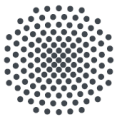

Universität Stuttgart

SimTech

Project Thesis

Analyzing the influence of non-neutral contact angle on the Cahn-Hilliard model

Jonathan Ulmer
Matriculation Number: 3545737

Supervisor: Tufan Ghosh

Department of Hydromechanics and Modelling of Hydrosystems

Completed 19.12.2024

Abstract

This project examines the effect of a simple Neumann boundary condition on a finite difference solver for the Cahn-Hilliard equation. It presents the results of the proposed method on different domains (for a square and a circule). There it is apparent, that the approach is able to affect the contact angle of the interface on the boundary. Additionally, this project gives a rudimentary technique to calculate this angle.

Contents

1 Introduction 1

2 Fundamentals 3
2.1 Notation 4

3 Boundary adaptation 7

4 Numerical solver 11

5 Numerical evaluation 13

6 Numerical evaluation on a circle 17

7 Angle 21
7.1 Circle 21
7.2 Square 22

8 Summary and outlook 25

Bibliography 27

List of Figures

3.1 Visualization of all grid-cells adjacent to the boundary $\partial\Omega_d$ of a square domain . . . 8
3.2 Visualization of all grid-cells adjacent to the boundary $\partial\Omega_d$ of a circular domain . . . 9

5.1 Phase-field ϕ after 100 time-steps with $C = 0$ emulating no-flux boundary. . . . 14
5.2 phase-field ϕ after 100 time steps with $C = 1$ . . . 14
5.3 phase-field ϕ after 100 time-steps with $C = -1$ . . . 15
5.4 phase-field ϕ after 500 time-steps with $C \in \{-1 + \frac{\sqrt{2}}{2}, -0.5, 0.5, 1 - \frac{\sqrt{2}}{2}\}$ 15

6.1 ϕ after 100 time steps on a circular domain with no-flux boundary-conditions after 100 time steps on a circular domain with no-flux . . 18
6.2 phase-field ϕ after 100 time-steps with $C = 1$ . . . 18
6.3 Phase-field ϕ after 100 time-steps with $C = -1$ . . . 19
6.4 Phase-field ϕ after 500 time-steps with $C \in \{-1+\frac{\sqrt{2}}{2}, -0.5, 0.5, 1-\frac{\sqrt{2}}{2}\}$ on a circular domain. . . . 20

7.1 Plot of the boundary and surface normals on a circular domain . . . 22
7.2 value for C and corresponding angle α after 200 time-steps . . . 23
7.3 Plot of the boundary and surface normals on a circular domain . . . 23
7.4 value for θ and corresponding angle α after 200 time-steps . . . 24

1 Introduction

This project thesis builds upon the work in our bachelor thesis, by introducing a simple boundary condition approach to a variation of the solver used therein. In Chapter 2 we introduce the Cahn-Hilliard equation in the formulation that we use for this project. This project used a coupled second order version of this CH equation rather than the usual single 4th order one, to simplify the numerical implementation. The solver itself builds upon a finite difference discretization of this equation. In Chapter 4, together with the discrete domains, on which we run our numerical solver, we introduce a Jacobi iteration to solve the linear system derived from the aforementioned discretization. The numerical solver in this thesis is GPU accelerated, and the discretization we chose to base our solver on is capable to calculate on all domains as long as a characteristic function is given. Therefore, we introduce two domains, on which we present our findings. The primary goal of this work is then the boundary condition approach in Chapter 3. Conceptionally the boundary condition we introduce simply consists of a constant value added in the linear system to all equations corresponding to grid-cells on the boundary. The actual implementation is capable of doing this for arbitrary domains. We present the results of this method on two Domains, a square one in Chapter 5 and a circular domain in Chapter 6 where we show the phase field for different boundary conditions which manifest in a variable contact angle of the interface on the boundary mimicking the behavior of hydrophobic/hydrophilic material. While we are unable to provide explicit formulae in relation to contact, in Chapter 7 we provide numerical insight in this relationship, and a table with precomputed values.

2 Fundamentals

This work concerns itself with boundary conditions on the Cahn-Hilliard equation. The Cahn-Hilliard (CH) equation is a fourth order partial differential equation (PDE) used to describe phase separation in binary mixtures. It models how a mixture of two components (e.g., two liquids or alloys) evolves over time to separate into distinct regions with different concentrations of each component. To achive this it provides a phase-field ϕ which is used for an implicit representation of the interface between both phases. The Cahn-Hilliard equation, in the formulation we use here, is derived from the **Ginzburg-Landau** energy (2.1), an example on how this is done is given by [3].

$$E^{\text{bulk}}[\phi] = \int_{\Omega} \frac{\varepsilon^2}{2} |\nabla \phi|^2 + W(\phi)\, dx, \tag{2.1}$$

There they introduce a chemical potential μ derived as derivative of the **Ginzburg-Landau** energy.

$$\mu = \frac{\delta E_{bulk}(\phi)}{\delta \phi} = -\varepsilon^2 \Delta \phi + W'(\phi), \tag{2.2}$$

Where $W(\phi)$ in the energy, is a double well potential. In our case we orient us at the work of[1], where they use

$$W(\phi) = \frac{(1-\phi^2)^2}{4}. \tag{2.3}$$

the Cahn-Hilliard equation in this thesis is then given as

$$\begin{aligned} \partial_t \phi(x,t) &= \Delta \mu \\ \mu &= -\varepsilon^2 \Delta \phi + W'(\phi). \end{aligned} \tag{2.4}$$

One thing to note is, that this way of writing the CH equation presents a second order, coupled system, rather than the one dimensional fourth order system (2.5) often given.

$$\partial_t \phi(\vec{x}, t) = \Delta(-\varepsilon^2 \Delta\phi + W'(\phi)) \tag{2.5}$$

This choice is deliberate, and aligns with the numerical implementation.

