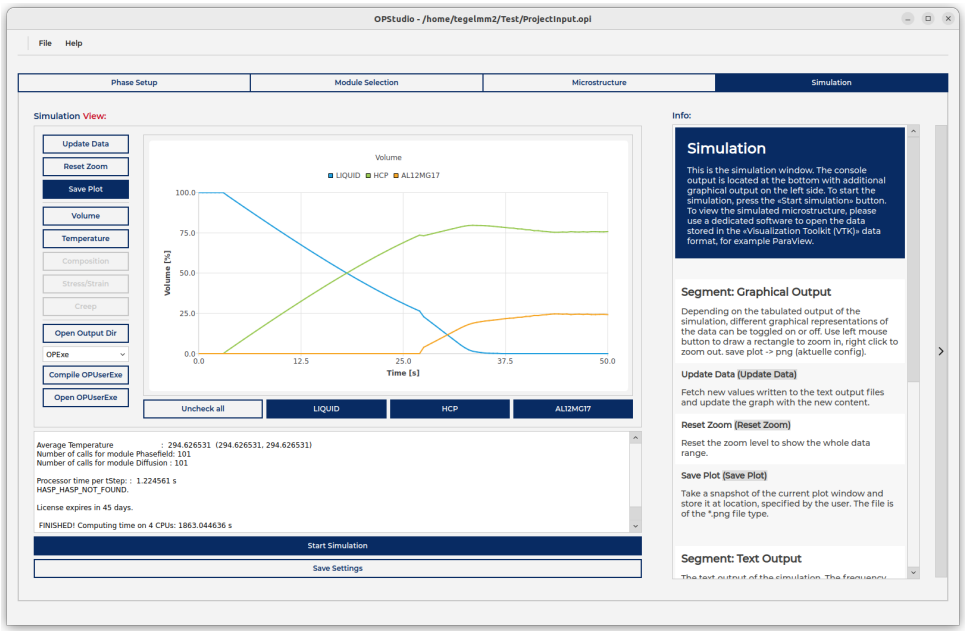


Figure 14: Phase fractions

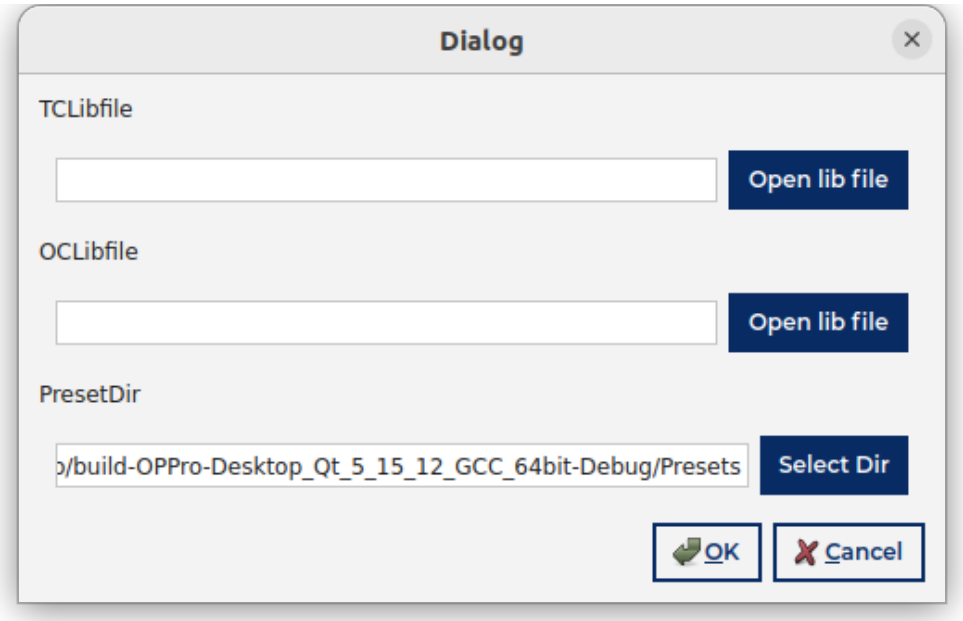


Figure 15: Select the TQ shared object or the TQ dll.

5.2 Thermo-Calc Coupling

The coupling with Thermo-Calc allows simulations with any number of elements and phases. It allows the direct utilization of thermodynamic and kinetic data from databases, without the need of creating linearized phase diagrams. Thermodynamic and kinetic data is pulled from database as needed during the simulation. In order to couple OPStudio to Thermo-Calc the TQ-Interface is required and needs to be linked in the graphical user interface of OPStudio. To setup the TQ-Interface open the GUI, and navigate to **File/Settings** and under TCLibFile click on "open" to either select the tq shared object on Linux or the tq dll on Windows as seen in figure 15. These files are usually located in

`Documents/ThermoCalc/[Version]/SDK/TQ`

but it can depend on your Thermo-Calc installation. The file will probably exist in your Thermo-Calc installation, but it requires a separate license. Without the TQ-Interface license, the coupling to Thermo-Calc will not work. You can verify the TQ-Interface license if you open Thermo-Calc and go to **Help/Show License Info**, if you see TC_TQ in the list you have access to the TQ-Interface.

In order to setup asimulation with Thermo-Calc coupling, start the OPStudio GUI and select **Custom Project**, then check **Thermodynamics** as seen in figure 16 and then choose a GES5 to load as seen in figure 17. In section 5.2.1 instruction for the creation of GES5 files are provided. The GES5 files can only be opened on a machine that also has access to license for the underlying database. At the moment the OPStudio GUI will close if the license is not found.

5.2.1 GES5 File Creation

The following shows the steps to create a GES5 file using Thermo-Calc. As an example we try to create a GES5 file for a ternary system of AL-CR-NI with the phases LIQUID and FCC_L12.

- Open Thermo-calc applicationand switch to console mode
- Execute the following commands:
  1. Go to thermodynamic database module
     - **SYS: go da**
  2. Switch the data base to see all the available data bases
     - **SYS: sw da**

