Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Materials Science and Engineering

Thomas Vatn Bjørge

Implementation of a Precipitation Model for Cylindrical-Shaped Particles

Master's thesis in Materials Science and Engineering
Supervisor: Bjørn Holmedal and Tomas Manik
June 2020

NTNU
Norwegian University of
Science and Technology

Thomas Vatn Bjørge

Implementation of a Precipitation Model for Cylindrical-Shaped Particles

Master's thesis in Materials Science and Engineering
Supervisor: Bjørn Holmedal and Tomas Manik
June 2020

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Materials Science and Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

In most age-hardenable alloys, the precipitation of second-phase particles is often of a non-spherical configuration. However, in Kampmann Wagner Numerical (KWN) models for predicting the particle size distribution, the particles are assumed to be spherical. In this thesis, a framework for modeling the coupled nucleation, growth, and coarsening of cylindrical-shaped particles was implemented. Using a Lagrangian approach, the particle size distribution was discretized into several particle classes where each class contained a number of particles with identical size. Further, a distinction between the cylindrical particle's end and side surface were made in terms of interfacial compositions and growth rates of the respective surfaces. This was achieved by solving the diffusion problem around the cylindrical particle to find the solute current into the respective surfaces. In addition, the axisymmetric Gibbs-Thomson effect was derived for the end and side surfaces of the particle. The evolution of each particle class was then tracked through time by the use of an ODE-solver and a mass balance equation to keep track of the amount of solute atoms in the matrix.

By implementing a function for the surface energy of the end surfaces that is dependent on the radius of the particle, the proposed KWN-model was successful in attaining elongated needles that achieve increasing aspect ratios throughout the time evolution.

Sammendrag

I de fleste utherdbare legeringer oppstår det partikler av sekundærfaser som ofte er av ikke sfærisk karakter. I Kampmann Wagner Numerical (KWN) modeller som brukes for å forutse partikkeldistribusjoner, er partiklene antatt å være sfæriske. I denne avhandlingen blir et rammeverk for å modellere de koblede prosessene nukleasjon, vekst og forgroving for sylinder-formede partikler utviklet. Ved å bruke en Lagrange tilnærming kan man diskretisere partikkeldistribusjonen i flere partikkelklasser hvor hver klasse inneholder et antall partikler med identisk størrelse. Videre blir det gjort en differensiering mellom ende- og sideflater for den sylindriske partikkelen med tanke på grenseflatekonsentrasjon og veksthastighetene til de respektive overflatene. Dette er mulig ved å løse diffusjonsproblemet rundt en sylindrisk partikkel for å finne strømmen av atomer inn gjennom de respektive overflatene. I tillegg blir den aksesymmetriske Gibbs-Thomson effekten utledet for ende- og sideflater. Evolusjonen av hver partikkelklasse gjennom tid blir deretter fulgt ved hjelp av en ordinær differensialligningløser og en massebalanse for å holde rede på hvor mye oppløste atomer det er i matrix.

Ved å innføre en funksjon for endeflateenergien som er avhengig av radiusen til partikkelen, er den foreslåtte KWN-modellen i stand til å utvikle forlengete nåler som har en økende aspekt rate gjennom tidsforløpet.

Preface

This thesis was submitted to the Norwegian University of Science and Engineering (NTNU) at the Department of Materials Science and Engineering(IMA). The thesis was submitted in June 2020 and was a part of the course *TMT4905 - Materials Technology, Master's Thesis*. The thesis builds on some elements introduced through a project work by the author in the fall semester, and some parts of the thesis will be inspired by the project report.

First and foremost, I would like to thank my life partner Maya Keilen, who made my life better in all aspects during the difficult times encountered this spring. Despite the consequences of the COVID-19 pandemic, I will look back at this spring as a time spent with my loved one.

I would also like to thank my supervisor Bjørn Holmedal and Tomas Manik for their great interest and enthusiasm for my thesis work. They were always available when I needed help and provided excellent guidance. This thesis would not be possible without them. Last but not least, I would like to thank my parents and friends who made these last five years a blast.

Contents

1 Introduction

Heat treatment of Aluminium and other heat-treatable alloys is of utmost importance due to the precipitation of particles of nanometer size. These second-phase particles are often of non-spherical shape due to their anisotropic nature with relation to misfit strains in the lattice and interfacial energies as seen for platelet-shaped θ' in Al-Cu [1], and needle-shaped β'' in Al-Mg-Si [2]. The key attribute of these non-spherical precipitates is their ability to inhibit dislocation movement as they act as pinning points [3]. This gives rise to a significant strengthening mechanism where the size, volume fraction, and morphology of the particle are important parameters. The need for predictive computer tools for these parameters is therefore of great interest for the industry.

Extensive research has been made to develop numerical models that can predict precipitation behavior for age-hardenable alloys. From the literature there exist two different approaches towards precipitation models, where the first approach is based on a direct detailed numerical approach that utilizes the phase-field method [4–6]. This approach have shown to give very detailed descriptions of phase transformations and their constituents, but due to the difficult implementation of such models and the exceedingly high computation times, they are not as suited for industrial purposes where time and usability are important factors. The other approach presented in the literature is one based on Frequency Distribution Function (FDF) approaches. These approaches rely on the work of Kampmann-Wagner [7] and Myhr-Grong [8], where they employ a statistical approach towards the evolution of the precipitate distribution in the material. Here, the particle distribution is divided into discrete size classes which are tracked over time to follow each classes' evolution. These models excel in providing solutions for precipitation

as they can handle several phenomena related to precipitation such as nucleation, growth, and coarsening. The FDF approaches have shown to yield promising results and offer a much easier implementation and the ability to address more complex systems.

In the Kampmann-Wagner Numerical (KWN) model [7], the diffusion problem is solved individually for each particle surrounded by an infinite matrix. Therefore, each particle interacts with one another through the interaction with the mean concentration field of the matrix. It is also assumed that the growth of each particle is entirely controlled by diffusion. One critical assumption of the KWN model is that the precipitates are of spherical morphology. As mentioned, the precipitates that form in a wide variety of alloys are not spherical and it, therefore, exists a demand for models that can generalize the model to account for non-spherical particles. To do so the diffusion problem has to be solved for the non-spherical configuration. This has been done by Holmedal *et al.* [9] for cuboid particles and the author [10] for cylindrical particles, where it is implemented two correction factors to account for the influence of the non-spherical configuration on the growth rate and the Gibbs-Thomson effect. These correction factors allow one to utilize the spherical KWN-model. However, this approach does not distinguish the particle surfaces in terms of interfacial concentration and is unable to describe the individual growth rates of the migrating surfaces of a non-spherical particle. In this thesis, a mathematical model is formulated for extending the KWN-model for a cylindrical particle where the growth of the respective end and side surfaces will be accounted for. The particle size distribution of the system will be evaluated using a Lagrangian approach.

2 Theory

The theoretical background regarding the Kampmann-Wagner Numerical model and its constituents will be examined in this chapter. First, to understand the nature of diffusional phase transformations in alloys, a review of the process of diffusion and the Gibbs-Thomson effect will be given in sections 2.1 and 2.2, respectively. Thereafter, a thorough review of the background of diffusional phase transformations in solids will be given in section 2.3, before finalizing the theory by examining different techniques of implementing the KWN-model as found from the literature.

2.1 Diffusion

The phenomenon of atomic migration within solids or other phases is known as diffusion. Diffusion is an important process within material science and plays a key role in many of the different processes and phase transformations that occur at the microscale in a material. As mentioned, diffusion is the process of atomic migration and is accountable for mass transport of atoms from high to low chemical potential regions. As with most mechanisms related to achieving chemical equilibrium in a material, the motivation for diffusion is to minimize the Gibbs free energy of the system [11].

There exist two types of diffusion, interstitial- and substitutional diffusion. Interstitial diffusion is relevant for smaller interstitial atoms, where these atoms can migrate through the material by interstitial sites in the lattice. Whereas substitutional diffusion occurs by the migration of larger substitutional atoms through vacancies in the material. An illustration of the two types of diffusion are

given in Figure 2.1. A significant feature of diffusion is its temperature dependency. The atomic migration is initiated by vibrations of the atoms which can cause an atomic jump. This is in turn related to the thermal energy present in the atoms. In addition, for substitutional diffusion, there has to exist vacancies for the atoms to jump. The number of vacancies in a material is greatly enhanced at elevated temperatures, which results in a higher probability of substitutional diffusion.

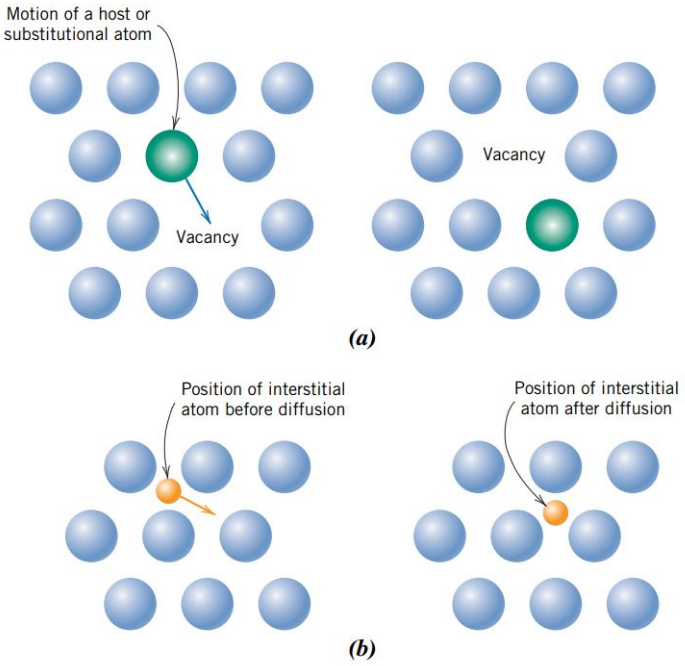


Figure 2.1: Illustration of (**a**) substitutional diffusion and (**b**) interstitial diffusion [12].

The rate at which the concentration in an alloy change over time, was first described by Fick's 2. law of diffusion and is depicted in equation (2.1), where D is the diffusion coefficient.

$$\frac{\partial c}{\partial t} = D\nabla^2 c \tag{2.1}$$

Equation (2.1) is the foundation of diffusional phase transformations and determines the rate at which the diffusional transformations advances. As a part of the KWN model, it is beneficial to review the steady-state diffusion equation which transforms equation (2.1) into equation (2.2).

$$\nabla^2 c = 0 \tag{2.2}$$

The above equation is also referred to as Laplace's equation which exhibits properties of a harmonic function [13]. As a consequence, solutions of equation (2.2) can be superposed to address more complex problems by summing up problems with simpler boundary conditions. The boundaries and extension of equation (2.2) to a cylindrical configuration will be given in the next section.

2.1.1 Steady State Diffusion For Cylindrical Particles

The steady-state diffusion problem for a cylindrical particle was solved in a project work by the author [10], where the concentration $c = c^i$ at the particle surface at both the side and end surfaces. When addressing the steady-state diffusion problem for a cylindrical shaped particle, equation (2.2) can be transformed into the cylindrical coordinate system. If one also assumes an axisymmetric specimen with angular symmetry, equation (2.2) can be rewritten as depicted below.

$$\frac{1}{r}\frac{\partial c}{\partial r} + \frac{\partial^2 c}{\partial r^2} + \frac{\partial^2 c}{\partial z^2} = 0 \tag{2.3}$$

Here, r is the radial component and z is the axial component. Due to the assumption of angular symmetry, the diffusion problem is effectively reduced into a 2-dimensional problem. An illustration of the 2D domain is shown in Figure 2.2. Here, an axisymmetric specimen of length L and radius R is shown, where the stapled lines indicate the boundaries of the domain.

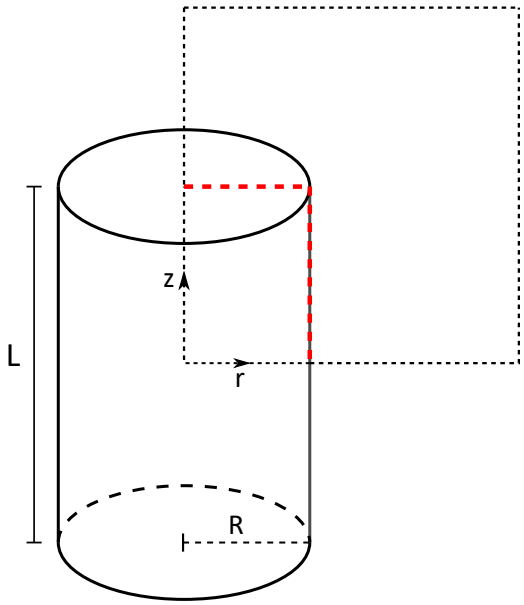


Figure 2.2: Illustration of the domain for the diffusion problem for an axisymmetric cylindrical particle.

Solutions to the 2D axisymmetric diffusion problem are not trivial and require a numerical treatment. Such a treatment was done in a project work by the author [10], which implemented a dimensionless solution to generalize the diffusion problem for all kinds of alloys. The dimensionless axial and radial components are shown in equation (2.4), in addition to an expression for the aspect ratio of the selected particle. The spatial coordinates are scaled by the particle radius.

$$\hat{r} = \frac{r}{R}, \quad \hat{z} = \frac{z}{R}, \quad \alpha = \frac{L}{2R} \tag{2.4}$$

Further, the dimensionless concentration can be expressed as shown below.

$$\hat{c} = \frac{c - c^m}{c^i - c^m} \tag{2.5}$$

Here, c^m is the concentration of the matrix, while c^i is the concentration of the particle at its interfaces. Also, a new expression for the 2D axisymmetric problem with the addition of dimensionless variables is needed.

$$\frac{1}{\hat{r}}\frac{\partial \hat{c}}{\partial \hat{r}} + \frac{\partial^2 \hat{c}}{\partial \hat{r}^2} + \frac{\partial^2 \hat{c}}{\partial \hat{z}^2} = 0 \tag{2.6}$$

From Figure 2.2, one can also assign boundary conditions to the diffusion problem as presented below.

$$\hat{c} = 0 \quad \text{when } \hat{r} \text{ or } \hat{z} \to \infty \tag{2.7}$$

$$\hat{c} = 1 \text{ at } \begin{cases} \hat{r} = 1, \ 0 \leq \hat{z} \leq \alpha \\ \hat{z} = \alpha, \ 0 \leq r \leq 1 \end{cases} \tag{2.8}$$

$$\frac{\partial \hat{c}}{\partial \hat{r}} = 0 \quad \text{when } \hat{r} = 0, \hat{z} > \alpha \tag{2.9}$$

$$\frac{\partial \hat{c}}{\partial \hat{z}} = 0 \quad \text{when } \hat{z} = 0, \hat{r} > 1 \tag{2.10}$$

Where the boundary conditions from equation (2.9) and (2.10) arise from the symmetry of the domain. From the numerical model implemented in the project work [10], the concentration fields were found for the diffusion problem around a cylindrical particle for an aspect ratio of 1 and 20, respectively. As evident from Figure 2.3, the concentration profile around a cylindrical particle degenerates to the spherical case further out from the particle.

Flux of Solute for Cylindrical Particles

After solving the diffusion problem around the cylindrical particle [10], acquired the dimensionless concentration flux at the interface. The flux of solute is crucial in the context of deriving growth rate equations for a cylindrical particle. For general shapes, the flux can be shown as the following.

$$I = \underset{\text{particle interface}}{\iint} \frac{\partial c}{\partial n} \, dS \tag{2.11}$$

By dividing the total flux into the particle as $I = I_{ends} + I_{side}$, one can achieve an expression for the total flux for a cylindrical particle [10].

$$I_{side} = \int_0^{2\pi} \int_{-\frac{L}{2}}^{\frac{L}{2}} D \left.\frac{\partial c}{\partial r}\right|_{r=R} R\, dz d\theta = 4\pi DR \left(c^m - c^i\right) \int_0^{\alpha} \left.\frac{\partial \hat{c}}{\partial \hat{r}}\right|_{\hat{r}=1} d\hat{z} \tag{2.12}$$

$$I_{ends} = \ 2 \int_0^{2\pi} \int_0^{R} D \left.\frac{\partial c}{\partial z}\right|_{z=\frac{L}{2}} r\, dr d\theta = 4\pi DR \left(c^m - c^i\right) \int_0^{1} \left.\frac{\partial \hat{c}}{\partial \hat{z}}\right|_{\hat{z}=\alpha} \hat{r} d\hat{r} \tag{2.13}$$

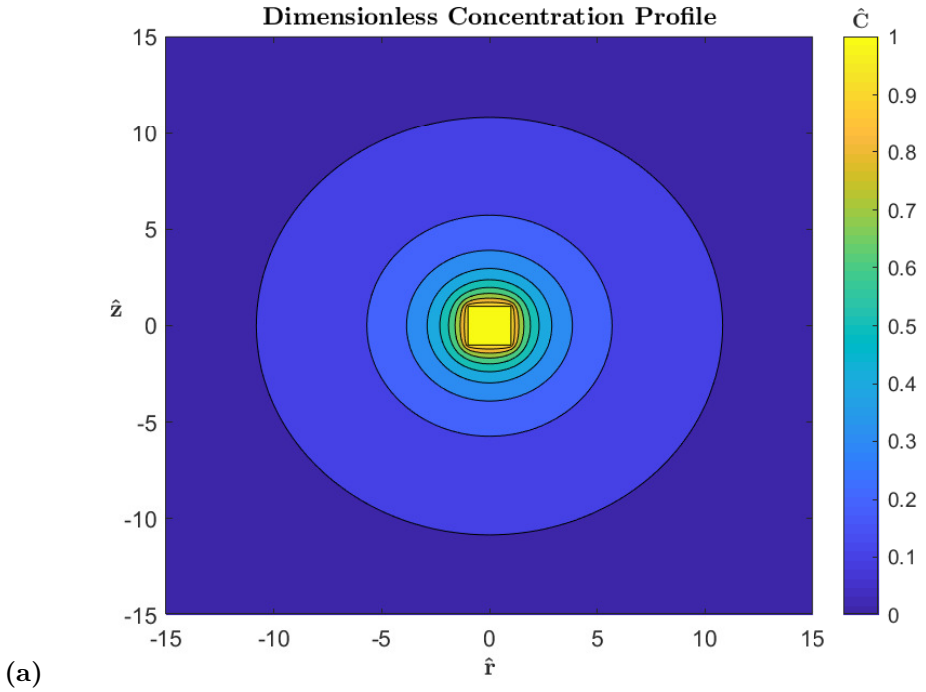
Further, one can introduce a dimensionless flux $\hat{I}$ which results in the following expression for the total flux.

$$I = RD \left(c^m - c^i\right) \hat{I}, \quad \hat{I} = \hat{I}_{side} + \hat{I}_{ends} \tag{2.14}$$

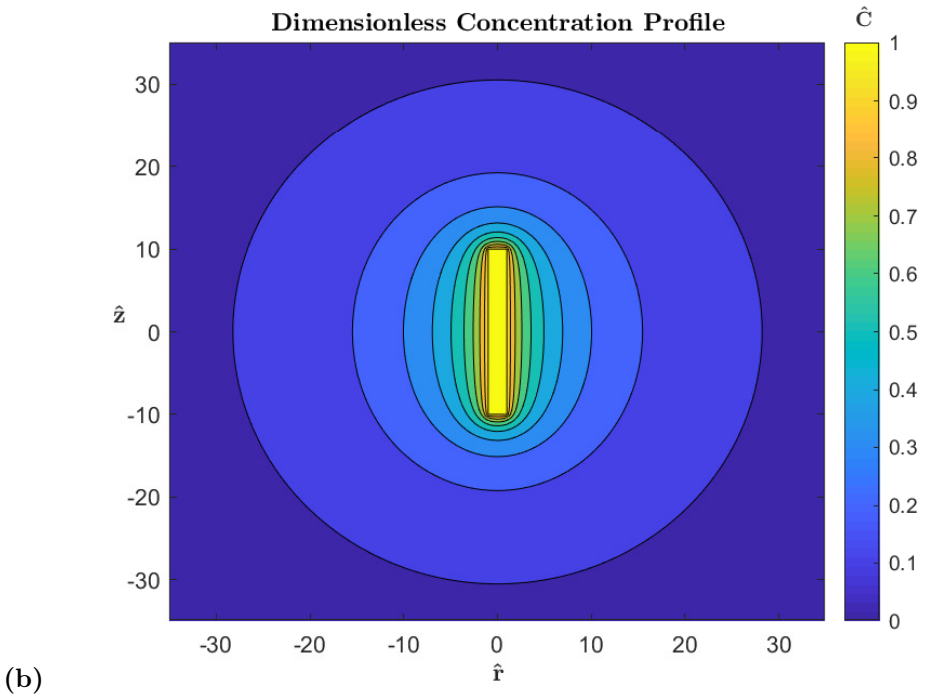
Hence, the dimensionless flux from the side and ends of the considered particle are defined in equation (2.15) and (2.16), respectively.

$$\hat{I}_{side} = 4\pi \int_0^{\alpha} \left.\frac{\partial \hat{c}}{\partial \hat{r}}\right|_{\hat{r}=1} d\hat{z} \tag{2.15}$$

$$\hat{I}_{ends} = 4\pi \int_0^{1} \left.\frac{\partial \hat{c}}{\partial \hat{z}}\right|_{\hat{z}=\alpha} \hat{r} d\hat{r} \tag{2.16}$$



(a)



(b)

Figure 2.3: **(a)** The concentration field for $\alpha = 1$. **(b)** The concentration field for $\alpha = 20$ [10].

2.2 The Gibbs-Thomson Effect for Axisymmetric Particles

An important factor in modeling phase transformations is to account for the Gibbs free energy of the system, and the significance the curvature of precipitates has on this energy. The influence of curvature on the Gibbs free energy is known as the Gibbs-Thomson effect and affects the interfacial composition given by the phase diagram [14]. This is crucial in the context of nucleation and coarsening where the Gibbs-Thomson effect might have a large influence. An illustration of this effect on a second-phase particle β in a primary α matrix is shown in Figure 2.4 [11]. As demonstrated by Figure 2.4, the Gibbs-Thomson effect alters the equilibrium concentration adjacent to the particle from X_e to X_r and effectively changes the concentration gradient that drives the diffusion-controlled growth of particles.

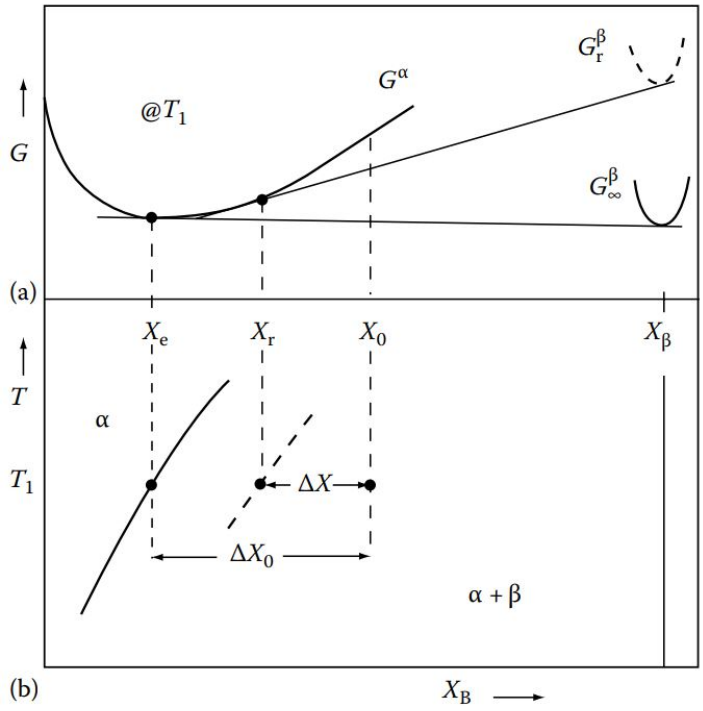


Figure 2.4: Schematic sketch of the Gibbs-Thomson effect [11]. **(a)** Gibbs free energy curves at T_1. **(b)** Corresponding phase diagram.

For spherical particles, the curvature is in essence constant around the interface which results in a constant interfacial concentration. For a spherical particle β in a diluted binary alloy, the Gibbs-Thomson effect can be expressed as shown in equation (2.17) [8, 11, 14].

$$X_r = X_e \exp\left(\frac{2\gamma v_m^\beta}{RkT}\right) \tag{2.17}$$

Here, X_r is the interfacial composition, X_e is the equilibrium composition in the matrix, γ is the surface energy, v_m^β is the molar volume of the β phase, k is the Boltzmann constant and T is the temperature.

Equation (2.17) is the most famous form of the Gibbs-Thomson equation, but it is only valid for a pure spherical precipitate β, i.e $X_p = 1$. However, the precipitate morphology of some alloys is often of non-spherical character and is often a compound [1, 15, 16]. For non-spherical particles, the interfacial curvature is essentially not constant around its interface and a rework of the classical Gibbs-Thomson equation is needed to account for a non-spherical morphology. Holmedal *et al.* [9] introduced a shape factor g to account for the non-spherical configuration which utilizes an extension of the spherical Gibbs-Thomson equation in equation (2.17). In order to extend the spherical Gibbs-Thomson relation, a set of assumptions is required. First, it is assumed that the precipitates take a prescribed shape, second, that their aspect ratio changes sufficiently slow so that one can consider the problem as quasi-constant when deriving the equation for the modified Gibbs-Thomson effect.

For an ordered precipitate β in a diluted binary alloy, the Gibbs energy of n^β atoms can be expressed as,

$$G^\beta = n^\beta \mu^\beta + \gamma\, S^\beta \tag{2.18}$$

where μ^β is the chemical potential of each atom, and S^β is the interface surface area. Additionally, the volume of the β particle can be expressed as the following.

$$V^\beta = n^\beta V_m^\beta \tag{2.19}$$

Here, V_m^β is the molar volume of the β phase. Further one can express the partial derivative of equation (2.18) with respect to n^β.

$$\frac{\partial G^\beta}{\partial n^\beta} = \mu^\beta + \gamma \frac{\partial S^\beta}{\partial n^\beta} \tag{2.20}$$

With the assumption of a prescribed shape, one can relate the volume and surface area of the precipitate which results in the following relations for a non-spherical particle.

$$\frac{\partial G^\beta}{\partial n^\beta} = \mu^\beta + \gamma \frac{\mathrm{d}S^\beta}{\mathrm{d}V^\beta} V_m^\beta = \mu^\beta + \frac{2g\gamma V_m^\beta}{R} \tag{2.21}$$

As evident from equation (2.21), a shape factor g is introduced. This shape factor is scaled so that the expression simplifies to the spherical case when $g = 1$. Therefore, the variable R is the radius of an equivalent sphere whose volume is identical to the non-spherical particle shape. The general expression for the shape factor is given below.

$$g = \frac{1}{2} R \frac{dS^\beta}{dV^\beta} \tag{2.22}$$

From equation (2.22), one can derive shape factors for all kinds of particle morphologies. Further, one can derive the Gibbs-Thomson equation by following the treatment done by Perez [14] on equation (2.21) which results in the following Gibbs-Thomson equation, where X_i is the interfacial concentration.

$$X_i = X_m \exp\left(\frac{2g\gamma v_m^\beta}{RkT}\right) \tag{2.23}$$

The shape factor for a volume equivalent cylindrical particle was derived during a project work [10]. For a cylindrical particle, there exists the following volume and surface area.

$$S_{cyl}^\beta = 2\pi R_{cyl}^2 + 4\pi\alpha R_{cyl}^2, \quad V_{cyl}^\beta = 2\pi\alpha R_{cyl}^3 \tag{2.24}$$

Consequently, the shape factor g can be derived from equation (2.22) by inserting the derivatives of the surface area and volume. The shape factor g for a cylinder is then depicted in equation (2.25) [10].

$$g(\alpha) = \frac{2\alpha + 1}{3\alpha \left(\frac{2}{3\alpha}\right)^{\frac{1}{3}}} \tag{2.25}$$

The presented shape factor and Gibbs-Thomson equation for a cylindrical configuration can easily be implemented in already existing models for spherical particles. However, these extensions of the spherical KWN model does not account for a distinction of solute concentration at the bases and sidewalls of the cylindrical particle.

2.3 Diffusional Transformations in Solids

Diffusional transformations are phase transformations that are initiated by the thermal migration of atoms. These transformations usually occur when heating the material to a region where more phases are stable, or when cooling down with a slow cooling rate. For the case of precipitation, the reaction is due to a supersaturated metastable α'-phase which decomposes into a stable α-phase and a precipitate β-phase [11]. An illustration of the corresponding phase diagram for a material that exhibits precipitation transformations is shown in Figure 2.5 [17]. As seen from the figure, precipitation reactions typically occur after heating the material into the single-phase region and then quench the material to obtain a supersaturated phase. By the following heat treatment in the two-phase region, the supersaturated phase decomposes and small precipitates are formed.

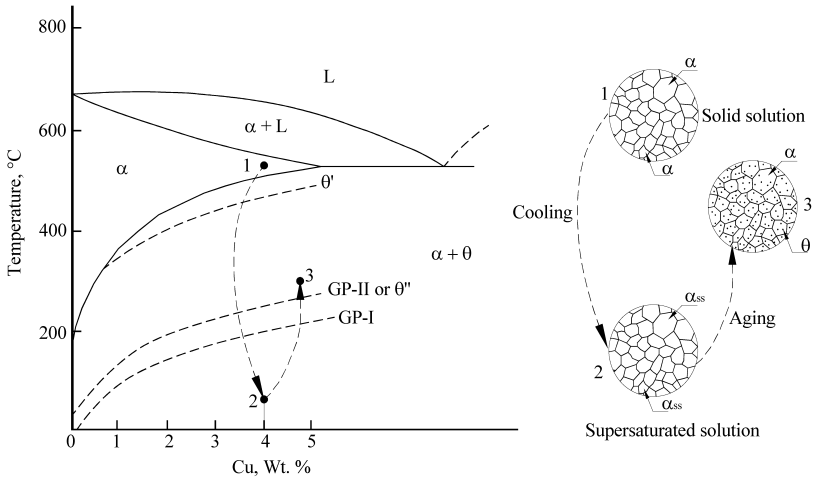


Figure 2.5: Phase diagram of the age-hardenable Al-Cu alloy [17].

As with all phase transformations, the process of diffusional transformations is dictated by the three coinciding events, nucleation, growth, and coarsening. It is these three processes that decide the distribution of precipitates over time in a material undergoing a diffusional phase transformation. A typical timeline that describes the evolution of the distribution of precipitates is shown in Figure 2.6. The evolution is categorized into four different sections [18]:

I. Incubation period τ. The time required to achieve steady-state nucleation.

II.. The steady-state nucleation with the number of particles increasing linearly.

III. Nucleation rate drops off due to a decrease in solute concentration. As the particles grow, more and more solute atoms are being tied up in the particles.

IV. The larger particles grow at the expense of smaller particles, also known as coarsening. The effect is driven by the Gibbs-Thomson effect, which affects the interfacial concentration of the particles and effectively changes the critical radius of the particles.

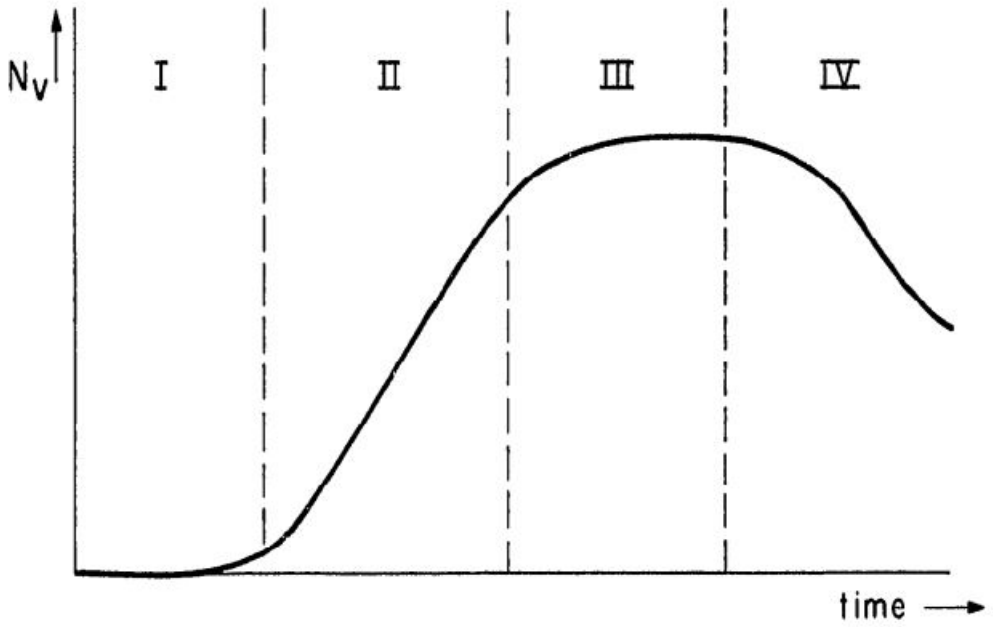


Figure 2.6: Schematic sketch of the evolution of the precipitate number density over time for a phase transformation. [18].

2.3.1 Nucleation

At the root of diffusional transformations lies the process of nucleation. The typical precipitation phase transformation consists of a metastable supersaturated α' phase which transforms into a stable α phase and a metastable or stable β phase as shown in Figure 2.5. For this reaction to take place, there has to arise small aggregates of the β phase in the α' phase. Therefore, nucleation is the process of which the smallest survivable aggregate of the more stable phase arises from the parent phase [19]. A special characteristic of the nucleation process is that it arises solely from fluctuations in the parent phase and is the only process that travels up a free energy gradient involving more than one atom [19]. This is possible due to the size scale of the process, which ranges from a few atoms and up. After an aggregate is formed, it can grow further by statistical fluctuations of solute atoms or dissolve back into the parent phase. The aggregate is said to be a nucleus when it reaches a critical size which is large enough to be stable [19]. The process of nucleation can occur in two different ways, homogeneous- and

heterogeneous nucleation. Whereas homogeneous nucleation appears in a perfect matrix in which there are no defects, heterogeneous nucleation appears at defects in the lattice. The two types of nucleation will be summarized in the two following sections.

Homogeneous Nucleation

As mentioned, homogeneous nucleation appears in a perfect matrix. While almost all nucleation in solids is heterogeneous [11], it is beneficial to review the homogeneous case first. As said, aggregates grow by a random fluctuation of solute atoms in the matrix. However, as the aggregate grows to form a stable nuclei, several factors concerning the free energy of the process occur.

1. The creation of a volume V of the more stable β phase results in a free energy reduction equal to $V\Delta G_v$ [11].
2. By assuming that the α/β interface is isotropic, there is an increase of free energy equal to $A\gamma$ due to the creation of new particle surface [11].
3. The creation of a volume of β phase in the matrix will in general give rise to lattice strains due to lattice misfit. Therefore, an increase of the free energy is proportional to the volume of the β phase equal to $V\Delta G_s$ [11].

Adding all of the terms above gives the general equation for the free energy change in homogeneous nucleation.

$$\Delta G = -V\Delta G_v + A\gamma + V\Delta G_s \tag{2.26}$$

In reality, the interfacial energy term should be a summation of all the interfaces of the particle as the surface energy might differ substantially depending on if the interfaces are coherent or incoherent. For simplicity, the interfaces are commonly assumed to exhibit the same surface energy. Further, it is commonly assumed that the nucleus is of spherical character. Equation (2.26) is then transformed into the following equation.

$$\Delta G = -\frac{4}{3}\pi r^3 \left(\Delta G_v - \Delta G_s\right) + 4\pi r^2 \gamma \tag{2.27}$$

An illustration of the variation of the Gibbs free energy, and its terms, as a function of the radius of the nucleus is shown in Figure 2.7.