2.1 Notation

This project solves the CH equation on a regular rectangular grid with grid-size h. The computational domain, is therefore discretized as

$$\vec{x}_{ij} := \frac{i}{h} * e_1 + \frac{j}{h} * e_2, \tag{2.6}$$

where $i, j \in [0, ... N]$ and N is chosen arbitrarily, such that the resulting rectangle $[0, Nh] \times [0, Nh]$ acts as bounding box of the domain Ω. For our implementation we use $N = 256$ as it gives a good compromise between resolution and compute time. We denote a discrete version of the domain Ω_d where

$$\Omega_d := \{x_{ij} | x_{ij} \in \Omega\} \tag{2.7}$$

On this discrete domain our solver calculates solutions for discrete fields

$$\phi^n_{ij} : \Omega_d \times \{0, ... \} \to \mathbb{R}\,, \tag{2.8}$$

$$\phi_{ij} := \phi(\vec{x}_{ij}) \qquad \vec{x}_{ij} \in \Omega_d \tag{2.9}$$

$$\mu^n_{ij} : \Omega_d \times \{0, ... \} \to \mathbb{R}\,, \tag{2.10}$$

$$\mu_{ij} := \mu(\vec{x}_{ij}) \tag{2.11}$$

We use the following differential quotients for field f_{ij}:

$$D_x f_{i+\frac{1}{2}j} = \frac{f_{i+1j} - f_{ij}}{h} \qquad D_y f_{ij+\frac{1}{2}} = \frac{f_{ij+1} - f_{ij}}{h} \tag{2.12}$$

And define a discrete gradient as.

$$\nabla_d f_{ij} = (D_x f_{i+1j},\ D_y f_{ij+1}) \tag{2.13}$$

And

$$\Delta_d f_{ij} = \nabla_d \cdot \nabla_d f_{ij} \tag{2.14}$$

See [2] Our solver implements the ansatz proposed by the authors [1].

$$\begin{aligned} \frac{\phi_{ij}^{n+1} - \phi_{ij}^{n}}{\Delta t} &= \nabla_d \cdot (G_{ij} \nabla_d \mu_{ij}^{n+\frac{1}{2}}) \\ \mu_{ij}^{n+\frac{1}{2}} &= 2\phi_{ij}^{n+1} - \varepsilon^2 \nabla_d \cdot (G_{ij} \nabla_d \phi_{ij}^{n+1}) + W'(\phi_{ij}^{n}) - 2\phi_{ij}^{n} \end{aligned} \tag{2.15}$$

This approach provides a semi implicit time discretization where linear terms are evaluated implicitly and the nonlinear double well potential is evaluated explicitly.

3 Boundary adaptation

The solver from [1], that we use as reference guaranties no flux boundary conditions at a discrete level by setting $\nabla\phi_{ij} = 0$ for $\phi_{ij} \in \partial\Omega_d$ this is done by multiplying the gradient $G_{ij}\nabla_d\phi_{ij}$ with the Characteristic function of Ω_d.

$$G_{ij} = \begin{cases} 1\,, x_{ij} \in \Omega \\ 0\,, x_{ij} \notin \Omega \end{cases} \tag{3.1}$$

To accommodate different boundary conditions, we modify $\nabla_d \cdot (G_{ij}\nabla_d\phi_{ij})$ with a constant term C on grid points next to the boundary. To do this, we introduce a boundary field B_{ij} that we add to μ_{ij}. We determine the value of B_{ij} using a central difference scheme on G. We note, that G is implemented as a function, rather than a discrete field.

$$B_{ij} = \max\Big(|G_{i+\frac{1}{2}j} - G_{i-\frac{1}{2}j}|, |G_{ij+\frac{1}{2}} - G_{ij-\frac{1}{2}}|\Big) * C \tag{3.2}$$

For example consider a 32x32 domain with $C = 1$ of the boundary fields $\mathbf{B}$ for a square domain Figure 3.1 and an example on a circular domain in Figure 3.2. In this Project we use the following adaptation of the discretization from [1].

$$\begin{aligned} \frac{\phi_{ij}^{n+1} - \phi_{ij}n}{\Delta t} &= \nabla_d \cdot (G_{ij}\nabla_d\mu_{ij}^{n+\frac{1}{2}}) \\ \mu_{ij}^{n+\frac{1}{2}} &= 2\phi_{ij}^{n+1} - \varepsilon^2\nabla_d \cdot (G_{ij}\nabla_d\phi_{ij}^{n+1}) + B_{ij} + W'(\phi_{ij}^n) - 2\phi_{ij}^n \end{aligned} \tag{3.3}$$

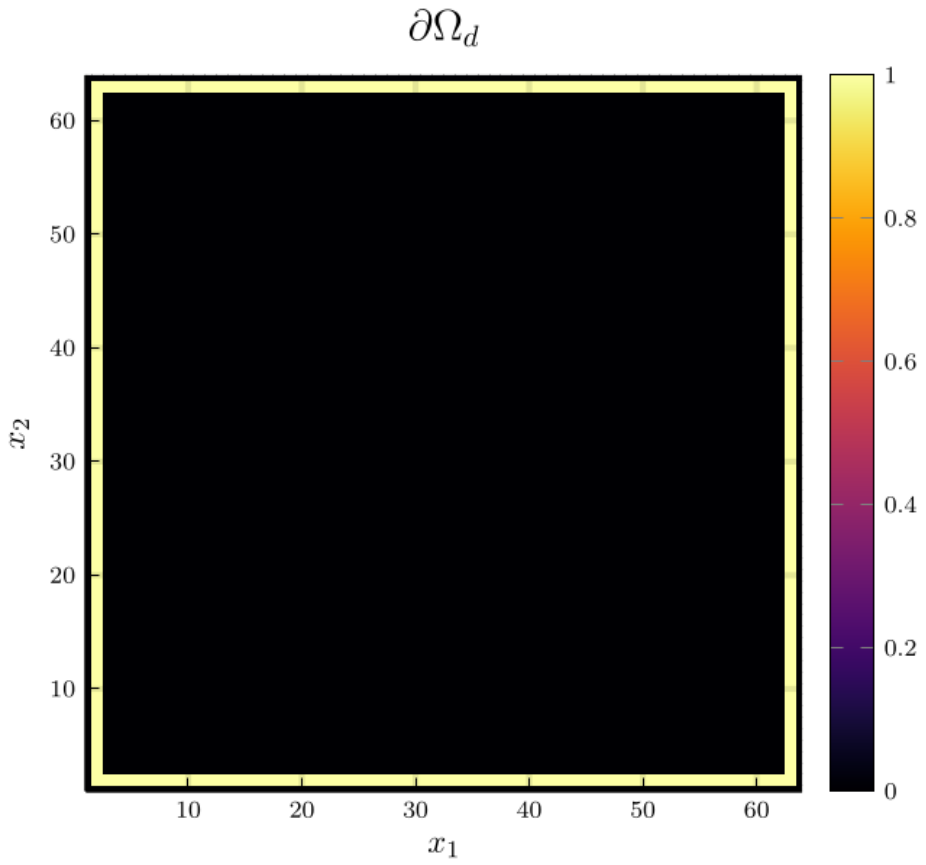

Figure 3.1: Visualization of all grid-cells adjacent to the boundary $\partial\Omega_d$ of a square domain