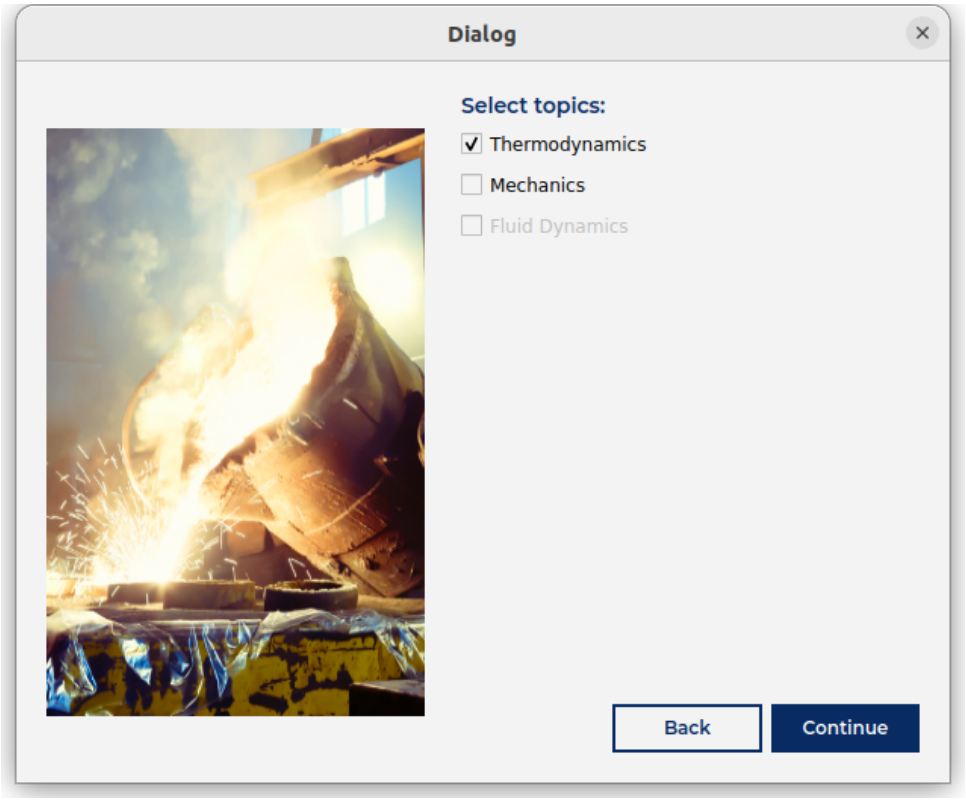


Figure 16: Activate Thermodynamics

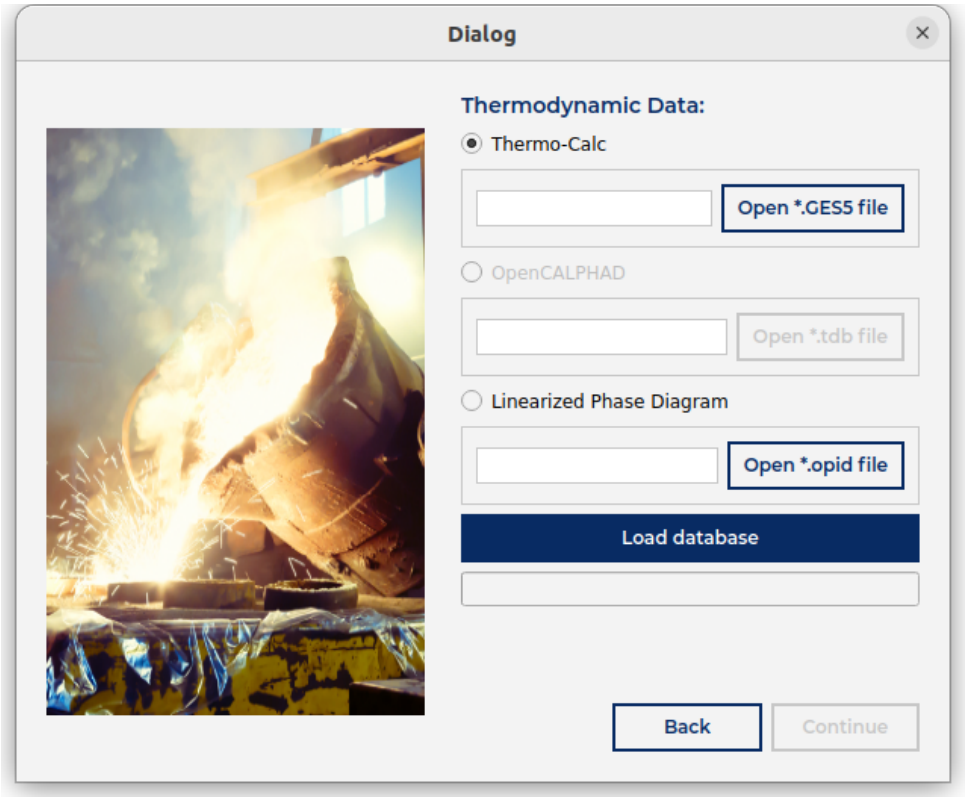


Figure 17: Choose a GES5 file and load database.

3. Select the database according to the alloy system
   - **DATABASE NAME: TCNI9**
4. Define the system that includes elements and phases.
   - **TDB_TCNI9: def-sys**
5. Select the alloying elements in alphabetical order.
   - **ELEMENTS: AL CR NI**
6. Reject all the phases to select the required phases in next step.
   - **TDB_TCNI9: rej ph***
7. Restore the phases of interest.
   - **TDB_TCNI9: res ph LIQUID:L FCC_L12**
8. Now get data for the selected system from the chosen database.
   - **TDB_TCNI9: get**
9. To include the kinetic mobility data in the GES5 file append the mobility data.
   - **TDB_TCNI9: App mob**
10. Select the appropriate mobility database.
    - **DATABASE NAME: MOBNI5**
11. Again define the system
    - **APP: def-sys**
    - **ELEMENTS: AL CR NI**
12. Reject all the phases to select the required phases in next step.
    - **TDB_MOBNI5: rej ph***
13. Restore the phases of interest.
    - **TDB_MOBNI5: res ph LIQUID:L FCC_L12**
14. Get the data
    - **APP: get**
15. Now, go to Gibbs Energy System module to save the GES file.
    - **APP: go to**
    - **MODULE NAME: GIBBS_ENERGY_SYSTEM**
16. Save the GES5 file at the required location.
    - **GES: save-GES**

5.3 OPUserExe

OPStudio allows the implementation of user defined functions and classes, that can be used in conjunction with the pre-compiled extension OPDiff and OPMech. Currently this can only supported on Linux machines, but can be used on Windows machines via virtual machines such as VirtualBox, see sections 5.4.

Part of the installation is the OPCore folder, that contains the source code for the academic part of OpenPhase. Changes or additional functions could be implemented directly in the source code of OPCore, however this might make support difficult if problems arise from changes to this source, also you might need to adapt your changes when new versions of OPStudio and OPCore are released. The recommended way to implement user functions and classes is the use of the **UserClass.h** and **OPUserExe.cpp** that are located in **usr/bin**, and contain boilerplate code.