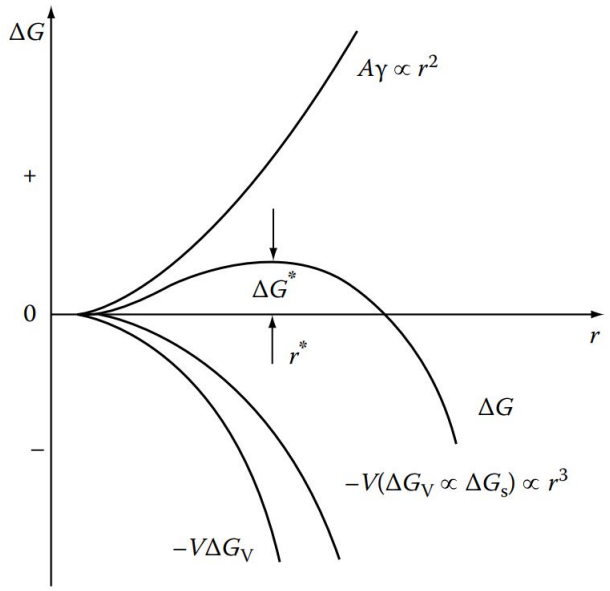


Figure 2.7: Illustration of the variation of the Gibbs free energy for homogeneous nucleation of a spherical nucleus [11].

As evident from Figure 2.7, there exists a critical radius r^* and a critical activation energy ΔG^* when $\partial \Delta G / \partial r = 0$. When an aggregate has reached this critical radius, it is denoted as a nucleus, as mentioned.

The rate at which these nuclei appear is of great importance in precipitation models. As shown by Aaronson *et al.* [19], the equilibrium concentration of critical nuclei is proportional to $\exp(-\Delta G^*/kT)$ and can be expressed as,

$$C^* = N_0 \exp\left(\frac{-\Delta G^*}{kT}\right) \tag{2.28}$$

where C_n^* is denoted as the equilibrium number of critical nuclei per unit volume, and N_0 is the number of atomic nucleation sites per unit volume. Then, if each nucleus can be made supercritical at a rate of f per second, one can express the homogeneous nucleation rate as depicted below [11].

$$\frac{dN_{hom}}{dt} = fC^* \tag{2.29}$$

As f depend on several factors like the rate of diffusion of solute to the critical nuclei, temperature and the size of the nuclei, one can rewrite the term as $\omega \exp\left(-\Delta G_m/kT\right)$ [11]. Here, ω is a variable related to the vibrational frequency of the atoms, and ΔG_m is the required Gibbs free energy for atomic migration per atom. Therefore, the homogeneous nucleation rate can be expressed in the form depicted below in equation (2.30) [11].

$$\frac{N_{hom}}{dt} = \omega N_0 \exp\left(-\frac{\Delta G_m}{kT}\right) \exp\left(-\frac{\Delta G^*_{hom}}{kT}\right) \tag{2.30}$$

In equation (2.30) the incubation period depicted in Figure 2.6 is neglected.

Heterogeneous Nucleation

Heterogeneous nucleation is the process in which the nuclei are formed on defects in the material, due to that it is energetically favorable. Nearly all nucleation taking place in a diffusional phase transformation of a solid, appear from heterogeneous sources. The most common heterogeneous nucleation sites are vacancies, dislocations, grain boundaries, stacking faults, inclusions, and free surfaces [11]. As with the homogeneous case, one can express an equation for the change in Gibbs free energy related to heterogeneous nucleation.

$$\Delta G_{het} = -V\left(\Delta G_v - \Delta G_s\right) + A\gamma - \Delta G_d \tag{2.31}$$

In this equation the new term ΔG_d has been added. Defects present in the matrix contributes to higher free energy of the system. Nucleation on these sites therefore effectively reduces the free energy of the system and gives a negative contribution in the form of ΔG_d. The sites which contribute to higher defect energies are the most favorable sites. In order from lower to higher ΔG_d are given below [11].

1. Homogeneous sites
2. Vacancies

3. Dislocations
4. Stacking faults
5. Grain boundaries and interphase boundaries.
6. Free surfaces.

Nucleation will most likely occur at the sites at the bottom of the list. However, the nucleation process is highly susceptible to the material and the circumstances. For instance, while homogeneous nucleation is the least likely nucleation site, still, the material experiences very high driving forces for nucleation, nearly every atom in the matrix can potentially be a homogeneous nucleation site. Compared to the homogeneous case, the rate equation for heterogeneous nucleation is altered due to a lower concentration of nucleation sites, but also a lower activation energy [11]. The heterogeneous nucleation rate is then depicted below in equation (2.32) [11].

$$\frac{dN_{het}}{dt} = \omega N_1 \exp\left(-\frac{\Delta G^*_{het}}{kT}\right) \exp\left(-\frac{\Delta G_m}{kT}\right) \tag{2.32}$$

An appropriate expression for the heterogeneous nucleation barrier ΔG^*_{het} has been used by Myhr *et al.* [8] for the case of spherical particles.

$$\Delta G^*_{het} = \frac{(A_0)^3}{(RT)^2 \left[\ln(c^m/c^{eq})\right]^2} \tag{2.33}$$

Here, A_0 is a constant related to how potent the heterogeneous nucleation sites are, and c^{eq} is the equilibrium solute content given by the phase diagram at the particle interface. A combination of equation (2.32) and (2.33), yields the following heterogeneous nucleation rate [8]:

$$\frac{dN_{het}}{dt} = j_0 \exp\left[-\left(\frac{A_0}{RT}\right)^3 \left(\frac{1}{\ln\left(c^m/c^{eq}\right)}\right)^2\right] \exp\left(-\frac{Q_d}{RT}\right) \tag{2.34}$$

Where, j_0 is a numerical constant, and Q_d is the activation energy for diffusion. As one can see from equation (2.34), the nucleation rate starts to decline as the concentration of the matrix is approaching the equilibrium concentration.

2.3.2 Growth

As emphasized in section 2.3.1, the nucleation of the β phase requires random fluctuations of solute atoms before acquiring a critical size, thereafter diffusion of solute atoms to the nucleus ensures further growth of the precipitate. After reaching a sufficient size of the nucleus, so that the probability of losing atoms to fluctuations are negligible, the nucleus is said to have entered the growth stage [19]. In contrast to the nucleation process, which occurs in a near-constant solute concentration, the growth stage is characterized by mass transport of solute atoms to the precipitate. While the shape of the nucleus is determined by the interfacial energies, the shape of the precipitate is determined by the relative growth rates of each interface [19]. In many diffusional phase transformations, the growth is said to be diffusion-controlled, i.e, limited by the flux of solute atoms to the migrating interface. This will be the assumption in this work.

Growth of Spherical and Non-spherical Precipitates

For a spherical precipitate, the growth rate was first proposed by Zener [20] for invariant field approximations, i.e, that one can use the steady-state equation as shown in equation (2.2), and an assumption of small supersaturation. The growth rate can be found by doing a simple flux balance at the particle surface. Figure 2.8 gives an illustration of the proposed flux balance. For a spherical particle with radius R, the incoming flux of solute atoms to the interface is as depicted in equation (2.35).

$$I = 4\pi D \left(c^m - c^i\right) R \tag{2.35}$$

Where c^m is the bulk concentration and c^i is the interfacial concentration. The amount of solute atoms entering the particle to yield growth can then be expressed as,

$$I dt = dV \left(c^p - c^i\right) \tag{2.36}$$

where c^p is the concentration of the particle. Consequently, the particle can expand in the radial direction by a volume element $dV = 4\pi R^2 dR$. Inserting for dV and

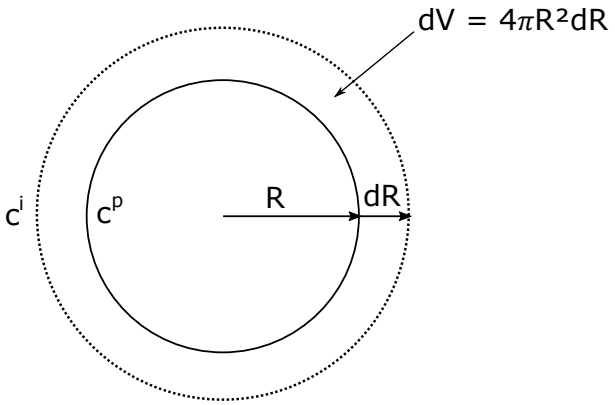


Figure 2.8: Flux balance at the interface of a spherical particle.

combining equation (2.35) and (2.36), results in the growth rate of a spherical particle under steady-state conditions.

$$\frac{dR}{dt} = \frac{D\left(c^m - c^i\right)}{R\left(c^p - c^i\right)} \tag{2.37}$$

The interfacial concentration c^i is determined by the Gibbs-Thomson effect which can have large impact on the growth rate, and consequently, the coarsening process. Equation (2.37) is frequently used in precipitation models for spherical precipitates [8, 21, 22].

An extension of the spherical precipitation models to account for non-spherical precipitates can be done by introducing a shape factor f to account for the changes in growth rates due to a non-spherical morphology. This methodology was, amongst others, introduced by Holmedal *et al.* [9] by utilizing an extension of the spherical growth rate in equation (2.37). For general shapes one can use equation (2.11) to express an equation for the flux from the considered non-spherical particle [9].

$$I = 4\pi D\left(c^m - c^i\right) fR \tag{2.38}$$

Here, f is the introduced shape factor depending on precipitate morphology and R is the radius of a volume equivalent sphere. A scaling factor 4π is also introduced

so that $f = 1$ yields the spherical case. Consequently, an expression for the growth rate of an equivalent spherical particle with an identical volume as the non-spherical particle can be found from combining equation (2.36) and (2.38) [9].

$$\frac{dR}{dt} = f\frac{D\left(c^m - c^i\right)}{R\left(c^p - c^i\right)} \tag{2.39}$$

Where a shape factor of 1 deduces equation (2.39) to the ordinary spherical growth rate as seen in equation (2.37).

An expression for the shape factor for a cylindrical particle was deduced from a project work [10]. For a cylinder, the following relationship exists for a volume equivalent sphere.

$$R = \sqrt[3]{\frac{3}{4}R_{cyl}^2\, L} = \sqrt[3]{\frac{3}{2}\alpha}R_{cyl} \tag{2.40}$$

Further, by inserting the radius R of a volume equivalent sphere into equation (2.14), one obtains the following relation.

$$I = D\left(\frac{2}{3\alpha}\right)^{\frac{1}{3}} R\left(c^m - c^i\right)\hat{I}\left(\alpha\right) \tag{2.41}$$

Comparing equation (2.38) and (2.41), one can obtain the shape factor f for a cylindrical particle as a function of its aspect ratio [10].

$$f\left(\alpha\right) = \frac{\left(\frac{2}{3\alpha}\right)^{\frac{1}{3}}\hat{I}\left(\alpha\right)}{4\pi} \tag{2.42}$$

As mentioned for the shape factor for the Gibbs-Thomson effect, the shape factor accounting for particle morphology on the growth rate can easily be implemented into already existing spherical particle models. However, this approach is unable to depict the individual growth rates of the side and end surfaces of a cylindrical precipitate. As the two types of surfaces might experience different surface energies and consequently, different interfacial concentrations. The distinction between the

surfaces becomes increasingly important as the coarsening phase arises where the Gibbs-Thomson effect is especially important.

2.3.3 Coarsening

The coarsening process is related to the final stage of the time evolution of the precipitate number density as seen in Figure 2.6. Here, the total number density of particles decreases with time. This effect is controlled by the Gibbs-Thomson effect which effectively changes the equilibrium concentration adjacent to the particle. An illustration of the nature of coarsening is given in Figure 2.9 [11].

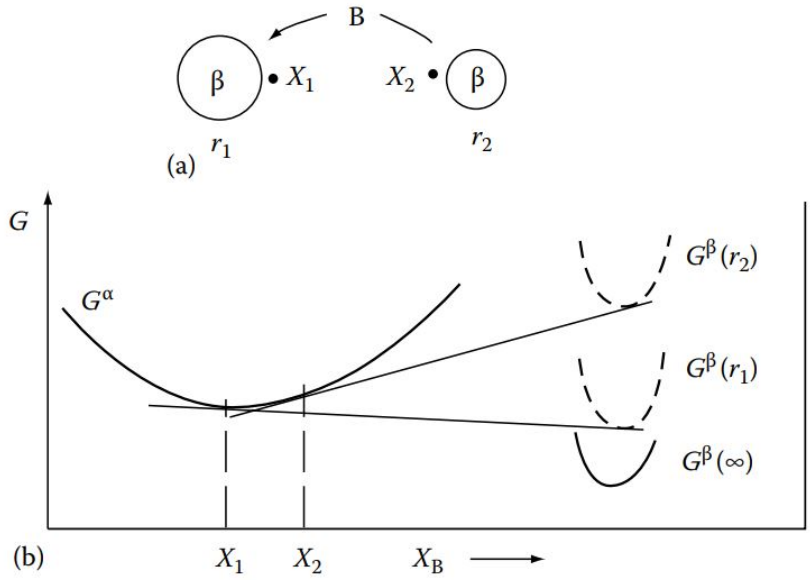


Figure 2.9: Illustration of the onset of coarsening [11].

In Figure 2.9a, two spherical precipitates of radius r_1 and r_2 exists. As a consequence of the Gibbs-Thomson effect, the equilibrium concentration adjacent to the smaller particle (r_2) will effectively be larger than that for the larger particle. Therefore, the larger particle will grow at the expense of the smaller one due to a concentration gradient which will cause solute atoms to migrate to the more stable particle with radius r_1 [11]. The coarsening effect leads to an increase in particle mean radius and a decrease of the particle number density.

2.4 Implementations of Nucleation and Growth Theories

To predict and better understand the underlying nature of diffusional phase transformations, the three coinciding processes nucleation, growth and, coarsening can be implemented in various mathematical models and solved numerically using computers. The most relevant parameters for precipitation in alloys are their number density, size, the precipitate volume fraction, morphology, and crystallography. One of the first models to implement the three processes as proposed by Langer and Schwartz for droplet formations in fluids [23], it utilized a so-called mean radius approach where the number density and mean radius of the droplets were tracked. Later, this was extended for precipitation in supersaturated alloys by Kampmann and Wagner, utilizing the same mean radius approach [24]. While the mean radius approach is successful in predicting the mean radius and the number density, it does not account for the precipitate size distribution. This was later resolved when Kampmann and Wagner introduced a new approach, where the continuous precipitate size distribution was divided into several size classes. Each class consisted of a set of identical spherical precipitates. The time evolution of the precipitate size distribution was tracked by calculating the size evolution of each size class. This multi-class approach is known as the Kampmann-Wagner Numerical model (KWN) [7]. The multi-class approach can further be divided into the Eulerian and the Lagrangian approaches. An introduction to the three types of implementation of classical nucleation and growth theories will be given in the next sections and is based on Perez *et al.* [21] and Myhr *et al.* [8].

2.4.1 The Mean Radius Approach

As the name suggests, the time evolution of the precipitates is tracked using the mean radius, $\bar{R}$, of the precipitates. As the size evolution is described by a single parameter $\bar{R}$, the spherical growth equation in equation (2.37) has to be modified to include the nucleation of new precipitates with a critical radius R^*. The simultaneous growth and nucleation of the mean radius are depicted as proposed by Perez *et al.* [21].

$$\left.\frac{d\bar{R}}{dt}\right|_{growth} = \frac{D}{\bar{R}}\frac{c^m - c^i}{\chi c^p - c^i} + \frac{1}{N}\frac{dN}{dt}\left(R^* - \bar{R}\right) \tag{2.43}$$

Here, χ is the ratio of matrix to precipitate atomic volume, N is the total number density of precipitates, and dN/dt is the nucleation rate. However, equation (2.43) does not take into account the coarsening effect. This is due to the fact that one only tracks the mean radius of the precipitates, hence, one does not track the relative radius between the precipitates in the particle size distribution. In order to account for the coarsening of the precipitates, a new expression for the evolution of the mean radius has to be introduced. The including of the coarsening mechanism to the growth rate reported in equation (2.37), was simultaneously discovered by Lifchitz and Slyosov [25] and Wagner [26]. They discovered that the particle size distribution reduces to a so-called Lifshitz–Slyozov–Wagner(LSW) distribution. From equation (2.41) and a linearized form of the Gibbs-Thomson equation, leads to the following expression for the growth rate of the mean radius in the coarsening regime [21].

$$\left.\frac{d\bar{R}}{dt}\right|_{coars} = \frac{4}{27}\frac{c^i}{\chi c^p - c^i}\frac{R_0 D}{\bar{R}^2} \tag{2.44}$$

$$R_0 = \frac{2\gamma v_p^\beta}{kT} \tag{2.45}$$

2.4.2 Multiclass Approach

In the multiclass approach, the particle size distribution is divided into several size classes. By assuming spherical precipitates, each size class exhibits a radius R_i and the number of particles of size R_i is N_i. The total number density of the particle size distribution is then given as:

$$N = \sum_i N_i \tag{2.46}$$

In addition, the mean radius is depicted as:

$$\bar{R} = \frac{\sum_i N_i R_i}{\sum_i N_i} \tag{2.47}$$

For spherical precipitates, the precipitate volume fraction can also be found as depicted below.

$$f_p = \frac{4}{3}\pi \sum_i N_i R_i^3 \tag{2.48}$$

To keep track of the bulk concentration as the particle size distribution progresses through time, a mass balance to depict the amount of solute atoms tied up in the particles is required.

$$c^m = \frac{c^0 - \chi c^p f_p}{1 - \chi f_p} \tag{2.49}$$

Where c_0 is the initial concentration of the matrix. As mentioned, for the multiclass approach there exist two choices for modeling the particle size distribution. The Eulerian or Lagrangian approach. The two approaches will be further explained in the two following sections.

Eulerian Approach

The use of a Eulerian coordinate system stems from fluid mechanics, where it implies observing the fluid properties at fixed positions [27]. Hence, for precipitation models, one considers the change of particle density for a fixed radius R, i.e a size class $R \pm dR/2$, with flux in and out at the infinitesimal control volume. In addition, nucleation occurs within the control volumes. Mathematically, the Eulerian approach can be expressed as proposed by Myhr *et al.* [8].

$$\frac{\partial N}{\partial t} + \frac{\partial (Nv)}{\partial R} = S \tag{2.50}$$

Here, N is the number density within the control volume $R \pm dR/2$, v is the particle growth or dissolution rate as defined by equation (2.37). Nv is then equal to the particle flux through the control volume. S is a source term that describes the

formation of new particles and is therefore equal to the nucleated particles per time.

Lagrangian Approach

In a Lagrangian coordinate system, the fluid properties are determined by observing the trajectories of individual pieces of the fluid [27]. Therefore, in precipitation models, the Lagrangian approach follows the time evolution of each particle, or a number of particles with the same size, i.e a particle size class. Each class is nucleated and given a constant set of particles. Mathematically, this corresponds to:

$$\frac{dN}{dt} = S, \qquad \frac{dR}{dt} = v \tag{2.51}$$

A translation between the Eulerian and Lagrangian coordinate system is then possible.

$$\underbrace{\frac{dN}{dt}}_{\text{Lagrangian}} = \underbrace{\frac{\partial N}{\partial t} + \frac{\partial \left(Nv\right)}{\partial R}}_{\text{Eulerian}} = S \tag{2.52}$$

A consequence of the Lagrangian approach is that it enables one to directly use growth equations for a spherical particle since it follows each particle size evolution through time. The methodology of this approach for precipitation models is inspired by Maugis *et al.* [22]. A schematic illustration of the nucleation and growth process in the Lagrangian approach is shown in Figure 2.10 [21].

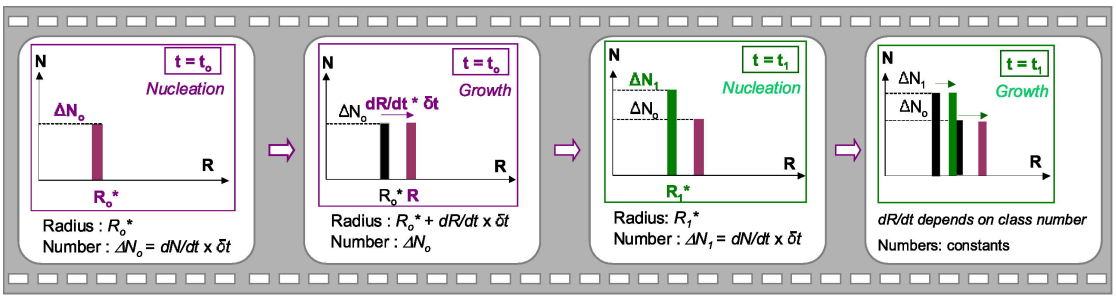


Figure 2.10: An illustration of the nucleation and growth in the Lagrangian-like approach [21].

Kernel Density Estimator

Due to the nature of the Lagrangian approach, the particle size distribution cannot readily be available from the size classes, in contrast to the Eulerian approach. As one knows how many particles of each size, to find the statistical density distribution one needs a density estimator. The simplest method is to use bins, and a viable method is utilizing a kernel density estimator(KDE). A KDE is a method of finding the probability density function of data set [28]. A kernel is placed on each data point in the set and is ultimately summed up to create a probability density function. The kernel estimator can be expressed as the following [28].

$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2.53}$$

Here, n is the number of data points, h is bandwidth, K is the kernel function and $x - x_i$ determines how far apart the data point x is from the point x_i. The kernel estimator may be thought of as a sum of narrow distributions placed at each observation [28]. Then the kernel determines the shape of the bumps and h determines the width. It is also noted that $\int K(t)dt = 1$, to ensure that $f(x)$ also integrates equal to 1. A common kernel function is the standard normal distribution. Therefore, the kernel density estimator can be described as shown below [29].

$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x - x_i}{h}\right)^2\right) \tag{2.54}$$

An illustration of the kernel density estimation with Gaussian kernels are shown in Figure 2.11 [29].

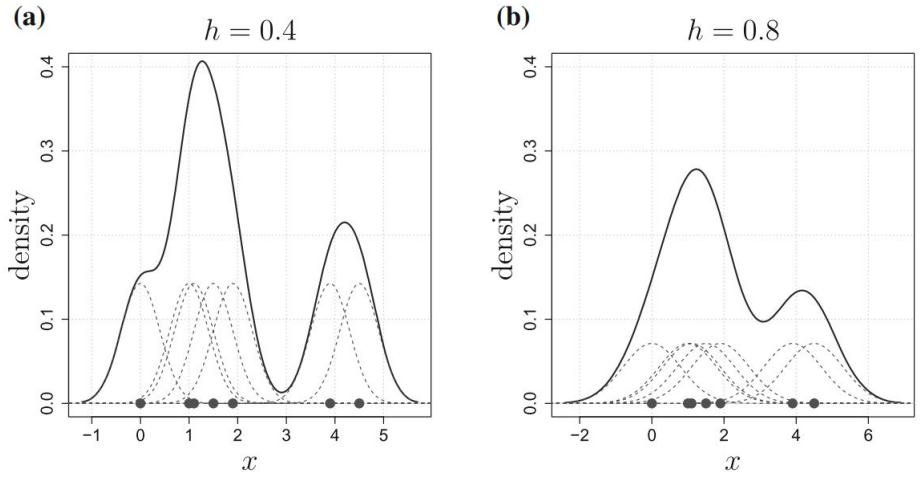


Figure 2.11: An illustration of the kernel density estimation with Gaussian kernels with two different h-values [29].

3 Modeling

In this thesis, a mathematical model is formulated and a numerical algorithm is formulated to solve the equations. The framework of the model is implemented in the language Fortran. The model uses a Lagrangian approach to evaluate the distribution in terms of the precipitates radius, length, and number densities.

3.1 Diffusion Problem and Rate Law

At the foundation of a numerical precipitation model lies the diffusion problem of solute around the particle which determines the growth speed of said particles. An illustration of the concentration field around a cylindrical particle was given in Figure 2.3. In this model, the particle growth will be assumed to be diffusion-controlled and will, therefore, be limited by the diffusion of solute atoms in solid solution to the particle's surface. The diffusion problem is further simplified by assuming that the particles are sufficiently spread out so that they do not have interacting diffusion fields, but rather interact through the same solute concentration in the matrix, which is said to be constant in some region between neighbouring particles. Moreover, the diffusion solutions near the particles and the concentration of the matrix in between the particles are assumed to change adequately slow so that one can treat the diffusion problem as quasi-steady state. The basis for the numerical model for solving the steady-state diffusion problem was solved for a non-dimensional system during a project work and will be further explained in section 3.4 [10]. As already mentioned in equation (2.2), the steady-state diffusion problem can be expressed as the following.

$$\nabla^2 c = 0 \tag{3.1}$$

Where C is the molar concentration in moles per volume of A atoms in a solid solution of A and B atoms in the primary matrix phase α. As mentioned, an axisymmetric specimen of length L and radius R is considered. A modification of the boundary conditions presented in section 2.1 is done by distinguishing the local concentrations at the end and side surfaces of the cylindrical particles. Hence, the new boundary conditions are presented below.

$$c = c^{side}, \qquad r = R, \ -\frac{L}{2} < z < \frac{L}{2} \tag{3.2}$$

$$c = c^{end}, \qquad z = \pm\frac{L}{2}, \ 0 < r < R \tag{3.3}$$

$$c = c^{m}, \qquad \sqrt{r^2 + z^2} >> R \tag{3.4}$$

As seen above, c^{side} is the local concentration at the sides of the particle, c^{end} is the local concentration at the two ends of the cylinder, and c^m is the concentration of the matrix far away from the particles. The same scaling into dimensionless spatial coordinates as in section 2.1.1 can be used to transform the system into a non-dimensional problem which allows a compact representation of the numerical solutions.

$$\hat{r} = \frac{r}{R}, \quad \hat{z} = \frac{z}{R}, \quad \alpha = \frac{L}{2R} \tag{3.5}$$

Due to the change of the boundary conditions in equation (3.2), a new expression for the dimensionless concentration will be defined.

$$c = c^m + \left(c^{end} - c^m\right)\hat{c}_{end} + \left(c^{side} - c^m\right)\hat{c}_{side} \tag{3.6}$$

As mentioned, equation (3.1) can be written as a linear combination of any function that satisfies Laplace's equation. Therefore, $\hat{c}_{end}$ and $\hat{c}_{side}$ can be solved independently. The notation $\hat{c}_{end}$ corresponds to a solution of the diffusion problem around a cylindrical particle where the ends of the cylinder have a concentration c^{end},

while the side walls exhibit a concentration $c = 0$. Moreover, $\hat{c}_{side}$ corresponds to a solution when the concentration of the side walls is equal to c^{side}, while the end surfaces correspond to a concentration $c = 0$. As mentioned in section 2.1.1, due to the symmetry of the domain, only half of the axisymmetric length needs to be calculated. A summary of the new boundary conditions and the governing equation for the two diffusion solutions are given below.

$$\nabla^2 \hat{c}_{end} = 0 \tag{3.7}$$

$$\hat{c}_{end} = 0, \qquad \hat{r} = 1,\ 0 \leq \hat{z} \leq \alpha \tag{3.8}$$

$$\hat{c}_{end} = 1, \qquad \hat{z} = \alpha,\ 0 \leq \hat{r} \leq 1 \tag{3.9}$$

$$\hat{c}_{end} = 0, \qquad \sqrt{\hat{r}^2 + \hat{z}^2} >> 1 \tag{3.10}$$

$$\frac{\partial \hat{c}_{end}}{\partial \hat{z}} = 0, \qquad \hat{z} = 0,\ \hat{r} \geq 0 \tag{3.11}$$

$$\frac{\partial \hat{c}_{end}}{\partial \hat{r}} = 0, \qquad \hat{r} = 0,\ \hat{z} \geq 0 \tag{3.12}$$

Similarly, for the diffusion problem of $\hat{c}_{side}$, the following boundaries and Laplace's equation exists.

$$\nabla^2 \hat{c}_{side} = 0 \tag{3.13}$$

$$\hat{c}_{side} = 0, \qquad \hat{r} = 1,\ 0 \leq \hat{z} \leq \alpha \tag{3.14}$$

$$\hat{c}_{side} = 1, \qquad \hat{z} = \alpha,\ 0 \leq \hat{r} \leq 1 \tag{3.15}$$

$$\hat{c}_{side} = 0, \qquad \sqrt{\hat{r}^2 + \hat{z}^2} >> 1 \tag{3.16}$$

$$\frac{\partial \hat{c}_{side}}{\partial \hat{z}} = 0, \qquad \hat{z} = 0,\ \hat{r} \geq 0 \tag{3.17}$$

$$\frac{\partial \hat{c}_{side}}{\partial \hat{r}} = 0, \qquad \hat{r} = 0,\ \hat{z} \geq 0 \tag{3.18}$$

The numerical implementation of the two diffusion problems will be further clarified in section 3.4. From the diffusion solution around the considered precipitate, one can find the current of solute through the particle surface. Therefore, the number of atoms per time moving through the particle interface can be expressed as shown below.

$$I = \iint\limits_{\text{particle interface}} D \frac{\partial c}{\partial n} \, dS \tag{3.19}$$

Consequently, by distinguishing the particles end and side surfaces, equation (3.19) can be transformed into the following equation.

$$I = 4\pi \int_{r=0}^{R} D \frac{\partial c}{\partial z}\bigg|_{z=\frac{L}{2}} r dr + 4\pi \int_{z=0}^{L} D \frac{\partial c}{\partial r}\bigg|_{r=R} dz \tag{3.20}$$

Further, by introducing the dimensionless variables previously defined in this section, the concentration flux is converted into the following relation.

$$\begin{aligned} I =& 4\pi RD \left(c^{side} - c^m\right) \int_{\hat{r}=0}^{1} \frac{\partial \hat{c}_{side}}{\partial \hat{z}}\bigg|_{\hat{z}=\alpha} \hat{r} d\hat{r} + 4\pi RD \left(c^{end} - c^m\right) \int_{\hat{r}=0}^{1} \frac{\partial \hat{c}_{end}}{\partial \hat{z}}\bigg|_{\hat{z}=\alpha} \hat{r} d\hat{r} \\ &+ 4\pi RD \left(c^{side} - c^m\right) \int_{\hat{z}=0}^{\alpha} \frac{\partial \hat{c}_{side}}{\partial \hat{r}}\bigg|_{\hat{r}=1} d\hat{z} + 4\pi RD \left(c^{end} - c^m\right) \int_{\hat{z}=0}^{\alpha} \frac{\partial \hat{c}_{end}}{\partial \hat{r}}\bigg|_{\hat{r}=1} d\hat{z} \\ =& RD \left(c^{side} - c^m\right) \left(\hat{I}_{side}^{end} + \hat{I}_{side}^{side}\right) + RD \left(c^{end} - c^m\right) \left(\hat{I}_{end}^{end} + \hat{I}_{end}^{side}\right) \end{aligned} \tag{3.21}$$

Here, $\hat{I}$ represents the dimensionless flux found from a numerical solution of the diffusion problem. The definition of the different flux terms is depicted below.

$$\hat{I}_{side}^{end} = 4\pi \int_{\hat{r}=0}^{1} \frac{\partial \hat{c}_{side}}{\partial \hat{z}}\bigg|_{\hat{z}=\alpha} \hat{r} d\hat{r} \tag{3.22}$$

$$\hat{I}_{end}^{end} = 4\pi \int_{\hat{r}=0}^{1} \frac{\partial \hat{c}_{end}}{\partial \hat{z}}\bigg|_{\hat{z}=\alpha} \hat{r} d\hat{r} \tag{3.23}$$

$$\hat{I}_{side}^{side} = 4\pi \int_{\hat{z}=0}^{\alpha} \frac{\partial \hat{c}_{side}}{\partial \hat{r}}\bigg|_{\hat{r}=1} d\hat{z} \tag{3.24}$$

$$\hat{I}_{end}^{side} = 4\pi \int_{\hat{z}=0}^{\alpha} \frac{\partial \hat{c}_{end}}{\partial \hat{r}}\bigg|_{\hat{r}=1} d\hat{z} \tag{3.25}$$

To clarify, $\hat{I}_{side}^{end}$ and $\hat{I}_{end}^{end}$ are the flux from the end surfaces of the considered particle with a $\hat{c}_{side}$-field and $\hat{c}_{end}$-field, respectively. Similarly, $\hat{I}_{side}^{side}$ and $\hat{I}_{end}^{side}$ are the flux from the side walls of the particle with a $\hat{c}_{side}$-field and $\hat{c}_{end}$-field, respectively.

Moreover, the current of solute to the cylindrical particle surface will yield growth rates. These growth rates can be found by applying a flux balance at the interfaces of the cylinder. An illustration of the two types of growth for a cylindrical particle is shown in Figure 3.1.

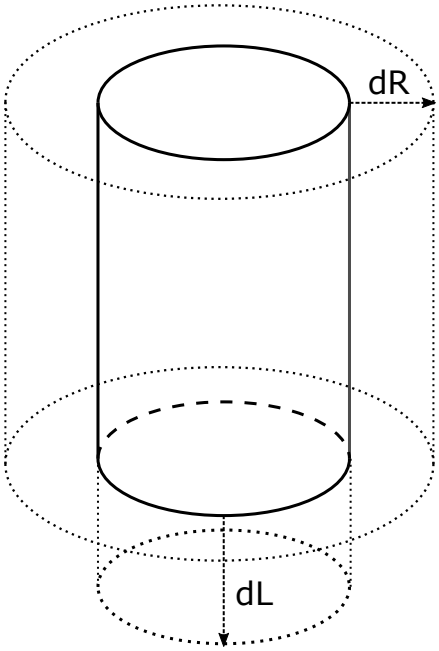


Figure 3.1: An illustration of the two migrating interfaces of a cylindrical particle.

As seen from Figure 3.1, the particle can either lengthen in the axial direction equal to $\pi R^2 dL$, or expand in the radial direction equal to $2\pi RLdR$. A flux balance at the end surfaces yields the following relation.

$$I_{end}\, dt = \left(\chi c^p - c^{end}\right) \pi R^2 dL \tag{3.26}$$

Here, I_{end} is denoted as the first term in equation (3.20) which denotes the current of solute atoms from the end surfaces, and $\chi = V_p^B / V_m^B$, where V_m^B is the average atomic volume per mole B atoms in the matrix, while V_p^B is the average atomic

volume per mole B atoms in the particle. Similarly, a flux balance at the sidewalls of the particle exists with I_{side} denoting the second term of equation (3.20).

$$I_{side}\, dt = \left(\chi c^{p} - c^{side}\right) 2\pi RL\, dR \tag{3.27}$$

By introducing equation (3.21) with the respective fluxes into equation (3.26) and (3.27), one can express the axial and radial growth rates of the considered particle.

$$\frac{dL}{dt} = \frac{D\left(c^{side} - c^{m}\right)\hat{I}^{end}_{side}(\alpha) + D\left(c^{end} - c^{m}\right)\hat{I}^{end}_{end}(\alpha)}{\pi R\left(\chi c^{p} - c^{end}\right)} \tag{3.28}$$

$$\frac{dR}{dt} = \frac{D\left(c^{side} - c^{m}\right)\hat{I}^{side}_{side}(\alpha) + D\left(c^{end} - c^{m}\right)\hat{I}^{side}_{end}(\alpha)}{2\pi L\left(\chi c^{p} - c^{side}\right)} \tag{3.29}$$

As seen from equation (3.28) and (3.29), in order to successfully calculate the growth rate of the particle one needs to both include the Gibbs-Thomson effect for c^{end} and c^{side}, but also solve the diffusion problem for a wide range of aspect ratios since the dimensionless flux is a function of the aspect ratio. The diffusion solution will be further explained in section 3.4, while the Gibbs-Thomson effect will be reviewed in section 3.2.

3.2 The Gibbs-Thomson Effect

To successfully implement a precipitation model for cylindrical particles, the Gibbs-Thomson equation for spherical particles which is shown in equation (2.17), needs to be modified to account for a cylindrical configuration. By following Perez's [14] work one can adjust the derivation of the Gibbs-Thomson equation to allow for a cylindrical particle with a distinction between the cylinder's side and end surfaces.

In the presented model there exists a primary α phase which consists of a binary solution of n_a mol A atoms and n_b mol B atoms. By assuming the free energy is due to the bond energy of adjacent atoms which is equivalent to a regular solution, one can express the Gibbs free energy of the phase as following,

$$G^{\alpha} = n_a \left(G_A + kT \ln \left(\frac{n_a}{n_a + n_b} \right) \right) + n_b \left(G_B + kT \ln \left(\frac{n_b}{n_a + n_b} \right) \right) + \frac{\Omega \, n_a n_b}{n_a + n_b} \tag{3.30}$$

where G_A and G_B are the molar free energies of the pure A and B phase, respectively, k is the Boltzmann constant, T is the temperature in kelvin and Ω is a regular solution constant depending on bond energies and coordination numbers.