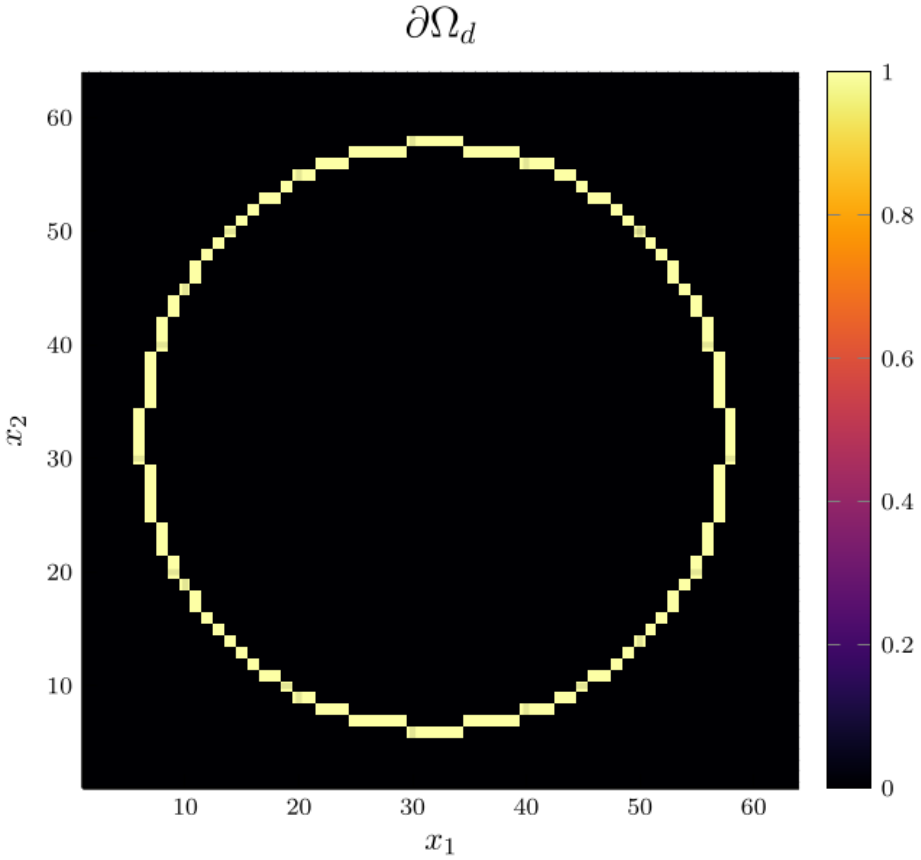

Figure 3.2: Visualization of all grid-cells adjacent to the boundary $\partial\Omega_d$ of a circular domain

4 Numerical solver

Contrary to the solver proposed in [2] we do not use a multi-grid Gauss-Seidel Solver to solve the linear system, instead we use a Jacobi solver, as this will eventually assist in parallelizing the computation. Similar to [2] we linearize (3.3) to

$$\begin{aligned} \frac{\phi_{ij}^{n+1}}{\Delta t} - \nabla_d \cdot (G_{ij} \nabla_d \mu_{ij}^{n+\frac{1}{2}}) &= \frac{\phi_{ij}^{n}}{\Delta t} \\ \mu_{ij}^{n+\frac{1}{2}} - 2\phi_{ij}^{n+1} + \varepsilon^2 \nabla_d \cdot (G_{ij} \nabla_d \phi_{ij}^{n+1}) &= 2\phi_{ij}^{n} - W'(\phi_{ij}^{n}) - B_{ij} \end{aligned} \tag{4.1}$$

One may note, that after rearranging some terms leads to a linear system with a right-hand side which is exclusively dependent on the previous time step. We use Jacobi's method to solve the resulting linear system given above, and the corresponding element wise representation of the same is given in the following.

Provided the mth Jacobi iteration has been computed, the $m+1th$ iteration is computed by solving

$$\begin{aligned} \frac{\phi_{ij}^{n+1,m+1}}{\Delta t} - \nabla_d \cdot (G_{ij} \nabla_d \mu_{ij}^{n+\frac{1}{2},m+\frac{1}{2}}) &= \frac{\phi_{ij}^{n}}{\Delta t} \\ \mu_{ij}^{n+\frac{1}{2},m} - 2\phi_{ij}^{n+1,m} + \varepsilon^2 \nabla_d \cdot (G_{ij} \nabla_d \phi_{ij}^{n+1,m+\frac{1}{2}}) + B_{ij} &= 2\phi_{ij}^{n} - W'(\phi_{ij}^{n}) \end{aligned} \tag{4.2}$$

For $\phi_{ij}^{n+1,m+1}, \mu_{ij}^{n+\frac{1}{2},m+1}$, where $\nabla_d \cdot (G_{ij} \nabla_d \mu_{ij}^{n+\frac{1}{2},m+\frac{1}{2}})$ and $\nabla_d \cdot (G_{ij} \nabla_d \phi_{ij}^{n+1,m+\frac{1}{2}})$ can be computed as follows. We use the results from the previous Jacoby step, m, for values off the center. e.g.