As an example we will implement a cooling and heating cycle following a cosine function.

$$T(x,t) = T_0 + \hat{T}(\cos(2\pi t) - 1), \tag{69}$$

with the start temperature T_0 and the amplitude $\hat{T}$.
For this we include **OPCore/include/Base/Includes.** to get the basic definitions of OpenPhase and **OPCore/include/Temperature.h** as we want to modify the temperature. We will use the namespace openphase and define the UserClass. Do not rename this class, it is a member of the **StudioClass.h** that is part of the pre-compiled extensions, see **OPCoreExtensions/include/StudioClass.h** In the private part we store two variables **StartTemperature** and **TemperatureAmplitude**, that are set in the **ReadInput** function. The ReadInput function takes a stringstream that contains the data in the opi file created by the OPStudioGUI. The **FindModuleLocation** function finds the index at which the definition of the **UserClass** begins in the opi file. The **ReadParameterD** functions find the next instance of **$StartTemperature** and **$TemperatureAmplitude** in the opi file and read double that is defined there. The 4th argument determines if the input is mandatory and the 5th argument is the default value if the input is not defined in the opi file.
The **Initialize** function can be used to allocate memory, but it is not needed for this example. The **SetTemperatureCycle** function sets the temperature according to equation 69. It also sets the boundary conditions and calculates the average, minimum and maximum temperature in the simulation domain, as these values are used in output and certain modules such as nucleation.
OMP_PARALLEL_STORAGE_LOOP_BEGIN and **OMP_PARALLEL_STORAGE_LOOP_END** are preprocessor macros that allow to loop over a Storage3D with the indices i, j, k. In this case the Storage3D is **Tx.Tx**, which is the storage of temperatures in grid points i, j, k and has a number of **Tx.Tx.Bcells()** grid points of boundary data. The loop is automatically subdivided by OpenMP and processed concurrently, the 6th argument allows additional prepressor instructions for OpenMP, for example to implement reductions.

```
#include "../../OPCore/include/Base/Includes.h"
#include "../../OPCore/include/Temperature.h"
namespace openphase
{
    class UserClass
    {
        public:
        void ReadInput(std::stringstream& inp)
        {
            int moduleLocation = UserInterface::FindModuleLocation(inp, "UserClass");
            StartTemperature = UserInterface::ReadParameterD(inp, moduleLocation, "StartTemperature", false, 0.);
            TemperatureAmplitude = UserInterface::ReadParameterD(inp, moduleLocation, "TemperatureAmplitude", false, 0.);
        }
        void Initialize() //Incase any storages need to be allocated
        {
        }
        void SetTemperatureCycle(double time, Temperature& Tx, BoundaryConditions& BC)
        {
            OMP\_PARALLEL\_STORAGE\_LOOP\_BEGIN(i,j,k,Tx.Tx,Tx.Tx.Bcells(),)
            {
                Tx(i, j, k) = StartTemperature - TemperatureAmplitude + TemperatureAmplitude*cos(2.*M\_PI*time);
                //M\_PI is approximately pi
            }
            OMP\_PARALLEL\_STORAGE\_LOOP\_END
            Tx.SetBoundaryConditions(BC);  //Update data in boundary.
            Tx.SetMinMaxAvg();
            /*
            Update the minimum, maximum and average in the system,
            used for example in output and nucleation
            */
        }
        private:
        double StartTemperature;
        double TemperatureAmplitude;
    };
}
```

Now we only need to modify the **OPUserExe.cpp** to include the **ReadInput** and **SetTemperatureCycle** functions. First we search for the **Tx.Set** function inside the **StudioClass::SolveUser** function and comment it out in order to deactivate the standard function that sets the temperature and instead call the **SetTemperatureCycle** function in the UserClass **UC**.

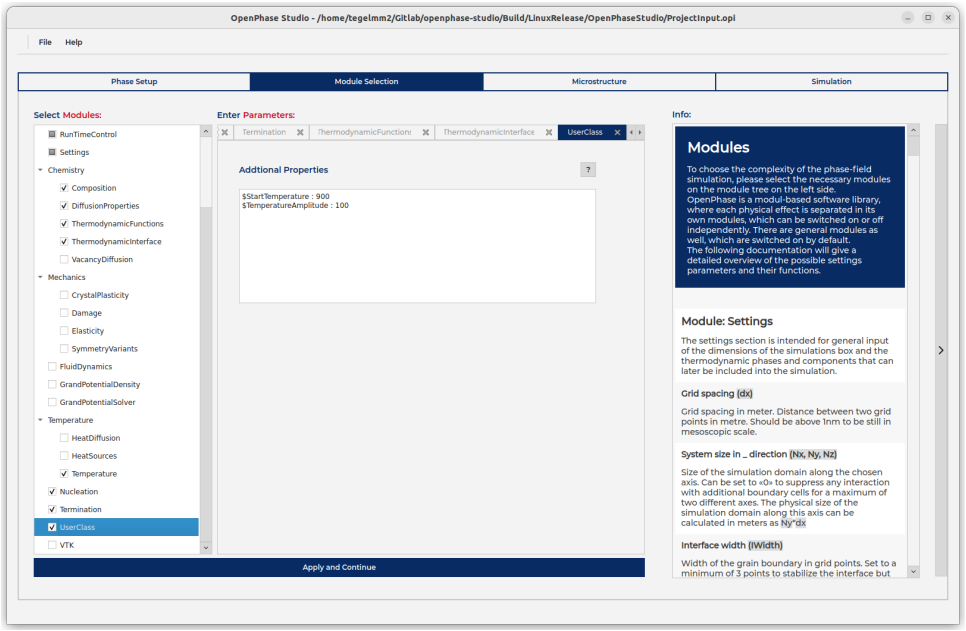


Figure 18: Add input paramters

```
if((actID[(int)activeID::Phasefield]) and (locID[(int)localeID::Phasefield] and dt != 0.0))
{
    //Tx.Set(BC[(int)bcID::Temperature],Phi,Cx,TP,time,dt);
    UC->SetTemperatureCycle(time,Tx, BC[(int)bcID::Temperature]);
    if (actID[(int)activeID::HeatDiffusion])
    {
        HS.Activate(Phi, Tx, RTC);
        HS.Apply(Phi, Tx, HD);
        if(actID[(int)activeID::Chemistry])
        {
            HD.SetEffectiveProperties(Phi, TP, Cx, Tx);
        }
        else
        {
            HD.SetEffectiveProperties(Phi, Tx);
        }
        HD.SolveImplicit(Phi,BC[(int)bcID::Temperature],Tx,dt);
        Timer.SetTimeStamp("HD.SolveImplicit");
    }
}
```

The ReadInput function is already implemented in the boilerplate code, it is called after **OPS.InitializeAndReadInput**, which initializes all needed modules and reads their input data.

```
OPS.InitializeAndReadInput(data);
OPS.UC->ReadInput(data);
```

Now we need to enter the input data, to do this go to **UserClass** in the Module Selection tab and add the two parameters under **Additional Properties**. The syntax for this is

```
$key description : value
```

The key has to be a word that does not contain any spaces. Anything between the first space and the colon can be used as a description and is printed out when reading the data. The value is given after the colon. In figure 18 the StartTemperature and TemperatureAmplitude are added. In order to run the simulation we first need to compile the OPUserExe by clicking **Compile User Exe** on the left side of the Simulation tab. The compilation log will be printed to the log on the bottom of the screen. This also recompiles OPCore with any changes that were made there. To use the OPUserExe in the simulation select OPUserExe instead of OPExe in the dropdown menu on the left as seen in figure 19. Then proceed to Start Simulation.

