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

Jonathan Ulmer

December 12, 2024

Contents

Abstract

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

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 two-dimensional second order version of this CH equation rather than the usual 1D 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 Jacoby 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 5 and a circular domain in 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 the constant, 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 seperation 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, two-dimensional 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 $i, j \in [0, \ldots N]$ 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

$$\phi_{ij}^n : \Omega_d \times \{0, \ldots\} \to \mathbb{R}, \tag{2.8}$$

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

$$\mu_{ij}^n : \Omega_d \times \{0, \ldots\} \to \mathbb{R}, \tag{2.10}$$

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

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 were 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 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

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

We present an example in case of a 32x32 domain with $C = 1$ of the boundary fields **B** for a square domain 3.1 and an example on a circular domain in 3.2. In this Project we use the following adaptation of the discretization from [1].

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

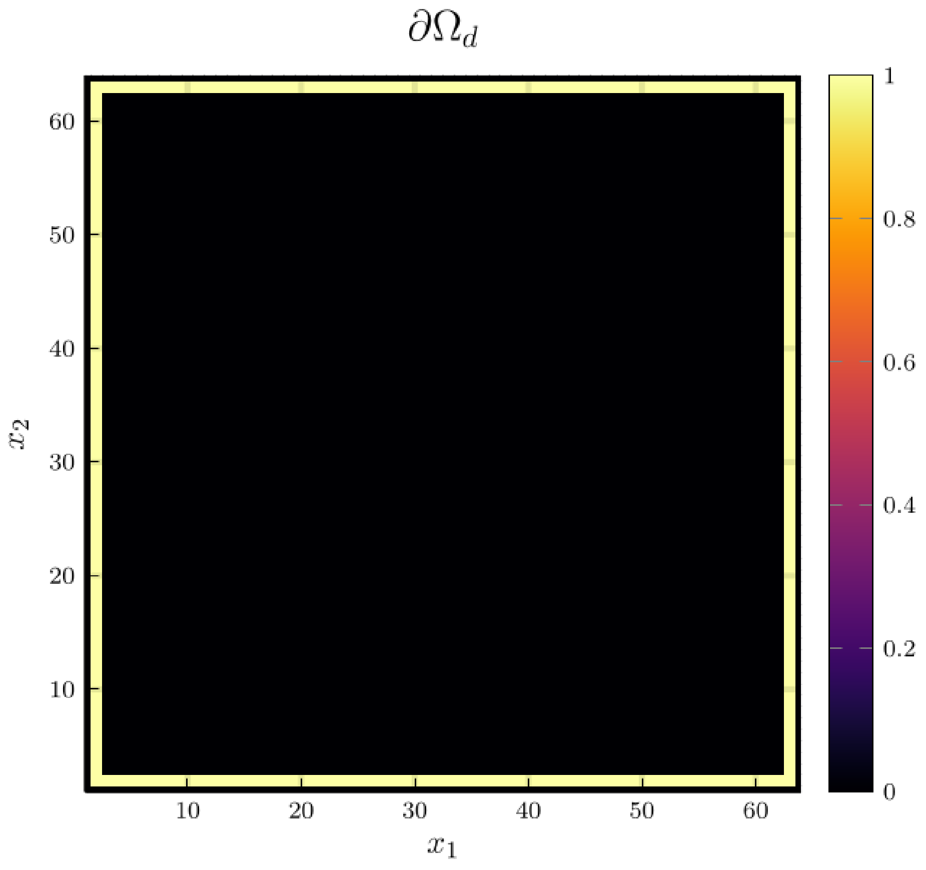


Figure 3.1: Visualization of all grid-cells adjacent to the boundary $\partial\Omega_d$ of a square domain