$$\nabla_d \cdot (G_{ij} \nabla_d \phi_{ij}^{n+1,m+\frac{1}{2}}) = \frac{1}{h^2} \sum_{\Delta \in S} G_{ij+\frac{1}{2}\Delta} \phi_{ij+\Delta}^{n+1,m} - \left(\sum_{\Delta \in S} G_{ij+\frac{1}{2}\Delta} \right) \phi_{ij}^{n+1,m+1} \tag{4.3}$$

where $S := \{(0,1),(0,-1),(1,0),(-1,0)\}$. Our implementation is done in julia to transmit the solution for each element in parallel on the GPU. In the following we described the complete implementation of the Jacobi's iteration.

```
@kernel function jacoby!(
    Φ,
    M,
    @Const(Ξ),
    @Const(Ψ),
    @Const(h),
    @Const(ε),
    @Const(Δt),
    @Const(iterations)
)
    I   = @index(Global, Cartesian)
    Id  = oneunit(I)
    Ids = CartesianIndices(M)
    Ix = CartesianIndex(1, 0)
    Iy = CartesianIndex(0, 1)
    if I in (Ids[begin]+Id:Ids[end]-Id)
        g = G(2 * I + Ix, Ids) + G(2 * I + Iy, Ids) + G(2 * I - Ix, Ids) + G(2 * I - Iy,
        ↪  Ids)
        a1 = 1/Δt
        a2 = -1* ε^2/h^2 * g  - 2
        b1 = 1/h^2 * g
        b2 = 1
        for _ = 1:iterations

            Σμ = G(2 * I + Ix, Ids) * M[I+Ix] + G(2 * I + Iy, Ids) * M[I+Iy] + G(2 * I -
            ↪  Ix, Ids) * M[I-Ix] + G(2 * I - Iy, Ids) * M[I-Iy]

            Σϕ = G(2 * I + Ix, Ids) * Φ[I+Ix] + G(2 * I + Iy, Ids) * Φ[I+Iy] +G(2 * I -
            ↪  Ix, Ids) * Φ[I-Ix] +G(2 * I - Iy, Ids) * Φ[I-Iy]

            c1 = Ξ[I] + 1/h^2   * Σμ
            c2 = Ψ[I] - ε^2/h^2 * Σϕ

            # stupid matrix solve
            @inline Φ[I] = (c1*b2 - c2*b1) / (a1*b2 - a2*b1)
            @inline M[I] = (a1*c2 - a2*c1) / (a1*b2 - a2*b1)
            #
            @synchronize()
        end

    end
end
```

5 Numerical evaluation

the numerical investigations in this capter and the following are done with the following hyperparameters:

parameter	ε	h	Δt
value	$8 * 10^{-5}$	$1 * 10^{-4}$	$1 * 10^{-4}$

We set constant values for B_{ij} on the boundary to begin with our evaluations. One may note that C = 0 is equivalent to the no-flux condition of the original solver introduced in the Bachelor thesis. Now, as a preliminary verification step we set C = 0 as our first choice. Consequently, for $C = 0$, the interface lies orthogonal on the boundary (see Fig. 5.1), which we expect for a CH solver with no-flux boundary conditions. For $B_{ij} \in \{-1, 1\}$ we observed behavior connected to hydrophobic / hydrophilic substances on the boundary, where $B_{ij} = 1$ resulted in the one phase pearling off the boundary, while the other seemed attracted. These certainly leads to the apparent contact angles of 180° and 0° respectively. Using $B_{ij} = -1$ results in the opposite behavior.

We show, that our solver is stable for values $C \neq 0$. In Fig. 5.2 we employ a constant value of $C = 1$ and observe the phase corresponding to $\phi = 1$ puling away from the boundary. The contact angle between phase 1 and the boundary approaches 180° i.e. the interface runs parallel to the boundary.

In Fig. 5.3 we try the reverse situation, and consequently we observe the corresponding behavior. When using a value of $C = -1$ we observe opposite behavior relative to the case in Fig. 5.2. Where the contact angle on the boundary lies at 0°, the interface runs parallel to the boundary again.

The most interesting behavior are noted for values between $(-1, 1)$, where we observe the contact angle of the interface at the boundary changes from parallel 0° to parallel 180°.

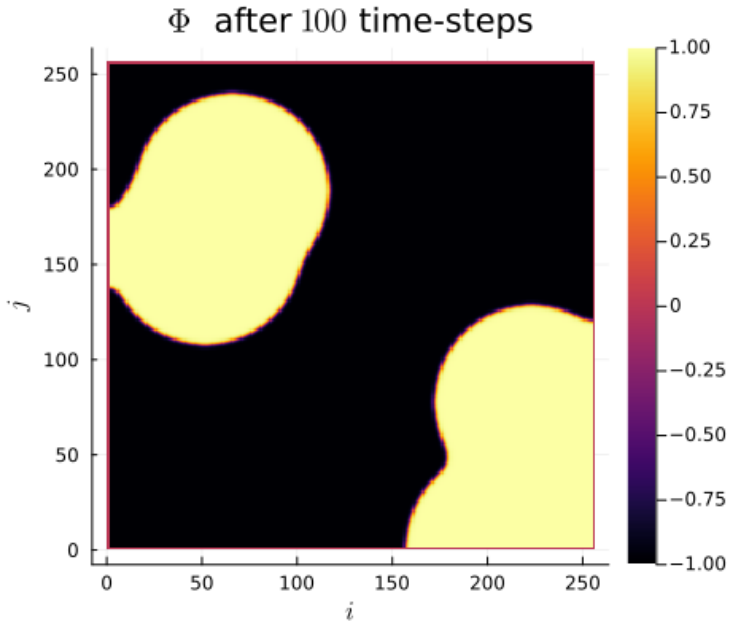

Figure 5.1: Phase-field ϕ after 100 time-steps with $C = 0$ emulating no-flux boundary.