Similarly there exists a second-phase particle β which consists of a known composition of $A_x B_x$, where the molar concentration of B in the β particle is equivalent to $X_p = y/(x+y)$. Consequently, an expression for the Gibbs free energy of the β phase can be made. Assuming that the β particle is perfectly ordered and without configurational entropy, the resulting Gibbs free energy including surface energy of the side and end of the cylindrical particle is shown below.

$$G^{\beta} = n^{\beta} G_n^{\beta} + 2\pi R L \gamma_{side} + 2\pi R^2 \gamma_{end} \tag{3.31}$$

Here, G_n^{β} is the Gibbs free energy per atom of β phase, n^{β} is the number of atoms in the β phase, R and L is the radius and length of the cylindrical particle, respectively and γ_{side} and γ_{end} are the surface energies of the side and end surfaces, respectively. Further, one can deduce that if the α phase is in equilibrium with the β phase, a small transfer of β phase molecules dn^{β} across a surface element with normal vector $\vec{n}$, from the particle into the surrounding α phase matrix as $\left(1 - X_p\right) dn^{\beta}$ A atoms and $X_p \, dn^{\beta}$ B atoms will not change the energy of the system. Consequently, one can show that the following expression in equation (3.32) exists.

$$\frac{\partial G^{\beta}}{\partial n^{\beta}} \, dn^{\beta} = \frac{\partial G^{\alpha}}{\partial n_A^{\alpha}} \left(1 - X_p\right) dn^{\beta} + \frac{\partial G^{\alpha}}{\partial n_B^{\alpha}} X_p \, dn^{\beta} \tag{3.32}$$

As seen from equation (3.32), an expression for the chemical potential for the particle, G_n^{β}, can be derived by considering that the β phase is in equilibrium with the primary α phase of composition X_{eq}^{∞}. By assuming a dilute regular solution, i.e $X_{eq}^{\infty} << 1$, the following expression for G_n^{β} can be derived.

$$G_n^\beta = \left(1 - X_p\right)\left(G_A^\alpha + kT\ln\left(1 - X_{eq}^\infty\right)\right) + X_p\left(G_B^\alpha + \Omega + kT\ln\left(X_{eq}^\infty\right)\right) \quad (3.33)$$

Additionally, the total volume of the particle is described as $V_p = \pi R^2 L = n^\beta V_p^\beta$. When the particle receives atoms across the sides, the length L remains constant, while the radius R is altered. Consequently, the radius of the particle can be expressed as shown in equation (3.34).

$$R = \sqrt{\frac{V_p^\beta n^\beta}{\pi L}} \quad (3.34)$$

Hence, the derivative of equation (3.31) with respect to n^β at a constant length, results in equation (3.35).

$$\left.\frac{\partial G^\beta}{\partial n^\beta}\right|_{L=const} - G_n^\beta = 2\pi\left(L\gamma_{side} + 2R\gamma_{end}\right)\frac{\partial R}{\partial n^\beta} = \frac{\left(\frac{L}{R}\gamma_{side} + 2\gamma_{end}\right)V_p^\beta}{L} \quad (3.35)$$

Similarly, when atoms are transferred across the ends of the particle, the radius R remains constant, while the length L is altered. The length of the particle can be expressed as following,

$$L = \frac{V_p^\beta n^\beta}{\pi R^2} \quad (3.36)$$

and ultimately, the derivative of equation (3.31) with respect to n^β at a constant radius, is given in equation (3.37).

$$\left.\frac{\partial G^\beta}{\partial n^\beta}\right|_{R=const} - G_n^\beta = 2\pi R\gamma_{side}\frac{\partial L}{\partial n^\beta} = \frac{2\gamma_{side}V_p^\beta}{R} \quad (3.37)$$

Further, by applying the same assumption of a regular solution as equation (3.30) and assumption of local equilibrium at each particle surface from equation (3.32), i.e diffusion-controlled interface motion, one can show that the following equilibrium conditions hold,

$$G_n^\beta + \frac{2\left(\frac{L}{R}\gamma_{side} + 2\gamma_{end}\right)V_p^\beta}{L} = \left(1 - X_p\right)\left(G_A^\alpha + kT\ln\left(1 - X_{eq}^{side}\right)\right) + X_p\left(G_B^\alpha + \Omega + kT\ln\left(X_{eq}^{side}\right)\right) \quad (3.38)$$

$$G_n^\beta + \frac{2\gamma_{side}V_p^\beta}{R} = \left(1 - X_p\right)\left(G_A^\alpha + kT\ln\left(1 - X_{eq}^{end}\right)\right) + X_p\left(G_B^\alpha + \Omega + kT\ln\left(X_{eq}^{end}\right)\right) \quad (3.39)$$

where X_{eq}^{side} and X_{eq}^{end} are the equilibrium compositions at the side and end surfaces of the particle, respectively. Also, one can obtain the general form of the Gibbs-Thomson equation, albeit modified to account for a cylindrical particle, by introducing equation (3.33) into equation (3.38) and (3.39). The general form is then given for the side and end surfaces below in equation (3.40) and (3.41), respectively.

$$\frac{2\left(\frac{L}{R}\gamma_{side} + 2\gamma_{end}\right)V_p^\beta}{LkT} = \left(1 - X_p\right)\ln\left(\frac{1 - X_{eq}^{side}}{1 - X_{eq}^{\infty}}\right) + X_p\ln\left(\frac{X_{eq}^{side}}{X_{eq}^{\infty}}\right) \quad (3.40)$$

$$\frac{2\gamma_{side}V_p^\beta}{RkT} = \left(1 - X_p\right)\ln\left(\frac{1 - X_{eq}^{end}}{1 - X_{eq}^{\infty}}\right) + X_p\ln\left(\frac{X_{eq}^{end}}{X_{eq}^{\infty}}\right) \quad (3.41)$$

In order to solve the general Gibbs-Thomson equations for X_{eq}^{side} and X_{eq}^{end}, one needs to introduce some approximations. Two simple and relevant approximations are when $X_p \approx 1$, i.e pure precipitates, or the case of diluted solid solutions where $X_{eq}^{side}, X_{eq}^{end} << 1$ and $X_{eq}^{\infty} << 1$ [14]. By applying the most common approximation of $X_p \approx 1$, equation (3.40) and (3.41) are reduced to the following equations for X_{eq}^{side} and X_{eq}^{end}.

$$X_{eq}^{side} = X_{eq}^{\infty}\exp\left(\frac{\left(\gamma_{side} + 2\frac{R}{L}\gamma_{end}\right)V_p^\beta}{RkT}\right) \quad (3.42)$$

$$X_{eq}^{end} = X_{eq}^{\infty} \exp\left(\frac{2\gamma_{side} V_p^{\beta}}{RkT}\right) \tag{3.43}$$

In addition, the approximation proposed by Perez [14] for the case of diluted solid solutions where $X_{eq}^{side}, X_{eq}^{end} << 1$ and $X_{eq}^{\infty} << 1$, can be used to derive an expression for the equilibrium concentrations at the surfaces of the particle. In this case the first term of equation (3.40) and (3.41) are negligible to the second one, except when $X_{eq}^{side}, X_{eq}^{end} \approx X_{eq}^{\infty}$. One can rearrange equation (3.40) as the following through doing a series expansion of the logarithmic terms.

$$\begin{aligned}\frac{2\left(\frac{L}{R}\gamma_{side} + 2\gamma_{end}\right) V_p^{\beta}}{LkT} = \\ \left(1 - X_p\right) \ln\left(1 + \frac{X_{eq}^{\infty} - X_{eq}^{side}}{1 - X_{eq}^{\infty}}\right) + X_p \ln\left(1 + \frac{X_{eq}^{side} - X_{eq}^{\infty}}{X_{eq}^{\infty}}\right) = \\ \left(1 - X_p\right) \frac{X_{eq}^{\infty} - X_{eq}^{side}}{1 - X_{eq}^{\infty}} + X_p \frac{X_{eq}^{side} - X_{eq}^{\infty}}{X_{eq}^{\infty}}\end{aligned} \tag{3.44}$$

By letting $\varepsilon = X_{eq}^{side} - X_{eq}^{\infty}$, equation (3.44) is then transformed into equation (3.45)

$$\frac{2\left(\frac{L}{R}\gamma_{side} + 2\gamma_{end}\right) V_p^{\beta}}{LkT} = \left(1 - X_p\right)(-\varepsilon) + X_p \frac{\varepsilon}{X_{eq}^{\infty}} \tag{3.45}$$

As can be seen from equation (3.45), the first term is also negligible here compared to the second one. A similar treatment can be done for equation (3.41). Therefore, equation (3.40) and (3.41) are deduced to the following two expressions for the X_{eq}^{side} and X_{eq}^{end} for the particle side and end surfaces, respectively.

$$X_{eq}^{side} = X_{eq}^{\infty} \exp\left(\frac{\left(\gamma_{side} + 2\frac{R}{L}\gamma_{end}\right) V_p^{\beta}}{X_p RkT}\right) \tag{3.46}$$

$$X_{eq}^{end} = X_{eq}^{\infty} \exp\left(\frac{2\gamma_{side} V_p^{\beta}}{X_p RkT}\right) \tag{3.47}$$

Equation (3.46) and (3.47) are the most versatile of the two proposed solutions as both equations reduce to equation (3.42) and (3.43) when $X_p = 1$. Thus, equation (3.46) and (3.47) are the ones implemented in the numerical model.

An implication rarely described in the literature is that the derivation of the Gibbs-Thomson equations yields an expression for atomic fraction X, while the growth equations demand volume fractions. The volume concentration c^m of atom type B in the matrix is expressed:

$$c^m = \frac{V_m^B\, n^B}{V_m^B\, n^B + V_m^A\, n^A} = \frac{V_m^B}{V_m^B + V_m^A \frac{n_A}{n_B}} = \frac{V_m^B X_{eq}^\infty}{V_m^B X_{eq}^\infty + V_m^A \left(1 - X_{eq}^\infty\right)} = \frac{X_{eq}^\infty}{X_{eq}^\infty + \frac{V_m^A}{V_m^B}\left(1 - X_{eq}^\infty\right)} \tag{3.48}$$

For dilute solutions, one can simplify the expression in equation (3.48):

$$c^m \approx \frac{V_m^B}{V_m^A} X_{eq}^\infty \tag{3.49}$$

Similar treatments can be done for the other concentration parameters. For simplification, this thesis will assume a V_m^B/V_m^A-ratio approximately equal to 1. This implies that one can directly use the atom fractions found from the Gibbs-Thomson equations in the mathematical model.

3.2.1 Non-Constant Surface Energies

The derivations in the previous section assumed constant surface energies. One can also imagine that the surface energy of the side or end surfaces changes with increasing length or radius. For needle-like precipitates, the growth rate in the length direction should be higher than the radial growth rate to facilitate long needles. As the growth rates are influenced by the Gibbs-Thomson relation, it is beneficial to derive a Gibbs-Thomson equation that can account for changes in the surface energies. For needles, the most relevant is when γ_{end} varies with the radius of the cylindrical particle. Hence, equation (3.35) is transformed into the following relation when γ_{end} is a function of R.

$$\left.\frac{\partial G^{\beta}}{\partial n^{\beta}}\right|_{L=const} - G_n^{\beta} = 2\pi \left(L\gamma_{side} + 2R\gamma_{end}(R) + R^2 \frac{d\gamma_{end}}{dR} \right) \frac{\partial R}{\partial n^{\beta}}$$
$$= \frac{\left(\frac{L}{R}\gamma_{side} + 2\gamma_{end}(R) + R\frac{d\gamma_{end}}{dR}\right) V_p^{\beta}}{L} \quad (3.50)$$

Using the same treatment as in the previous section, one can obtain an expression for the new Gibbs-Thomson equation with a non-constant end surface energy.

$$X_{eq}^{side} = X_{eq}^{\infty} \exp\left(\frac{\left(\gamma_{side} + \frac{R^2}{L}\frac{\mathrm{d}\gamma_{end}}{\mathrm{d}R} + 2\frac{R}{L}\gamma_{end}(R)\right) V_p^{\beta}}{X_p RkT} \right) \quad (3.51)$$

3.2.2 Surface Energy Function

A proposed equation for γ_{end} is one that ramps up when the radius of the precipitate increases. This is due to the fact that the interfacial energy of, for instance β'' in Al-Mg-Si, is size dependent due to gradual losses of coherency [30]. As seen from Murayama and Hono [31], the nuclei are of spherical character and later grows into needles for the Al-Mg-Si alloys. Therefore, the γ_{end} function should be equal to γ_{side} at nucleation to facilitate a more spherical character. From there γ_{end} can be ramped up to a threshold value. An illustration of the proposed upramp of γ_{end} and the accompanying suggestion for $\gamma_{end}(R)$ is shown in Figure 3.2. Here, the linear upramp and $\gamma_{end}(R)$ is as described below:

$$\gamma = \begin{cases} \gamma_1, & R < R_1 \\ \gamma_1 + \frac{\gamma_2 - \gamma_1}{R_2 - R_1} R, & R_1 \leq R \leq R_2 \\ \gamma_2, & R > R_2 \end{cases} \quad (3.52)$$

$$\gamma(R) = \gamma_1 + \frac{1}{2}(\gamma_2 - \gamma_1)\left(1 + \tanh\left(\frac{(\gamma_2 - \gamma_1)\left(R - \frac{1}{2}R_1 - \frac{1}{2}R_2\right)}{2(R_2 - R_1)}\right)\right) \quad (3.53)$$

Equation (3.53) can then be used in the model for varying γ_{end}. The proposed equation is only an educated guess and is not a representation of the actual surface

energy of the end surfaces. However, this can be used to create qualitative results for needle-like precipitates.

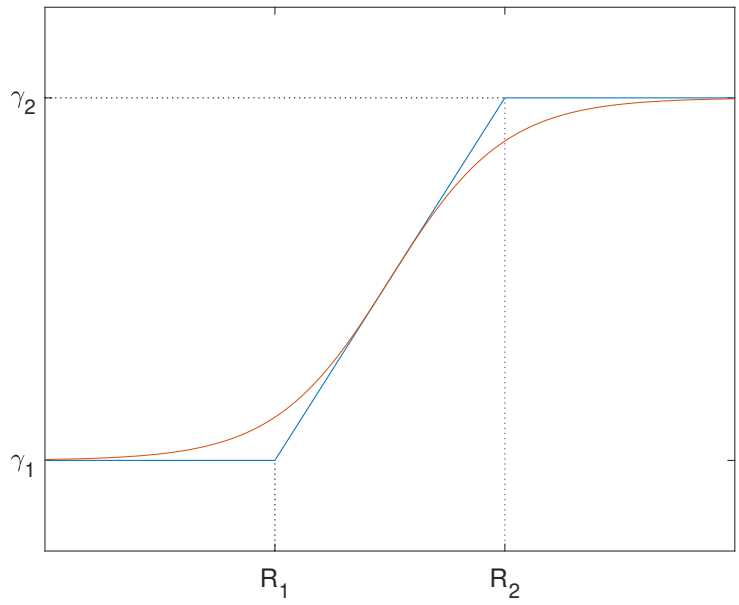


Figure 3.2: An illustration of the linear upramp of γ_{end} and proposed equation for γ_{end}.

3.3 Critical Nucleus

The critical radius R^* and length L^* can both be found from the growth equations. They are defined as the minimum radius and length of the precipitate to achieve a stable critical nucleus. If the radius or length is below R^* or L^*, the nuclei will achieve a negative growth rate and will readily shrink and disappear. From equation (3.28) and (3.29), the following condition has to be satisfied for R^* and L^*.

$$0 = \frac{D\left(c^{side} - c^m\right)\hat{I}^{end}_{side}(\alpha) + D\left(c^{end} - c^m\right)\hat{I}^{end}_{end}(\alpha)}{\pi R\left(\chi c^p - c^{end}\right)} \tag{3.54}$$

$$0 = \frac{D\left(c^{side} - c^{m}\right)\hat{I}^{side}_{side}(\alpha) + D\left(c^{end} - c^{m}\right)\hat{I}^{side}_{end}(\alpha)}{2\pi L\left(\chi c^{p} - c^{side}\right)} \tag{3.55}$$

An obvious solution arise when $c^m = c^{side} = c^{end}$ which satisfies both equations. Thus, the following relation exist when inserting equation (3.47) and $c^m = c^{end}$.

$$c^{m} = c^{eq}\exp\left(\frac{2\gamma_{side}V_p^{\beta}}{c^{p}R^{*}kT}\right) \tag{3.56}$$

Which leads to an appropriate equation for R^*.

$$R^{*} = \frac{2\gamma_{side}V_p^{\beta}}{\ln\left(\frac{c^{m}}{c^{\infty}_{eq}}\right)c^{p}kT} \tag{3.57}$$

Similarly, an expression for L^* is obtainable by inserting equation (3.46) or equation (3.51) depending on if the surface energy γ_{end} is constant or not. For a constant γ_{end} the following expression arises when $c^{side} = c^{end}$.

$$\gamma_{side} + 2\frac{R^{*}}{L^{*}}\gamma_{end} = 2\gamma_{side} \tag{3.58}$$

Further, introducing the aspect ratio α into equation (3.58) yields an expression for the critical aspect ratio, and consequently, an expression for the critical length.

$$\alpha^{*} = \frac{\gamma_{end}}{\gamma_{side}}, \qquad L^{*} = 2\,\alpha^{*}R^{*} \tag{3.59}$$

For a non-constant γ_{end}, equation (3.58) transforms into the following equation.

$$\gamma_{side} + \frac{{R^{*}}^{2}}{L^{*}}\left.\frac{d\gamma_{end}}{dR}\right|_{R=R^{*}} + 2\frac{R^{*}}{L^{*}}\gamma_{end}(R^{*}) = 2\gamma_{side} \tag{3.60}$$

Which results in a new expression for L^* as shown in equation (3.61).

$$L^* = \frac{{R^*}^2}{\gamma_{side}} \frac{d\gamma_{end}}{dR}\bigg|_{R=R^*} + 2\frac{\gamma_{end}(R^*)}{\gamma_{side}} R^* \tag{3.61}$$

As one can observe from equation (3.61), L^* reduces to equation (3.59) when γ_{end} is constant.

3.4 Algorithm for Numerical Diffusion Solution

The foundation of the numerical model used to calculate the resulting concentration field around a cylindrical particle was built in the computing language Fortran during a project work [10], and will only be briefly summarized in this section with the new boundary conditions set in section 3.1. To numerically solve the diffusion field around a cylindrical particle, a finite difference scheme was implemented to solve equation (3.7) or (3.13) by the use of a line-by-line method where the domain is swept column- and row-wise. Each sweep is then calculated using a tridiagonal matrix solver. In addition, the concentration flux at the interfaces was calculated using a trapezoid method. As mentioned, an assumption of quasi-steady conditions is used which enables one to use the Laplace form of the diffusion equation.

3.4.1 Grid and Computational Domain

For evaluating the diffusion problem, a computer domain consisting of a grid of nodes in which each node exhibits a concentration value. From the boundary conditions in section 3.1, the concentration values range from 1 at the particle interface to 0 in the matrix far away from the considered particle. An illustration of the computation domain is shown in Figure 3.3 [10].

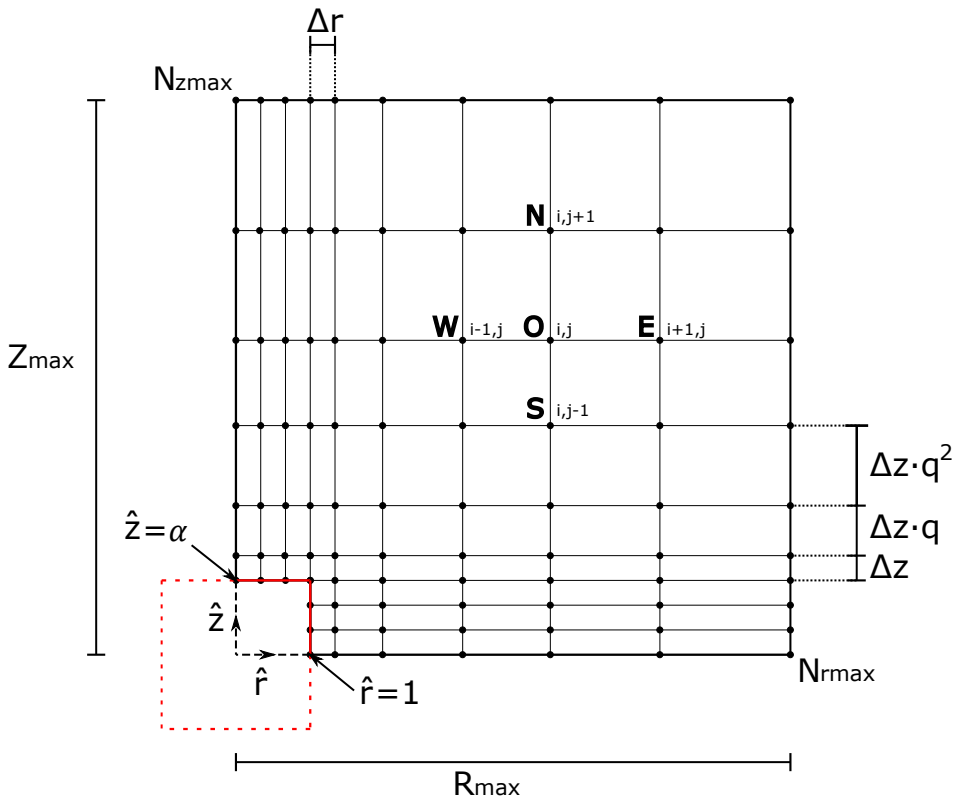


Figure 3.3: An illustration of the computational domain and grid for the diffusion model [10].

Here, R_{max} and Z_{max} is the length of the grid in the radial and axial direction, respectively. Additionally, N_{rmax} and N_{zmax} is the number of nodes in the respective directions. To accurately satisfy the outer boundary condition presented in equation (3.10) and (3.16), a stretched grid is used where the distance between neighbouring nodes is increased by a factor q, where Δr and Δz is the initial nodal spacing used near the particle interface.

3.4.2 Finite Difference Scheme

To numerically obtain the diffusion field, equation (3.7) and (3.13) has to be discretized into nodal equations. As shown in section 2.1.1, the Laplace equation becomes the following equation in dimensionless cylindrical coordinates with angular symmetry.

$$\frac{1}{\hat{r}}\frac{\partial \hat{c}}{\partial \hat{r}} + \frac{\partial^2 \hat{c}}{\partial \hat{r}^2} + \frac{\partial^2 \hat{c}}{\partial \hat{z}^2} = 0 \tag{3.62}$$

Due to the boundary conditions of the outer nodes, a distinction between the inner and outer nodes is necessary. The governing equations of the diffusion problem can be categorized into the outer and inner nodal equations and will be reviewed in the following two sections.

Inner Nodal Equations

An illustration of the stencil for the inner nodes of the domain is shown in Figure 3.4. For the inner nodes a Taylor series expansion of $\hat{c}_{i+1,j}$ and $\hat{c}_{i-1,j}$ around $\hat{c}_{i,j}$ is done. From the Taylor series expansion, an expression for the second derivative with respect to $\hat{r}$ was found to be as shown in equation (3.63).

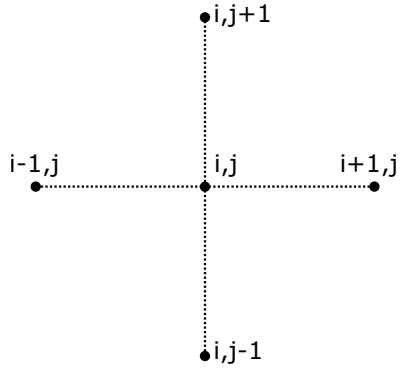


Figure 3.4: An illustration of the stencil for the inner nodes [10].

$$\begin{aligned}\left.\frac{\partial^2 \hat{c}}{\partial \hat{r}^2}\right|_{i,j} =& \frac{2}{\Delta\hat{r}_E\left(\Delta\hat{r}_E + \Delta\hat{r}_W\right)}\,\hat{c}_{i+1,j} + \frac{2}{\Delta\hat{r}_W\left(\Delta\hat{r}_E + \Delta\hat{r}_W\right)}\,\hat{c}_{i-1,j} \\ & - \frac{2}{\Delta\hat{r}_E\,\Delta\hat{r}_W}\,\hat{c}_{i,j}\end{aligned} \tag{3.63}$$

Here, $\Delta\hat{r}_E = \hat{r}_{i+1,j} - \hat{r}_{i,j}$ and $\Delta\hat{r}_W = \hat{r}_{i,j} - \hat{r}_{i-1,j}$. A truncation error appearing from the Taylor series expansion was found to be as depicted below.

$$\varepsilon = \frac{1}{3}\left(\Delta\hat{r}_E - \Delta\hat{r}_W\right)\left.\frac{\partial^3\hat{c}}{\partial\hat{r}^3}\right|_{i,j} + \cdots \tag{3.64}$$

Similarly, the same treatment can be done for the second derivative with respect to $\hat{z}$. The Taylor series expansion was then made of $\hat{c}_{i,j+1}$ and $\hat{c}_{i1,j-1}$ around $\hat{c}_{i,j}$. The resulting nodal equation was then found to be:

$$\begin{aligned}\left.\frac{\partial^2\hat{c}}{\partial\hat{z}^2}\right|_{i,j} =& \frac{2}{\Delta\hat{z}_N\left(\Delta\hat{z}_N + \Delta\hat{z}_S\right)}\,\hat{c}_{i,j+1} + \frac{2}{\Delta\hat{z}_S\left(\Delta\hat{z}_N + \Delta\hat{z}_S\right)}\,\hat{c}_{i,j-1} \\ &- \frac{2}{\Delta\hat{z}_N\,\Delta\hat{z}_S}\,\hat{c}_{i,j}\end{aligned} \tag{3.65}$$

Where $\Delta\hat{z}_N = \hat{r}_{i,j+1} - \hat{r}_{i,j}$ and $\Delta\hat{z}_S = \hat{r}_{i,j} - \hat{r}_{i,j-1}$. The related truncation error is depicted below.

$$\varepsilon = \frac{1}{3}\left(\Delta\hat{z}_N - \Delta\hat{z}_S\right)\left.\frac{\partial^3\hat{c}}{\partial\hat{z}^3}\right|_{i,j} + \cdots \tag{3.66}$$

From the same Taylor series expansion as used for equation (3.63), an expression for the first derivative of equation (3.62) can be found as shown in equation (3.67).

$$\begin{aligned}\left.\frac{\partial\hat{c}}{\partial\hat{r}}\right|_{i,j} =& \frac{\Delta\hat{r}_W}{\Delta\hat{r}_E\left(\Delta\hat{r}_E + \Delta\hat{r}_W\right)}\,\hat{c}_{i+1,j} - \frac{\Delta\hat{r}_E}{\Delta\hat{r}_W\left(\Delta\hat{r}_E + \Delta\hat{r}_W\right)}\,\hat{c}_{i-1,j} \\ &- \frac{\Delta\hat{r}_W - \Delta\hat{r}_E}{\Delta\hat{r}_W\Delta\hat{r}_E}\,\hat{c}_{i,j}\end{aligned} \tag{3.67}$$

With the corresponding truncation error as shown below.

$$\varepsilon = \Delta\hat{r}_W\Delta\hat{r}_E\left.\frac{\partial^3\hat{c}}{\partial\hat{r}^3}\right|_{i,j} + \cdots \tag{3.68}$$

Adding equation (3.63), (3.65) and (3.67) together yields the discretized form of equation (3.62). The discretized equation is shown in equation (3.69) in the form used in the Fortran program.

$$\begin{aligned}
&\frac{3\hat{r}_i - \hat{r}_{i-1}}{\hat{r}_i\left(\hat{r}_{i-1} - \hat{r}_{i+1}\right)\left(\hat{r}_i - \hat{r}_{i+1}\right)}\,\hat{c}_{i+1,j} + \\
&\frac{3\hat{r}_i - \hat{r}_{i+1}}{\hat{r}_i\left(\hat{r}_{i-1} - \hat{r}_{i+1}\right)\left(\hat{r}_{i-1} - \hat{r}_i\right)}\,\hat{c}_{i-1,j} + \\
&\frac{2}{\left(\hat{z}_{j+1} - \hat{z}_j\right)\left(\hat{z}_{j+1} - \hat{z}_{j-1}\right)}\,\hat{c}_{i,j+1} + \\
&\frac{2}{\left(\hat{z}_j - \hat{z}_{j-1}\right)\left(\hat{z}_{j+1} - \hat{z}_{j-1}\right)}\,\hat{c}_{i,j-1} + \\
&\left(\frac{\hat{r}_{i-1} - 4\hat{r}_i + \hat{r}_{i+1}}{\hat{r}_i\left(\hat{r}_{i-1} - \hat{r}_i\right)\left(\hat{r}_i - \hat{r}_{i+1}\right)} - \frac{2}{\left(\hat{z}_{j+1} - \hat{z}_j\right)\left(\hat{z}_j - \hat{z}_{j-1}\right)}\right)\hat{c}_{i,j} = 0
\end{aligned} \tag{3.69}$$

Outer Nodal Equations

As a consequence of the symmetry of the domain, the governing equations at $\hat{r} = 0$ and $\hat{z} = 0$ exhibits a Neumann boundary as described in equation (3.11) and (3.12). However, as can be seen from equation (3.62), a singularity appears when $\hat{r}$ and $\partial\hat{c}/\partial\hat{r}$ reaches zero. A solution was to evaluate the first derivative term as it approaches zero. One can achieve this by applying L'Hôpitals rule.

$$\lim_{\hat{r}=0} \frac{1}{\hat{r}} \frac{\partial \hat{C}}{\partial \hat{r}} = \lim_{\hat{r}=0} \frac{\frac{\partial}{\partial \hat{r}}\left(\frac{\partial \hat{C}}{\partial \hat{r}}\right)}{\frac{\partial}{\partial \hat{r}}\left(\hat{r}\right)} = \left.\frac{\partial^2 \hat{C}}{\partial \hat{r}^2}\right|_{\hat{r}=0} \tag{3.70}$$

As a consequence, the governing equation is transformed into the following form for the boundary nodes at $\hat{r} = 0$:

$$2\frac{\partial^2 \hat{C}}{\partial \hat{r}^2} + \frac{\partial^2 \hat{C}}{\partial \hat{z}^2} = 0 \tag{3.71}$$

The stencil for the leftmost nodes at $\hat{r} = 0$ is shown in Figure 3.5. Through a "ghost" point technique, one can assign a central difference scheme to the boundary nodes as done with the inner nodes. Utilizing the boundary condition presented in equation (3.12), one can express the $\hat{c}_{0,j}$ node, as seen from Figure 3.5, to be equal to the $\hat{c}_{2,j}$ node. Therefore, the resulting discretized equation for the leftmost nodes are as depicted in equation (3.72).

$$\begin{aligned}
&\frac{4}{\left(\hat{r}_{2,j}-\hat{r}_{1,j}\right)^2}\hat{C}_{2,j}- \\
&\frac{2}{\left(\hat{z}_{1,j+1}-\hat{z}_{1,j}\right)\left(\hat{z}_{1,j+1}-\hat{z}_{1,j-1}\right)}\hat{C}_{1,j+1}+ \\
&\frac{2}{\left(\hat{z}_{1,j}-\hat{z}_{1,j-1}\right)\left(\hat{z}_{1,j+1}-\hat{z}_{1,j-1}\right)}\hat{C}_{1,j-1}+ \\
&\left(\frac{4}{\left(\hat{r}_{2,j}-\hat{r}_{1,j}\right)^2}+\frac{2}{\left(\hat{z}_{1,j+1}-\hat{z}_{1,j}\right)\left(\hat{z}_{1,j}-\hat{z}_{1,j-1}\right)}\right)\hat{C}_{1,j}=0
\end{aligned} \tag{3.72}$$

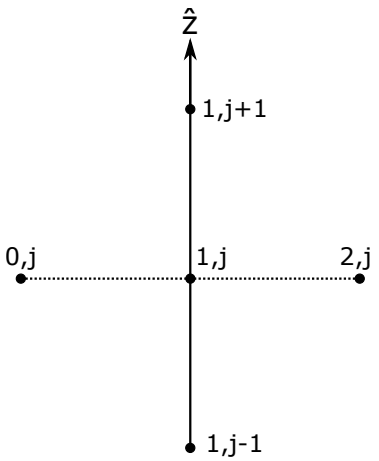


Figure 3.5: An illustration of the stencil for the leftmost nodes [10].

The same "ghost" point technique can be utilized on the bottom nodes at $\hat{z}=0$. The stencil for the bottom nodes are shown in Figure 3.6. As seen from the figure, the boundary conditions of $\partial\hat{c}/\partial\hat{z}=0$, yields that $\hat{c}_{i,2}=\hat{c}_{i,0}$, and the discretized equation for the bottom nodes are as depicted in equation (3.73).

$$\begin{aligned}
&\frac{2}{\left(\hat{z}_2-\hat{z}_1\right)^2}\hat{C}_{i,2}+\\
&\frac{3\hat{r}_i-\hat{r}_{i-1}}{\hat{r}_i\left(\hat{r}_{i-1}-\hat{r}_{i+1}\right)\left(\hat{r}_i-\hat{r}_{i+1}\right)}\hat{C}_{i+1,1}+\\
&\frac{3\hat{r}_i-\hat{r}_{i+1}}{\hat{r}_i\left(\hat{r}_{i-1}-\hat{r}_{i+1}\right)\left(\hat{r}_{i-1}-\hat{r}_i\right)}\hat{C}_{i-1,1}+\\
&\left(\frac{\hat{r}_{i-1}-4\hat{r}_i+\hat{r}_{i+1}}{\hat{r}_i\left(\hat{r}_{i-1}-\hat{r}_i\right)\left(\hat{r}_i-\hat{r}_{i+1}\right)}-\frac{2}{\left(\hat{z}_2-\hat{z}_1\right)^2}\right)\hat{C}_{i,1}=0
\end{aligned} \tag{3.73}$$

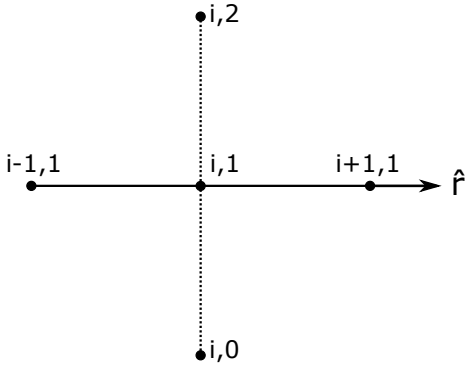


Figure 3.6: An illustration of the stencil for the bottom nodes [10].

3.4.3 Dimensionless Concentration Flux

After obtaining a concentration field around the cylindrical particle, the concentration flux at the interface of the particle is achievable. As mentioned in section 3.1, the dimensionless flux can expressed as:

$$\hat{I}^{end}=4\pi\int_{\hat{r}=0}^{1}\left.\frac{\partial\hat{c}}{\partial\hat{z}}\right|_{\hat{z}=\alpha}\hat{r}d\hat{r} \tag{3.74}$$

$$\hat{I}^{side}=4\pi\int_{\hat{z}=0}^{\alpha}\left.\frac{\partial\hat{c}}{\partial\hat{r}}\right|_{\hat{r}=1}d\hat{z} \tag{3.75}$$

The flux is then obtainable from the concentration field. For calculating fluxes at an interface, a forward Taylor series expansion is done which results in equation

(3.76) and (3.76) for $\partial\hat{c}/\partial\hat{z}$ and $\partial\hat{c}/\partial\hat{r}$. As noted from Figure 3.3, the stretching of the grid is constant at the two adjacent nodes next to the interface.

$$\left.\frac{\partial\hat{c}}{\partial\hat{r}}\right|_{N_R,\,j} = \frac{-3\hat{c}_{N_R,\,j} + 4\hat{c}_{N_R+1,\,j} - \hat{c}_{N_R+2,\,j}}{2\Delta\hat{r}_0} \tag{3.76}$$

$$\left.\frac{\partial\hat{c}}{\partial\hat{z}}\right|_{i,N_Z} = \frac{-3\hat{c}_{i,N_Z} + 4\hat{C}_{i,N_Z+1} - \hat{c}_{i,N_Z+2}}{2\Delta\hat{z}_0} \tag{3.77}$$

With the following truncation error.

$$\varepsilon = -\frac{1}{3}\Delta\,\hat{r}_0^2\,\frac{\partial^3\hat{c}}{\partial\hat{r}^3} \tag{3.78}$$

The stencil used for the side flux is shown in Figure 3.7.