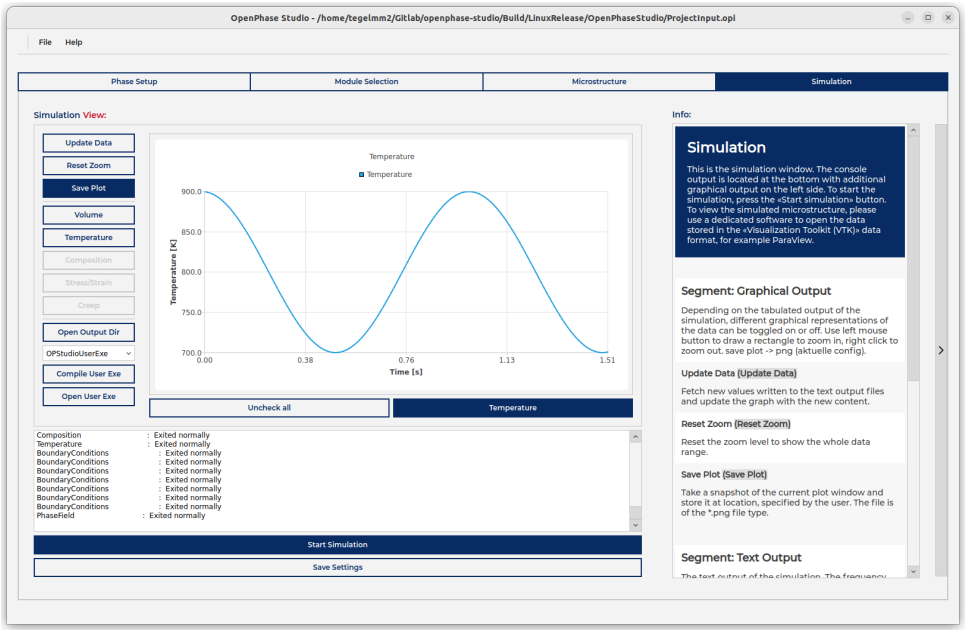


Figure 19: Compile and run OPUserExe

5.4 OPStudio and VirtualBox

If you have a windows machine and want to make use of the OPUserExe a possible solution is to install Linux in a virtual machine. OPStudio has been tested on Ubuntu 22.04 LTS (`https://ubuntu.com/` on VirtualBox 7.0 (`https://www.virtualbox.org/`), also see `https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using-virtualbox#1-overview`. In order to use OPStudio you might need to activate the license dongle in the virtual machine, by selecting SafeNet Sentinel under Devices/USB in the control panel of VirtualBox.

6 Input Parameters

6.1 Module: Settings

The settings section is intended for general input of the dimensions of the simulations box and the thermodynamic phases and components that can later be included into the simulation.

- **Grid spacing (dx)**
  - Grid spacing in meter. Distance between two grid points in real units. Should be above 1nm to be still in mesoscopic scale.
- **System size in [X, Y, Z] direction (Nx, Ny, Nz)**
  - Size of the simulation domain along the chosen axis. Can be set to ”0” to suppress any interaction with additional boundary cells for a maximum of two different axes. The physical size of the simulation domain along this axis can be calculated in meters as Nx*dx.
- **Interface width (IWidth)**
  - Width of the grain boundary in grid points. Set to a minimum of 3 points to stabilize the interface but should not be greater than 5. Defaulted to 4.5, allowing the interface to vary between 4 and 5 grid points.
- **Move frame direction (MoveFrameDirection)**

- MoveFrameDirection is used for simulating directional solidification. It can be set to a coordinate axis, which should coincide with the growth direction of the dendrite.

- **Move frame position (MoveFramePosition)**
  - MoveFramePosition sets the position, which the phase defined in (MoveFramePhase) needs to reach in order for the frame to move in MoveFrameDirection.

- **Move frame phase (MoveFramePhase)**
  - MoveFramePhase determines the index of the phase that is used for moving the frame, see description of (MoveFrameDirection) and (MoveFramePhase).

- **Consider Nucleus Volume(ConsiderNucleusVolume)**
  - Considers the volume of nuclei in the calculation of the prefactor of the phase field equation. Use with caution.

- **Aggregate State (State)**
  - Sets the aggregate state for each thermodynamic phase. This changes how the phase is treated in nucleation, calculation of interface properties and fluid dynamics.

6.2 Module: RunTimeControl

The section titled RunTimeControl contains the different input parameters to control the simulation time and the frequency of the desired output, as well as information about the parallelization of OpenPhase.

- **Simulation Title (SimTtl)**
  - Title of the simulation. Can be a string with blank spaces and special characters.

- **Number of timesteps (nSteps)**
  - Number of total iterations for the phase-field simulation. nSteps multiplied with dt will give the total simulated time.

- **Output to disk every (FTime)**
  - Frequency to write VTK/VTS files to disk in _ number of time steps.

- **Output to screen every (STime)**
  - Frequency to write Screen output to the terminal. Can be set to a higher value than "1" to reduce the amount of output.

- **Timestep (dt)**
  - Time in real units between each phase-field iteration. Value in seconds. Chose a value that is small enough that it still satisfies the different stability criteria but high enough to get the desired simulation speed.

- **OpenMP threads (nOMP)**
  - Number of OpenMP threads for parallelization. Do not exceed number of threads of your local working machine for optimal performance.

- **Restart output every (tRstrt)**
  - Frequency to write restart files to disk in number of time steps.

- **Restart switch (Restrt)**
  - Restart switch. When set to "Yes", simulation will be restarted from iteration tStart if restart files are available. Defaults to "No".

- **Restart timestep (tStart)**
  - See description for Restrt. If Restrt is set to "Yes", this value will determine at which point to restart. Binary RAW-Data has to be available for exactly the same time step.
- **Synchronize all active modules(SynchronizeTimeSteps)**
  - Activating this option synchronizes all active modules to the smallest time step. This is especially useful for situations in which the phase-field time step is a lot larger than the diffusion time step and compositions between phases are far apart. In that case, every phase-field step releases a lot of composition, which has to be taken care of by the diffusion.
- **Refinement of the phase field step(PhaseFieldTimeFactor)**
  - The PhaseFieldTimeFactor determines how much more often the phase-field is solved than the built-in stability criterion determines. The factor should only be used if stability problems in the phase-field occur.
- **Refinement of the diffusion step(DiffusionTimeFactor)**
  - The DiffusionTimeFactor determines how much more often the diffusion is solved than the built-in stability criterion determines. The factor should only be used if stability problems in the diffusion occur.
- **Refinement of the heat diffusion step(HeatDiffusionFactor)**
  - The HeatDiffusionFactor determines how much more often the heat diffusion is solved than the built-in stability criterion determines. The factor should only be used if stability problems in the heat diffusion occur.
- **Heart beat (HeartBeat)**
  - A status report will be printed to the console every _ seconds. "0" to deactivate.