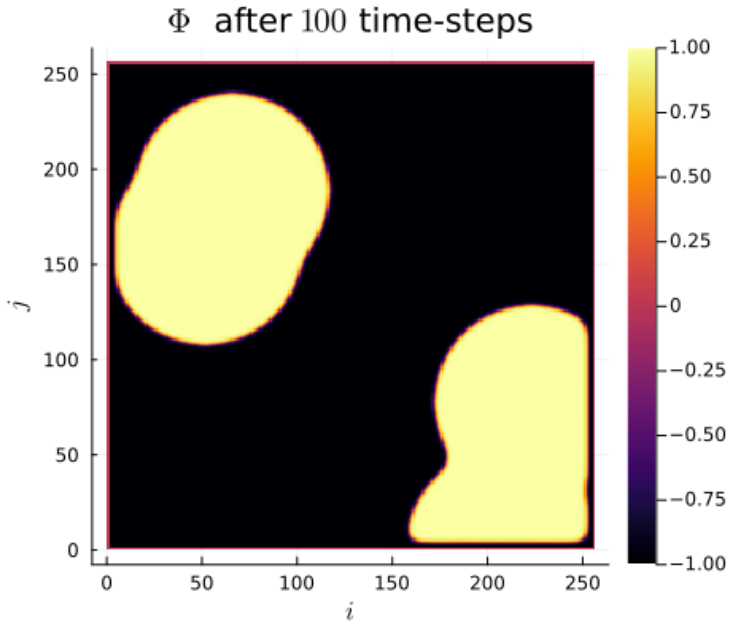

Figure 5.2: phase-field ϕ after 100 time steps with $C = 1$

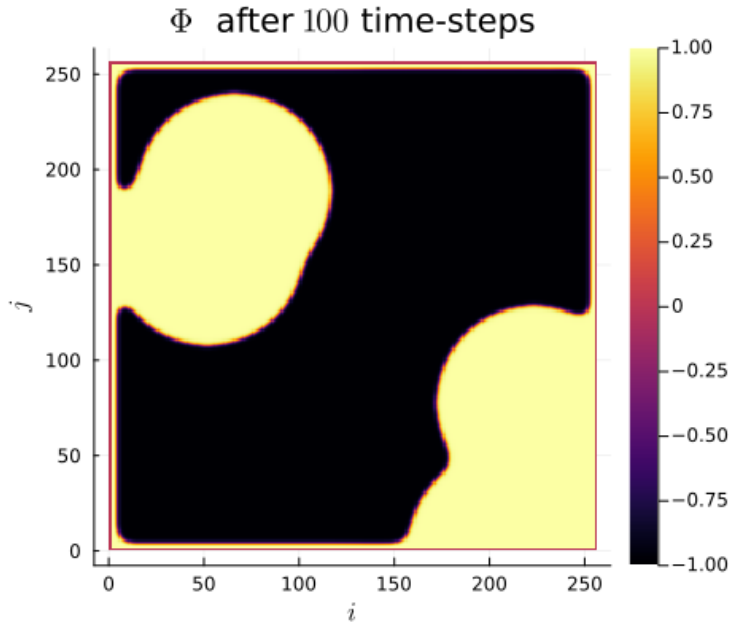

Figure 5.3: phase-field ϕ after 100 time-steps with $C = -1$

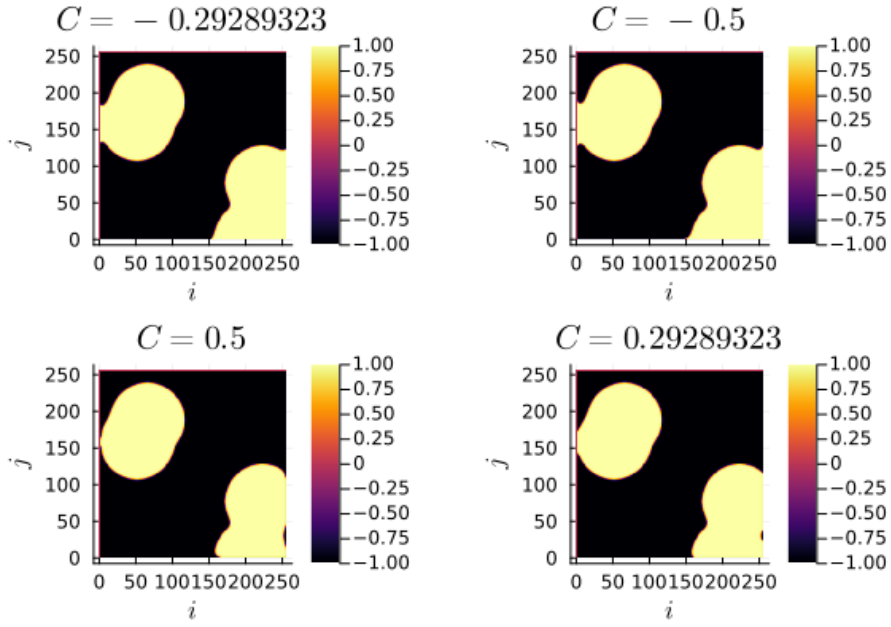

Figure 5.4: phase-field ϕ after 500 time-steps with $C \in \{-1 + \frac{\sqrt{2}}{2}, -0.5, 0.5, 1 - \frac{\sqrt{2}}{2}\}$

6 Numerical evaluation on a circle

The original solver presented in [1] was able to solve the CH equation on arbitrary domains. Since the addition of our boundary function depends solely on the characteristic function of the discrete domain, we are able to use our approach on different domains, by providing a different characteristic function. We present the results of which in this chapter. To show the behavior of the CH solver in Fig. 6.1, we first employ no-flux boundary conditions on a circular domain. We observe the interface perpendicular on the boundary, as we expect.

```
@inline function G(I::CartesianIndex , Ids::CartesianIndices)::Float32
    @inline r  = Ids[end] - I
    m = maximum(Tuple(Ids[end]))
    if norm(Tuple(r)) < 0.8 * m
        return 1.
        end
    return 0.
end

h::Float32 = 3f-4
Δt::Float32 = 1e-4
ε::Float32 = 2e-4
W′(x) = -x * (1 - x^2)
```

```
W′ (generic function with 1 method)
```

The results we observe in Fig. 6.2 are similar to the results on a square domain in Fig. 5.2. The contact angle is 180° i.e. the interface does not touch the boundary and runs parallel to it.

The results for $C = -1$ in Fig. 6.3 on the circular domain, are similar to the results in Fig. 5.3 on the square domain as well, where the interface touches the boundary and runs parallel with a contact angle of 0°.