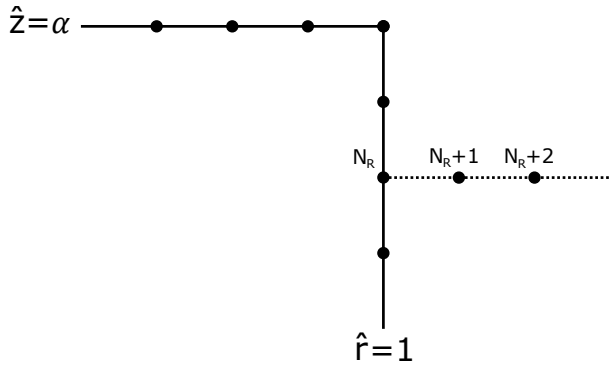


Figure 3.7: An illustration of the stencil used for flux calculations of the side of the particle [10].

3.5 Algorithm for Lagrangian for KWN-Model

The model presented in this thesis is utilizing a Lagrangian-like approach for calculating the resulting particle size distribution for the cylindrical particles. As mentioned, in the Lagrange-like approach the particle size distribution is divided into several sub-classes where each class exhibits a constant number of particles, and a radius R and length L. The mean radius, mean length and total number density of the particle size distribution are then as depicted below.

$$N = \sum_i N_i, \qquad \bar{R} = \frac{\sum_i N_i R_i}{\sum_i N_i}, \qquad \bar{L} = \frac{\sum_i N_i L_i}{\sum_i N_i} \tag{3.79}$$

The volume fraction of the cylindrical precipitates in the material can also be expressed.

$$f_p = \pi \sum_i N_i R_i^2 L_i \tag{3.80}$$

Also, a continuity equation exists which keeps track of how much solute atoms are tied up in the precipitates and how much is left in the matrix. For cylindrical precipitates, this mass balance is equal to equation (2.49) and is repeated here for the sake of readability.

$$c^m = \frac{c^0 - \chi c^p f_p}{1 - \chi f_p} \tag{3.81}$$

As mentioned in section 2.4.2, a new class is nucleated each time step with a set number density and a critical nucleus with a radius R^* and length L^*. As R^* and L^* signifies the critical size for zero growth, a growth margin $\Delta R^*/R^*$ has to be imposed to ensure that the nucleated particles initiate growth. A typical growth margin found from Myhr *et al.* [8], suggests a reasonable value of 0.05 for $\Delta R^*/R^*$. As L^* is dependent on R^*, the growth margin is also transferred to the critical length. The number density of the newly nucleated class is determined from equation (3.82).

$$N_i = \frac{dN}{dt}\Delta t \tag{3.82}$$

Where the nucleation rate dN/dt is given by equation (2.34). Typically a new class is only created if N_i is in the magnitude of 10^{-10} times or higher than the total number density N. At each time step, the growth of all current classes is also evaluated. The new radius and length of the precipitate class are then given as:

$$L(t + \Delta t) = L(t) + \frac{dL}{dt}\Delta t \tag{3.83}$$

$$R(t + \Delta t) = R(t) + \frac{dR}{dt}\Delta t \tag{3.84}$$

Where dL/dt and dR/dt is given by equation (3.28) and (3.29), respectively. Equation (3.83) and (3.84) are calculated using a variable-coefficient ordinary differential equation (VODE) solver developed by Brown *et al.* [32]. The current version implemented in the Fortran model utilize the double precision extension named DVODE.

After conducting a time step, the model checks whether any classes have disappeared by comparing R and L to a set threshold value where the precipitate is deemed to be small enough to have disappeared from the system. Therefore, in the Lagrangian-like approach, the number of classes in the system is continuously changing. During the nucleation stage the number of classes increases, while it decreases once the coarsening stage has been met. A flowchart describing the pseudo-code for the model implemented in Fortran can be viewed in Figure 3.8.

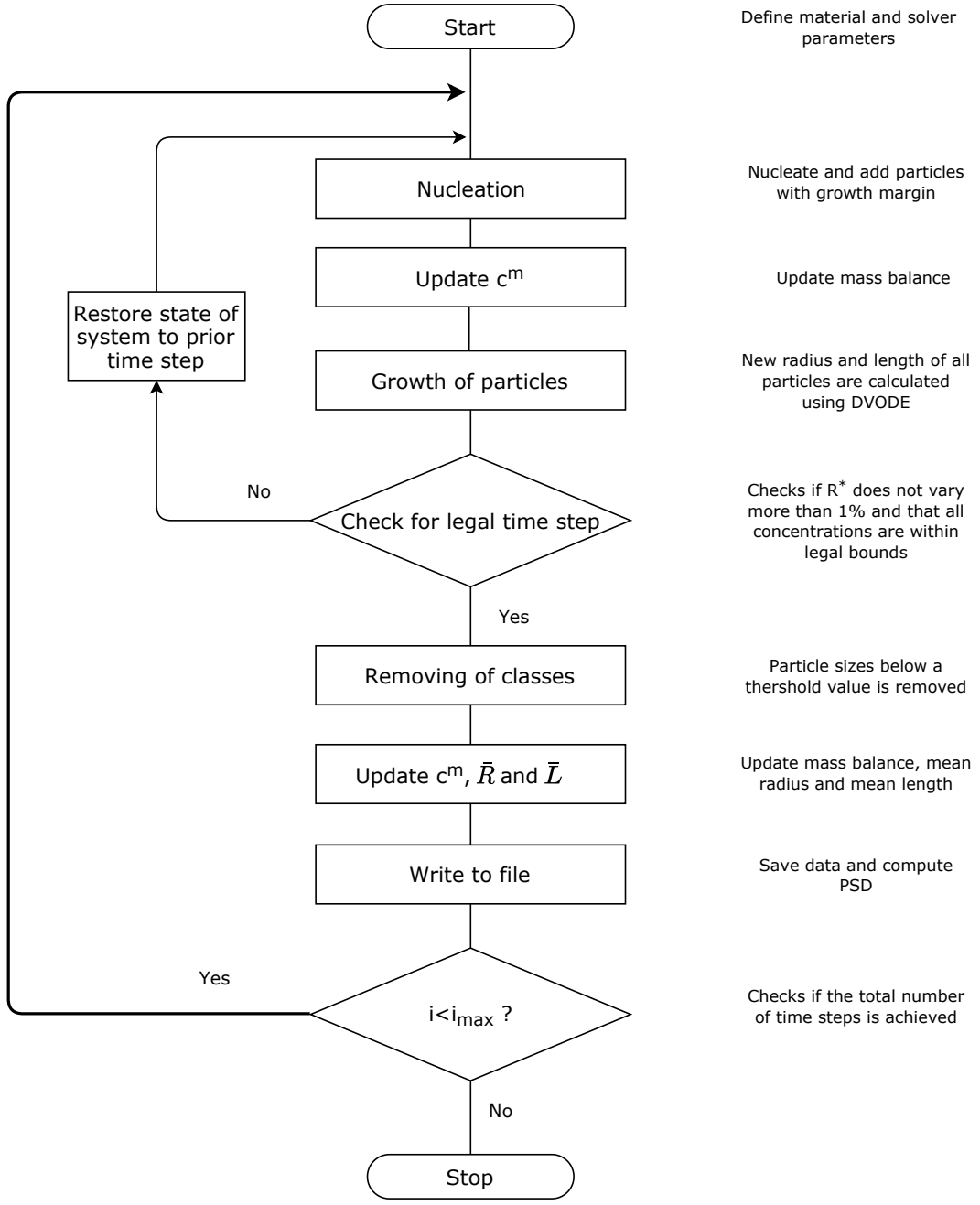


Figure 3.8: A flowchart describing the calculation steps for the precipitation model.

3.5.1 Adaptive Time Step

Each time step of the simulation is described by a time increment Δt. For the Lagrangian-like approach, the time step is of considerable importance as it controls the number of classes introduced to the system, and how often the mass balance in equation (3.81) is recalculated. The time increment therefore effectively affects the precision of the simulations. It is beneficial to introduce a logarithmic time increment as the simulation is solved for a large span in time. To ensure that special occurrences such as coarsening or other processes are accurately described, the time step is validated by requiring that R^* do not change more than a maximum of 1% and that all solute concentrations are within legal limits between 0 and 1 [21]. If these requirements are not met, a while-loop in the model is activated and the time step is run again at a lower time step, typically $\Delta t = \Delta t/2$. This requires that the simulation is saved before the time step so that one can refresh the state of the system when conducting the time step again.

3.5.2 Kernel Density Estimator in 2D

For visualizing the shape of the particle size distribution, a Gaussian kernel density estimator is implemented in the model. As mentioned, the distribution consists of classes with N_i precipitates and radius R_i and length L_i. The mean radius, mean length, and total number density are previously defined in equation (3.79). From these parameters, one can express the standard deviation σ_R and σ_L for R and L, respectively.

$$\sigma_R = \sqrt{\sum_i N_i \frac{\left(R - \bar{R}\right)^2}{N-1}}, \qquad \sigma_L = \sqrt{\sum_i N_i \frac{\left(L - \bar{L}\right)^2}{N-1}} \tag{3.85}$$

For a 1-dimensional representation of the size distribution for R, the kernel density estimator converts into the following expression by superimposing the normal distributions.

$$f(R) = \sum_i \frac{N_i}{h\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{R - R_i}{h}\right)^2\right) \tag{3.86}$$

Since the integral of each normal distribution is equal to 1, $\int f(R)dR = N$. A suitable choice of the parameter h must also be done. A simple and quick way of finding the bandwidth is to use Scott's estimate [28].

$$h \approx i_{tot}^{-0.2}\sigma \tag{3.87}$$

Where i_{tot} is the total number of classes. A similar expression for the 1-dimensional distribution for L can be found. Additionally, one can extend the KDE to account for 2-dimensional data. This way one can visualize the 2D particle size distribution as a function of R and L. The extension to 2D data is shown in the equation below.

$$f(R,L) = \sum_i \frac{N_i}{h_R h_L \sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{R-R_i}{h_R}\right)^2 - \frac{1}{2}\left(\frac{L-L_i}{h_L}\right)^2\right) \tag{3.88}$$

Similarly, h_R and h_L can be found using Scott's estimate.

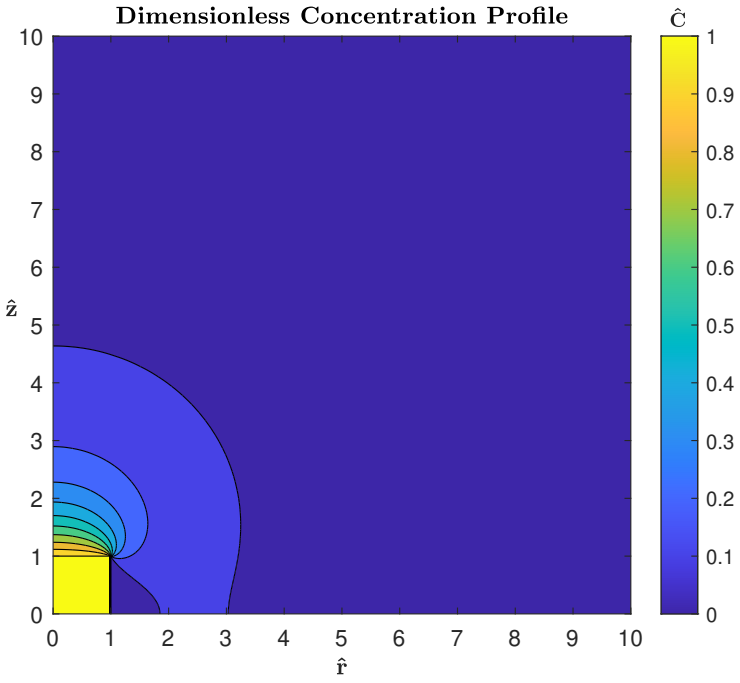
$$h_R \approx i_{tot}^{-0.2}\sigma_R, \qquad h_L \approx i_{tot}^{-0.2}\sigma_L \tag{3.89}$$

4 Results

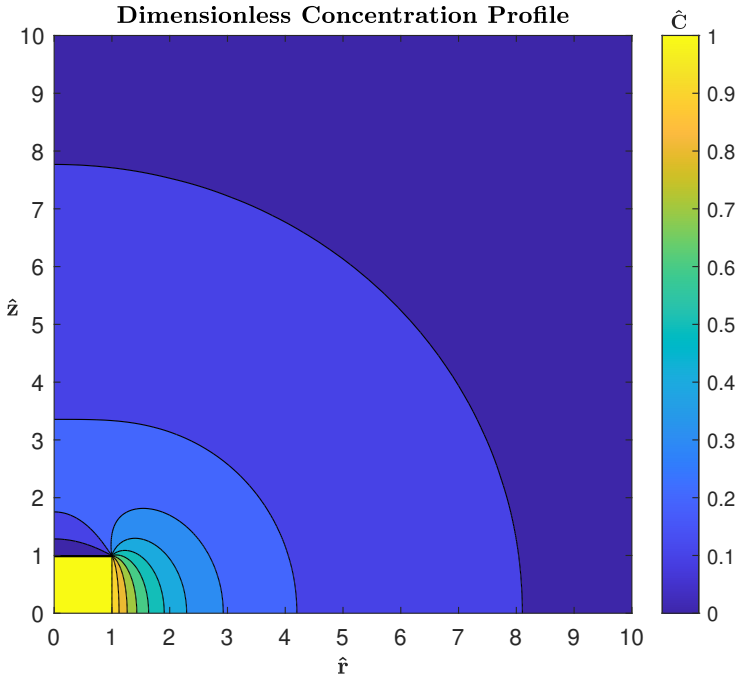
In this section, the concentration fields for $\hat{c}_{end}$ and $\hat{c}_{side}$ will be shown, in addition to the dimensionless flux found from solving the diffusion problem for a large variety of aspect ratios. The proposed solution to the Gibbs-Thomson effect for cylindrical particles will also be investigated. At last, the proposed KWN-model will be validated by comparing it to the literature and used in several simulations relevant for needle-like precipitates.

4.1 Concentration Fields

A key assumption of the proposed model is that the steady-state diffusion equation can be superposed and that one can solve for a $\hat{c}_{end}$- and $\hat{c}_{side}$-field independently. Therefore, it is of importance to confirm this assumption. In Figure 4.1, the dimensionless concentration fields for both $\hat{c}_{end}$ and $\hat{c}_{side}$ are shown. Additionally, the superposed concentration field is shown in Figure 4.2. As observed from Figure 4.2, the superposed concentration field is identical to the one where $c = c^i$ at both surfaces as seen in Figure (2.3) which was found from a project work [10].



(a)



(b)

Figure 4.1: **(a)** The concentration field for $\hat{c}_{end}$ with $\alpha = 1$. **(b)** The concentration field for $\hat{c}_{side}$ with $\alpha = 1$.

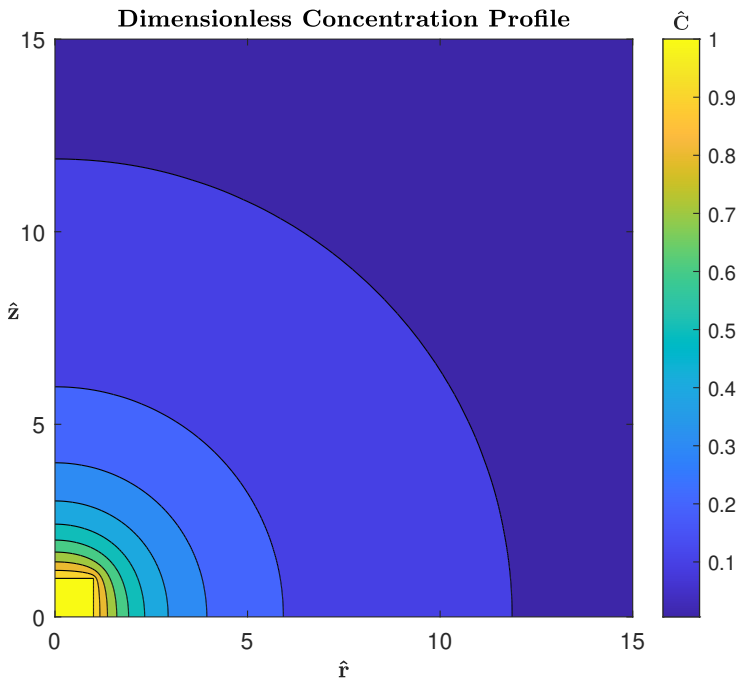


Figure 4.2: Plot of the dimensionless concentration field when combining $\hat{c}_{end}$- and $\hat{c}_{side}$-field.

4.2 Diffusion Solution For Needle-like Particles

To acquire the growth rates of the side and end surfaces of the cylindrical particle, the diffusion solution around particle has to be found. Once found, the dimensionless flux into the respective surfaces of the particle is also attainable which can be utilized in the growth equations (3.28) and (3.29). Since this flux depends on the aspect ratio of the considered particle, the diffusion problem was solved for a wide variety of aspect ratios. The computations were carried out using resources provided by the NTNU IDUN/EPIC computing cluster [33]. In Figures 4.3 and 4.4 one can see the resulting flux for $\hat{I}^{side}_{side}(\alpha)$ and $\hat{I}^{end}_{end}(\alpha)$, respectively. In addition, a line fitting performed in Matlab yielded equation (4.1) and (4.2) for $\hat{I}^{side}_{side}(\alpha)$ and $\hat{I}^{end}_{end}(\alpha)$, respectively.

$$\hat{I}_{side}^{side}(\alpha) = -\Big[1.6259 \times 10^{-5}\alpha^5 - 0.004801\,\alpha^4 + 2.5067\,\alpha^3 + 83.9911\,\alpha + 379.6569\,\alpha + 170.6609\Big] \Big/ \Big[\alpha^2 + 14.7673\,\alpha + 14.6144\Big] \tag{4.1}$$

$$\hat{I}_{end}^{end}(\alpha) = -\frac{4.2618 \times 10^{-5}\alpha^3 + 15.9496\,\alpha^2 + 266.1683\,\alpha + 17.0092}{\alpha^2 + 16.6917\,\alpha + 1.3963} \tag{4.2}$$

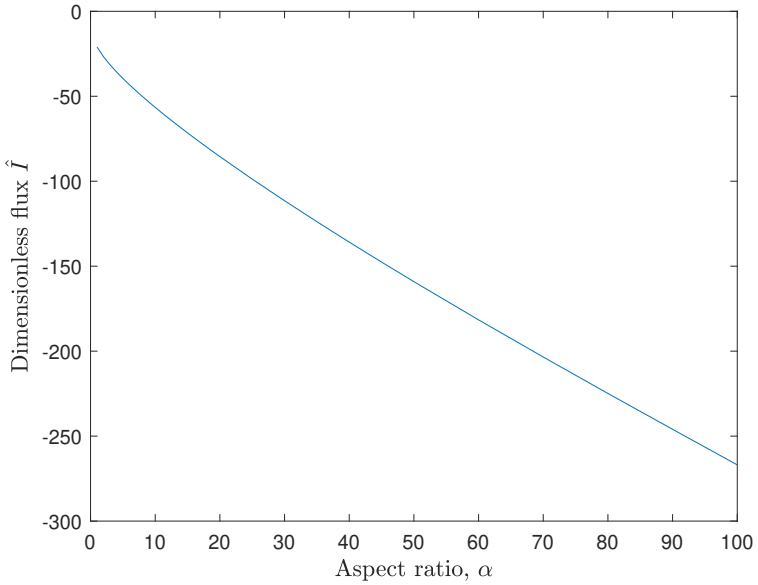


Figure 4.3: Plot of dimensionless flux at the side with a $\hat{c}_{side}$-field.

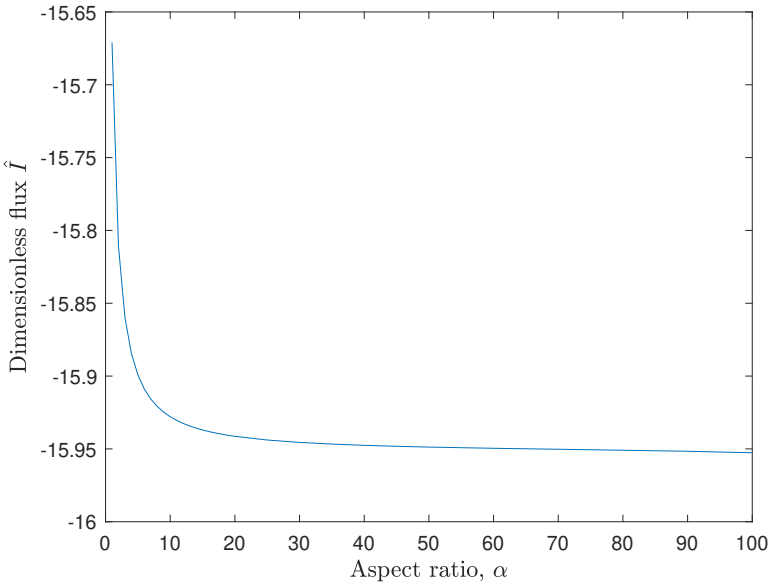


Figure 4.4: Plot of dimensionless flux at the ends with a $\hat{c}_{end}$-field.

Further, the results for $\hat{I}^{side}_{end}(\alpha)$ and $\hat{I}^{end}_{side}(\alpha)$ is shown in Figure 4.5. Consequently, a line fitting produced the following equations for $\hat{I}^{side}_{end}(\alpha)$ and $\hat{I}^{end}_{side}(\alpha)$.

$$\begin{aligned}\hat{I}^{side}_{end}(\alpha) = -\Big[&3.0154\times10^{-5}\alpha^4 - 0.005534\,\alpha^3 - 12.8684\,\alpha^2 \\ &- 81.3180\,\alpha - 20.7693\Big] \Big/ \Big[\alpha^2 + 6.9381\,\alpha + 2.5473\Big]\end{aligned} \tag{4.3}$$

$$\begin{aligned}\hat{I}^{end}_{side}(\alpha) = -\Big[&0.07117\,\alpha^4 - 27.1158\,\alpha^3 - 3.7298\times10^4\alpha^2 \\ &- 3.0131\times10^5\alpha - 1.0421\times10^5\Big] \Big/ \\ \Big[&\alpha^3 + 2.9140\times10^3\alpha^2 + 2.5590\times10^4\alpha + 1.2121\times10^4\Big]\end{aligned} \tag{4.4}$$

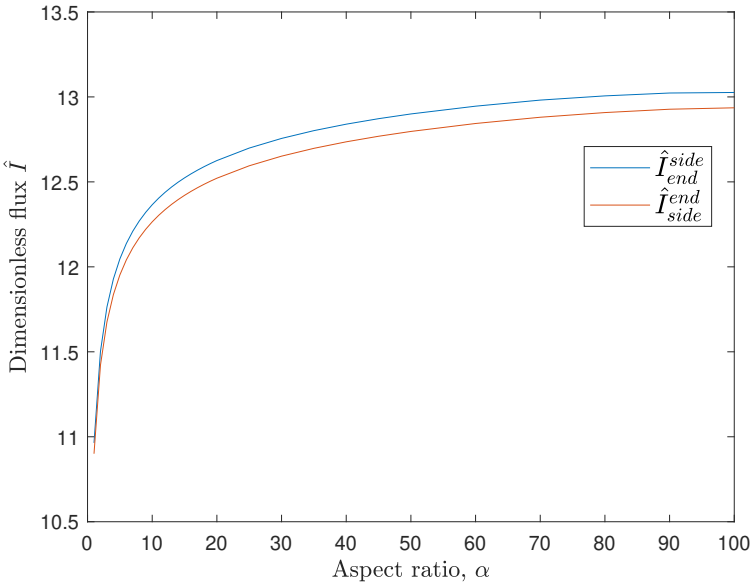


Figure 4.5: Plot of dimensionless flux at the ends with a $\hat{c}_{side}$-field, and at the sides with a $\hat{c}_{end}$-field.

4.3 Diffusion Solution For Disk-like Particles

The diffusion solution were also computed for disk-like particles were the aspect ratio is below 1. The dimensionless flux for $\hat{I}^{side}_{side}(\alpha)$ and $\hat{I}^{end}_{end}(\alpha)$ are shown in Figure 4.6. To easily compare the flux values for needle- and disk-like particles, the inverse aspect ratio is plotted on the x-axis. The resulting line fitting yielded equation (4.5) and (4.6) for $\hat{I}^{side}_{side}(\alpha)$ and $\hat{I}^{end}_{end}(\alpha)$, respectively.

$$\hat{I}^{side}_{side}\left(\frac{1}{\alpha}\right) = -\frac{-1.0498 \times 10^{-4}\alpha^2 + 16.5676\,\alpha + 6.1457}{\alpha + 0.0832} \tag{4.5}$$

$$\begin{aligned}\hat{I}^{end}_{end}\left(\frac{1}{\alpha}\right) = -\Big[&9.4596 \times 10^{-4}\alpha^5 + 23.7154\,\alpha^4 + 580.3917\,\alpha^3 \\ &-2.6179 \times 10^3\alpha^2 + 1.0822 \times 10^3\alpha + 4.9366 \times 10^3\Big] \Big/ \\ &\Big[\alpha^4 + 28.5534\,\alpha^3 - 103.3915\,\alpha^2 - 64.8390\,\alpha + 394.2475\Big]\end{aligned} \tag{4.6}$$

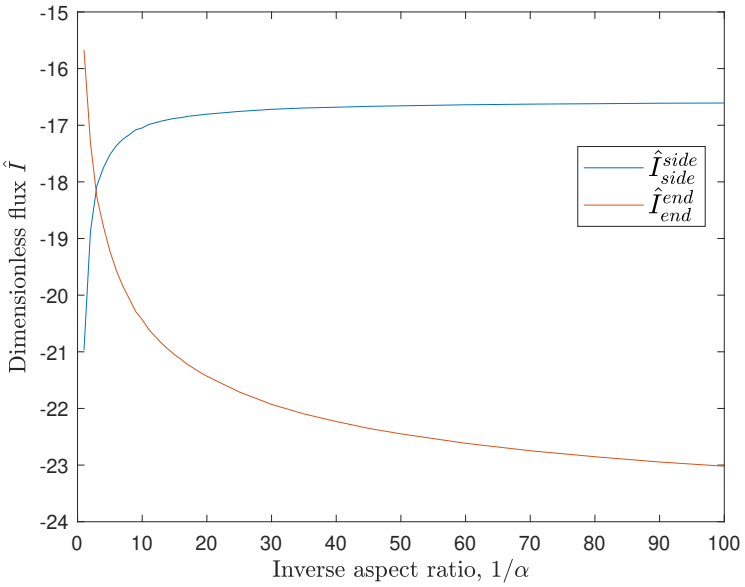


Figure 4.6: Plot of dimensionless flux at the ends with a $\hat{c}_{end}$-field, and at the side with a $\hat{c}_{side}$-field.

Similarly, the results for $\hat{I}^{side}_{end}(\alpha)$ and $\hat{I}^{end}_{side}(\alpha)$ for disk-like particles is shown in Figure 4.7. The line fitted equation for $\hat{I}^{side}_{end}(\alpha)$ and $\hat{I}^{end}_{side}(\alpha)$ are depicted below.

$$\begin{aligned}\hat{I}^{side}_{end}\left(\frac{1}{\alpha}\right) = -\Big[&-16.7103\,\alpha^4 - 5.6667\times10^3\alpha^3 - 4.4500\times10^4\alpha^2 \\ &+ 1.5558\times10^5\alpha + 8.0314\times10^4\Big] \Big/ \Big[\alpha^4 + 360.2312\alpha^3 \\ &+ 3.5066\times10^3\alpha^2 - 9.9724\times10^3\alpha - 1.0834\times10^4\Big]\end{aligned} \tag{4.7}$$

$$\begin{aligned}\hat{I}^{end}_{side}\left(\frac{1}{\alpha}\right) = -\Big[&-3.1704\times10^{-4}\alpha^5 - 16.1876\,\alpha^4 - 422.8250\,\alpha^3 \\ &+ 1.9161\times10^3\alpha^2 - 984.1365\alpha - 3.0602\times10^3\Big] \Big/ \\ &\Big[\alpha^4 + 29.6572\,\alpha^3 - 109.8029\,\alpha^2 - 45.5430\,\alpha + 360.2118\Big]\end{aligned} \tag{4.8}$$

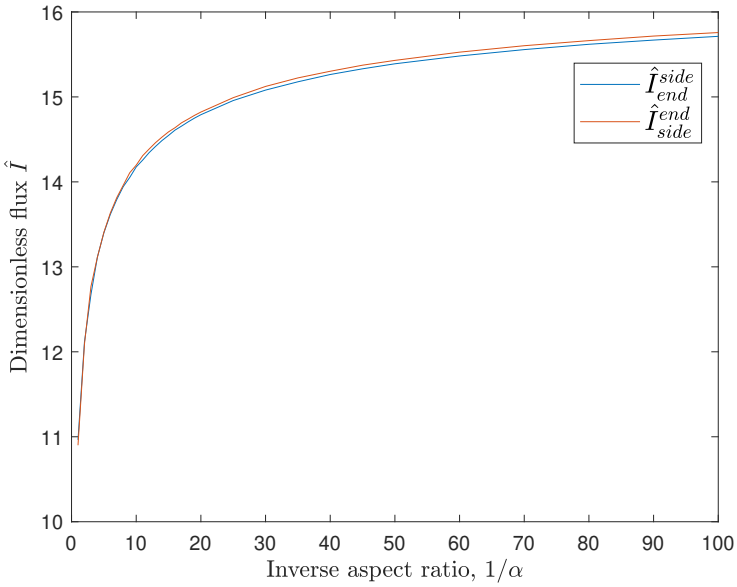


Figure 4.7: Plot of dimensionless flux at the ends with a $\hat{c}_{end}$-field, and at the side with a $\hat{c}_{side}$-field.

4.4 The Gibbs-Thomson Effect

In the work of implementing a precipitation model for cylindrical precipitates, a new expression for the Gibbs-Thomson effect for the end and side surfaces of the precipitate was proposed in equations (3.47) and (3.46), respectively. An interesting case to examine is the new equation for c^{side}. The input parameters to the Gibbs-Thomson equations and later for the particle size distributions are from Myhr *et al.* [8] and are shown in Table 4.1. The solution of equation (3.46) for low to high aspect ratios, i.e disk-like and needle-like precipitates, are shown in Figure 4.8.

Table 4.1: Input data at 180°C from Myhr *et al.* [8].

Parameter	Value
c^p	0.634
c^{eq}	3.54×10^{-5}
c^0	0.0063
$D\ [m^2 s]$	2.278×10^{-19}
$A_0\ [J/mol]$	16220
$j_0\ [\#/m^3 s]$	9.66×10^{34}
$Q_d\ [J/mol]$	130000
$\gamma_{side}\ [J/m^2]$	0.2
$\gamma_{end}\ [J/m^2]$	0.2
$V_p^\beta\ [m^3]$	6.559×10^{-29}
χ	1

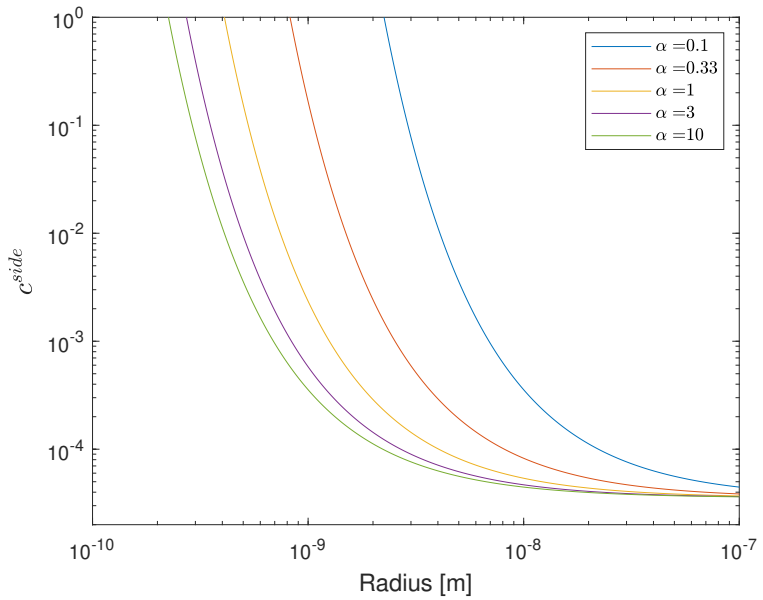


Figure 4.8: Plot of c^{side} with varying α.

The Gibbs-Thomson equation with varying end surface energy was also proposed.

As equation (3.47) does not depend on γ_{end}, only the equation for the interfacial concentration at the side surface were altered and is shown in equation (3.51). For equation (3.51), the Gibbs-Thomson effect is also dependent on the function for γ_{end}. The variation in c^{side} with γ_{end}-functions with increasing $\gamma_{end}/\gamma_{side}$ ratio is shown in Figures 4.9 with $\alpha = 1$ and 4.10 with $\alpha = 1.2$. The respective γ_{end}-functions are shown in Figure 4.11 and is based on equation (3.53).

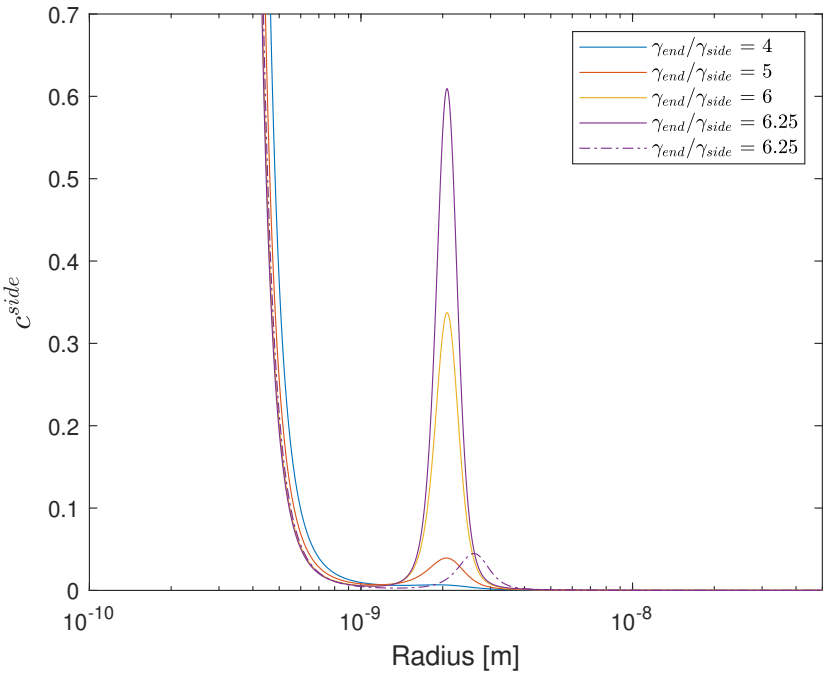


Figure 4.9: Plot of c^{side} with increasing $\gamma_{end}/\gamma_{side}$ ratio and $\alpha = 1$.