6.3 Module: BoundaryConditions

For simulations with repeating periodic microstructure or other effects that require complex boundary condition, this section includes all necessary input.

- **[X, Y, Z] axis beginning boundary condition (BC0X, BC0Y, BC0Z)**
  - Boundary condition along chosen axis. Can bet set to "Fixed", "NoFlux" and "Periodic". With "Periodic" boundary conditions the domain will be repeated along this axis. With "NoFlux" boundary conditions the first derivative of any field will be zero at the boundary. The third condition ("Fixed") will keep the initial values, for example to force a certain composition on a certain layer.
- **[X, Y, Z] axis far end boundary condition (BCNX, BCNY, BCNZ)**
  - For more information, see BC0X, BC0Y, BC0Z above. If "Periodic" was specified above, it has to be chosen here as well. The other boundary conditions can be intermixed, but "Periodic" needs to be applied for the beginning and far end condition.

6.4 Module: DrivingForce

DrivingForce can be used to individually tune the stability of the interface region and therefore the stability of the whole simulation. Driving-force averaging can be turned on/off and the maximum allowed local value for the driving-force can be changed.

- **Driving force averaging (Averg)**

  - Average the local driving force values perpendicular to the interface to avoid interface-spreading and further stabilize the interface region. Default to Yes.

- **Driving force cutoff (CutOff)**
  - The factor, by how much of the allowed driving force the local values are cut in order to stabilize the interface. The allowed driving force depends on the interface energy. Default to 0.95.

- **Noise (NoiseMode)**
  - Adds a random noise on the driving force. This is mostly used for simulating small disturbances in the composition in solidification simulations, in oder to provoke branching of dendrites. Modes are Zero, Relative and Absolute. Zero switches Noise off, Relative adds a random value between - (Noise) and (Noise) to the driving force, Absolute multiplies the driving force with a random value between 0 and (Noise).

- **Noise intensity (Noise)**
  - Intensity of the random noise for each phase pair, see (NoiseMode).

- **Noise intensity (Noise)**
  - Intensity of the random noise for each phase pair, see (NoiseMode).

- **Phase-Field Stabilization (Stabilization)**
  - Experimental mode, that does not limit the driving force. May give incorrect results with triple or higher order junctions.

- **Sharp Phase-Field (SharpPhaseField)**
  - Sharp phase field model proposed by Alphonse Finel, allows for an reduced interface width. Still in experimental mode.

6.5 Module: InterfaceProperties

This section contains the information on the interfacial properties, energy and mobility, between the different phases and grain-boundaries of the same phases. Anisotropic behaviour can be switched on, as well as Arrhenius-type temperature dependencies.

- **Stiffness Model (EnergyModel)**
  - Allows the selection of interface stiffness model, see section 3.1.3.

- **Interface energy (Sigma)**
  - Interface energy for each phase pair.

- **Interface anisotropy (Eps)**
  - Anisotropy factor, should be chosen between 0.0 and 1.0. The interface anisotropy depends on the selected crystal system. Defaults to 0.

- **Mobility Model (MobilityModel)**
  - Allows the selection of interface mobilitymodel, see section 3.1.3.

- **Interface mobility (Mu)**
  - Interface mobility for each phase pair.

- **Interface anisotropy (Eps)**

- Anisotropy factor, should be chosen between 0.0 and 1.0. The interface anisotropy depends on the selected crystal system. Defaults to 0.

- **Activation energy (AE)**
  - If set to a different value than 0.0, the interface mobility is calculated with an Arrhenius-equation. Mu_PHASE0PHASE1 is then multiplied with $e^{\frac{-AE}{RT}}$. Defaults to 0.

- **Mobility Diffusion Limited (MobilityDiffusionLimit)**
  - Experimental mobility estimation for the diffusion limited case, from Steinbach, Topical Review: Phase-field models in materials science, 2009.

- **Mobility Diffusion Limited Stoichiometry(MobilityDiffusionLimitStoichiometry)**
  - Experimental mobility estimation for the diffusion limited case, from Steinbach, Topical Review: Phase-field models in materials science, 2009. Additional support for stoichiometry.

- **Adaptive Mobility(AdaptiveMobility)**
  - Automatically adapts the interface mobility to be as large as possible, while avoiding an oscillating driving force. Uses the given interface mobility as a starting value. Experimental.

6.6 Module: UserDrivingForce

The UserDrivingForce can be used to apply a constant driving force or a simple driving force using

$$-\frac{T - T_{eq}}{T_{eq}} L_{\alpha\beta}, \tag{70}$$

With temperature T, equilibrium temperature T_{eq} and latent heat $L_{\alpha\beta}$ for the phase transition between phases α and β.

- **User DrivingForce Mode (UDF_Mode)**
  - Selection of the User Driving Force Mode for a phase pair. Use value for a constant driving force or formula for the above formula.

- **Value (UDF_Value)**
  - Value of the constant driving force.

- **Latent Heat (UDF_LatentHeat)**
  - Latent heat $L_{\alpha\beta}$ for the phase transition between phases α and β.

- **Equilibrium Temperature (UD_T0)**
  - The equilibrium temperature that is used in above formula.

6.7 Module: Composition

The composition section holds all the necessary values for initialization of the complete composition field.

- **Initial Concentration (C0)**
  - For all phases listed in Settings a phase-composition has to be given for each element. The unit is in mole-fraction, and all phase-compositions of a phase have to sum up to unity. If the finite-interface-model is chosen initially, the site-fractions of each element have to be specified for each phase.

- **Composition profile type(ProfileType)**

  - A composition profile can be set for each phase. This is especially useful for simulating diffusion couples. Profile types are: Uniform, StepWise, Linear and DoubleLinear. StepWise is a step function between C0 and C1, Linear initializes a composition gradient from C0 to C1 and when using DoubleLinear, the composition is initialized with two linear gradients from C0 to C1 and back to C0.

- **Second concentration for profile(C1)**
  - Second concentration for initializing (ProfileType) other than Uniform.
- **Axis for composition profile(Axis)**
  - Axis, along which the composition profile defined in (ProfileType) is set.
- **Profile region global(Global)**
  - Select this in order for the composition profile defined in (ProfileType) to be calculated relative to the whole simulation domain. If this is not selected, the composition profile is calculated relative to the region in which the phase is present.
- **Point of interest on axis(VolumeLeft)**
  - This parameter determines where the point of interest (the step for profile StepWise, the lowest/highest point for profile DoubleLinear) for the profile selected in (ProfileType) is located.
- **Minimum mole fractions (Cmin)**
  - Minimum value of each phase-composition. Defaulted to 0. If CALPHAD coupling is used, minimum and maximum values are calculated from the sublattice models given from CALPHAD-coupling. Can be set to a more strict value to force a higher minimum amount of said element.
- **Maximum mole fractions (Cmax)**
  - Same as Cmin, defaulted to 1.
- **LocalCompositions**
  - Local Composition defines cuboids from points Upper limit and Lower limit in which a source or sink for the specific concentrations applied. The value given is the rate of change for each element and has to sum up to zero.