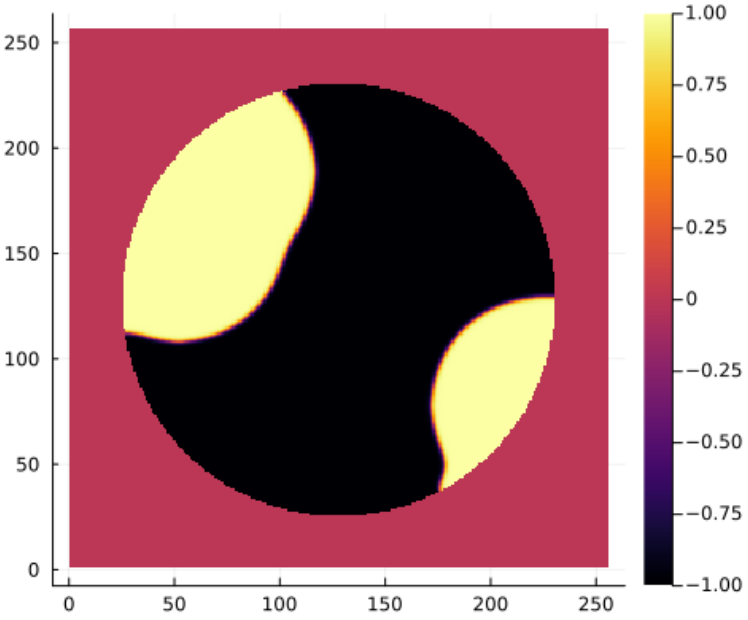

Figure 6.1: ϕ after 100 time steps on a circular domain with no-flux boundary-conditions after 100 time steps on a circular domain with no-flux

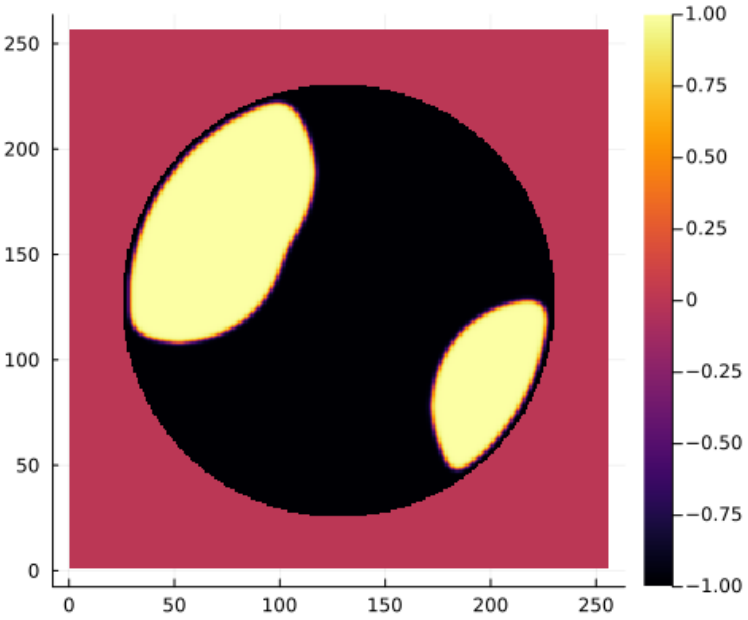

Figure 6.2: phase-field ϕ after 100 time-steps with $C = 1$

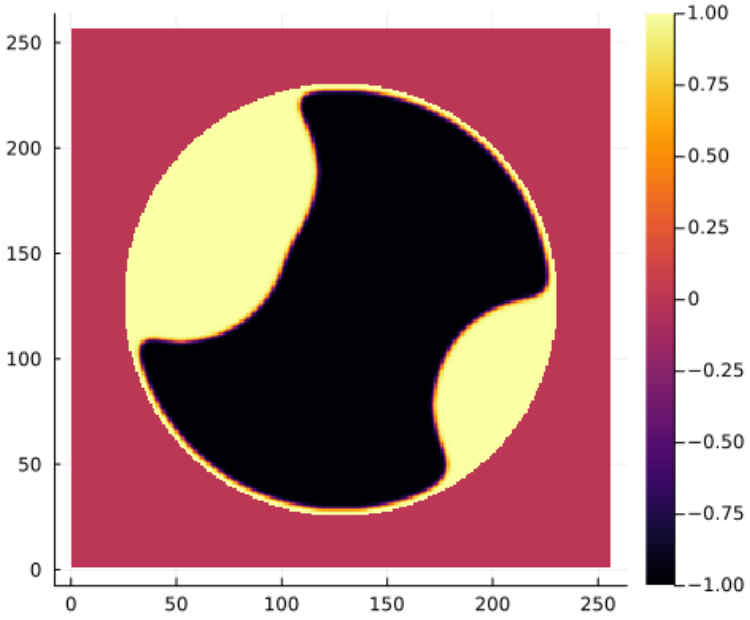

Figure 6.3: Phase-field ϕ after 100 time-steps with $C = -1$

When evaluating intermediate contact angles in Fig. 6.4, the results are similar to the square domain again, however, especially for shallow angles, we observe some artifacts of one phase appearing in places where previously was none. We observe similar behavior on square domains only in the corners, i.e. points where the boundary has high curvature. Note that this effect can be circumvented with different values for ε howeverthis has an effect on the boundary angle, that we did not investigate.

When using random initial phase-fields, the results look the comparable to the square domain, and exhibit the for the CH equation expected behavior, whereas time goes on, the many small parts coalesce into larger parts.

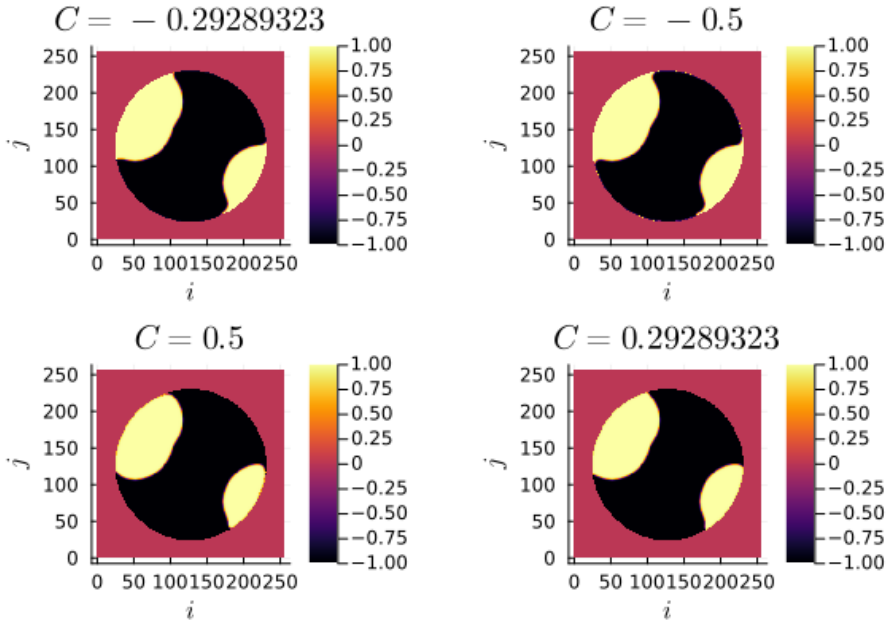

Figure 6.4: Phase-field ϕ after 500 time-steps with $C \in \{-1 + \frac{\sqrt{2}}{2}, -0.5, 0.5, 1 - \frac{\sqrt{2}}{2}\}$ on a circular domain.