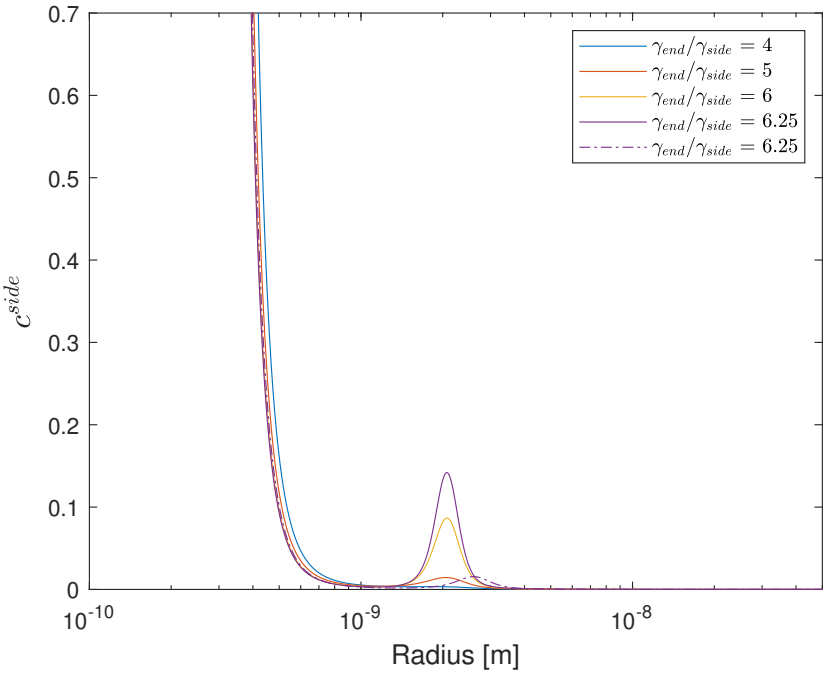


Figure 4.10: Plot of c^{side} with increasing $\gamma_{end}/\gamma_{side}$ ratio and $\alpha = 1.2$.

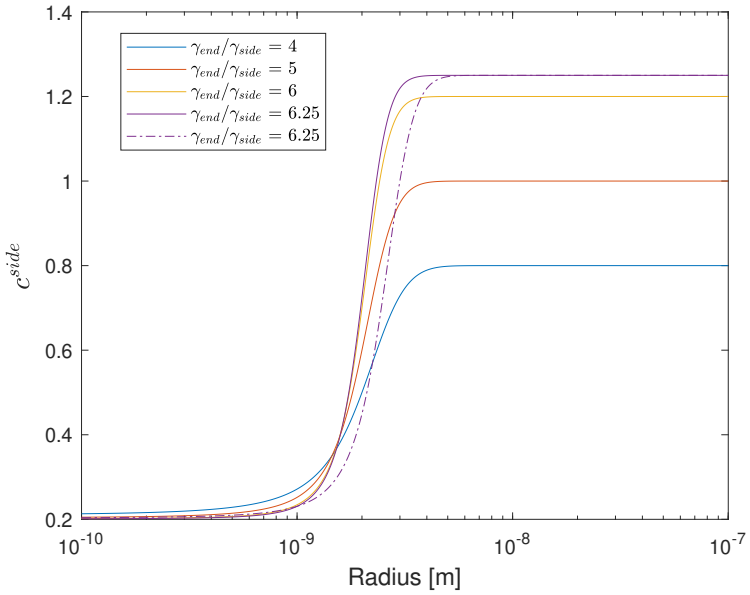


Figure 4.11: Plot of γ_{end} as a function of R with increasing $\gamma_{end}/\gamma_{side}$ ratio.

4.5 Particle Distributions by the KWN-Model

4.5.1 Validation of Model

To validate that the proposed KWN-model yields satisfactory results, a comparison to the KWN-model utilized in Myhr *et al.* [8] for spherical particles in an Al-Mg-Si alloy is done. Here, the quasi-binary Al-Mg_2Si section of the phase diagram is used. Letting $\gamma_{side} = \gamma_{end}$ results in a critical aspect ratio of 1 for the nucleated particles as seen in equation (3.59), which yields comparable results to a spherical particle where the surface energy is constant around its interface. The input parameters to the KWN-model are summarized in Table 4.1. The nucleation rate and total particle number density are shown in Figure 4.12, mean and critical radius are shown in Figure 4.13, and the particle size distribution is shown in Figure 4.14. All three figures show a very close resemblance to the results presented in Myhr *et al.* [8] for the isothermal case of spherical particles at 180°C.

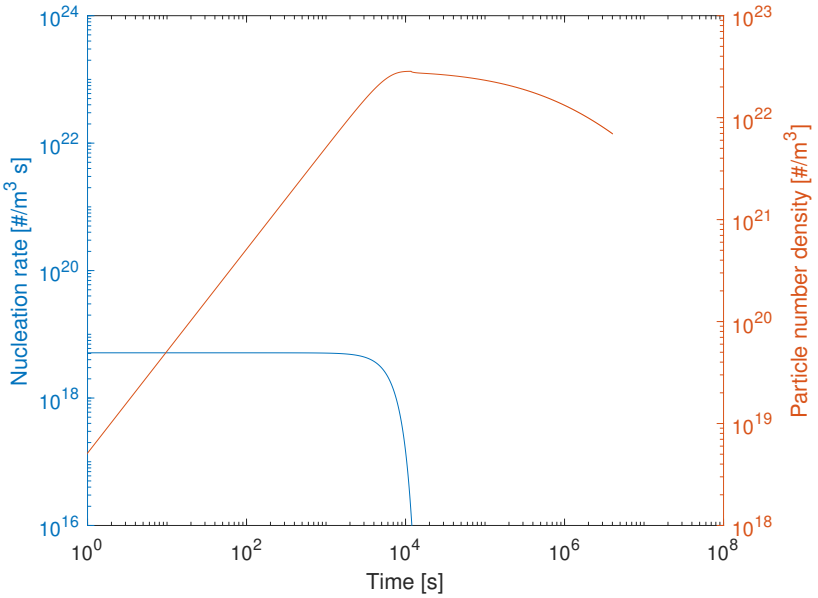


Figure 4.12: Change of nucleation rate dN/dt and total particle number density N with time.

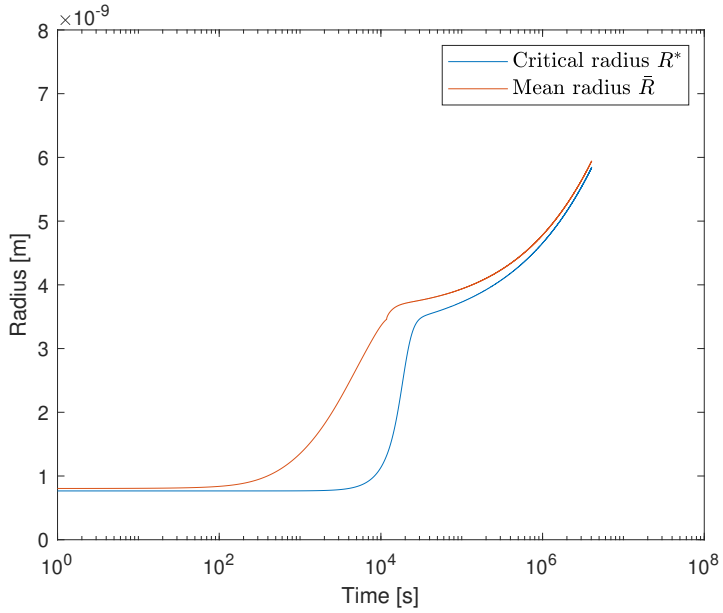


Figure 4.13: Change of critical radius R^* and mean radius $\bar{R}$ with time.

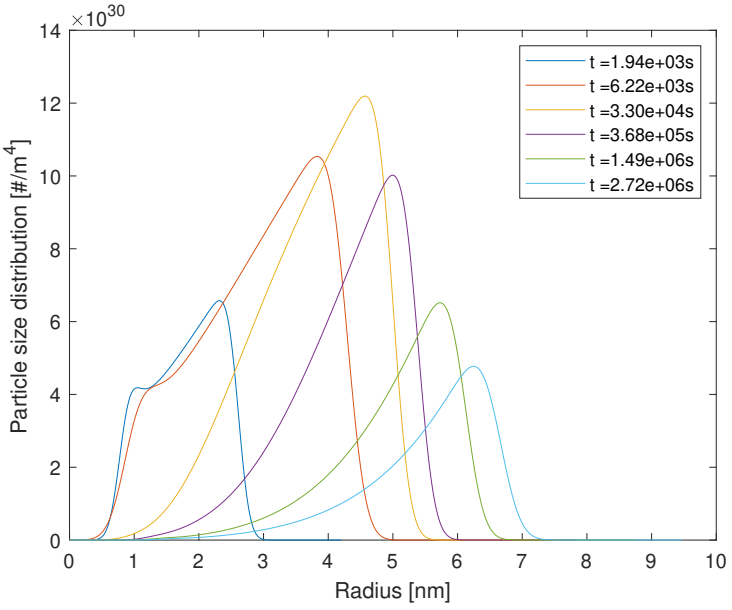


Figure 4.14: Particle size distribution at various times.

4.5.2 Dependence of Number of Classes

In the Lagrangian approach, the number of classes in the particle size distribution is dependent on how large the time step dt of the system is. The accuracy of the resulting particle size distribution is therefore linked to the time step. The simulations for the particle size distribution in Figure 4.14, contained close to 1600 size classes before the onset of coarsening. When operating with a large number of classes, the calculation times increase drastically. Therefore, it is of importance to check the influence of implementing a larger time step dt, which effectively reduces the total number of classes in the system. In Figure 4.15, one can see a comparison between the results for a system containing 1600 classes at most, to a system containing around 500 classes at most. The simulation with a higher time step is indicated with stapled lines. Although the simulations with a larger time step had considerably fewer size classes, the resulting particle size distribution shows a close resemblance to the case with a higher amount of size classes.

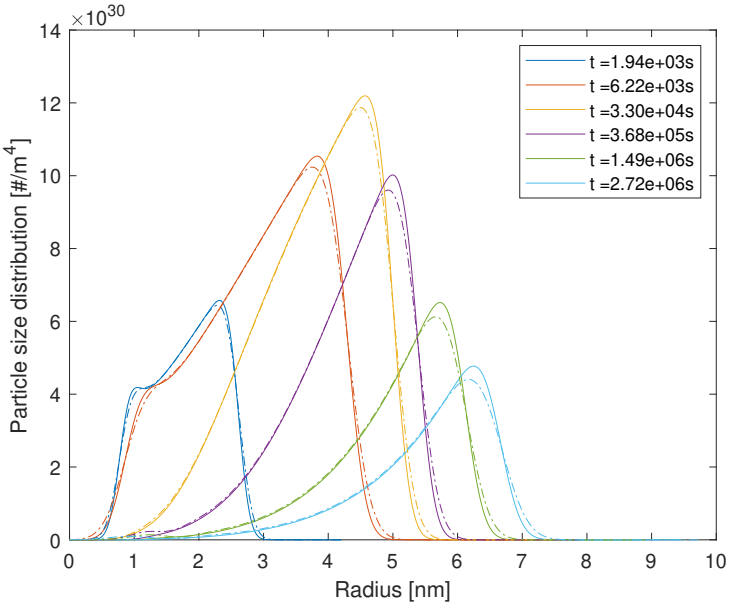


Figure 4.15: Particle size distribution at various times. The dotted lines represents the distribution using a larger dt.

4.5.3 Constant Surface Energy

Needle-Like Precipitates

The surface energy of the side or end surfaces of the particle can be changed to stimulate needle- or disk-like particles. By increasing γ_{end}, one increases the critical aspect ratio as seen by equation (3.59) which effectively results in needle-like precipitates. The following simulations was run with the same parameters as Myhr *et al.* [8], which is shown in Table 4.1, except for the surface energy of the ends, where $\gamma_{end} = 10\,\gamma_{side}$. The results for R^* and $\bar{R}$ is shown in Figure 4.16 which shows a much lower and slower growth compared to the case where $\gamma_{end} = \gamma_{side}$. Further, L^* and $\bar{L}$ is shown in Figure 4.17. An interesting note from Figure 4.17 is that L^* that is given from R^* and α^*, is surpassing the mean length of the precipitates, yet the precipitates continue to lengthen.

It is also beneficial to review the aspect ratio of the precipitates. A parameter β is defined as $\beta = \bar{L}/2\bar{R}$, and the evolution of β through time is shown in Figure 4.18. An interesting remark is that the aspect ratio of the precipitates decreases during the nucleation and growth stage as seen in Figure 4.18. As the critical aspect ratio is 10, the nucleated particles start with an aspect ratio of 10 but readily decrease to half that when nucleation stops. At the onset of coarsening, the aspect ratio of the precipitates increases again up to nearly the initial value.

Finally, the 2D particle size distribution is plotted at various times in Figure 4.19. The particle size distributions seem to mostly appear in a straight line in the R-L plane.

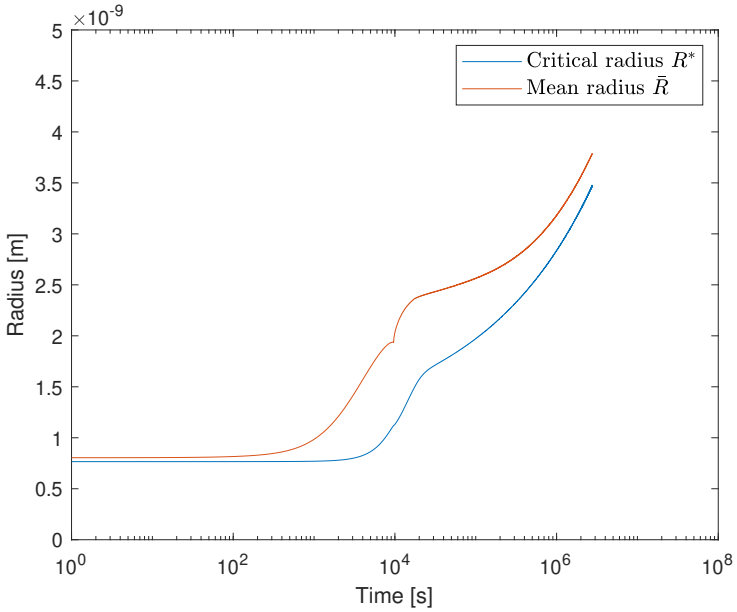


Figure 4.16: Change of critical radius R^* and mean radius $\bar{R}$ with time for $\gamma_{end} = 10\,\gamma_{side}$.

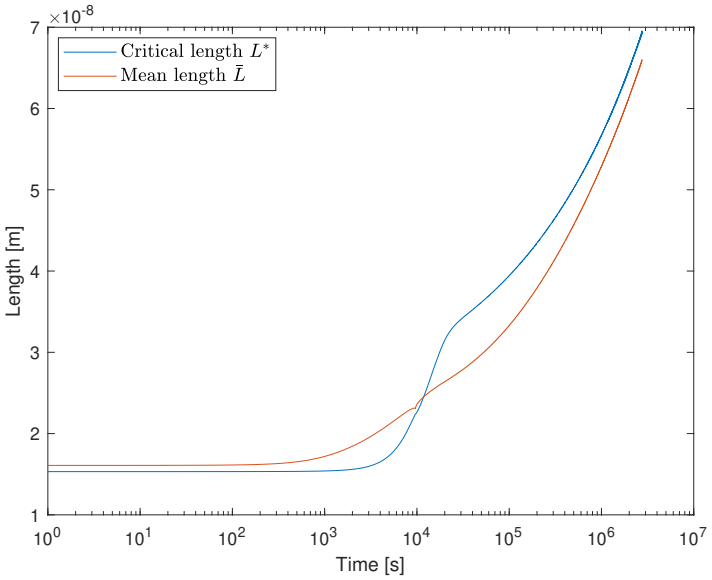


Figure 4.17: Change of critical length L^* and mean length $\bar{L}$ with time for $\gamma_{end} = 10\,\gamma_{side}$.

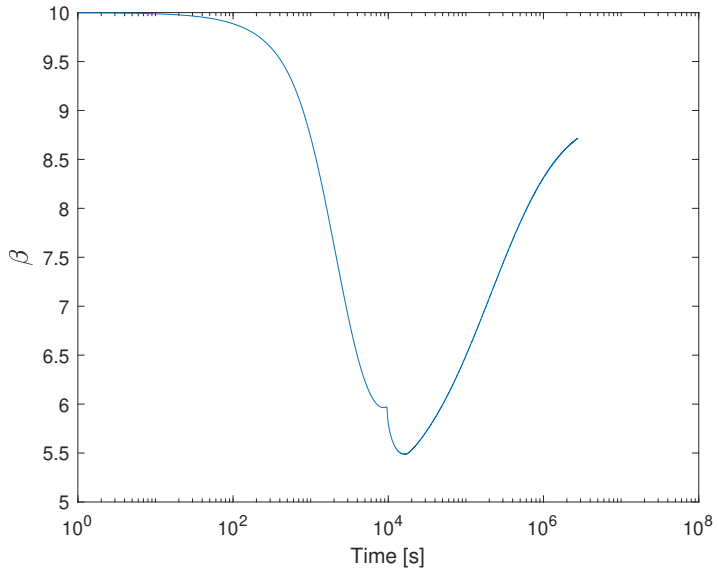


Figure 4.18: Change of $\beta = \bar{L}/2\bar{R}$ with time for $\gamma_{end} = 10\,\gamma_{side}$.

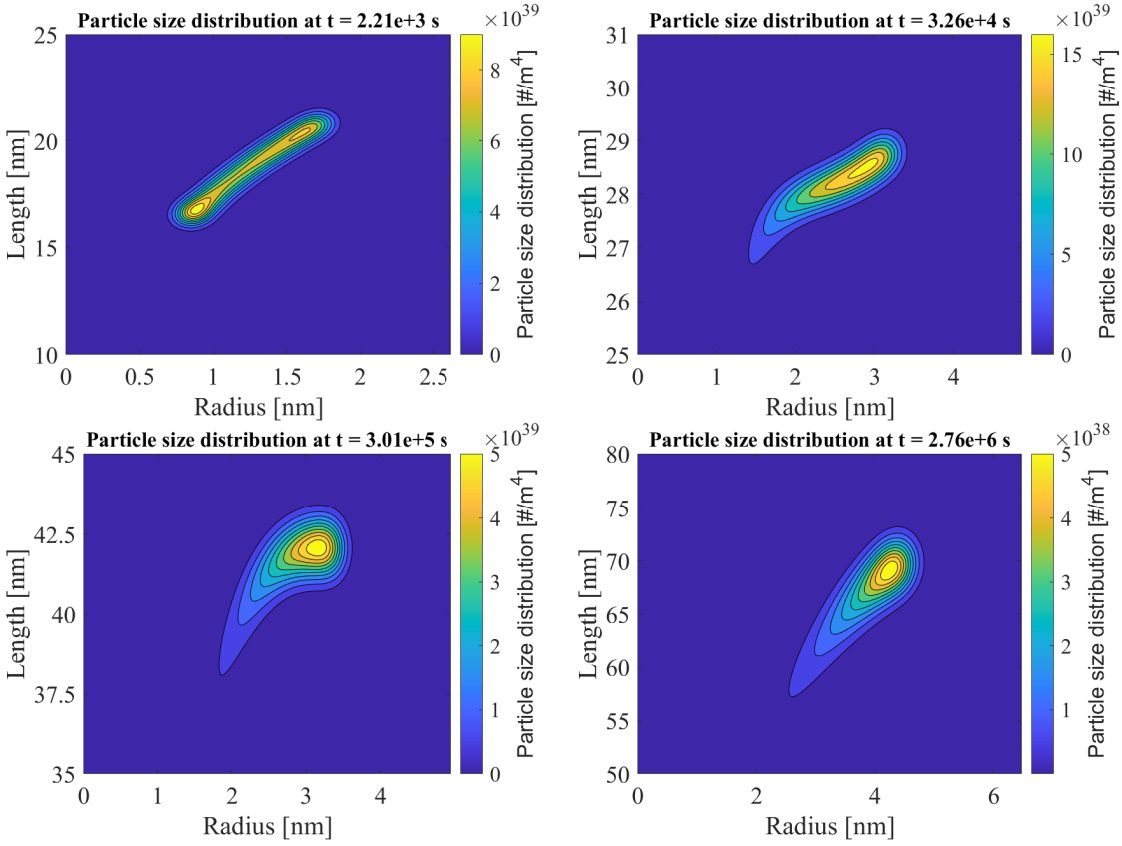


Figure 4.19: The particle size distribution at various times when $\gamma_{end} = 10\,\gamma_{side}$.

Disk-Like Precipitates

In addition to needles, one can facilitate the creation of disk-like particles by adjusting γ_{side} relative to γ_{end}. The simulation for disk-like particles was run by letting $\gamma_{side} = 10\,\gamma_{end}$, and the other parameters as presented in Table 4.1. The mean and critical radii of the disk-like particles are shown in Figure 4.20. Unsurprisingly, both R^* and $\bar{R}$ has a much more profound growth for this case. In contrast to the needle-case, the critical radius does not surpass the mean radius, but rather follows it closely. Further, L^* and $\bar{L}$ is shown in Figure 4.21.

The disk-like particles also show a similar trend as the needle-like with regards to β. From Figure 4.22 one can see that the aspect ratio increases during nucleation and growth, before decreasing again at the onset of coarsening to approximately the critical aspect ratio of 0.1.

At last, the 2D particle size distributions are shown at various times in Figure 4.23. As for the needles, the disk-like particles tend to arrange in a nearly straight line in the R/L plane.

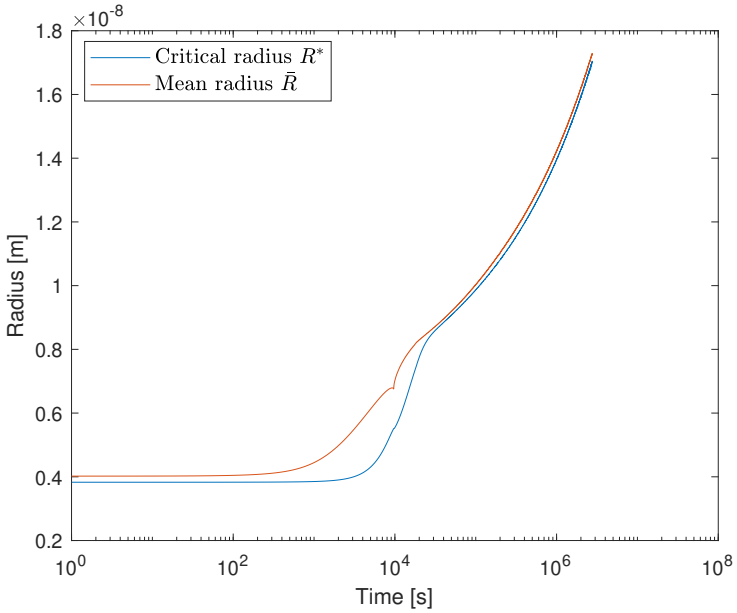


Figure 4.20: Change of critical radius R^* and mean radius $\bar{R}$ with time for $\gamma_{side} = 10\,\gamma_{end}$.

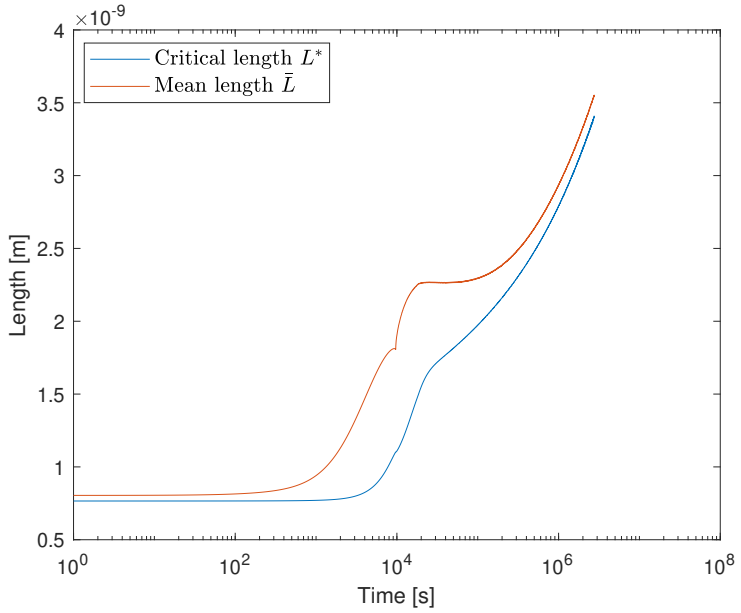


Figure 4.21: Change of critical length L^* and mean length $\bar{L}$ with time for $\gamma_{side} = 10\,\gamma_{end}$.

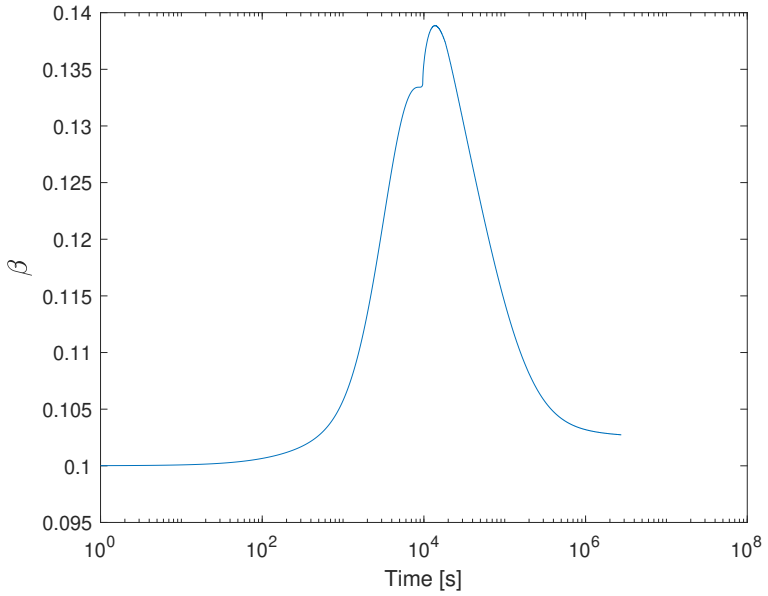


Figure 4.22: Change of $\beta = \bar{L}/2\bar{R}$ with time for $\gamma_{side} = 10\,\gamma_{end}$.

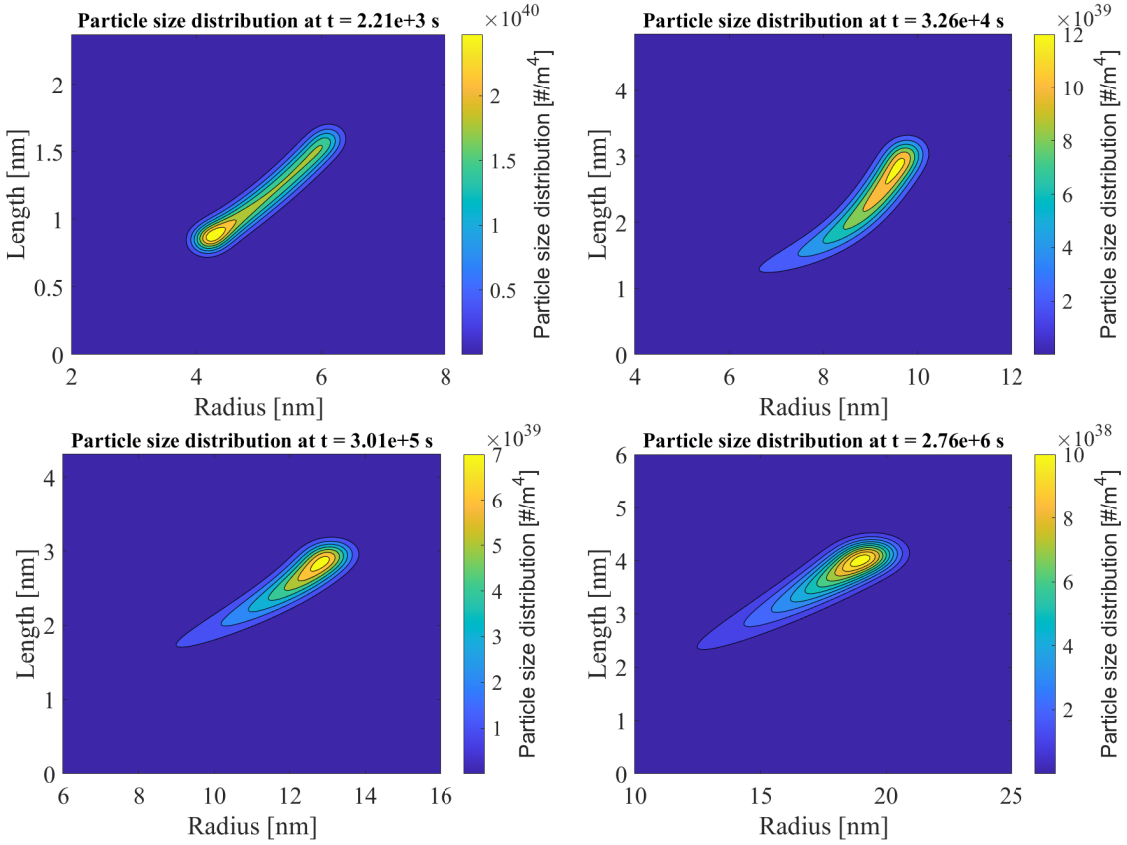


Figure 4.23: The particle size distribution at various times when $\gamma_{end} = 0.1\,\gamma_{side}$.

4.5.4 Variation of Surface Energy

While introducing a constant γ_{end} gives needle-like particles, it is not realistic to nucleate very long needles as in the case when $\alpha^* = 10$. Also, the change in the aspect ratio of the needle-like particles with constant γ_{end} does not seem realistic. By introducing an equation $\gamma_{end}(R)$ as described in section 3.2.2, one can facilitate both the nucleation of near-spherical nuclei and an increase in aspect ratio as the needles grow. The surface energy function for γ_{end} used in the following simulations are on the form presented in equation (4.9). The proposed parameters for the equation are shown in Table 4.2. Additionally, the other material parameters are shown in Table 4.1.

Table 4.2: Input data to γ_{end}-function.

Parameter	Value	Comment
γ_{side}	0.2	From Ref. [8]
γ_{max}	$15 \times \gamma_{side}$	Threshold value for γ_{end}
R_{st}	1×10^{-9} m	Start of rampup
R_{sp}	4×10^{-9} m	End of rampup

$$\gamma_{end}(R) = \gamma_{side} + \frac{1}{2}\left(\gamma_{max} - \gamma_{side}\right)\left(1 + \tanh\left(\frac{\left(\gamma_{max} - \gamma_{side}\right)\left(R - \frac{1}{2}R_{st} - \frac{1}{2}R_{sp}\right)}{0.3\left(R_{sp} - R_{st}\right)}\right)\right) \tag{4.9}$$

With an up ramping function, the simulations show a very different story than for the constant surface energies presented in the previous section. In Figure 4.24, one can see the change of R^* and $\bar{R}$ over time. Here, the radius of the precipitates does not grow to the same extent as previously and the rapid growth of R occurs sooner in the time evolution. The mean radius also shrinks slightly before following the critical radius of the system at the onset of coarsening.

A promising feature by introducing the γ_{end} -function can be seen in Figure 4.25. Here, both the mean length and mean aspect ratio is shown over time. With increasing surface energy of the end surface, the precipitates start to lengthen to become needles. As evident from Figure 4.25, the increase in β is mainly due to the increase in length, as the radius of the precipitates changes rather slowly in the coarsening regime.

At last, the 2D particle size distributions are plotted at various times in Figure 4.26. The particle size distributions show a nearly straight line in the R/L plane as seen for the particle size distributions for constant surface energies. However, the up leftmost particle size distribution at an earlier stage in the nucleation and growth process shows a different shape. It appears to divide into two lines in the plane. One from roughly 1-1.5 nm radius, and the other from 1.5 up to 2 nm.

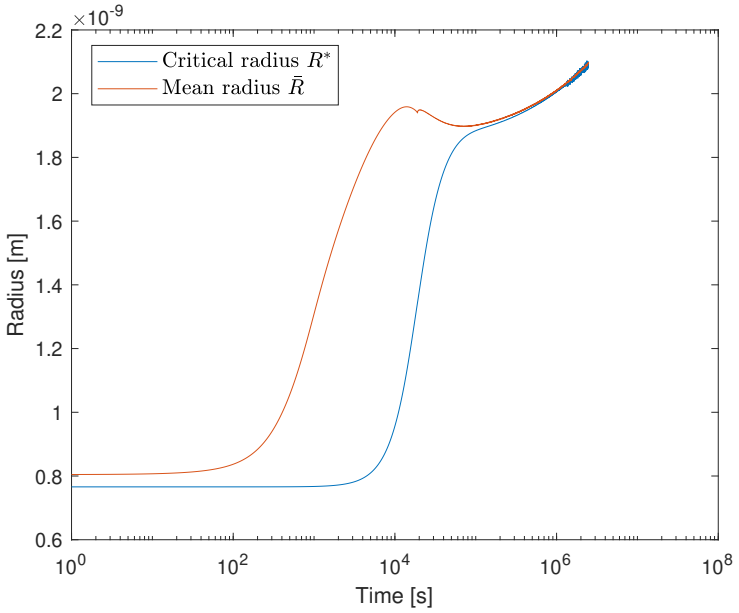


Figure 4.24: Change of critical radius R^* and mean radius $\bar{R}$ with time with a non-constant γ_{end}.

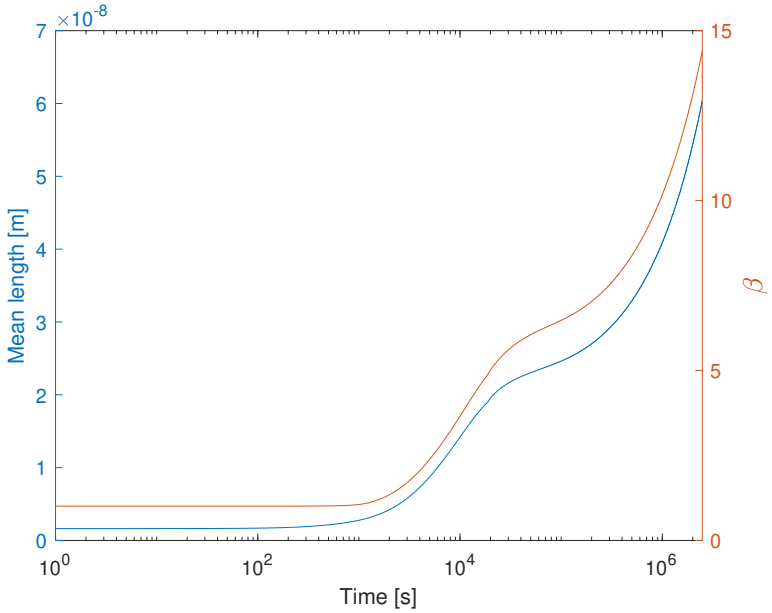


Figure 4.25: Change of mean length $\bar{L}$ and β with time with a non-constant γ_{end}.

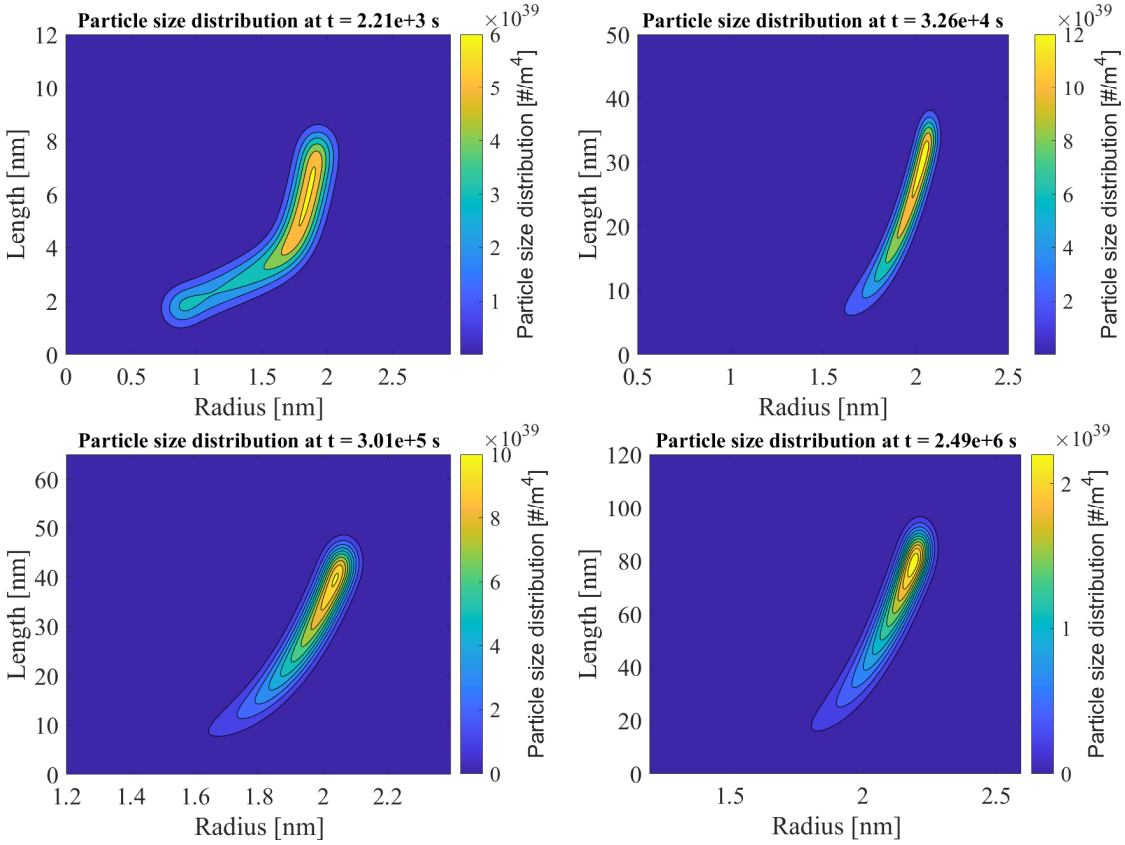


Figure 4.26: The particle size distribution at various times with γ_{end}-function.