6.8 Module: Elasticity

Input for elasticity module from loading conditions to elastic properties.

- **Strain Accuracy (StrainAccuracy)**
  - This parameter describes the accuracy of strain convergence, that means the iterations stop when max strain change in a given iteration falls below this number.
- **Maximum Iterations (MAXIterations)**
  - Maximum Iterations prevents infinite iterations if the solver cannot converge to desired accuracy.
- **Elasticity Model (EModel)**
  - Homogenization of elastic coefficients in the interface region between multiple different phases. Available homogenization modes are "Khachaturyan" and "Steinbach".
- **Strain Mode (SMode**
  - Option to select the strain mode. Available options are "Small", "GreenLagrange", "Hencky" and "Bazant". "GreenLagrange", "Hencky" and "Bazant" require purchase of OPDiff.

- **Boundary Conditions [X, Y, Z, XY, XZ, YZ] (BCX, BCY, BCZ, BCXY, BCXZ, BCYZ)**
  - Loading state of the simulation domain, default to "FreeBoundaries", but can be loaded with "AppliedStrain", "AppliedStrainRate" or "AppliedStress". Default to "FreeBoundaries".
- **BCValue [X, Y, Z, XY, XZ, YZ] (BCValueX, BCValueY, BCValueZ, BCValueXY, BCValueXZ, BCValueYZ)**
  - If strain or stress is applied in the boundary cells, the value is given through this parameter. Default to 0.0.
- **Restrict (Restrict)**
  - Options are "None", "AspectRatio" and "Shear".
- **Elastic Constant Mode (ElasticConstantMode)**
  - The elastic coefficients can be entered in two modes, either directly as the full "tensor" or via "elastic moduli", "Bulk modulus", "Young's modulus", "First Lame", "Shear modulus", "Poisson's ratio", "P-wave modulus". If "elastic moduli" is selected two of these have to be provided.
- **Elastic coefficients (C)**
  - Elastic coefficients matrix entries. For $I = 1, 2, \ldots, 6$ and $J = 1, 2, \ldots, 6$ in Pascal.
- **Transformation Stretches (U)**
  - Transformation- or Eigenstrain of the phase written in matrix form $3x3$.
- **Consider External Forces (ConsiderExternalForces)**
  - Activates external forces such as magnetic forces.
- **Chemo-Mechanical Coupling (ChemoMechanicalCoupling)**
  - Composition change can affect elastic coefficients – and stress can affect diffusion fluxes.
- **Reference concentration (Cref)**
  - Reference composition, used for calculating the change in Eigendeformation and elastic coefficients is calculated, when Lambda or Kappa values are specified.
- **Kappa (Kappa)**
  - Coefficients of linear dependency of the elastic coefficients on composition. The calculation uses the equation $C^*_{ij} = C_{ij} + \kappa_{ij}(Cx - Cref)$.
- **Lambda (Lambda)**
  - Coefficients of linear dependency of the Eigendeformation on the composition. The calculation uses the equation $U^*_{ij} = U_{ij} + \lambda_{ij}(Cx - Cref)$.
- **Thermo Mechanical Coupling (ThermoMechanicalCoupling)**
  - Temperature change can affect elastic coefficients – and stress can affect diffusion fluxes.
- **Reference temperature (Tref)**
  - Reference composition, used for calculating the change in Eigendeformation and elastic coefficients is calculated, when Lambda or Kappa values are specified.
- **Gamma (Gamma)**

  - Change of elastic constants by a change in the temperature. The calculation uses the equation $C_{ij} = \gamma_{ij}(Tx - Tref)$.

- **Alpha (Alpha)**
  - Change of Eigendeformation by a change in the temperature. The calculation uses the equation $U_{ij} = \alpha_{ij}(Tx - Tref)$.

- **Neuber correction (NeuberCorrection)**
  - The Neuber-rule is a simple model to correct mechanical stresses when simulating without crystal plasticity. The model is described in detail in Neuber HH. Theory of Stress Concentration for Shear-Strained Prismatical Bodies with Arbitrary Nonlinear Stress-Strain Law. ASME. J. Appl. Mech. 1961;28(4):544-550. Here a simplified plastic behavior has to be specified.

- **Neuber Youngs modulus (YoungsModulus)**
  - Young's modulus of the simplified plastic behavior.

- **Neuber Yield strength (YieldStrength)**
  - Yield strength of the simplified plastic behavior.

- **Neuber hardening (NeuberHardening)**
  - Hardening coefficient of the simplified plastic behavior.

6.9 Module: Nucleation

Nucleation input handles nucleation location, density and distribution.

- **Nucleation of origin phase in target phase (Allowed)**
  - "Yes", "No", "Bulk", "GB" or "bottom". Specified whether a phase is allowed to nucleate in a matrix phase. "Bulk" enables nucleation only in bulk points, "GB" only in the interface (GrainBoundary) between the two respective phases. "Bottom" is designed for directional solidification, with this option, nucleation happens only for z-coordinate = 0. Only if a nucleation event is allowed, the remaining parameters for the phase pair need to be specified. Default to No.

- **InputMode (IMode)**
  - Determines the number of Nucleation either as a number of "Sites" or as a "density".

- **Nucleation Sites (Nsites)**
  - Determines the number of nucleation sites if "Sites" is selected as the Input Mode.

- **Particle density (Density)**
  - Nucleation density of Nucleus in Matrix in m^{-3}.

- **Nucleation density relative to phase volume (RelDensity)**
  - If yes, the nucleation density is relative to the volume of the target phase, otherwise it is relative to the whole simulation domain.

- **Minimum temperature (Tmin)**
  - Minimum temperature for nucleation of Nucleus in Matrix.

- **Maximum temperature (Tmax)**

  - Maximum temperature for nucleation of Nucleus in Matrix.

- **Minimum distance between nuclei (Shielding)**
  - Distance from a nucleation site in grid points, where no additional nucleation is permitted. Defaulted to two times interface width. Default to 2*iWidth.

- **Grain orientation mode (Orientation)**
  - Options are "Random", "Parent" and "Reference". With "Random" the nucleus gets a random orientation, with "Parent" it inherits the orientation of the matrix phase. "Reference" does not rotate the nucleus and uses the orientation of the reference frame.

- **MobilityReduction (MobilityReduction)**
  - Reduces mobility of Nuclei with the specified factor (e.g. 0.5) in order to stabilize the nucleation event. Use if Nuclei tend to spread. Default to 1.0.

- **Nucleation interval in timesteps (NucleateEvery)**
  - Every global timestep a new generation of nuclei is scheduled. This regeneration of nucleation sites can be suppressed by increasing this value.