7 Angle

We calculate the angle using the folowing hyperparameters

parameter	ε	h	Δt
value	$2 * 10^{-5}$	$1 * 10^{-4}$	$1 * 10^{-4}$

In previous experiments we noted that the angle of the interface changes with different input parameters. While we do not have a mathematical derivation of this relation, we aim to provide numerical insight in this chapter. We calculate this angle using the gradient of the phase-field $\nabla\phi_{ij}$ and the normal of our domains' boundary.

$$\frac{\nabla_d \phi_{ij} \cdot \mathbf{n}_{ij}}{\|\nabla_d \phi_{ij}\|} = \cos(\theta), \qquad \text{where } \phi_{ij} \in \partial\Omega_d \tag{7.1}$$

For a single point $\vec{x}_{ij}$ on the interface and near the boundary. Since we need a finite difference to evaluate (7.1), we do not select a point directly on the boundary and since we need a point on the interface, where $\nabla\phi_{ij}$ is large, we calculate the angle at

$$P_{ij} = arg\max_{\vec{x}_{ij}} \nabla\phi_{ij} \qquad \text{where} \quad \phi_{ij} \in \partial\Omega \tag{7.2}$$

7.1 Circle

The normal of the circular domain in our second example is

$$\mathbf{n}_{ij} := \mathbf{n}(\vec{x}_{ij}) = \frac{\vec{c} - \vec{x}_{ij}}{\|\vec{c} - \vec{x}_{ij}\|} \tag{7.3}$$

Where $\vec{c}$ is the center of the domain. In Fig. 7.1 we present the results of a calculated angle, together with the normals and the point it is calculated from.

The method we use to calculate the interface-boundary angle is not stable under small changes in C as seen in Fig. 7.2 this is unsurprising, since the selection criterion for the point, where we calculate the angle is prone to change as the maximum derivative of the phase-field is approximately the same everywhere on the phase-field. To combat this effect, we chose the point to evaluate at once for all iterations.

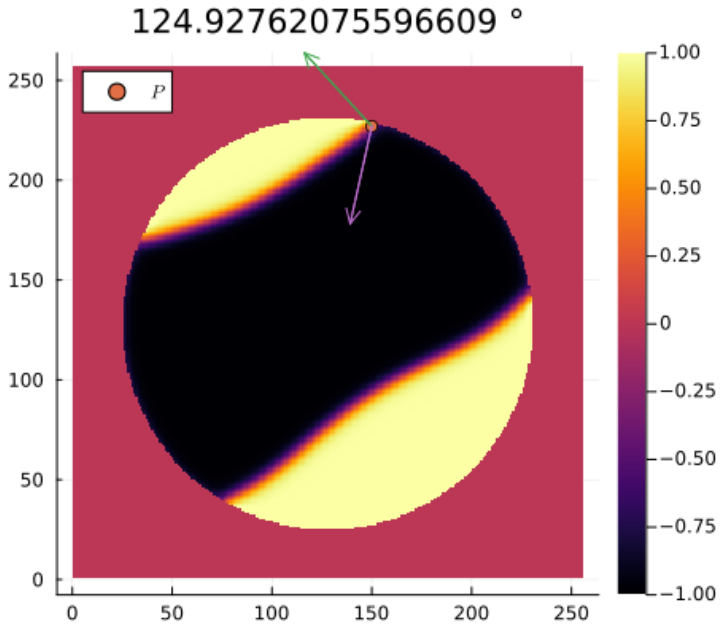

Figure 7.1: Plot of the boundary and surface normals on a circular domain

This however only works if the interface doesn't move from the point selected. The results in Fig. 7.2 shows, that the contact angle is related to the value we use for C. Since the other hyperparameters (ε and h) may have an effect on the angel as well, we cannot give an exact relation.

7.2 Square

The normal vector on a square domain is a little bit more complicated than the normal for the circle. In this case we use the normal

$$\mathbf{n}_{ij} = \mathbf{n}(\vec{x}_{ij}) = \max(\vec{c} - \vec{x}_{ij}) e_{\arg\max_{i,j}(\vec{c}-\vec{x}_{ij})} \tag{7.4}$$

The angle on a square domain is presented in Fig. 7.3

The relation between the contact angle and C, as seen in Fig. 7.4 is about the same as in the circular domain