Influence of Ramp up Function

The ramp-up function for γ_{end} can have a large influence on the particle size distribution. For instance, as shown in Section 4.4, the Gibbs-Thomson equation is sensitive to the slope of the ramp-up function. Therefore, it is of interest to examine the influence of the γ_{end}-function on the particle distribution. In Figure 4.27, β is plotted for various ramp up functions. The parameters used are the same as in Tables 4.1 and 4.2, except for R_{sp} that signifies the radius where the γ_{end}-function starts to flatten and become constant. Therefore, the three cases in Figure 4.27 shows the influence of the slope of the ramp-up function on β. As observed, a higher slope yields larger aspect ratios of the precipitates.

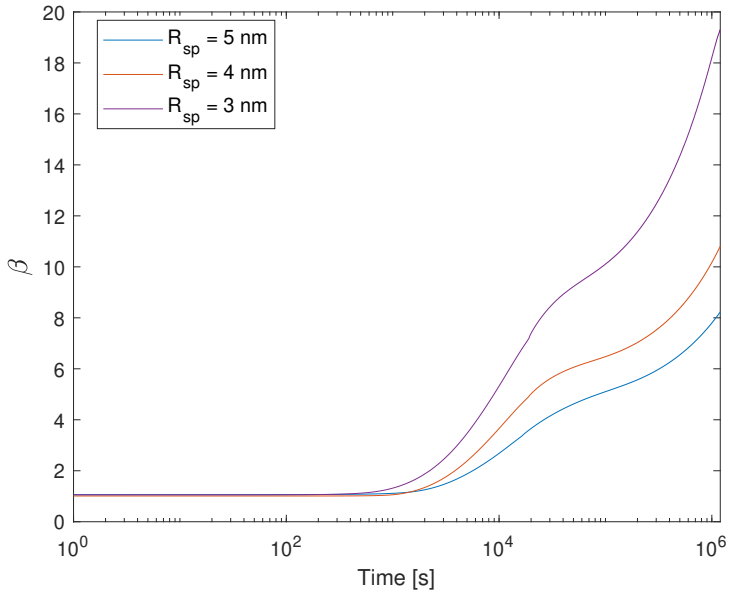


Figure 4.27: Influence of the steepness of the ramp up function for γ_{end} on the aspect ratio of the precipitates.

Influence of Nucleation Rate

Another important aspect of the resulting particle distributions is the nucleation rate of the system. In this section, the influence of the nucleation rate is examined. The parameters used in the following simulations is shown in Table 4.1, except for the numerical constant j_0 that is used in equation (2.34). By varying this constant, one can achieve different nucleation rates and ultimately change the number of particles in the system. In Figure 4.28 one can see the influence of a higher nucleation rate on the mean radius of the precipitates. As seen, the growth in the radial direction is suppressed when the system contains a higher amount of particles. Further, β is plotted for the same three cases in Figure 4.29. Here, the aspect ratio is mostly affected in the early coarsening regime and ultimately leads to the same aspect ratio at longer times.

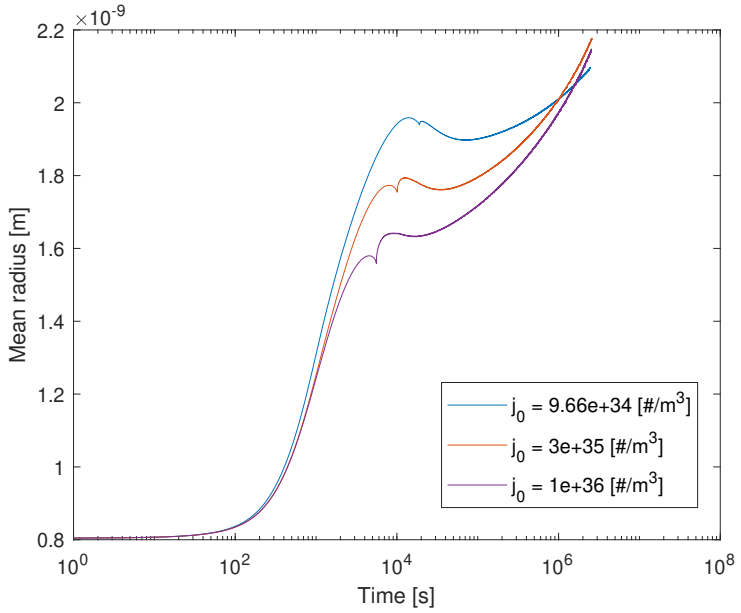


Figure 4.28: Change of mean radius $\bar{R}$ for different nucleation rates with a non-constant γ_{end}.

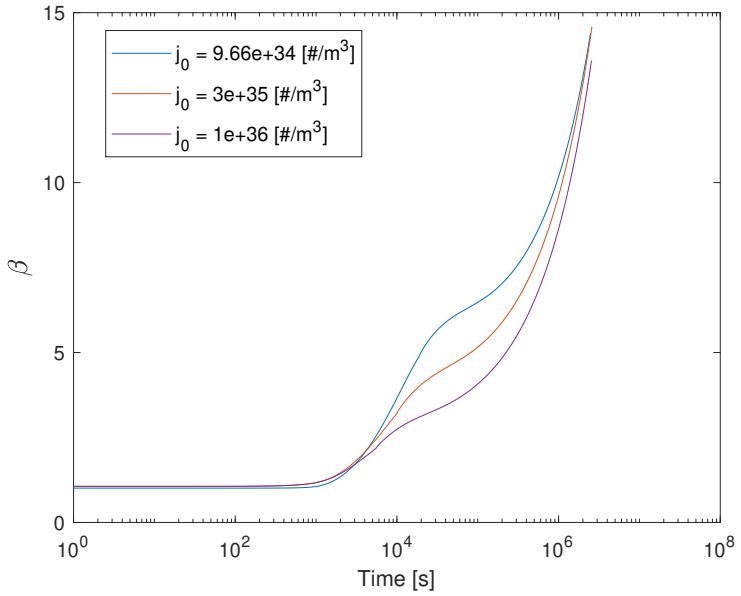


Figure 4.29: Change of β for different nucleation rates with a non-constant γ_{end}.

Comparison to Experimental Data

To see if the proposed KWN-model yields realistic results, the model will be compared to experimentally found data. In Du and Holmedal *et al.* [30], the mean cross-section area and mean length of precipitates in an Al-0.52wt%Mg-0.75wt%Si was found as shown in Figure 4.30. The plot displays a proportional relationship between the mean cross-section area and the mean length of the precipitates. In Figure 4.31, the resulting mean cross-section area and mean length is plotted from the proposed model. The input parameters used are the same as in Tables 4.1 and 4.2, except for R_{sp} which is reduced to 3 nm. As seen from Figure 4.31, when $\bar{L}$ is around 30 nm, the model also yields a proportional relationship. However, before $\bar{L} = 30$ nm, the model does not show the same relationship. In addition, the experimentally found cross-section area is smaller for shorter needles in Figure 4.30, and the linear relation has a higher slope than our model. However, the reported cross-section area in Du and Holmedal *et al.* [30] might be too low for short needles as Sunde *et al.* [34] report much higher cross-section area at $10 \pm 1\, nm^2$ for a mean length of $13 \pm 1\, nm$.

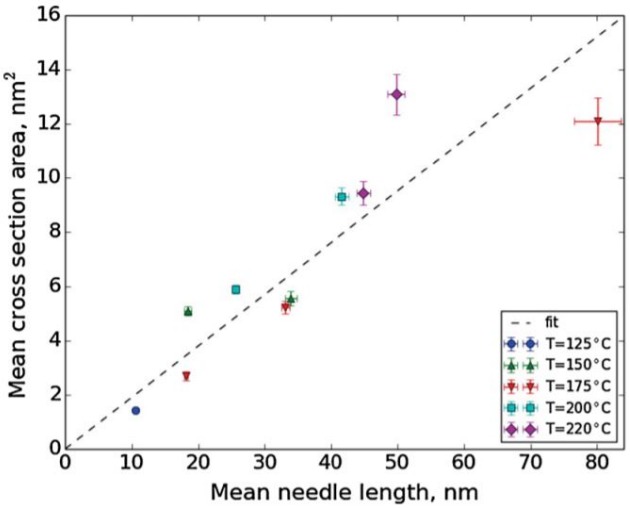


Figure 4.30: Experimental values of mean cross section area and mean length of β'' precipitates [30].

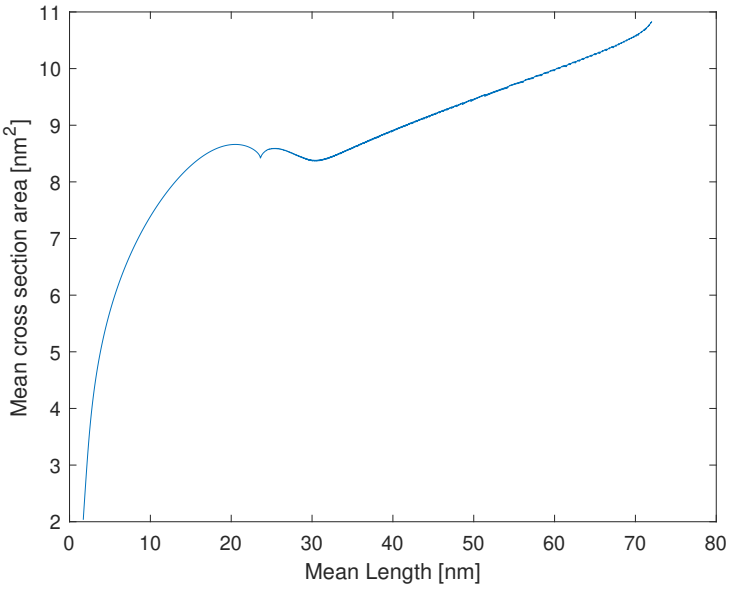


Figure 4.31: Plot of mean cross section area and mean length of the precipitates.

5 Discussion

5.1 Diffusion Solutions

The solutions for the dimensionless fluxes in section 4.2, are crucial in determining the growth rate of the respective surfaces of a cylindrical particle. As observed from Figures 4.3-4.5, by dividing the linear Laplace equation into two cases of $\hat{c}_{end}$ and $\hat{c}_{side}$, the particle will experience two currents with opposite signs. For instance, with a $\hat{c}_{side}$-field, the side surface will experience a negative flux, while the end surfaces experience a positive flux. Due to the definition of $\hat{c}_{side}$ and the dimensionless fluxes in section 3.1, the negative flux signifies a net current into the particle. Therefore, with a $\hat{c}_{side}$-field, there is a net current of atoms into the side surface, and a net current of atoms out of the particle at the end surfaces. The same trend is observed for a $\hat{c}_{end}$-field, where there is a current of atoms into the end surfaces and a current of atoms out of the side surface. This is however intuitive as the surface either has a concentration higher or lower than the surrounding concentration field.

Another remark on the diffusion solution is that $\hat{I}_{side}^{side}$ for a needle-like particle shows a different trend than the other dimensionless fluxes for needles. While the other fluxes seemingly reaches an asymptotic solution at higher aspect ratios, $\hat{I}_{side}^{side}$ does not seem to converge to an asymptotic solution. However, this is due to the scaling of the system. The model introduced in section 3.4 scales with the aspect ratio in the axial direction. This can be seen in equation (3.14) and (3.15), where $\hat{c}_{side} = 0$ at $\hat{r} = 1$, and $\hat{c}_{end} = 1$ at $\hat{z} = \alpha$. Hence, the needle becomes longer and longer with higher aspect ratios. Instead, the dimensionless flux divided by α reaches a constant value when $\alpha \to \infty$. An illustration of this is shown in Figure

5.1, where $\hat{I}^{side}_{side}/\alpha$ is plotted as a function of α.

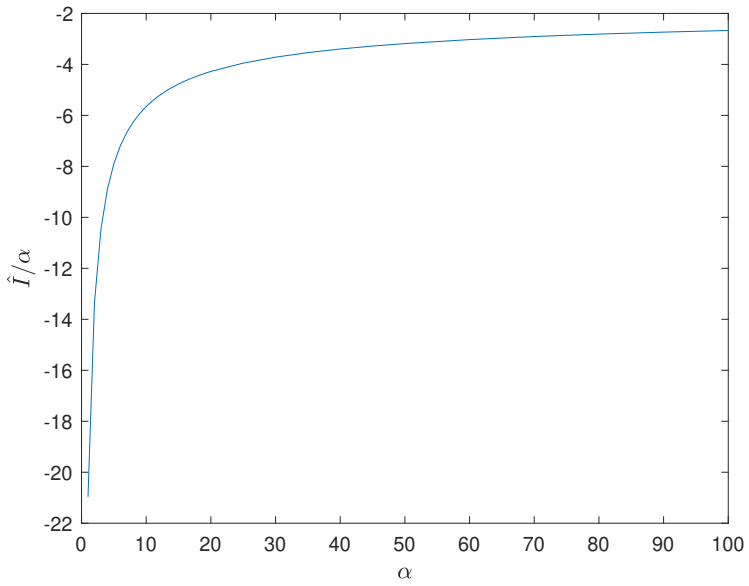


Figure 5.1: Plot of $\hat{I}^{side}_{side}/\alpha$ as a function of α.

For disks, however, the radius is constant while the length is decreased until it diminishes. The contribution of flux from its sides is negligible as the aspect ratio reaches zero, while the flux from its ends is constant. Therefore, the dimensionless fluxes for a disk reach a constant value as $\alpha \to 0$.

5.2 The Gibbs-Thomson Effect

The proposed Gibbs-Thomson equation for c^{side} was investigated for both constant and non-constant γ_{end}. With a constant γ_{end}, the influence of aspect ratio on equation (3.46) was investigated. As seen in Figure 4.8, α has a large influence on the interfacial concentration at the side surface. Higher aspect ratios shifted the Gibbs-Thomson curves towards the left side of Figure 4.8. This signifies that the interfacial concentration at the side surface reaches the equilibrium concentration at a lower radius. This can readily be seen from equation (3.46) as the γ_{end} term scales with $1/L$ and a higher aspect ratio gives a smaller contribution from γ_{end}. Consequently, a smaller aspect ratio increases the contribution from γ_{end}.

An inherent problem with the Gibbs-Thomson equations, which is rarely addressed in the literature, is that the equations readily increase to concentration values which are not valid when R is too low. For instance, this is shown in Figure 4.8, where the interfacial concentration quickly starts to increase when $R < 1$ nm for an aspect ratio of 1. At a radius of around 4.3×10^{-10} nm, c^{side} has nearly reached c^p and at a even lower radius c^{side} will exceed a concentration of 1. An interfacial concentration over 1 is, however, not possible. Also, a complication appears in the coarsening regime if $c^{side} > c^p$, since equation (3.29) changes sign and the fast-shrinking precipitates with low radii suddenly switch to an enormous growth. To avoid this complication, the threshold value at which the precipitates are removed from the system must not be too low and preferably at several burgers vector length.

The Gibbs-Thomson equation for c^{side} with a non-constant γ_{end} was also investigated for $\alpha = 1$ and $\alpha = 1.2$. As seen from Figure 4.9, the influence of the different $\gamma_{end}(R)$ functions from Figure 4.11 can have a large influence on the Gibbs-Thomson effect. Figure 4.9 depicts the influence of several $\gamma_{end}(R)$ with increasing $\gamma_{end}/\gamma_{side}$-ratios where the start and endpoint of the up ramp is approximately the same. By using a γ_{end} function with a high derivative as for the case of $\gamma_{end}/\gamma_{side} = 6.25$, the interfacial concentration rapidly increases as the up ramp starts. Using a γ_{end} function with a slightly less derivative decreases the sharp increase for c^{side}. To illustrate that equation (3.51) is most sensitive to the steepness of the γ_{end} function and not the threshold γ_{end} value, Figure 4.9 also depicts a γ_{end} function with the same $\gamma_{end}/\gamma_{side}$-ratio, but with a less steep gradient as shown by the stapled lines. This is correlated to the fact that the derivative is scaled with R/L, whilst γ_{end} is scaled with $1/L$ in addition to the derivative being of a much higher magnitude. Therefore, the same complication of too high interfacial concentration discussed in the previous paragraph also arises if the ramp-up of γ_{end} is too high as seen in Figure 4.9. However, as seen in Figure 4.10, the aspect ratio also have a large effect on equation (3.51). With a slightly higher aspect ratio at $\alpha = 1.2$, suppresses the quickly increasing c^{side} at the up ramping of γ_{end}. Thus, the problem of invalid interfacial concentrations only appears if the aspect ratio of the considered precipitate reaches 1. However, this can be a problem depending on the coarsening characteristic of the system. If the

precipitates shrink in the axial direction and are not removed before approaching an aspect ratio of 1, the interfacial concentration might reach invalid values.

5.3 Particle Distributions by the KWN-Model

To ensure that the framework around the proposed KWN-model is viable, a comparison to the Eulerian approach of Myhr *et al.* was shown in section 4.5.1. As seen in Figure 4.12, the nucleation rate is approximately constant for most of the time-span which is a consequence of neglecting the incubation period. The nucleation rate then quickly decreases as c^m approaches c^{eq} as can be seen in equation (2.34). As the nucleation rate quickly fades away, the total number of particles in the system reaches a maximum value before the system enters the coarsening regime. From Figure 4.13 one can see that the precipitates experience rapid growth under the nucleation and growth phase, before gradually reaching the critical radius of the system in which the total number of particles begins to decrease due to coarsening.

5.3.1 Constant Surface Energy

To facilitate the creation of needle-like particles, the surface energy of the end surfaces were increased in section 4.5.3. The increase of surface energy implies a higher resistance to create new surface area. Therefore, an increased γ_{end} relative to γ_{side} would imply a lower motivation to grow in the radial direction. However, as seen in Figure 4.18, the aspect ratio of the precipitates readily decreases after nucleation. This is a consequence of a higher growth rate in the radial direction which is also visible in Figures 4.16 and 4.17 where $\bar{R}$ has increased by a factor 2.5 while $\bar{L}$ only has grown a factor 1.5. The increased surface energy of the end surfaces only becomes prominent when the system has reached the coarsening regime where the effect of the Gibbs-Thomson equation sets in.

The implementation of a constant γ_{end} also influences the nucleation process as mentioned in section 4.5.3. Due to a much higher end surface energy, the critical nuclei will have to be nucleated with a high aspect ratio to ensure growth in both directions. However, as stated, this is not realistic as the nuclei would need to arrange in long needles before entering the growth stage. Another remark regarding

α^* and L^* is that one can see from Figure 4.17 that the critical length surpasses the mean length of the system. Still, the particles keep growing in the axial direction. While this may seem strange, the solution for the critical dimensions as proposed in section 3.3 is only one of multiple possible solutions. Another solution to consider as seen from equation (3.28), is when $\left(c^{end} - c^m\right) \hat{I}^{end}_{end} + \left(c^{side} - c^m\right) \hat{I}^{end}_{side} = 0$. As seen from Figures 4.4 and 4.5, $\hat{I}^{end}_{end}$ and $\hat{I}^{end}_{side}$ exhibit opposite signs and can yield alternative solutions depending on c^{end}, c^{side} and the aspect ratio of the considered particle.

A case for disk-like particles with constant surface energies was also simulated. As for the case of needles, the disk-like particles also deviate from the nucleated aspect ratio as the precipitates grow. As seen in Figures 4.20 and 4.21, the disk-like particles achieves a higher growth in the axial direction compared to the radial and therefore increases in aspect ratio during the nucleation and growth stage. At the onset of coarsening, the growth rate of the side surface increases more rapidly and the aspect ratio approaches its initial value.

5.3.2 Variation of Surface Energy

To succumb to the difficulties that arrive in utilizing constant surface energies, the model was extended to account for a function for γ_{end} that depends on the radius of the precipitate. Implementing an equation for γ_{end} resulted in facilitating both the nucleation of precipitates of spherical character, i.e $\alpha \approx 1$, and a subsequent increase of aspect ratio for the particles as they grow. As seen from Figure 4.24, the growth of the radius of the precipitates is suppressed when applying an increasing γ_{end} -function. This is reasonable as an increased γ_{end} inhibits growth in the radial direction. The critical radius of the system also appears to grow slower which signifies that the concentration of the matrix also decreases slower than for the case of constant surface energies shown in Figure 4.16. The mean length of the precipitates also seem to grow slowly at first, but as seen in Figures 4.25 and 4.26, the growth rate in the axial direction suddenly increases when the radius of the particles reach a certain point. This is easily seen in Figure 4.26 where the up leftmost particle size distribution experience an upward bend at around 2 nm. This could be because $\gamma_{end}(R)$ has a steep slope at this point in which the interfacial concentration of the sidewalls increases as seen from Figures 4.9 and

4.10.

The importance of the slope of the function for γ_{end} is further strengthened when assessing the aspect ratios of the precipitates for various up ramp functions. In Figure 4.27, three cases with increasing slopes are shown. The three functions for γ_{end} is essentially the same except for R_{sp}. For the function with the greatest slope, i.e $R_{sp} = 3$ nm, β has a much higher value throughout the evolution. As noted, the interfacial concentration at the sidewalls increases rapidly when γ_{end} has a greater slope which constrains the growth of the particle in the radial direction.

Another important element of the evolution of the particles through time is the nucleation rate which determines how many particles are introduced to the system. As seen in Figure 4.28, higher nucleation rates yields a lower mean radius of the particles at intermediate times. This can be linked to the fact that the equilibrium precipitate volume fraction is sooner met for the high nucleation case. Therefore, the precipitates have less time for growth before the coarsening regime enters. While the radius of the precipitates is suppressed with higher nucleation rates, β is also lower as seen in Figure 4.29 which signifies that the growth in the axial direction is also restrained.

The comparison to the experimental data in terms of mean cross-section area and mean length of the precipitates yields promising results for the proposed KWN-model in this thesis as seen in Figure 4.31. A linear relationship between mean cross-section area and mean length was also seen for the implemented model, albeit not for low $\bar{L}$. As previously discussed, the function for γ_{end} is of great influence on the resulting particle size distribution. Therefore, it can be imagined that one can tailor the input parameters to the model to achieve an even better correlation with experimental data. For instance, the slope of the linear relationship can be adjusted by adjusting the function for γ_{end}.

5.3.3 Improvements to the Model

Several improvements can be done to either increase the accuracy of the proposed model, or its computational efficiency. For instance, in the Lagrangian approach of this model, a new class is introduced to the system at each time step until nucleation has stopped. Therefore, the amount of equations to solve for the growth

of the radius and length of the precipitates increases by a factor 2 at each time step. However, one can implement an algorithm that checks if the newly nucleated class does not differ from the class nucleated at the prior time step. If the two classes are reasonably similar, one can combine the classes to effectively reduce the number of equations that need to be solved in the system.

Another important aspect to consider is that accurate values for γ_{end} are not easily found and the values used in this thesis are essentially fitted parameters. As such, the presented results in this thesis are qualitative in nature and accurate values for the surface energies are needed to accurately describe the particle size distribution. A possible way to acquire reasonable values can be to do extensive atomistic simulations to find the appropriate surface energies for a precipitate.

The model implemented in this thesis also utilizes a simple heterogeneous nucleation equation. Bear in mind that the adopted equations for the nucleation rate do not reflect the actual complexity of the nucleation event and is in essence a fitted equation.

Due to time limitations, an expression for γ_{side} as a function of L was not implemented. But, the framework for the model has been built. By deriving new equations for interfacial concentration at the end and side surface with a non-constant γ_{side}, one can test the proposed KWN-model for the growth of disk-like particles.

6 Conclusion

In this thesis, the Kampmann Wagner Numerical model was successfully implemented for a cylindrical particle using a Lagrangian approach. A distinction between the particle's end and side surfaces were made in terms of interfacial compositions and growth rates. This was achieved by performing a detailed numerical solution of the surrounding diffusion field around the particle, and a derivation of the axisymmetric Gibbs-Thomson equation for the end and side surfaces of the cylindrical particle. Further, an equation for the surface energy of the sidewalls of the considered particle was suggested to facilitate the growth of elongated needles.

7 Further Work

To further extend the proposed KWN-model for a cylindrical particle, the following measures can be done.

- Obtaining accurate values for the surface energies as a function of the particle radius and/or length.
- Extending the proposed KWN-model by deriving new Gibbs-Thomson equations for c^{side} and c^{end} with a non-constant γ_{side}.
- Implement more sophisticated nucleation equations to the model.
- Testing the model for non-isothermal cases such as welding or other non-isothermal transformations.

References

[1] Aniruddha Biswas, Donald J. Siegel, C. Wolverton, and David N. Seidman. Precipitates in Al–Cu alloys revisited: Atom-probe tomographic experiments and first-principles calculations of compositional evolution and interfacial segregation. *Acta Materialia*, 59(15):6187–6204, September 2011.

[2] G.A. Edwards, K. Stiller, G.L. Dunlop, and M.J. Couper. The precipitation sequence in Al–Mg–Si alloys. *Acta Materialia*, 46(11):3893–3904, July 1998.

[3] P.M. Kelly. The effect of particle shape on dispersion hardening. *Scripta Metallurgica*, 6(8):647–656, August 1972.

[4] Long-Qing Chen. Phase-Field Models for Microstructure Evolution. *Annual Review of Materials Research*, 32(1):113–140, August 2002.

[5] G. Wang, D.S. Xu, N. Ma, N. Zhou, E.J. Payton, R. Yang, M.J. Mills, and Y. Wang. Simulation study of effects of initial particle size distribution on dissolution. *Acta Materialia*, 57(2):316–325, January 2009.

[6] Britta Nestler and Abhik Choudhury. Phase-field modeling of multi-component systems. *Applications of Phase Field Modeling in Materials Science and Engineering*, 15(3):93–105, June 2011.

[7] Richard Wagner, Reinhard Kampmann, and Peter W. Voorhees. Homogeneous Second-Phase Precipitation. In *Phase Transformations in Materials*, pages 213–303. John Wiley & Sons, Ltd, 1991.

[8] O. R. Myhr and Ø Grong. Modelling of non-isothermal transformations in alloys containing a particle distribution. *Acta Materialia*, 48(7):1605 – 1615, 2000.

[9] Bjørn Holmedal, Elisa Osmundsen, and Qiang Du. Precipitation of Non-Spherical Particles in Aluminum Alloys Part I: Generalization of the Kampmann–Wagner Numerical Model. *Metallurgical and Materials Transactions A*, 47(1):581–588, January 2016.

[10] Thomas Vatn Bjørge. Modelling of precipitation of non-spherical particles in alloys. Project report in TMT4500, Department of Materials Science and Engineering, NTNU – Norwegian University of Science and Technology, Dec. 2019.

[11] David A. Porter, Kenneth E. Easterling, and Mohamed Y. Sherif. *Phase Transformations in Metals and Alloys (Revised Reprint).* CRC Press, February 2009.

[12] W.D. Callister and D.G. Rethwisch. *Materials Science and Engineering.* Wiley, 2014.

[13] Robert Alexander Adams and Christopher Essex. *Calculus: a complete course.* Pearson, Toronto, eighth edition edition, 2013.

[14] Michel Perez. Gibbs–Thomson effects in phase transformations. *Scripta Materialia*, 52(8):709–712, April 2005.

[15] C.D Marioara, S.J Andersen, J Jansen, and H.W Zandbergen. The influence of temperature and storage time at RT on nucleation of the β'' phase in a 6082 Al–Mg–Si alloy. *Acta Materialia*, 51(3):789–796, February 2003.

[16] L. Sagalowicz, G. Lapasset, and G. Hug. Transmission electron microscopy study of a precipitate which forms in the Al-Mg-Si system. *Philosophical Magazine Letters*, 74(2):57–66, August 1996. Publisher: Taylor & Francis.

[17] T. Gladman. Precipitation hardening in metals. *Materials Science and Technology*, 15(1):30–36, January 1999.

[18] Kenneth C. Russell. Nucleation in solids: The induction and steady state effects. *Advances in Colloid and Interface Science*, 13(3):205–318, September 1980.

[19] Hubert I. Aaronson, Masato Enomoto, Jong K. Lee, Masato Enomoto, and Jong K. Lee. *Mechanisms of Diffusional Phase Transformations in Metals and Alloys*. CRC Press, April 2016.

[20] Clarence Zener. Theory of Growth of Spherical Precipitates from Solid Solution. *Journal of Applied Physics*, 20(10):950–953, 1949. Publisher: American Institute of Physics.

[21] M. Perez, M. Dumont, and D. Acevedo-Reyes. Implementation of classical nucleation and growth theories for precipitation. *Acta Materialia*, 56(9):2119–2132, 2008.

[22] Philippe Maugis and Mohamed Gouné. Kinetics of vanadium carbonitride precipitation in steel: A computer model. *Acta Materialia*, 53(12):3359–3367, July 2005.

[23] J. S. Langer and A. J. Schwartz. Kinetics of nucleation in near-critical fluids. *Physical Review A*, 21(3):948–958, March 1980. Publisher: American Physical Society.

[24] R. Kampmann and R. Wagner. Kinetics of Precipitation in Metastable Binary Alloys: Theory and Application to Cu-1. 9 at.% Ti and Ni-14 at.% Al. *Decomposition of alloys: the early stages*, pages 91–103, 1983.

[25] I.M. Lifshitz and V.V. Slyozov. The kinetics of precipitation from supersaturated solid solutions. *Journal of Physics and Chemistry of Solids*, 19(1):35–50, April 1961.

[26] Carl Wagner. Theorie der alterung von niederschlägen durch umlösen (Ostwald-reifung). *Zeitschrift für Elektrochemie, Berichte der Bunsengesellschaft für physikalische Chemie*, 65(7-8):581–591, 1961. ISBN: 0005-9021 Publisher: Wiley Online Library.

[27] James F. Price. *Lagrangian and eulerian representations of fluid flow: Kinematics and the equations of motion*. MIT OpenCourseWare, 2006.

[28] Bernard W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[29] Artur Gramacki. *Nonparametric Kernel Density Estimation and Its Computational Aspects*. Studies in Big Data. Springer International Publishing, 2018.

[30] Qiang Du, Bjørn Holmedal, Jesper Friis, and Calin D. Marioara. Precipitation of Non-spherical Particles in Aluminum Alloys Part II: Numerical Simulation and Experimental Characterization During Aging Treatment of an Al-Mg-Si Alloy. *Metallurgical and Materials Transactions A*, 47(1):589–599, January 2016.

[31] M Murayama and K Hono. Pre-precipitate clusters and precipitation processes in Al–Mg–Si alloys. *Acta Materialia*, 47(5):1537–1548, March 1999.

[32] Peter N. Brown, George D. Byrne, and Alan C. Hindmarsh. VODE: A Variable-Coefficient ODE Solver. *SIAM Journal on Scientific and Statistical Computing*, 10(5):1038–1051, September 1989. Publisher: Society for Industrial and Applied Mathematics.

[33] Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019.

[34] Jonas K. Sunde, Calin D. Marioara, and Randi Holmestad. The effect of low Cu additions on precipitate crystal structures in overaged Al-Mg-Si(-Cu) alloys. *Materials Characterization*, 160:110087, February 2020.