- **Distribution Mode (Distribution)**
  - Options are "None", "Normal", "Cauchy" and "Uniform". Each nucleus is assigned a size and the nucleus is planted if the driving force exceeds the counteracting curvature of a nucleus of the size. The selection then determines the distribution of nuclei sizes.

- **Radius (Radius)**
  - Each nucleus for the phase pair is assigned this radius if "None" is selected as a distrubution mode.

- **Distribution Center (Center)**
  - Center of the size distribution for "Normal" and "Cauchy".

- **Normal Distribution Sigma (Deviation)**
  - Standard deviation for the normal distribution.

- **Cauchy Distribution Gamma (HalfWidth)**
  - Scale parameter which specifies the half-width at half-maximum in the Cauchy distribution.

- **Uniform Distribution Minimum Radius (RadiusMIN)**
  - Lower bound for the radius in the uniform distrbution.

- **Uniform Distribution Maximum Radius (RadiusMAX)**
  - Upper bound for the radius in the uniform distrbution.

- **Number of Variants (Variants)**
  - Number of variants that are tested for each nucleation site.

- **Variant Mode (VariantsMode)**
  - Options are "Random" and "LowestEnergy".

6.10 Module: Temperature

The input of the system temperature, temperature gradients or cooling/heating-behaviour can be set in this section.

- **Obtain temperature curve from file (ReadfromFile)**
  - Obtain temperature curve (over time) from an input file. The file needs to contain lines with time and temperature, separated by spaces. Default to No.
- **Data file containing temperature curve (DataFile)**
  - Text file with two columns of data, first column for time in seconds, second column for temperature in Kelvin. Interpolation in between.
- **Initial System Temperature (T0)**
  - Initial temperature in the simulation domain.
- **Cooling(-) or heating(+) rate (DT_Dt)**
  - Change of temperature per time. Constant value for the whole simulation duration in K/s.
- **[X, Y, Z] coordinate of the reference point (R0X,R0Y,R0Z)**
  - Base point for definition of a temperature gradient. Default to 0.0.
- **[X, Y, Z] component of the temp. gradient (DT_DRX,DT_DRY,DT_DRZ)**
  - Vector for definition of a temperature gradient. Default to 0.0.
- **Latent heat mode (LatentHeat)**
  - Use latent heat calculation. Can be set to "Global", which applies the generated heat to the whole simulation domain. This assumes infinite heat diffusion rate. Using the heat diffusion solver, LatentHeat can be set to "Local", which applies the generated heat locally and then solves the heat diffusion equation. No latent heat for option "No".
- **LocalTemperature**
  - Local Temperature defines cuboids from points Upper limit and Lower limit in which the Cooling(-) or heating(+) rate is applied. Deprecated: Use HeatSources instead.
- **Extension_[X, Y, Z]_[Upper, Lower]**
  - Extends the domain in this direction by the chosen number of grid points, useful in conjunction with heat diffusion.
- **Stop cooling temperature (StopMin)**
  - Sets DT_Dt to zero, when the minimum temperature in the domain is lower or equal to this temperature.
- **Stop heating temperature (StopMax)**
  - Sets DT_Dt to zero, when the maximum temperature in the domain is higher or equal to this temperature.

6.11 Module: ThermodynamicInterface

If thermodynamic values are to be used in the simulation, this section provides the input on where to find the data and how to process it. Different third-party CALPHAD software interfaces can be used to provide thermodynamic data.

- **Composition shield (Shield)**
  - Cutoff to prevent asking for thermodynamic values with too low composition.
- **Database file OP (OPfile)**
  - File name for OpenPhase Input Database *.opid files.
- **Database file TC (TCfile)**
  - File name for Thermo-Calc *.GES5 files.
- **Database file OC (OCfile)**
  - File name for Open Calphad *.tdb files.

6.12 Module: ThermodynamicFunctions

This section contains the input for the thermodynamic module, which determines the handling of the thermodynamic interfaces and frequency of recalculation of thermodynamic values.

- **Extrapolation mode (Expo)**
  - Use extrapolation of the thermodynamic values. Default "Active". Not using extrapolation increases precision, but severely impacts performance.
- **Maximum composition deviation (MDev)**
  - Recalculate a new set of thermodynamic values if the average composition in the interface (in a local point – if the FID model is chosen) exceeds an amount of MDev from when the old set was calculated. Default to 1E-4.
- **Maximum temperature deviation (TDev)**
  - Recalculate a new set of thermodynamic values, if the average temperature in the interface (in a local point – if the FID model is chosen) exceeds an amount of TDev from when the old set was calculated. Default to 1.
- **Driving Force Type (DrivingForceType)**
  - The thermodynamic driving force can be calculated in different ways. In "Standard" the contribution from both sides of the dual phase region are considered as a arithmetic mean. In "LowestSlope" only the contribution by the side with lower absolute slope is considered. In "Weighted" both sides are considered but are weighted by the inverse slope. And "User" allows the selection of slope with which to calculate the driving force. These settings can be useful to improve stability with stoichiometric and near stoichiometric phases, which have very steep slopes.
- **Consider Driving Force (DGContributionMAP)**
  - Select which driving force contribution to calculate if Driving Force Type is set to "User".

6.13 Module: Damage

Input for the damage module. Damage depends on the norm of local plastic strain and is linearly interpolated between the initial value k0_linear_PHASE and the maximum value kc_linear_PHASE. The model includes a non-local regularization in order to make the model mesh independent.

- **Damage active (damageflag)**
  - Enables or disables damage and therefore weakening of local points for chosen phase.
- **Plastic strain damage start (k0_linear)**
  - Starting plastic strain norm required for damage.
- **Plastic strain damage end (kc_linear)**
  - Plastic strain norm for which the maxDmg value is reached.
- **Non-local regularization (alphanonlocal)**
  - Internal length scale of the damage model see [2].
- **Maximum damage (maxDmg))**
  - Maximum damage value, this is recommended to be taken ¡ 0.95, in order to reduce contrast in stiffness and thus ensure convergence of the spectral elastic solver.

6.14 Module: HeatDiffusion

Solve heat diffusion using the implicit solver. The simulation domain can be extended by a 1D extension in arbitrary direction, the effective boundary condition at the end of the extension is then fixed. Molar heat capacity in J/(mol*K) needs to be supplied in the *.opid file in ThermodynamicInterface / can be fetched from a thermodynamic interface.

- **Thermal Conductivity (ThermalDiffusivity)**
  - Thermal conductivity for each phase in $\frac{J}{msK}$.
- **Volumetric Heat Capacity (VolumetricHeatCapacity)**
  - Volumetric heat capacity for each pahse in $\frac{J}{Km^3}$
- **Tolerance (Tolerance)**
  - Maximum residual for the implicit solution of heat diffusion. Default to 1E-8.
- **Maximum Iteration Warning (MaxIterations)**
  - A warning will be printed to the console, when the chosen number of iterations is reached.
- **Solver (Solver)**
  - Selection between a simple Jacobi iteration, a Gauss-Seidel (GS) solver or a matrix based BiCGStab.