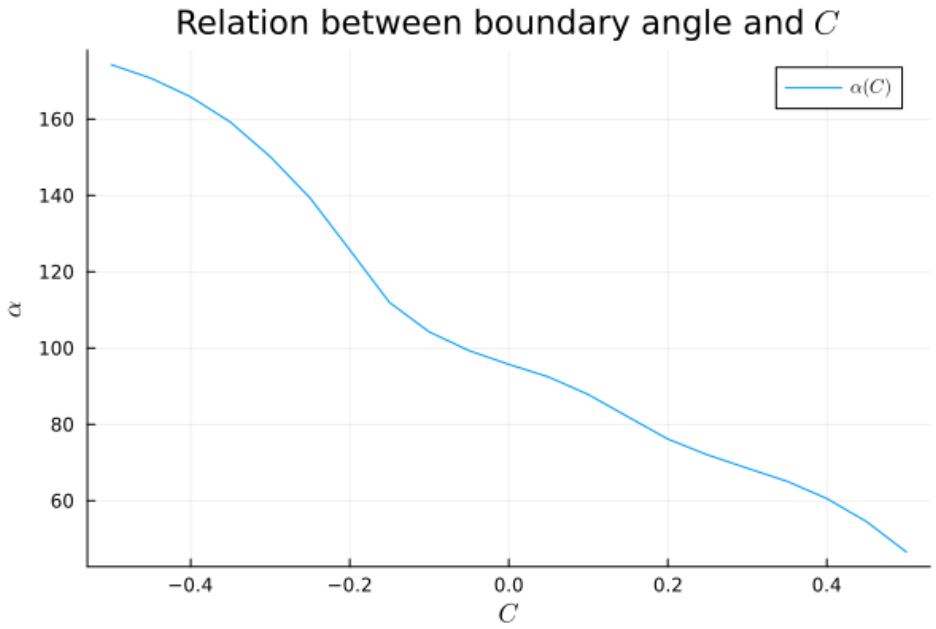

Figure 7.2: value for C and corresponding angle α after 200 time-steps

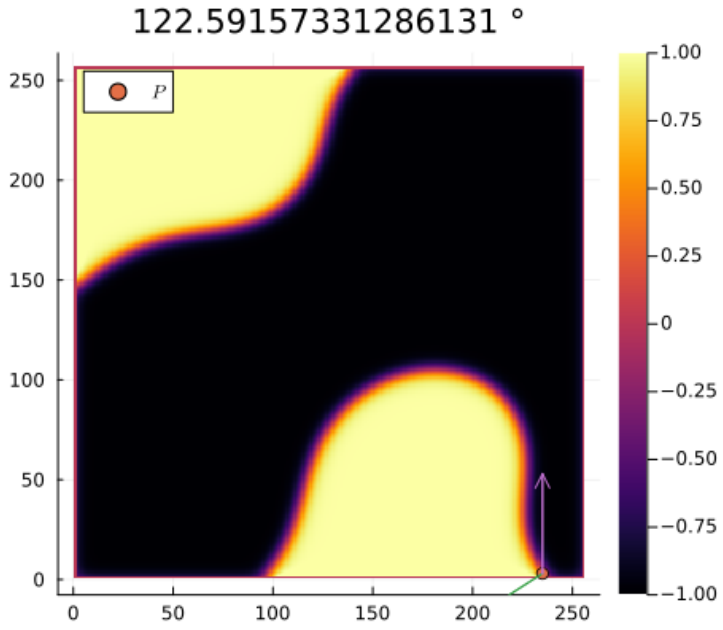

Figure 7.3: Plot of the boundary and surface normals on a circular domain

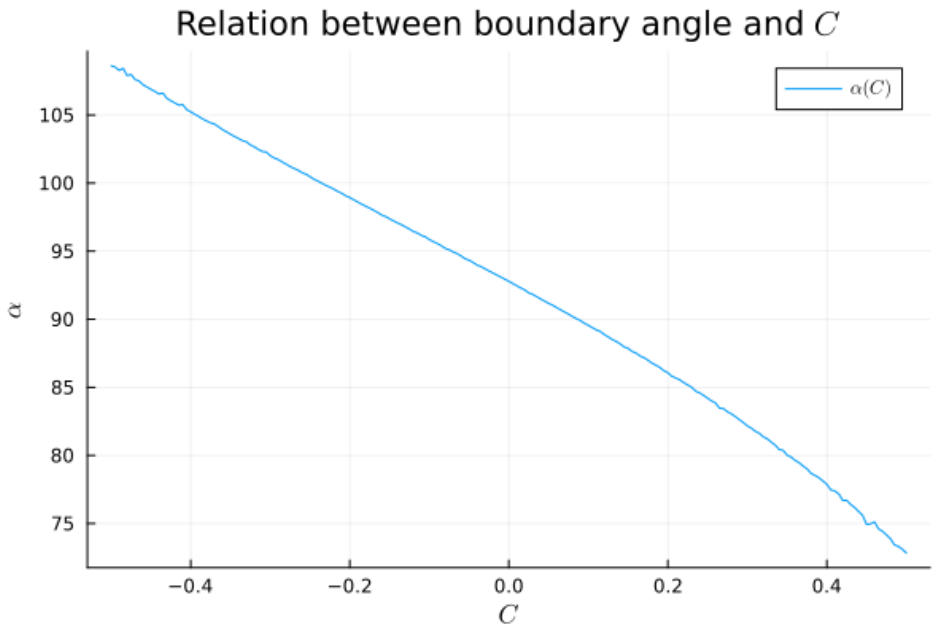

Figure 7.4: value for θ and corresponding angle α after 200 time-steps

8 Summary and outlook

In this project we examined a numerical model of the CH equation, with simple Neumann boundary conditions. We introduced a simplified version of the solver used in [2] and derived from [1]. Which due to GPU acceleration is significantly faster on our available hardware. We have shown a simple Neumann boundary approach that runs stable on both tested domains. The approach introduced by us is able to freely affect the angle of the phase interface on the boundary, The results of which we have shown on a circular and square domain. We introduced a rudimentary method to calculate the contact angle programmatically however we acknowledge that the results are unreliable. Further research would require a more consistent approach. One such method may be a filter that averages the angle calculation over more than one point, another should be a consistent selection of the point of interest, as the current approach cannot guaranty that the same (or a similar) point is selected if the input parameters change slightly.

Further research may concern itself, with the following topics. First and foremost, we observed inconsistent behavior when changing the hyperparameters ε and grid-size h. However, the methods we used for evaluation were inconsistent at best. Due to the aforementioned unpredictability in the angle calculation the resulting data series was erratic and no trends where apparent. Further research would require investigation of those effects on the boundary. Additionally, in our bachelor thesis, which served as preliminary work to this project, we investigated an analytical relaxation. The solver used therein for the relaxed problem is compatible with the boundary approach introduced herein. Initial tests with the solver for the relaxed system where promising, and further research may investigate those results.

Bibliography

[1] Jaemin Shin, Darae Jeong, and Junseok Kim. "A conservative numerical method for the Cahn–Hilliard equation in complex domains". In: *Journal of Computational Physics* 230.19 (2011), pp. 7441–7455. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2011.06.009. URL: https://www.sciencedirect.com/science/article/pii/S0021999111003585.

[2] Jonathan Ulmer. "A numerical method on the Cahn-Hilliard equation and its relaxed variation." Bachelor Thesis. University of Stuttgart, 2024. URL: https://github.com/ProceduralTree/CahnHilliardJulia/blob/af39a13f44caa1de1ee293b43e61e0c315d5aa0f/Thesis_jl.pdf.

[3] Hao Wu. "A review on the Cahn–Hilliard equation: classical results and recent advances in dynamic boundary conditions". In: *Electronic Research Archive* 30.8 (2022), pp. 2788–2832. DOI: 10.3934/era.2022143. URL: https://doi.org/10.3934%2Fera.2022143.