A Fortran Code

A.1 Algorithm for Numerical Diffusion Solution

```
1  !Masteroppgave.f90
2  program Masteroppgave
3  use DISPMODULE  !module used for plotting matrices
4  implicit none
5  !Declaration of variables
6  integer, parameter :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
7  real (kind=ikind), parameter :: pi = 3.141592653589793
8  integer :: i, j, allstat, count, N_R, N_Z, N_Rmax, N_Zmax   !N = grid size, N_R/N_Z nodes in
       particle in r and z direction
9  real (kind=ikind), allocatable :: dcr(:), dcz(:), a(:), s(:), r(:), z(:), d(:), Q(:), dtemp(:), C(:,:),
       temp(:,:), errLaplace(:,:)
10 real (kind=ikind) :: alpha, Rmax, Zmax, r0, z0, q_z, q_r, err, I_end, I_side, I_sum, temp2
11
12 read(*,*) alpha      !Read in aspect ratio(L/2R)
13 !Define variables related to grid
14 !N_R = 81
15 N_Z = 81    !Either assign value to N_R or N_Z
16 Rmax = 150 !length of grid, set equal to 1 if q is known
17 Zmax = 150
18 !r0 = 1.D0/(N_R-1)  !For needles
19 !N_Z = ceiling((alpha/r0)+1) !For needle
20 z0 = alpha/(N_Z-1)  !initial grid spacing at particle + 2 node
21 N_R = ceiling((1/z0)+1) !for plate
22 r0 = 1.D0/(N_R-1) !initial grid spacing at particle + 2 node
23 q_r = 1.05      !stretching factors
24 q_z = 1.05
25
```

```
26 N_Rmax = ceiling(log(-N_R*q_r+((Rmax*(q_r-1))/r0)+N_R+1)/log(q_r)+N_R+1)    !
       calculate the required number of nodes in each direction
27 N_Zmax = ceiling(log(-N_Z*q_z+((Zmax*(q_z-1))/z0)+N_Z+1)/log(q_z)+N_Z+1)
28 allocate(C(N_Zmax,N_Rmax),temp(N_Zmax,N_Rmax), errLaplace(N_Zmax,N_Rmax),r(
       N_Rmax),z(N_Zmax),d(N_Rmax), &
29     Q(N_Rmax),dtemp(N_Rmax),a(N_Rmax-1),s(N_Rmax-1),dcr(N_Z),dcz(N_R), STAT =
       allstat)  !allocate memory according to input
30 If (allstat /= 0) STOP "*** Not enough memory ***"
31
32 call grid2(N_R,N_Rmax,r0,q_r,Rmax,r) !Defines the grid vector in each direction.
33 call grid2(N_Z,N_Zmax,z0,q_z,Zmax,z)
34
35 !initial C with dirichlet boundaries
36 C(1:N_Z-1,1:N_R-1) = 1. !particle
37 C(N_Z,1:N_R-1) = 1. !top of particle
38 C(1:N_Z-1,N_R) = 0. !side of particle
39 C(N_Z,N_R) = (C(N_Z-1,N_R)+C(N_Z,N_R-1))/2 !corner node of particle -->  avg of top
       and side
40 errLaplace(1:N_Z,1:N_R) = 0
41
42 I_sum = 15 !first guess of flux
43 err = 1.
44 count = 0
45
46 !-------------------------------------------------------
47 do while (err > 5e-6)
48
49    do i = 1,N_Rmax  !updating the boundaries of domain with the flux calculated.
50       C(N_Zmax,i) = I_sum/(4*pi*sqrt(r(i)**2+z(N_Zmax)**2))
51    end do
52
53    do i = 1,N_Zmax  !updating the boundaries of domain with the flux calculated.
54       C(i,N_Rmax) = I_sum/(4*pi*sqrt(r(N_Rmax)**2+z(i)**2))
55    end do
56
57    temp = C    !to check if diverged
58    temp2 = I_sum
59    count = count + 1
60    !-------------------------------------------------------
61    !Row-wise sweep of grid
62    !a - subdiagonal -->  C_i-1
```

```
63  !d - diagonal -->  C_i,j
64  !s - superdiagonal -->  C_i+1
65  !Q - right hand side -->  C_i,j+1 + C_i,j-1
66  !a and s is independent of j and can be found for the whole row sweep
67  do i = 2,N_Rmax-1
68     a(i-1) = (3*r(i)-r(i+1))/       &           !C_i-1
69              (r(i)*(r(i-1)-r(i))*(r(i-1)-r(i+1)))
70
71     s(i) = (3*r(i)-r(i-1))/       &           !C_i+1
72           (r(i)*(r(i-1)-r(i+1))*(r(i)-r(i+1)))
73
74     dtemp(i) = (r(i-1)-4*r(i)+r(i+1))/      &   !part of d that is independent of j
75                (r(i)*(r(i-1)-r(i))*(r(i)-r(i+1)))
76  end do
77
78  a(N_Rmax-1) = 0.     !since last element of row has dirichlet boundary
79  s(N_R) = 0.     !Dirichlet at particle
80  s(1) = 4/((r(2)-r(1))**2)       !at r=0
81  d((/N_R, N_Rmax/)) = 1.      !Dirichlet boundary
82  !-------------------------------------------------------------
83  !Bottom row
84  Q(N_R) = C(1,N_R)
85  Q(N_Rmax) = C(1,N_Rmax)
86  do i = N_R+1,N_Rmax-1
87     d(i) = dtemp(i)-2/((z(2)-z(1))**2)             !C_i,j
88
89     Q(i) = -(2*C(2,i))/((z(2)-z(1))**2)            !C_i,j+1
90  end do
91  call tdma(N_Rmax-N_R+1,a(N_R:N_Rmax-1),d(N_R:N_Rmax),s(N_R:N_Rmax-1),Q(N_R:
      N_Rmax),C(1,N_R:N_Rmax))       !call subroutine to calculate C
92  !-------------------------------------------------------------
93  !Second row up to N_Z
94  do j = 2,N_Z
95     Q(N_R) = C(j,N_R)
96     Q(N_Rmax) = C(j,N_Rmax)
97     do i = N_R+1,N_Rmax-1
98        d(i) = dtemp(i)-2/((z(j+1)-z(j))*(z(j)-z(j-1)))                  !C_i,j
99
100       Q(i) = -(2*C(j+1,i))/((z(j+1)-z(j))*(z(j+1)-z(j-1)))    &          !C_i,j+1 + C_i,j
      -1
101                -(2*C(j-1,i))/((z(j)-z(j-1))*(z(j+1)-z(j-1)))
```

```
102       end do
103       call tdma(N_Rmax-N_R+1,a(N_R:N_Rmax-1),d(N_R:N_Rmax),s(N_R:N_Rmax-1),Q(
          N_R:N_Rmax),C(j,N_R:N_Rmax))
104    end do
105    !-----------------------------------------------------------------
106    !Row N_Z+1 up to N-1
107    s(N_R) = (3*r(N_R)-r(N_R-1))/       &                              !Need to recreate S
          at N_R, s not 0
108           (r(N_R)*(r(N_R-1)-r(N_R+1))*(r(N_R)-r(N_R+1)))
109    do j = N_Z+1,N_Zmax-1
110       d(1) = -(4/((r(2)-r(1))**2)+2/((z(j+1)-z(j))*(z(j)-z(j-1))))          !Neumann
          boundary
111       Q(1) = -(2*C(j+1,1))/((z(j+1)-z(j))*(z(j+1)-z(j-1)))    &
112            -(2*C(j-1,1))/((z(j)-z(j-1))*(z(j+1)-z(j-1)))
113       Q(N_Rmax) = C(j,N_Rmax)
114       do i = 2,N_Rmax-1
115          d(i) = dtemp(i)-2/((z(j+1)-z(j))*(z(j)-z(j-1)))                  !C_i,j
116
117          Q(i) = -(2*C(j+1,i))/((z(j+1)-z(j))*(z(j+1)-z(j-1)))     &       !C_i,j+1 + C_i,j
          -1
118               -(2*C(j-1,i))/((z(j)-z(j-1))*(z(j+1)-z(j-1)))
119       end do
120       call tdma(N_Rmax,a,d,s,Q,C(j,1:N_Rmax))
121    end do
122    deallocate(d,Q,dtemp,a,s, STAT = allstat)  !deallocate memory from row-wise sweep
123    If (allstat /= 0) STOP "*** Deallocation not succesful ***"
124    !-----------------------------------------------------------------
125    !Column-wise sweep of grid
126    allocate(d(N_Zmax),Q(N_Zmax),dtemp(N_Zmax),a(N_Zmax-1),s(N_Zmax-1), STAT =
          allstat)  !allocate memory according to input
127    If (allstat /= 0) STOP "*** Not enough memory ***"
128    !a and s is independent of i and can be found for the whole column sweep
129    do j = 2,N_Zmax-1
130       a(j-1) = 2/((z(j)-z(j-1))*(z(j+1)-z(j-1)))  !C_j-1
131
132       s(j) = 2/((z(j+1)-z(j))*(z(j+1)-z(j-1)))    !C_j+1
133
134       dtemp(j) = 2/((z(j+1)-z(j))*(z(j)-z(j-1)))  !part of d that is independent of i
135    end do
136    a(N_Zmax-1) = 0.     !since last element of column has dirichlet boundary
137    s(N_Z) = 0.     !Dirichlet at particle
```

```
138   s(1) = 2/((z(2)-z(1))**2)   !at z=0
139   d((/N_Z, N_Zmax/)) = 1.      !Dirichlet boundary
140   !-------------------------------------------------------
141   !Left column
142   Q(N_Z) = C(N_Z,1)
143   Q(N_Zmax) = C(N_Zmax,1)
144   do j = N_Z+1,N_Zmax-1
145      d(j) = -(dtemp(j)+4/((r(2)-r(1))**2))              !C_i,j
146
147      Q(j) = -(4*C(j,2))/((r(2)-r(1))**2)                !C_i,j+1
148   end do
149   call tdma(N_Zmax-N_Z+1,a(N_Z:N_Zmax-1),d(N_Z:N_Zmax),s(N_Z:N_Zmax-1),Q(N_Z:
        N_Zmax),C(N_Z:N_Zmax,1))
150   !-------------------------------------------------------
151   !Second column to N_R
152   do i = 2,N_R
153      Q(N_Z) = C(N_Z,i)
154      Q(N_Zmax) = C(N_Zmax,i)
155      do j = N_Z+1,N_Zmax-1
156         d(j) = -dtemp(j)+(r(i-1)-4*r(i)+r(i+1))/      &          !C_i,j
157              (r(i)*(r(i-1)-r(i))*(r(i)-r(i+1)))
158
159         Q(j) = -((3*r(i)-r(i+1))/       &                    !C_i-1 + C_i+1
160              (r(i)*(r(i-1)-r(i))*(r(i-1)-r(i+1))))*C(j,i-1)      &
161              -((3*r(i)-r(i-1))/       &
162              (r(i)*(r(i-1)-r(i+1))*(r(i)-r(i+1))))*C(j,i+1)
163      end do
164      call tdma(N_Zmax-N_Z+1,a(N_Z:N_Zmax-1),d(N_Z:N_Zmax),s(N_Z:N_Zmax-1),Q(N_Z
        :N_Zmax),C(N_Z:N_Zmax,i))
165   end do
166   !-------------------------------------------------------
167   !Column N_R+1 up to N-1
168   s(N_Z) = 2/((z(N_Z+1)-z(N_Z))*(z(N_Z+1)-z(N_Z-1)))          !s not equal to 0
        anymore
169   do i = N_R+1,N_Rmax-1
170      d(1) = (r(i-1)-4*r(i)+r(i+1))/      &                   !Neumann boundary
171           (r(i)*(r(i-1)-r(i))*(r(i)-r(i+1)))-      &
172           2/((z(2)-z(1))**2)
173      Q(1) = -((3*r(i)-r(i+1))/       &                       !C_i-1 + C_i+1
174           (r(i)*(r(i-1)-r(i))*(r(i-1)-r(i+1))))*C(1,i-1)      &
175           -((3*r(i)-r(i-1))/       &
```

```
176             (r(i)*(r(i-1)-r(i+1))*(r(i)-r(i+1))))*C(1,i+1)
177         Q(N_Zmax) = C(N_Zmax,i)
178         do j = 2,N_Zmax-1
179             d(j) = -dtemp(j)+(r(i-1)-4*r(i)+r(i+1))/      &              !C_i,j
180                 (r(i)*(r(i-1)-r(i))*(r(i)-r(i+1)))
181
182             Q(j) = -((3*r(i)-r(i+1))/       &                          !C_i-1 + C_i+1
183                 (r(i)*(r(i-1)-r(i))*(r(i-1)-r(i+1))))*C(j,i-1)      &
184                 -((3*r(i)-r(i-1))/      &
185                 (r(i)*(r(i-1)-r(i+1))*(r(i)-r(i+1))))*C(j,i+1)
186         end do
187         Call tdma(N_Zmax,a,d,s,Q,C(1:N_Zmax,i))
188     end do
189     deallocate(d,Q,dtemp,a,s, STAT = allstat)  !deallocate memory from row-wise sweep
190     If (allstat /= 0) STOP "*** Deallocation not succesful ***"
191     !-------------------------------------------------------------
192     !Error of laplace equation
193     allocate(d(N_Rmax),Q(N_Rmax),dtemp(N_Rmax),a(N_Rmax-1),s(N_Rmax-1), STAT =
          allstat)  !allocate memory according to input
194     If (allstat /= 0) STOP "*** Not enough memory ***"
195
196     a(N_Rmax-1) = 0.     !since last element of row has dirichlet boundary
197     s(N_R) = 0.     !Dirichlet at particle
198     s(1) = 4/((r(2)-r(1))**2)       !at r=0
199     d((/N_R, N_Rmax/)) = 1.      !Dirichlet boundary
200     !-------------------------------------------------------------
201     !Bottom row
202     Q(N_R) = C(1,N_R)
203     Q(N_Rmax) = C(1,N_Rmax)
204     do i = N_R+1,N_Rmax-1
205         a(i-1) = (3*r(i)-r(i+1))/       &                           !C_i-1
206             (r(i)*(r(i-1)-r(i))*(r(i-1)-r(i+1)))*C(1,i-1)
207
208         s(i) = (3*r(i)-r(i-1))/       &                             !C_i+1
209             (r(i)*(r(i-1)-r(i+1))*(r(i)-r(i+1)))*     &
210             C(1,i+1)
211
212         d(i) = ((r(i-1)-4*r(i)+r(i+1))/       &
213             (r(i)*(r(i-1)-r(i))*(r(i)-r(i+1)))-2/((z(2)-z(1))**2))*C(1,i)     !C_i,j
214
215         Q(i) = -(2*C(2,i))/((z(2)-z(1))**2)                           !C_i,j+1
```

```
216
217       errLaplace(1,i) = a(i-1)+d(i)+s(i)-Q(i)
218     end do
219     !-------------------------------------------------------------
220     !Second row up to N_Z
221     do j = 2,N_Z
222       Q(N_R) = C(j,N_R)
223       Q(N_Rmax) = C(j,N_Rmax)
224       do i = N_R+1,N_Rmax-1
225         a(i-1) = (3*r(i)-r(i+1))/       &                                   !C_i-1
226                  (r(i)*(r(i-1)-r(i))*(r(i-1)-r(i+1)))*C(j,i-1)
227
228         s(i) = (3*r(i)-r(i-1))/         &                                   !C_i+1
229                (r(i)*(r(i-1)-r(i+1))*(r(i)-r(i+1)))*    &
230                C(j,i+1)
231         d(i) = ((r(i-1)-4*r(i)+r(i+1))/      &
232                (r(i)*(r(i-1)-r(i))*(r(i)-r(i+1)))-2/((z(j+1)-z(j))*(z(j)-z(j-1))))*C(j,i)      !
    C_i,j
233
234         Q(i) = -(2*C(j+1,i))/((z(j+1)-z(j))*(z(j+1)-z(j-1)))    &                   !
    C_i,j+1 + C_i,j-1
235                -(2*C(j-1,i))/((z(j)-z(j-1))*(z(j+1)-z(j-1)))
236
237         errLaplace(j,i) = a(i-1)+d(i)+s(i)-Q(i)
238       end do
239     end do
240     !-------------------------------------------------------------
241     !Row N_Z+1 up to N-1
242     s(N_R) = (3*r(N_R)-r(N_R-1))/       &                                 !Need to recreate S
    at N_R, s not 0
243            (r(N_R)*(r(N_R-1)-r(N_R+1))*(r(N_R)-r(N_R+1)))
244     do j = N_Z+1,N_Zmax-1
245       d(1) = -(4/((r(2)-r(1))**2)+2/((z(j+1)-z(j))*(z(j)-z(j-1))))          !Neumann
    boundary
246       Q(1) = -(2*C(j+1,1))/((z(j+1)-z(j))*(z(j+1)-z(j-1)))    &
247              -(2*C(j-1,1))/((z(j)-z(j-1))*(z(j+1)-z(j-1)))
248       Q(N_Rmax) = C(j,N_Rmax)
249       do i = 2,N_Rmax-1
250         a(i-1) = (3*r(i)-r(i+1))/       &                                   !C_i-1
251                  (r(i)*(r(i-1)-r(i))*(r(i-1)-r(i+1)))*C(j,i-1)
252
```

```
253         s(i) = (3*r(i)-r(i-1))/       &                                   !C_i+1
254               (r(i)*(r(i-1)-r(i+1))*(r(i)-r(i+1)))*    &
255               C(j,i+1)
256         d(i) = ((r(i-1)-4*r(i)+r(i+1))/      &
257               (r(i)*(r(i-1)-r(i))*(r(i)-r(i+1)))-2/((z(j+1)-z(j))*(z(j)-z(j-1))))*C(j,i)      !
    C_i,j
258
259         Q(i) = -(2*C(j+1,i))/((z(j+1)-z(j))*(z(j+1)-z(j-1)))    &                    !
    C_i,j+1 + C_i,j-1
260               -(2*C(j-1,i))/((z(j)-z(j-1))*(z(j+1)-z(j-1)))
261
262         errLaplace(j,i) = a(i-1)+d(i)+s(i)-Q(i)
263       end do
264   end do
265   !-------------------------------------------------------
266   !Flux calculations
267   !Side of cylinder
268   do j = 1,N_Z
269       dcr(j) = (-3*C(j,N_R)+4*C(j,N_R+1)-C(j,N_R+2))/(2*r0)
270   end do
271
272   I_side = -4*pi*r0*(0.5*(dcr(1)+dcr(N_Z))+sum(dcr(2:N_Z-1)))          !dimensionless flux
    from cylinder side. Trapezoid method
273
274   !End of cylinder
275   do i = 1,N_R
276       dcz(i) = ((-3*C(N_Z,i)+4*C(N_Z+1,i)-C(N_Z+2,i))/(2*z0))*r(i)
277   end do
278
279   I_end = -4*pi*z0*(0.5*(dcz(1)+dcz(N_R))+sum(dcz(2:N_R-1)))           !dimensionless
    flux from cylinder end. Trapezoid method
280   I_sum = I_side + I_end
281   !-------------------------------------------------------
282   !Error term with sum of error + the highest error in conc.field
283   err = 1./(min(N_Rmax,N_Zmax)**2.)*sum(abs(C-temp))+ maxval(abs(C-temp)) +
    1./15*abs((I_sum-temp2)) + &
284       1./(min(N_Rmax,N_Zmax)**2.)*sum(abs(errLaplace)) + maxval(abs(errLaplace))
285   !Prints iterations, flux and error each 100 iteration
286   if ((count/100)*100 == count) then
287   print *, count, I_sum, err
288   end if
```

```
289 end do
290 !-----------------------------------------------------------
291 open(10,file='C_field2.txt')
292 call disp(C,UNIT=10,DIGMAX=15)        !module that prints matrix in regular format
293
294 open(11,file='r_vec.txt')
295 call disp(r,UNIT=11,DIGMAX=15)
296
297 open(12,file='z_vec.txt')
298 call disp(z,UNIT=12,DIGMAX=15)
299
300 print *, N_Rmax, N_Zmax, alpha, I_side, I_end, I_sum, count, r0, q_r, Rmax, Zmax
301
302 end program Masteroppgave
303
304 !-----------------------------------------------------------
305 subroutine grid(N_R,N_max,delta,q,Rmax,r) !function for finding Rmax/q
306 implicit none
307 !Declaration of variables
308 !N_R - #nodes for particle
309 !N_max - max # nodes
310 !delta - initial grid spacing
311 !q - stretch factor for grid
312 !Rmax - length of grid
313 !r - grid coordinate vector
314 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
315 integer, intent(in) :: N_R, N_max
316 real (kind=ikind), intent(in) :: delta
317 real (kind=ikind), intent(inout) :: Rmax, q
318 real (kind=ikind), dimension(N_max), intent(out) :: r
319 !local variables
320 real (kind=ikind) :: const, q_old, f_old, f, test
321 integer :: i
322
323 if (Rmax == 1 ) then !When Rmax is not given
324     Rmax = delta*((N_R+1)+(q**(N_max-N_R-1)-q)/(q-1))
325 else if (q == 1) then
326     Rmax = delta*(N_max-1)
327 else
328     q_old = 1.1 !Initial guess
329     f_old = Rmax - delta*((N_R+1)+(q_old**(N_max-N_R-1)-q_old)/(q_old-1))
```

```
330     q = 1.01
331     f = 1.
332     do while (abs(f) > 1e-8) !secant method
333        if (abs(q-1.) < 1e-10) then
334           f = Rmax - delta*((N_R+1)+(N_max-N_R-2))        !Limit when q --> 1
335        else
336           f = Rmax - delta*((N_R+1)+(q**(N_max-N_R-1)-q)/(q-1))   !else when q is not
       too near 1
337        end if
338        if (f == f_old) then
339           const = 0.
340           f = 0.
341        end if
342        const = -f*(q-q_old)/(f-f_old)
343        q_old = q
344        q = q + const
345        f_old = f
346     end do
347     test = delta*((N_R+1)+(q**(N_max-N_R-1)-q)/(q-1)) !Have to test if solution has
       diverged (because of poor initial values or non-meaningful input)
348     if (abs(Rmax-test)>1) then
349        STOP 'ERROR DETECTED! SUBROUTINE GRID NOT SUCCESSFUL'
350     end if
351  end if
352
353  !Define grid coordinates
354  r(1) = 0
355  do i = 2,N_R+2
356     r(i) = r(i-1) + delta
357  end do
358  do i = N_R+3, N_max
359     r(i) = r(i-1) + delta*q**(i-N_R-2)
360  end do
361  end subroutine grid
362  !-----------------------------------------------------------
363  subroutine grid2(N_R,N_max,delta,q,Rmax,r) !function for finding Rmax/q
364  implicit none
365  !Declaration of variables
366  !N_R - #nodes for particle
367  !N_max - max # nodes
368  !delta - initial grid spacing
```

```
!q - stretch factor for grid
!Rmax - length of grid
!r - grid coordinate vector
integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
integer, intent(in) :: N_R, N_max
real (kind=ikind), intent(in) :: delta
real (kind=ikind), intent(inout) :: Rmax, q
real (kind=ikind), dimension(N_max), intent(out) :: r
!local variables
real (kind=ikind) :: const, q_old, f_old, f, test
integer :: i

Rmax = delta*((N_R+1)+(q**(N_max-N_R-1)-q)/(q-1))
!Define grid coordinates
r(1) = 0
do i = 2,N_R+2
   r(i) = r(i-1) + delta
end do
do i = N_R+3, N_max
   r(i) = r(i-1) + delta*q**(i-N_R-2)
end do
end subroutine grid2
!-----------------------------------------------------------
subroutine tdma(N,a,d,s,Q,phi) !tridiagonalsolver
implicit none
!Declaration of variables
!N - size of diagonal
!a - subdiagonal
!d - central diagonal
!s - superdiagonal
!Q - right hand side
!phi - unknowns
integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
integer, intent(in) :: N
real (kind=ikind), intent(in), dimension(N-1) :: a,s
real (kind=ikind), intent(in), dimension(N) :: d,Q
real (kind=ikind), intent(out) :: phi(N)
!Local variables
integer :: i
real (kind=ikind) :: const
real (kind=ikind), dimension(N) :: d_new,Q_new
```

```
410 d_new = d
411 Q_new = Q
412 !Forward elimination
413 do i = 2,N
414     const = a(i-1)/d_new(i-1)
415     d_new(i) = d_new(i) - const*s(i-1) !diagonal
416     Q_new(i) = Q_new(i) - const*Q_new(i-1) !right hand side
417 end do
418 phi(N) = Q_new(N)/d_new(N) !last equation of matrix
419 !Backward substitusion
420 do i = N-1, 1, -1
421     phi(i) = (Q_new(i)-s(i)*phi(i+1))/d_new(i)
422 end do
423 end subroutine tdma
```