6.15 Module: DiffusionProperties (EquilibriumPartitioning)

When the diffusion model is set to EQP in the ChemicalProperties section, this section provides further input for the diffusion model with equilibrium partitioning.

- **Activate grain boundary diffusion (GrainBoundaryDiffusion)**
  - Diffusion increase in the grain boundary gridpoints.
- **Grain boundary diffusion factor (GBValue)**
  - Factor to multiply the diffusivities in the GB area with. Keep in mind that the interface is diffuse in phase-field simulations, therefore the grain boundary diffusion factor should be chose accordingly. The diffuseness of the interface is defined by interface width (IWidth) and grid spacing (dx) in module Settings.
- **Diffusion matrix modifier (DiffusionMode)**
  - Diffusion mode with "Full" diffusivities matrix, or only "Diagonals". Using only diagonal components of the diffusion matrix accelerates the simulation, but cross-diffusional effects are neglected.
- **AntiTrapping (AntiTrapping)**
  - Activate anti trapping current.

6.16 Module: VacancyDiffusion

Enables the simulation of vacancies in addition to other elements.

- **Vacancy Diffusion Coefficient (VacancyDiffusionCoefficient)**
  - Diffusion coefficent of vacancies independent of phase.
- **Vacancy Equilibrium Concentration (VacancyEquilibriumConcentration)**
  - Equilibrium concentration of vacancies in a void, the equilibrium concentration of vacancies in non-void phases will be 1 minus this concentration.
- **Vacancy Driving Force Prefactor (VacancyEntropy)**
  - Scaling factor for the driving force contribution by vacancies for solid-void interfaces.
- **Minimum Vacancy Concentration in Void (VacancyMinVoid)**
  - Lower limit for the vacancy phase contration in the void phase.
- **Maximum Vacancy Concentration in Solid (VacancyMaxSolid)**
  - Upper limit for the vacancy phase contration in each non-void phase.
- **Target Vacancy Concentration in Solid-Solid Grain Boundaries (VacancyGBConcentration)**
  - Grain boundaries between non-void phases can act as sources or sinks for the vacancy concentration. This value gives the target concentration of vacancies in the grain boundaries.
- **Rate of Vacancy Creation and Destruction in Solid-Solid Grain Boundaries (VacancyGBSpeed)**
  - This parameter determines how fast the concentration will reach the vacancy concentration target in the grain boundaries. Automatically limited to 0.1/dt.

6.17 Module: Grand Potential Density

Module repesenting the grand presenting density of the bulk material in the energy function.

- **Grand Potential Density Phase Model (OMEGA)**
  - Each thermodynamic phase can be modeled by its own state function, omega(mu). The parabolic model is the Legendre transformed parabolic approximation of the free energy, f(rho). Other models will be available in future releases.
- **Energy Coefficient (EPS)**
  - Energy Coefficient of parabolic free expansion. Determines compressibility of the material with respect to the component and is solubility.
- **Equilibrium Density (C0)**
  - Energy Coefficient of parabolic free expansion. Determines compressibility of the material with respect to the component and is solubility.

6.18 Module: Grand Potential Solver

Solves the differential equation for the chemical potential which was defined by the grand potential density. Also calculates the phase-transformation according to the change to the chemical potential.

- **Molar Mass (Mass)**
  - Molar mass of component in kilogram per mole
- **Initial Density (CI)**
  - Initial density of component in mole per cubic meter.
- **Chemical Mobility (M0)**
  - Mobility of component in phase in square mole per joule, meters and second.
- **Semi Implicit Euler (Implicit)**
  - Solves differential equation for the chemical potential using an implicit Euler scheme.
- **Enforce Mass Conservation (TOC)**
  - The local chemical potential is changed at every point by the same factor so that the total mass of the simulation domain is enforced to be constant.

References

[1] C.K. Aidun and Y. Lu. Lattice boltzmann simulation of solid particles suspended in fluid. *Journal of Statistical Physics*, 1995.

[2] M. Boeff, F. Gutknecht, P. Engels, A. Ma, and A. Hartmaier. Formulation of nonlocal damage models based on spectral methods for application to complex microstructures. *Engineering Fracture Mechanics*, 2015.

[3] B. Böttger, J. Eiken, and I. Steinbach. Phase field simulation of equiaxed solidification in technical alloys. *Acta Materialia*, 2006.

[4] S. Chen and G. D. Doolen. Lattice boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 1998.

[5] A.L. Kupershtokh, D.A. Medvedev, and D.I. Karpov. On equations of state in a lattice boltzmann method. *Computers & Mathematics with Applications*, 2009.

[6] A. Monas, O. Shchyglo, D. Höche, and M. Tegeler et al. Dual-scale phase-field simulation of mg-al alloy solidification. *IOP Conference Series: Materials Science and Engineering*, 2015.

[7] A. Monas, O. Shchyglo, S. J. Kim, and C. Yim et al. Divorced eutectic solidification of mg-al alloys. *JOM*, 2015.

[8] Y. H. Qian, D. D'Humières, and P. Lallemand. Lattice bgk models for navier-stokes equation. *Europhysics Letters*, 1992.

[9] Xiaowen Shan and Hudong Chen. Lattice boltzmann model for simulating flows with multiple phases and components. *Physical Review E*, 1993.

[10] Xiaowen Shan and Hudong Chen. Simulation of nonideal gases and liquid-gas phase transitions by the lattice boltzmann equation. *Physical Review E*, 1994.

[11] I. Steinbach. Phase-field models in materials science; a tutorial review. *Modelling and Simulation in Materials Science and Engineering*, 2009.

[12] Ingo Steinbach. Phase-Field Model for Microstructure Evolution at the Mesoscopic Scale. *Annual Review of Materials Research*, 43(1):89–107, 2013.

[13] D. Y. Sun, M. I. Mendelev, and C. A. Becker et al. Crystal-melt interfacial free energies in hcp metals: A molecular dynamics study of mg. *Physical Review B*, 2006.

[14] F. Varnik, M. Gross, and N. Moradi et al. Stability and dynamics of droplets on patterned substrates: insights from experiments and lattice boltzmann simulations. *Journal of Physics: Condensed Matter*, apr 2011.

[15] S. Vedantam and B. S. V. Patnaik. Efficient numerical algorithm for multiphase field simulations. *Physical Review E*, 2006.

[16] D.A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction.* Springer Berlin Heidelberg, 2004.

[17] M. Yang, S.-M. Xiong, and Z. Guo. Characterisation of the 3-d dendrite morphology of magnesium alloys using synchrotron x-ray tomography and 3-d phase-field modelling. *Acta Materialia*, 2015.

[18] S. Zhao, J. Li, and L. Liu et al. Eutectic growth from cellular to dendritic form in the undercooled ag-cu eutectic alloy melt. *Journal of Crystal Growth*, 2009.