A.2 Algorithm for Lagrangian for KWN-Model

```
1  !Precipitation_model.f90
2  program Precipitation_model
3  implicit none
4  external JEX, precipitation
5  !-----------------------------------------------------------
6
7  !Declaration of parameters:
8  integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
9  real(kind=ikind), parameter :: k = 1.380648813D-23, Rg = 8.314462175  !Boltzmann constant
       [J/K] and universal gas constant [J/K mol]
10 real(kind=ikind), parameter :: pi = 3.14159265358979323846264433
11
12 integer, parameter :: idist = 600    !Number of points for kernel density estimator(kde)
13 integer, parameter :: imax = 3900    !Number of timesteps
14
15 logical C_e_test, C_s_test, negative_check  !Logicals for if tests
16
17 integer, parameter :: NEQ = imax*2   !Number of ODEs
18 integer, parameter :: NWDIM=22+9*NEQ+2*NEQ**2, IWDIM=30+NEQ  !ODE solver
       parameters
19
20 !strings for write to file
21 Character(len = 25) :: str_r, str_l, str_i, str_g, str_rate_r,      &
```

```
22                      str_rate_l, str_N, str_r_crit, str_t,        &
23                      str_dist_r, str_dist_l, str_dist_2d
24
25 !Integers for solver and do loops
26 integer :: n_step, i_step, i, j, m, ITASK, ISTATE, IOPT, ITOL, MF,  &
27           LRW, LIW, step, IFLAG, index, a, b, count, N_check,      &
28           t_var, index_temp, gamma_funct
29
30 ! Real Constants and parameters
31 real (kind=ikind) :: asp, I_end_end0, I_end_side0, I_side_side0,    &
32                      I_side_end0, C_m, C_p, C_0, C_eq, gamma_s,     &
33                      gamma_e, xi, t0, q, t_res, Temp, V, khi, D,    &
34                      dtime, t, t_out, j0, A0, Q_d, f_p, r_check,    &
35                      asp_check, check, C_m_res, C_m_temp, r_stop,   &
36                      gamma_stop, d_gamma, r_start, gamma_e_r, rand_nucl
37
38 !Integer arrays
39 integer :: IWORK(IWDIM), IPAR(2)
40
41 !Real arrays
42 real (kind=ikind) :: r(imax+1), l(imax+1), N(imax+1), RPAR(17+NEQ),   &
43                      ATOL(1), RWORK(NWDIM), f_p_save(imax+1),         &
44                      RTOL(1), part(NEQ), part_der(NEQ),               &
45                      r_save(imax+1), l_save(imax+1), t_save(imax+1),  &
46                      l_check(imax+1), r_crit(imax+1), N_rate(imax+1), &
47                      mean_r(imax+1), mean_l(imax+1), N_tot(imax+1),   &
48                      C_e(imax+1), C_s(imax+1), C_m_save(imax+1),      &
49                      r_crit_nucl(imax+1)
50
51 !Arrays for kde
52 real (kind=ikind) :: phi(idist,idist), phi_r(idist), phi_l(idist),    &
53                      r_dist(idist), l_dist(idist)
54 !-----------------------------------------------------------------
55
56 !Material constans and parameters
57 C_0 = 0.0063D0          !initial solute conc.
58 C_m = C_0               !solte conc in matrix, vol-fraction
59 C_p = 0.634D0           !solute conc in particle
60 C_eq = 3.54D-05         !equilibrium solute conc
61 gamma_s = 0.2*0.6D0     !Surface energy of side, J/m^2
62 gamma_e = 0.2*0.6D0     !Surface energy of ends, J/m^2
```

```
63 Temp = 180D0+273.15D0   !Temperature, K
64 V = 6.559D-29           !Atomic volume precipitate, assume same as in matrix -->
       xi = V_B_p/V_B_m = 1
65 D = 2.278D-19           !Diffusion coefficient, m^2/s
66 j0 = 9.66D34            !Numerical constant for nucleation rate, #/m^3 s
67 A0 = 16220.0D0          !Energy barrier in nucleation equation, J/mol
68 Q_d = 130000.0D0        !Activation energy for diffusion, J/mol
69 xi = 1.0D0              !V_B_p/V_B_m
70 N = 0.0D0               !Initializing N and N_tot
71 N_tot(1) = 0.
72
73
74 !First order diff.eq solver constants and parameters, See DVODE-documentation for more
       information
75 t0 = 1.0D0             !t0 for strecthed timesteps
76 q = 1.002D0            !Stretching factor for timestep
77 ATOL(1) = 1.D-14       !Absolute tolerance parameter (scalar or array)
78 RTOL(1) = 1.D-10       !Relative tolerance parameter (scalar)
79 ITOL = 1               !1 if ATOL is scalar and 2 is ATOL is array
80 ITASK = 1              !1 for normal computation of output values of Y at t = TOUT
81 ISTATE = 1             !Integer flag (input and output).  Set ISTATE = 1
82 IOPT = 0               !0 to indicate no optional input used
83 MF = 22                !Method flag, 22 is standard
84 LRW = NWDIM            !Declared length of RWORK
85 LIW = IWDIM            !Declared length of IWORK
86
87 t = 0.0D0              !Initial time
88 t_out = 0.0D0          !Initializing t_out
89
90
91 !Variables for printing output, while loop and, gamma_end function
92 b = 30
93 check = 1
94 N_check = 0
95 index = 0
96 gamma_funct = 1
97 r_start = 1D-9
98 r_stop = 4D-9
99 gamma_stop = 15*gamma_s
100
101
```

```
102
103 !Defining IPAR_vector for inout in ODE-solver and other subroutines
104 IPAR(1) = imax
105
106 !Defining RPAR-vector for input in ODE-solver and other subroutines
107 RPAR(1) = D
108 RPAR(2) = C_m
109 RPAR(3) = V
110 RPAR(4) = xi
111 RPAR(5) = C_p
112 RPAR(6) = r_start
113 RPAR(7) = A0
114 RPAR(8) = gamma_funct
115 RPAR(9) = Temp
116 RPAR(10) = C_eq
117 RPAR(11) = Q_d
118 RPAR(12) = gamma_s
119 RPAR(13) = gamma_e
120 RPAR(14) = gamma_stop
121 RPAR(15) = r_stop
122 !-----------------------------------------------------------
123
124 !Loop which each time step nucleates a new class and calculates the growth of all the current
        classes
125 do i = 1,imax
126
127    !Calculate timestep and saving system if dt is too large
128    t_out = t_out + t0*q**(i-1)
129    t_res = t
130    step = 2
131    C_m_res = C_m
132
133
134    !Checks if timestep was succesful
135    do while (check == 1)
136
137       N(i) = (t_out-t)*j0*exp(-((A0/(Rg*Temp))**3)*((1/(log(C_m/C_eq)))**2))*exp(-Q_d
       /(Rg*Temp)) !Number of nucleated classes
138
139       N_rate(i) = j0*exp(-((A0/(Rg*Temp))**3)*((1/(log(C_m/C_eq)))**2))*exp(-Q_d/(Rg*
       Temp))   !dN/dt
```

```
140
141     r_crit(i) = (2.0*gamma_s*V)/(log(C_m/C_eq)*C_p*k*Temp)  !Critical radius for particle
142
143     !if N(i) is too low no class is created !
144     if (i > 1 .and. N(i) < N_tot(i)*1.0D-10 .or. N_check == 1) then
145         N(i) = 0
146         r(i) = 0
147         l(i) = 0
148         N_check = 1
149     else
150
151         r(i) = 1.05*(2.0*gamma_s*V)/(log(C_m/C_eq)*C_p*k*Temp)      !Overcritical radius
    to ensure growth
152
153         !Critical L is dependent on if gamma_e is constant or not
154         if (gamma_funct == 1) then
155
156             gamma_e_r = gamma_s + 0.5*(gamma_stop-gamma_s)* &
157             (1+tanh(((gamma_stop-gamma_s)*(r(i)-0.5*r_stop-0.5*r_start)) &
158             /(0.3*(r_stop-r_start))))
159
160             d_gamma = -(1.66667*(gamma_s - gamma_stop)**2*  &
161             (1/(cosh((3.33333*(-gamma_s + gamma_stop)*      &
162             (-0.5*r_start - 0.5*r_stop + r(i)))/            &
163             (-r_start + r_stop))**2)))/(r_start - r_stop)
164
165             l(i) = (d_gamma*r(i)**2)/gamma_s + (2*r(i)*gamma_e_r)/gamma_s
166             asp = l(i)/(2*r(i))
167
168         else
169
170             asp = gamma_e/gamma_s       !Critical aspect ratio
171
172             l(i) = 2.0*asp*r(i)     !L determined by critical aspect ratio
173
174         end if
175
176
177         !Update part-vector and RPAR-vector. Part is the output vector from DVODE.
    RPAR is the input-vector to the growth subroutine.
178         part(i*2-1) = r(i)
```

```
179         RPAR(15+i*2-1) = r(i)
180
181         part(2*i) = l(i)
182         RPAR(15+i*2) = l(i)
183
184       end if
185
186
187       !Finding index of first zero in N
188       index = findloc(N, VALUE = 0., DIM = 1) - 1
189
190
191       !Running ODE-solver. Checks if any classes has dissapeared or if no new class was
      nucleated
192       if (index /= 0 .and. index < i) then
193
194         !Classes dissapeared/not nucleated. Reset part-vector if dt was too big
195         part(1:(index*2-1):2) = r(1:index)
196         part(2:(index*2):2) = l(1:index)
197         RPAR(16:15+index*2-1:2) = r(1:index)
198         RPAR(17:15+index*2:2) = l(1:index)
199         IPAR(2) = index
200
201         !Precipitate volume fraction
202         f_p = pi*sum(N(1:index)*(part(1:(index*2-1):2)**2)*part(2:(index*2):2))
203
204         !Mass balance equation
205         C_m = (C_0-xi*f_p*C_p)/(1-xi*f_p)
206         RPAR(2) = C_m
207
208         r_crit_nucl(i) = (2.0*gamma_s*V)/(log(C_m/C_eq)*C_p*k*Temp)
209
210         ISTATE = 1  !ISTATE needs to be reset to 1 since number of equations increases/
      decreases
211
212         !DVODE calculate growth in R and L direction
213         call DVODE(precipitation,index*2,part(1:index*2),t,t_out,ITOL,RTOL,ATOL,  &
214             ITASK,ISTATE,IOPT,RWORK,LRW,IWORK,LIW,JEX,MF,RPAR,IPAR)
215
216         !Finds dR/dt and dL/dt
217         call DVINDY(t,1,RWORK(21),index*2,part_der,IFLAG)
```

```
218
219         !Subroutine which removes too small classes, sorts the vectors and calculates new
        C_m for the timestep test
220         call sort_calc(index, part(1:(index*2-1):2), part(2:(index*2):2), N(1:index), &
221             C_e(1:index), C_s(1:index), index_temp, C_m_temp, C_0, xi, C_p, C_eq,     &
222             temp, gamma_s, gamma_e, V, gamma_stop, r_stop, r_start, gamma_funct)
223
224         !Returns a logical statement if C_e/C_s is within legal limits
225         C_e_test = all(0 < C_e(1:index_temp) < 1,1)
226         C_s_test = all(0 < C_s(1:index_temp) < 1,1)
227
228         !Checks that no classes has dissolved into a particle with negative values
229         negative_check = all(0 < part(1:index),1)
230
231       else
232
233         !Reset part-vector if dt was too big
234         part(1:(i*2-1):2) = r(1:i)
235         part(2:(i*2):2) = l(1:i)
236         RPAR(16:15+i*2-1:2) = r(1:i)
237         RPAR(17:15+i*2:2) = l(1:i)
238         IPAR(2) = i
239
240         !Precipitate volume fraction
241         f_p = pi*sum(N(1:i)*(part(1:(i*2-1):2)**2)*part(2:(i*2):2))
242
243         !Mass balance equation
244         C_m = (C_0-xi*f_p*C_p)/(1-xi*f_p)
245         RPAR(2) = C_m
246
247         r_crit_nucl(i) = (2.0*gamma_s*V)/(log(C_m/C_eq)*C_p*k*Temp)
248
249         ISTATE = 1  !ISTATE needs to be reset to 1 since number of equations increases/
        decreases
250
251         !DVODE calculate growth in R and L direction
252         call DVODE(precipitation,i*2,part(1:i*2),t,t_out,ITOL,RTOL,ATOL,ITASK,   &
253             ISTATE, IOPT,RWORK,LRW,IWORK,LIW,JEX,MF,RPAR,IPAR)
254
255         !Finds dR/dt and dL/dt
256         call DVINDY(t,1,RWORK(21),i*2,part_der,IFLAG)
```

```
257
258         !Subroutine which removes too small classes, sorts the vectors and calculates new
        C_m for the timestep test
259         call sort_calc(i, part(1:(i*2-1):2), part(2:(i*2):2), N(1:i), C_e(1:i), C_s(1:i),&
260             index_temp, C_m_temp, C_0, xi, C_p, C_eq, temp, gamma_s, gamma_e, V,        &
261             gamma_stop, r_stop, r_start, gamma_funct)
262
263         !Returns a logical statement if C_e/C_s is within legal limits
264         C_e_test = all(0 < C_e(1:index_temp) < C_p,1)
265         C_s_test = all(0 < C_s(1:index_temp) < C_p,1)
266
267         !Checks that no classes has dissolved into a particle with negative values
268         negative_check = all(0 < part(1:i),1)
269
270       end if
271
272
273       !Calculate difference between old and new r_crit
274       r_check = abs(1-(r_crit_nucl(i)/((2.0*gamma_s*V)/    &
275               (log(C_m_temp/C_eq)*C_p*k*Temp))))
276
277
278       !Checking if all solute conc. are within accepted regions and that r* does not vary more
        than 1%
279       if (C_e_test == .true. .and. C_s_test == .true. .and. negative_check == .true. &
280         .and. 0 < C_m_temp < 1 .and. r_check < 0.01) then
281
282         !Timestep was succesful and program exits while
283         check = 0
284
285       else
286
287         !If infinite loop print out values
288         if (step > 500) then
289
290           open(100,file='GrowthErrorR.dat')
291           open(101,file='GrowthErrorL.dat')
292           open(102,file='GrowthErrorN.dat')
293           open(103,file='GrowthErrorRateR.dat')
294           open(104,file='GrowthErrorR_crit.dat')
295           open(105,file='GrowthErrorTime.dat')
```

```
296             open(106,file='GrowthErrorN_tot.dat')
297             open(107,file='GrowthErrorC_e.dat')
298             open(108,file='GrowthErrorC_s.dat')
299             open(109,file='GrowthErrorRateL.dat')
300             open(110,file='GrowthErrorN_Rate.dat')
301             open(111,file='GrowthErrorC_m.dat')
302             open(112,file='GrowthErrorR_crit2.dat')
303
304             do j = 1,i
305                 write(100,'(ES24.17)') r(j)
306                 write(101,'(ES24.17)') l(j)
307                 write(102,'(ES24.17)') N(j)
308                 write(103,'(ES24.17)') part_der(j*2-1)
309                 write(104,'(ES24.17)') r_crit(j)
310                 write(105,'(ES24.17)') t_save(j)
311                 write(106,'(ES24.17)') N_tot(j)
312                 write(107,'(ES24.17)') C_e(j)
313                 write(108,'(ES24.17)') C_s(j)
314                 write(109,'(ES24.17)') part_der(j*2)
315                 write(110,'(ES24.17)') N_rate(j)
316                 write(111,'(ES24.17)') C_m_save(j)
317                 write(112,'(ES24.17)') r_crit_nucl(j)
318             end do
319
320             print *, i
321             check = 0
322
323         else
324
325             !Timestep too big. Reset time and assign new timestep
326             check =  1
327             t =  t_res
328             t_out = t + (t0*q**(i-1))/step
329             step = step + 2
330
331             !Have to reset C_m since dt was to big
332             C_m = C_m_res
333             RPAR(2) = C_m
334
335         end if
336
```

```
337         end if
338
339     end do
340
341
342     !Reset check for while loop
343     check = 1
344
345
346     !Saving t and C_m for printing to output
347     t_save(i) = t
348     C_m_save(i) = C_m
349
350
351     !Updating r- and l-vector
352     if (index /= 0 .and. index < i) then
353
354         r(1:index) = part(1:(index*2-1):2)
355         l(1:index) = part(2:(index*2):2)
356
357     else
358
359         r(1:i) = part(1:(i*2-1):2)
360         l(1:i) = part(2:(i*2):2)
361
362     end if
363
364
365     if (asp >= 1) then
366
367         !Checking if classes has dissapeared
368         if (r_crit_nucl(i)*1.05 > 0.6D-9) then
369
370             !When r* is larger than a threshold value the removing requirment shift
371             where (l<0.6D-9 .or. r<0.6D-9)
372                 N = 0
373                 r = 0
374                 l = 0
375             end where
376
377         else
```

```
378
379         !When r* is small at the nucleation phase, most likely no particles dissapears in this
        phase
380         where (l<r_crit_nucl(i)/2 .or. r<r_crit_nucl(i)/2)
381             N = 0
382             r = 0
383             l = 0
384         end where
385
386       end if
387
388     else
389
390       !Checking if classes has dissapeared
391       if (2*asp*r_crit_nucl(i) > 0.6D-9) then
392
393         !When l* is larger than a threshold value the removing requirment shift
394         where (l<0.6D-9 .or. r<0.6D-9)
395             N = 0
396             r = 0
397             l = 0
398         end where
399
400       else
401
402         !When l* is small at the nucleation phase, most likely no particles dissapears in this
        phase
403         where (l<asp*r_crit_nucl(i) .or. r<asp*r_crit_nucl(i))
404             N = 0
405             r = 0
406             l = 0
407         end where
408
409       end if
410
411     end if
412
413
414     !Sorting algorithm to move all zeroes(removed classes) to end of array
415     count = 1
416
```

```
417   do a = 1,size(N)
418       if (N(a) /= 0) then
419
420           N(count) = N(a)
421           r(count) = r(a)
422           l(count) = l(a)
423           count = count + 1
424
425       end if
426   end do
427
428   do while (count <= size(N))
429       N(count) = 0
430       r(count) = 0
431       l(count) = 0
432       count = count + 1
433   end do
434
435
436   !Finding index of first zero in N
437   index = findloc(N, VALUE = 0., DIM = 1) - 1
438
439   !Checking if the aspect ratio of any of the current classes are outside the accepted region
440   do m = 1,i
441
442       if (r(i)/=0 .and. l(i)/=0) then
443
444           asp_check = l(i)/(2*r(i))
445           if (asp_check > 100 .or. asp_check < 0.01)  STOP '*** ASPECT RATIO IS
      OUTSIDE ALLOWED REGION ***'
446
447       end if
448
449   end do
450
451   !Updating concentration in matrix
452   if (index /= 0 .and. index < i) then
453
454       f_p = pi*sum(N(1:index)*(r(1:index)**2)*l(1:index))      !Precipitate volume fraction
455       C_m = (C_0-xi*f_p*C_p)/(1-xi*f_p)
456
```

```
457     else
458
459         f_p = pi*sum(N(1:i)*(r(1:i)**2)*l(1:i))      !Precipitate volume fraction
460         C_m = (C_0-xi*f_p*C_p)/(1-xi*f_p)
461
462     end if
463
464     !Save for output
465     f_p_save(i) = f_p
466
467     !Updating total number of prarticles
468     N_tot(i+1) = sum(N)
469
470     !Turns off nucleation if classes is starting to dissapear
471     if (index < i) then
472         N_check = 1
473     end if
474
475     !Saving mean radius/length of particles
476     mean_r(i) = sum(r*N)/sum(N)
477     mean_l(i) = sum(l*N)/sum(N)
478
479     !Prints out values every 100th iteration
480     if ((i/100)*100 == i) then
481         print *, i, t, f_p, N_tot(i)
482
483         !Defining strings for output names
484         str_g = 'Growth'
485         str_r = 'r.dat'
486         str_l = 'l.dat'
487         str_N = 'N.dat'
488         str_rate_r = 'RateR.dat'
489         str_rate_l = 'RateL.dat'
490         str_r_crit = 'r_crit.dat'
491         str_t = 'time.dat'
492         str_dist_r = 'r_dist.dat'
493         str_dist_l = 'l_dist.dat'
494         str_dist_2d = 'dist_2D.dat'
495
496         write(str_i,'(i0)') i
497
```

```
498     !Opening output files
499     open(b,file=trim(str_g)//trim(str_i)//trim(str_r))
500     open(b+1,file=trim(str_g)//trim(str_i)//trim(str_l))
501     open(b+2,file=trim(str_g)//trim(str_i)//trim(str_N))
502     open(b+3,file=trim(str_g)//trim(str_i)//trim(str_rate_r))
503     open(b+4,file=trim(str_g)//trim(str_i)//trim(str_rate_l))
504     open(b+5,file=trim(str_g)//trim(str_i)//trim(str_r_crit))
505     open(b+6,file=trim(str_g)//trim(str_i)//trim(str_t))
506     open(b+7,file=trim(str_g)//trim(str_i)//trim(str_dist_r))
507     open(b+8,file=trim(str_g)//trim(str_i)//trim(str_dist_l))
508     open(b+9,file=trim(str_g)//trim(str_i)//trim(str_dist_2d))
509
510     !Writing to files if classes has dissapeard
511     if (index /= 0 .and. index < i) then
512
513         !Calculating kde for PSD
514         call kde(index, r(1:index), mean_r(i), l(1:index), mean_l(i), N(1:index), N_tot(i),  &
515             phi, phi_r, phi_l, r_dist, l_dist)
516
517         do j = 1,index
518
519             write(b,'(ES24.17)') r(j)
520             write(b+1,'(ES24.17)') l(j)
521             write(b+2,'(ES24.17)') N(j)
522             write(b+3,'(ES24.17)') part_der(j*2-1)
523             write(b+4,'(ES24.17)') part_der(j*2)
524
525         end do
526
527     !Writing to files when no classes has dissapeared
528     else
529
530         !Calculating kde for PSD
531         call kde(i, r(1:i), mean_r(i), l(1:i), mean_l(i), N(1:i), N_tot(i), phi, phi_r,       &
532             phi_l, r_dist, l_dist)
533
534         do j = 1,i
535
536             write(b,'(ES24.17)') r(j)
537             write(b+1,'(ES24.17)') l(j)
538             write(b+2,'(ES24.17)') N(j)
```

```
539             write(b+3,'(ES24.17)') part_der(j*2-1)
540             write(b+4,'(ES24.17)') part_der(j*2)
541
542         end do
543
544     end if
545
546     do j = 1,i
547
548         write(b+5,'(ES24.17)') r_crit_nucl(j)
549         write(b+6,'(ES24.17)') t_save(j)
550
551     end do
552
553     !Writing 1D and 2D PSD to file
554      do j = 1,idist
555
556             write(b+7,'(ES28.17E3)', ADVANCE="NO") phi_r(j)
557             write(b+7,'(ES28.17E3)', ADVANCE="NO") r_dist(j)
558             write(b+8,'(ES28.17E3)', ADVANCE="NO") phi_l(j)
559             write(b+8,'(ES28.17E3)', ADVANCE="NO") l_dist(j)
560
561             do m = 1,idist
562                 write(b+9,'(ES28.17E3)') phi(m,j)
563             end do
564
565             write(b+7,*)
566             write(b+8,*)
567
568      end do
569
570     !counting variable for output files
571     b = b + 10
572
573   end if
574
575 end do
576
577 !Opening and writing to files in vector-format
578 open(14,file='R_final.dat')
579 open(15,file='L_final.dat')
```

```
580 open(16,file='t_final.dat')
581 open(17,file='N_final.dat')
582 open(18,file='N_rate_final.dat')
583 open(19,file='r_crit_final.dat')
584 open(20,file='N_tot_final.dat')
585 open(21,file='GrowthRateR_final.dat')
586 open(22,file='GrowthRateL_final.dat')
587 open(23,file='Cm_final.dat')
588 open(24,file='mean_r_final.dat')
589 open(26,file='mean_l_final.dat')
590 open(27,file='f_p_final.dat')
591
592 if (index /= 0 .and. index < i) then
593
594     do j = 1,index
595
596         write(14,'(ES24.17)') r(j)
597         write(15,'(ES24.17)') l(j)
598         write(17,'(ES24.17)') N(j)
599         write(21,'(ES24.17)') part_der(j*2-1)
600         write(22,'(ES24.17)') part_der(j*2)
601
602     end do
603
604 else
605
606     do j = 1,i
607
608         write(14,'(ES24.17)') r(j)
609         write(15,'(ES24.17)') l(j)
610         write(17,'(ES24.17)') N(j)
611         write(21,'(ES24.17)') part_der(j*2-1)
612         write(22,'(ES24.17)') part_der(j*2)
613
614     end do
615 end if
616
617 do j = 1,i
618
619     write(16,'(ES24.17)') t_save(j)
620     write(18,'(ES24.17)') N_rate(j)
```

```
621     write(19,'(ES24.17)') r_crit_nucl(j)
622     write(20,'(ES24.17)') N_tot(j)
623     write(23,'(ES24.17)') C_m_save(j)
624     write(24,'(ES24.17)') mean_r(j)
625     write(26,'(ES24.17)') mean_l(j)
626     write(27,'(ES24.17)') f_p_save(j)
627
628 end do
629
630 !Prints out input data to the model
631 RPAR(14) = i
632 open(25,file="input_precipitation.dat")
633 do j = 1,14
634
635     write(25,'(ES24.17)') RPAR(j)
636
637 end do
638
639 print *, t, C_m, C_eq, ISTATE
640
641 end program Precipitation_model
642 !-----------------------------------------------------------
643
644 !Growth subroutine used in DVODE-solver
645 subroutine precipitation(neqn, t, y, ydot, RPAR, IPAR)
646 !Declaration of variables
647 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
648 integer, intent(in) :: neqn, IPAR(2)
649 real (kind=ikind), intent(in) :: t, RPAR(17+neqn)
650 real (kind=ikind), intent(in) :: y(neqn)
651 real (kind=ikind), intent(inout) :: ydot(neqn)
652 !Local Variables
653 real(kind=ikind), parameter :: k = 1.380648813D-23, Rg = 8.314462175     !Boltzmann
        constant [J/K] and universal gas constant [J/K mol]
654 real (kind=ikind), parameter :: pi = 3.14159265358979323846264 33
655 integer :: i, j, gamma_funct
656
657 real (kind=ikind) :: D, C_m, C_e, C_s, C_p, I_end_side0, I_end_end0, I_side_side0, xi,    &
658                      I_side_end0, j0, A0, temp, C_eq, Q_d, gamma_s, gamma_e, r, l, asp,   &
659                      r_stop, gamma_stop, d_gamma
660
```

```
661 !Input from RPAR and IPAR
662 j = IPAR(2)
663
664 D = RPAR(1)
665 C_m = RPAR(2)
666 V = RPAR(3)
667 xi = RPAR(4)
668 C_p = RPAR(5)
669 r_start = RPAR(6)
670 A0 = RPAR(7)
671 gamma_funct = RPAR(8)
672 temp = RPAR(9)
673 C_eq = RPAR(10)
674 Q_d = RPAR(11)
675 gamma_s = RPAR(12)
676 gamma_e = RPAR(13)
677 gamma_stop = RPAR(14)
678 r_stop = RPAR(15)
679
680
681 !Calculates dr/dt and dl/dt
682 do i = 1,j
683
684    r = RPAR(15+i*2-1)
685    l = RPAR(15+i*2)
686
687    !New gamma_e to be calculated if non-constant
688    if (gamma_funct == 1) then
689
690       gamma_e = gamma_s+0.5*(gamma_stop-gamma_s)*(1+tanh(((gamma_stop-gamma_s)*
          &
691             (r-0.5*r_stop-0.5*r_start))/(0.3*(r_stop-r_start))))
692
693       d_gamma =-(1.66667*(gamma_s - gamma_stop)**2*  &
694             (1/(cosh((3.33333*(-gamma_s + gamma_stop)    &
695            *(-0.5*r_start - 0.5*r_stop + r))/      &
696             (-r_start + r_stop))**2)))/(r_start - r_stop)
697
698    end if
699
700    asp = l/(2*r)
```

```
701
702   if (1 <= asp < 100) then
703
704       !Subroutines which calculate numerically found flux from a fitted equation
705       call I_end_end0fit(asp,I_end_end0)
706       call I_end_side0fit(asp,I_end_side0)
707       call I_side_end0fit(asp,I_side_end0)
708       call I_side_side0fit(asp,I_side_side0)
709
710   elseif (0.01 < asp < 1) then
711
712       !Subroutines which calculate numerically found flux from a fitted equation
713       call I_disc_end_end0fit(asp,I_end_end0)
714       call I_disc_end_side0fit(asp,I_end_side0)
715       call I_disc_side_end0fit(asp,I_side_end0)
716       call I_disc_side_side0fit(asp,I_side_side0)
717
718   end if
719
720
721   if (r>0 .AND. l>0) then
722
723       C_e = C_eq*exp((2*gamma_s*V)/(r*C_p*k*temp)) !Gibbs-Thomson effect interfacial
      concentration at ends
724
725       if (gamma_funct == 1) then
726
727           C_s = C_eq*exp(((gamma_s+2*(r/l)*gamma_e+((r**2)/l)*d_gamma)*V)  &
728                /(r*C_p*k*temp)) !Gibbs-Thomson effect interfacial concentration at side
729
730       else
731
732           C_s = C_eq*exp(((gamma_s+2*(r/l)*gamma_e)*V)/(r*C_p*k*temp))  !Gibbs-
      Thomson effect interfacial concentration at side
733
734       end if
735
736   end if
737
738
739   ydot(i*2-1) = (D*(C_s-C_m)*I_side_end0 + D*(C_e-C_m)*I_side_side0)/  &
```

```
740                 (2*pi*l*(xi*C_p-C_s)) !dR/dt
741
742     ydot(i*2) = (D*(C_e-C_m)*I_end_side0 + D*(C_s-C_m)*I_end_end0)/   &
743                 ((pi*r)*(xi*C_p-C_e)) !dL/dt
744
745
746 end do
747
748 end subroutine precipitation
749 !-----------------------------------------------------------------
750
751 !Removes too small classes, sorts the vectors and calculates new C_m
752 subroutine sort_calc(NEQ, r, l, N, C_e, C_s, index, C_m, C_0, xi, C_p, C_eq, temp, gamma_s, &
753           gamma_e, V, gamma_stop, r_stop, r_start, gamma_funct)
754
755 !Declaration of variables
756 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
757 integer, intent(in) :: NEQ, gamma_funct
758 real (kind=ikind), intent(in) :: r(NEQ), l(NEQ), N(NEQ)
759 real (kind=ikind), intent(in) :: C_0, xi, C_p, C_eq, temp, gamma_s, gamma_e, V,   &
760                                  gamma_stop, r_stop, r_start
761 real (kind=ikind), intent(out) :: C_e(NEQ), C_s(NEQ)
762 real (kind=ikind), intent(out) :: C_m
763 integer, intent(out) :: index
764
765 !Local Variables
766 real(kind=ikind), parameter :: k = 1.380648813D-23      !Boltzmann constant [J/K]
767 real (kind=ikind), parameter :: pi = 3.14159265358979323846264433
768 real (kind=ikind) :: r_temp(NEQ), l_temp(NEQ), N_temp(NEQ), f_p, d_gamma,     &
769                      gamma_e_r(NEQ)
770 integer :: a, count, allstat, i
771
772 r_temp = r
773 l_temp = l
774 N_temp = N
775
776 !Checking if classes has dissapeared
777 where (l_temp<1.5D-10 .or. r_temp<1.5D-10)
778     N_temp = 0
779     r_temp = 0
780     l_temp = 0
```

```
781 end where
782
783 !Sorting algorithm to move all zeros to end of array
784 count = 1
785
786 do a = 1,size(N_temp)
787    if (N_temp(a) /= 0) then
788
789       N_temp(count) = N_temp(a)
790       r_temp(count) = r_temp(a)
791       l_temp(count) = l_temp(a)
792       count = count + 1
793
794    end if
795 end do
796
797 do while (count <= size(N_temp))
798    N_temp(count) = 0
799    r_temp(count) = 0
800    l_temp(count) = 0
801    count = count + 1
802 end do
803
804 index = findloc(N_temp, VALUE = 0., DIM = 1) - 1
805
806 if (index > 1 .and. index < NEQ) then
807
808    !Calculate conc. at interfaces to check that they are within legal limits
809    C_e(1:index) = C_eq*exp((2*gamma_s*V)/(r_temp(1:index)*C_p*k*temp)) !Gibbs-
       Thomson effect interfacial concentration at ends
810
811    do i = 1,index
812
813       if (gamma_funct == 1) then
814
815          !Gamma_end(R)
816          gamma_e_r = gamma_s + 0.5*(gamma_stop-gamma_s)*  &
817          (1+tanh(((gamma_stop-gamma_s)*                   &
818          (r_temp(i)-0.5*r_stop-0.5*r_start))/             &
819          (0.3*(r_stop-r_start))))
820
```

```
821             !dgamma_end/dR
822             d_gamma = -(1.66667*(gamma_s - gamma_stop)**2*   &
823             (1/(cosh((3.33333*(-gamma_s + gamma_stop)*       &
824             (-0.5*r_start - 0.5*r_stop + r_temp(i)))/        &
825             (-r_start + r_stop))**2)))/(r_start - r_stop)
826
827             !Gibbs-Thomson effect interfacial concentration at side
828             C_s(i) = C_eq*exp(((gamma_s+2*(r_temp(i)/l_temp(i))*gamma_e_r(i)+ &
829             ((r_temp(i)**2)/l_temp(i))*d_gamma)*V)/(r_temp(i)*C_p*k*temp))
830
831         else
832
833             !Gibbs-Thomson effect interfacial concentration at side
834             C_s(i) = C_eq*exp(((gamma_s+2*(r_temp(i)/l_temp(i))*gamma_e)*V)/  &
835             (r_temp(i)*C_p*k*temp))
836
837         end if
838
839     end do
840
841     !Calculate C_m to check that it is within legal limits
842     f_p = pi*sum(N_temp(1:index)*(r_temp(1:index)**2)*l_temp(1:index))
843     C_m = (C_0-xi*f_p*C_p)/(1-xi*f_p)
844
845 else
846
847     !Calculate conc. at interfaces to check that they are within legal limits
848     C_e(1:NEQ) = C_eq*exp((2*gamma_s*V)/(r_temp(1:NEQ)*C_p*k*temp)) !Gibbs-Thomson
          effect interfacial concentration at ends
849
850     do i = 1,NEQ
851
852         if (gamma_funct == 1) then
853
854             !Gamma_end(R)
855             gamma_e_r = gamma_s + 0.5*(gamma_stop-gamma_s)*  &
856             (1+tanh(((gamma_stop-gamma_s)*                   &
857             (r_temp(i)-0.5*r_stop-0.5*r_start))/             &
858             (0.3*(r_stop-r_start))))
859
860             !dgamma_end/dR
```

```
861             d_gamma = -(1.66667*(gamma_s - gamma_stop)**2*   &
862             (1/(cosh((3.33333*(-gamma_s + gamma_stop)*       &
863             (-0.5*r_start - 0.5*r_stop + r_temp(i)))/        &
864             (-r_start + r_stop))**2)))/(r_start - r_stop)
865
866             !Gibbs-Thomson effect interfacial concentration at side
867             C_s(i) = C_eq*exp(((gamma_s+2*(r_temp(i)/l_temp(i))*gamma_e_r(i)+  &
868             ((r_temp(i)**2)/l_temp(i))*d_gamma)*V)/(r_temp(i)*C_p*k*temp))
869
870         else
871
872             !Gibbs-Thomson effect interfacial concentration at side
873             C_s(i) = C_eq*exp(((gamma_s+2*(r_temp(i)/l_temp(i))*gamma_e)*V)/   &
874             (r_temp(i)*C_p*k*temp))
875
876         end if
877
878     end do
879
880     !Calculate C_m to check that it is within legal limits
881     f_p = pi*sum(N_temp(1:NEQ)*(r_temp(1:NEQ)**2)*l_temp(1:NEQ))     !Precipitate volume
          fraction
882     C_m = (C_0-xi*f_p*C_p)/(1-xi*f_p)
883
884     index = NEQ
885
886 end if
887
888 end subroutine sort_calc
889 !-----------------------------------------------------------
890
891 !Kernel density estimation for both 1D and 2D
892 subroutine kde(index, r, mean_r, l, mean_l, N, N_tot, phi, phi_r, phi_l, r_dist, l_dist)
893 implicit none
894 !Declaration of variables
895 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
896 integer, parameter :: idist = 600
897 real (kind=ikind), parameter :: pi = 3.14159265358979323846264 33
898 integer, intent(in) :: index
899 real (kind=ikind), intent(in) :: r(index), l(index), N(index), mean_r, mean_l, N_tot
900 real (kind=ikind), intent(out) :: phi(idist,idist), phi_r(idist), phi_l(idist), r_dist(idist), l_dist(
```

```
      idist)
!Local variables
real (kind=ikind) :: sigma_r, sigma_l, hr, hl, rmax, lmax
integer :: i, j, k

rmax = maxval(r)
lmax = maxval(l)

!Calculating the std for r and l
sigma_r = sqrt((sum(N*(r-mean_r)**2)/(N_tot-1.)))
sigma_l = sqrt((sum(N*(l-mean_l)**2)/(N_tot-1.)))

!Calculating steps size h for kde
hr = sigma_r*index**(-0.2)
hl = sigma_l*index**(-0.2)

!1D kde for r and l
do i = 1,idist

   r_dist(i) = (rmax+3.*sigma_r)*(i-1)/(idist-1)
   l_dist(i) = (lmax+3.*sigma_l)*(i-1)/(idist-1)

   phi_r(i) = sum((N/(hr*sqrt(2*pi)))*exp(-0.5*((r_dist(i)-r)/hr)**2))
   phi_l(i) = sum((N/(hl*sqrt(2*pi)))*exp(-0.5*((l_dist(i)-l)/hl)**2))

end do

!2D kde for r and l
do j = 1,idist

   do k = 1,idist

      phi(k,j) = sum((N/(hr*hl*2*pi))*exp(-0.5*((l_dist(k)-l)/hl)**2-  &
            0.5*((r_dist(j)-r)/hr)**2))

   end do

end do


end subroutine kde
```

```
941 !-----------------------------------------------------------
942
943 subroutine I_disc_side_side0fit(y,I_side_side0) !subroutine for finding I_end_end0 values for asp
        < 1
944 implicit none
945 !Declaration of variables
946 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
947 real (kind=ikind), intent(inout) :: y
948 real (kind=ikind), intent(out) :: I_side_side0
949 !Local variables
950 real (kind=ikind) :: p1, p2, p3, p4, p5, q1, q2, q3, q4, x
951
952 !curvefit based on inversed aspect ratio
953 x = 1./y
954
955 !Fitting equation coefficients
956 p1 = -16.7103
957 p2 = -5.6667e+03
958 p3 = -4.4500e+04
959 p4 = 1.5558e+05
960 p5 = 8.0314e+04
961 q1 = 360.2312
962 q2 = 3.5066e+03
963 q3 = -9.9724e+03
964 q4 = -1.0834e+04
965
966 ! NB! only valid for aspect ratio 1 - 1/100
967 I_side_side0 = -(p1*x**4 + p2*x**3 + p3*x**2 + p4*x + p5)/(x**4 + q1*x**3 + q2*x**2 +
        q3*x + q4)
968 end subroutine I_disc_side_side0fit
969 !-----------------------------------------------------------
970
971 subroutine I_disc_side_end0fit(y,I_side_end0) !subroutine for finding I_end_end0 values for asp <
        1
972 implicit none
973 !Declaration of variables
974 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
975 real (kind=ikind), intent(inout) :: y
976 real (kind=ikind), intent(out) :: I_side_end0
977 !Local variables
978 real (kind=ikind) :: p1, p2, p3, q1, x
```

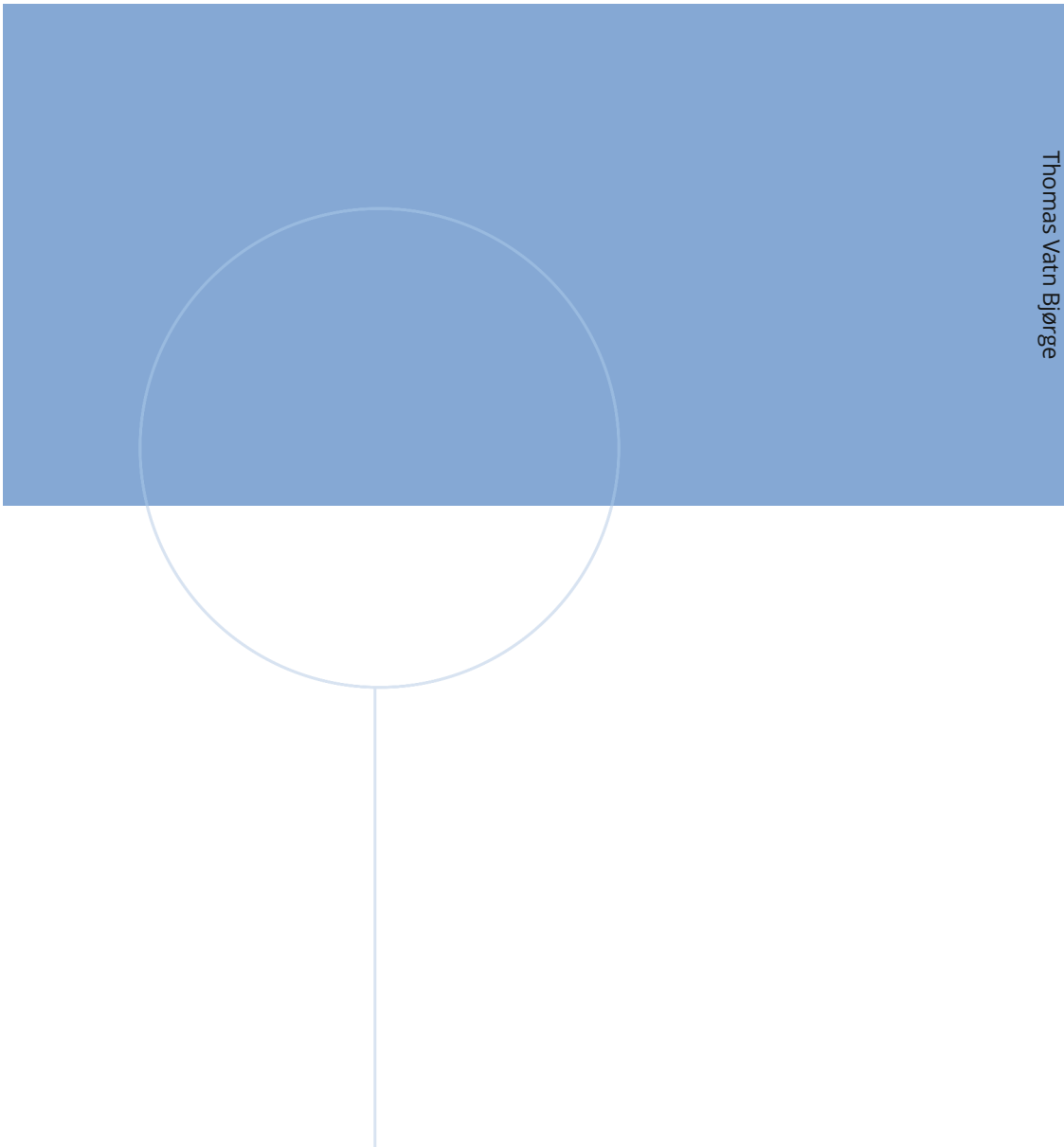
```
979
980 !curvefit based on inversed aspect ratio
981 x = 1./y
982
983 !Fitting equation coefficients
984 p1 = -1.0498e-04
985 p2 = 16.5676
986 p3 = 6.1457
987 q1 = 0.0832
988
989 ! NB! only valid for aspect ratio 1 - 1/100
990 I_side_end0 = -(p1*x**2 + p2*x + p3)/(x + q1)
991 end subroutine I_disc_side_end0fit
992 !-----------------------------------------------------------------
993
994 subroutine I_disc_end_side0fit(y,I_end_side0) !subroutine for finding I_end_end0 values for asp <
        1
995 implicit none
996 !Declaration of variables
997 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
998 real (kind=ikind), intent(in) :: y
999 real (kind=ikind), intent(out) :: I_end_side0
1000 !Local variables
1001 real (kind=ikind) :: p1, p2, p3, p4, p5, p6, q1, q2, q3, q4, x
1002
1003 !curvefit based on inversed aspect ratio
1004 x = 1./y
1005
1006 !Fitting equation coefficients
1007 p1 = 9.4596e-04
1008 p2 = 23.7154
1009 p3 = 580.3917
1010 p4 = -2.6179e+03
1011 p5 = 1.0822e+03
1012 p6 = 4.9366e+03
1013 q1 = 28.5534
1014 q2 = -103.3915
1015 q3 = -64.8390
1016 q4 = 394.2475
1017
1018 ! NB! only valid for aspect ratio 1 - 1/100
```

```
1019 I_end_side0 = -(p1*x**5 + p2*x**4 + p3*x**3 + p4*x**2 + p5*x + p6)/(x**4 + q1*x**3 +
       q2*x**2 + q3*x + q4)
1020 end subroutine I_disc_end_side0fit
1021 !-----------------------------------------------------------------
1022
1023 subroutine I_disc_end_end0fit(y,I_end_end0) !subroutine for finding I_end_end0 values for asp <
       1
1024 implicit none
1025 !Declaration of variables
1026 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
1027 real (kind=ikind), intent(in) :: y
1028 real (kind=ikind), intent(out) :: I_end_end0
1029 !Local variables
1030 real (kind=ikind) :: p1, p2, p3, p4, p5, p6, q1, q2, q3, q4, x
1031
1032 !curvefit based on inversed aspect ratio
1033 x = 1./y
1034
1035 !Fitting equation coefficients
1036 p1 = -3.1704e-04
1037 p2 = -16.1876
1038 p3 = -422.8250
1039 p4 = 1.9161e+03
1040 p5 = -984.1365
1041 p6 = -3.0602e+03
1042 q1 = 29.6572
1043 q2 = -109.8029
1044 q3 = -45.5430
1045 q4 = 360.2118
1046
1047 ! NB! only valid for aspect ratio 1 - 1/100
1048 I_end_end0 = -(p1*x**5 + p2*x**4 + p3*x**3 + p4*x**2 + p5*x + p6)/(x**4 + q1*x**3 +
       q2*x**2 + q3*x + q4)
1049 end subroutine I_disc_end_end0fit
1050 !-----------------------------------------------------------------
1051
1052 subroutine I_end_end0fit(x,I_end_end0) !subroutine for finding I_end_end0 values for asp > 1
1053 implicit none
1054 !Declaration of variables
1055 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
1056 real (kind=ikind), intent(in) :: x
```

```
1057 real (kind=ikind), intent(out) :: I_end_end0
1058 !Local variables
1059 real (kind=ikind) :: p1, p2, p3, p4, p5, q1, q2, q3
1060
1061 !Fitting equation coefficients
1062 p1 = 0.07117
1063 p2 = -27.1158
1064 p3 = -3.7298e+04
1065 p4 = -3.0131e+05
1066 p5 = -1.0421e+05
1067 q1 = 2.9140e+03
1068 q2 = 2.5590e+04
1069 q3 = 1.2121e+04
1070
1071 ! NB! only valid for aspect ratios 1-100
1072 I_end_end0 = -(p1*x**4 + p2*x**3 + p3*x**2 + p4*x + p5)/(x**3 + q1*x**2 + q2*x + q3)
1073 end subroutine I_end_end0fit
1074 !-----------------------------------------------------------------
1075
1076 subroutine I_end_side0fit(x,I_end_side0) !subroutine for finding I_end_side0 values for asp > 1
1077 implicit none
1078 !Declaration of variables
1079 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
1080 real (kind=ikind), intent(in) :: x
1081 real (kind=ikind), intent(out) :: I_end_side0
1082 !Local variables
1083 real (kind=ikind) :: p1, p2, p3, p4, q1, q2
1084
1085 !Fitting equation coefficients
1086 p1 = 4.2618e-05
1087 p2 = 15.9496
1088 p3 = 266.1683
1089 p4 = 17.0092
1090 q1 = 16.6917
1091 q2 = 1.3963
1092
1093 ! NB! only valid for aspect ratios 1-100
1094 I_end_side0 = -(p1*x**3 + p2*x**2 + p3*x + p4)/(x**2 + q1*x + q2)
1095 end subroutine I_end_side0fit
1096 !-----------------------------------------------------------------
1097
```

```
1098 subroutine I_side_side0fit(x,I_side_side0) !subroutine for finding I_side_side0 values for asp > 1
1099 implicit none
1100 !Declaration of variables
1101 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
1102 real (kind=ikind), intent(in) :: x
1103 real (kind=ikind), intent(out) :: I_side_side0
1104 !Local variables
1105 real (kind=ikind) :: p1, p2, p3, p4, p5, q1, q2
1106
1107 !Fitting equation coefficients
1108 p1 = 3.0154e-05
1109 p2 = -0.005534
1110 p3 = -12.8684
1111 p4 = -81.3180
1112 p5 = -20.7693
1113 q1 = 6.9381
1114 q2 = 2.5473
1115
1116 ! NB! only valid for aspect ratios 1-100
1117 I_side_side0 = -(p1*x**4 + p2*x**3 + p3*x**2 + p4*x + p5)/(x**2 + q1*x + q2)
1118 end subroutine I_side_side0fit
1119 !-----------------------------------------------------------------
1120
1121 subroutine I_side_end0fit(x,I_side_end0) !subroutine for finding I_side_end0 values for asp > 1
1122 implicit none
1123 !Declaration of variables
1124 integer, parameter  :: ikind=selected_real_kind(p=15) !15digit precision(6 is default)
1125 real (kind=ikind), intent(in) :: x
1126 real (kind=ikind), intent(out) :: I_side_end0
1127 !Local variables
1128 real (kind=ikind) :: p1, p2, p3, p4, p5, p6, q1, q2
1129
1130 !Fitting equation coefficients
1131 p1 = 1.6259e-05
1132 p2 = -0.004801
1133 p3 = 2.5067
1134 p4 = 83.9911
1135 p5 = 379.6569
1136 p6 = 170.6609
1137 q1 = 14.7673
1138 q2 = 14.6144
```

```
1139
1140 ! NB! only valid for aspect ratios 1-100
1141 I_side_end0 = -(p1*x**5 + p2*x**4 + p3*x**3 + p4*x**2 + p5*x + p6)/(x**2 + q1*x + q2)
1142 end subroutine I_side_end0fit
1143 !-----------------------------------------------------------------
1144
1145 !If supplying jacobi directly. Not relevant here, only dummy subroutine
1146 SUBROUTINE JEX (NEQ, T, Y, ML, MU, PD, NRPD, RPAR, IPAR)
1147 DOUBLE PRECISION PD, RPAR, T, Y
1148 DIMENSION Y(NEQ), PD(NRPD,NEQ)
1149
1150 !      PD(1,1) = -.04D0
1151 !      PD(1,2) = 1.D4*Y(3)
1152 !      PD(1,3) = 1.D4*Y(2)
1153 !      PD(2,1) = .04D0
1154 !      PD(2,3) = -PD(1,3)
1155 !      PD(3,2) = 6.D7*Y(2)
1156 !      PD(2,2) = -PD(1,2) - PD(3,2)
1157
1158 end
```



Thomas Vatn Bjørge

NTNU
Norwegian University of
Science and Technology