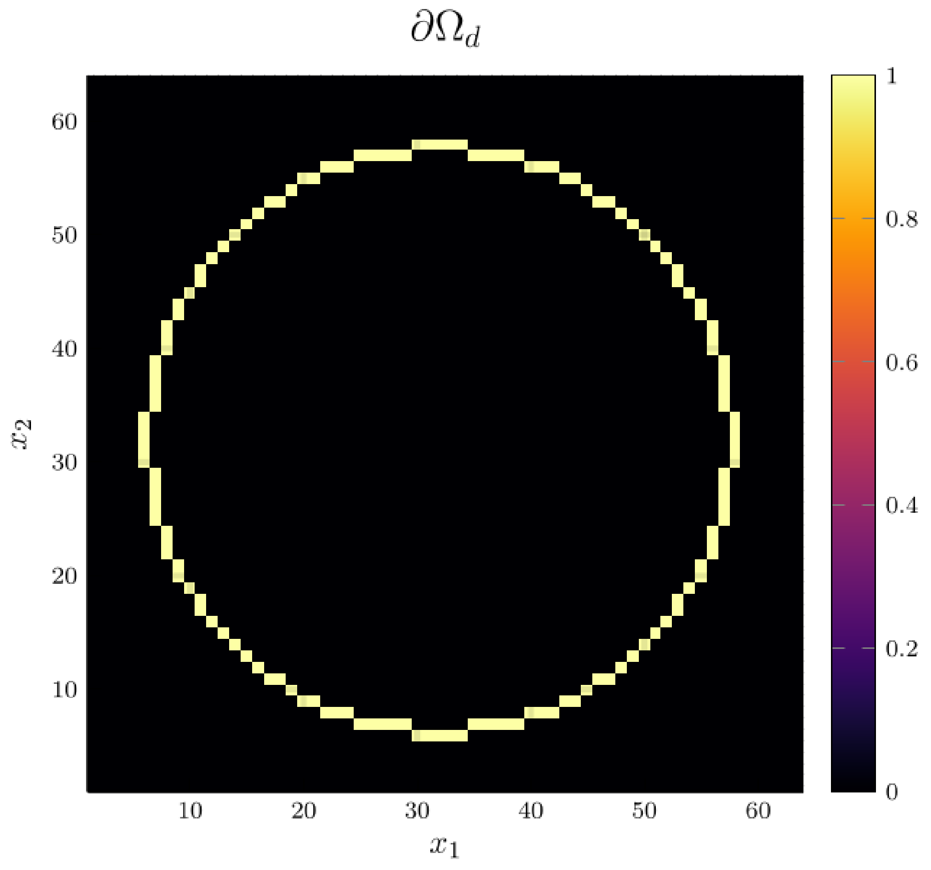


Figure 3.2: Visualization of all grid-cells adjacent to the boundary $\partial\Omega_d$ of a circular domain

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 Jacoby solver, as this will eventually assist in parallelizing the computation. Similar to [2] we linearize (3.3) to

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

One may note, that after rearanging some terms leads to a linear system with a right-hand side wich is exclusively dependant 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 Jacoby 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 $\nabla_d \cdot (G_{ij} \nabla_d \phi_{ij}^{n+1,m+\frac{1}{2}})$. Use the results from the previous Jacoby step for values off the center. e.g.

$$\begin{aligned} \nabla_d \cdot (G_{ij} \nabla_d \phi_{ij}^{n+1,m+\frac{1}{2}}) =& \frac{1}{h^2} (G_{i+\frac{1}{2}j} \phi_{i+1j}^{n+1,m} + G_{i-\frac{1}{2}j} \phi_{i-1j}^{n+1,m} \\ &+ \quad G_{ij+\frac{1}{2}} \phi_{ij+1}^{n+1,m} + G_{ij-\frac{1}{2}} \phi_{ij-1}^{n+1,m}) \\ &- \left(G_{i+\frac{1}{2}j} + G_{i-\frac{1}{2}j} + G_{ij+\frac{1}{2}} + G_{ij-\frac{1}{2}}\right) \phi_{ij}^{n+1,m+1} \end{aligned} \tag{4.3}$$

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.

```julia
@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

We set constant values for $\mathrm{B_{ij}}$ 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.

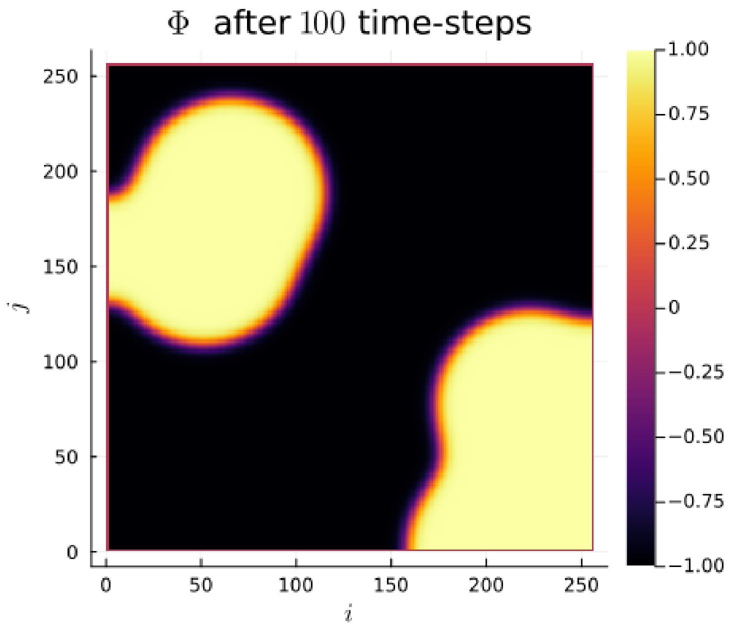


Figure 5.1: phase-field ϕ after 100 time-steps with $C = 0$ emmulating no-flux boundary.

We show, that our solver is stable for values $C \neq 0$. In 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.

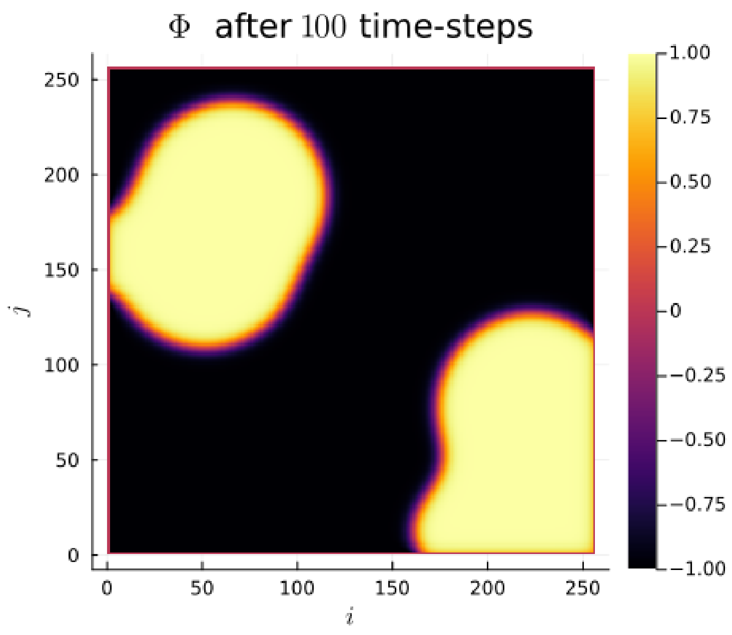


Figure 5.2: phase-field ϕ after 100 time steps with $C = 1$

In 5.3 we try the reverse situation. And we observe 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°.

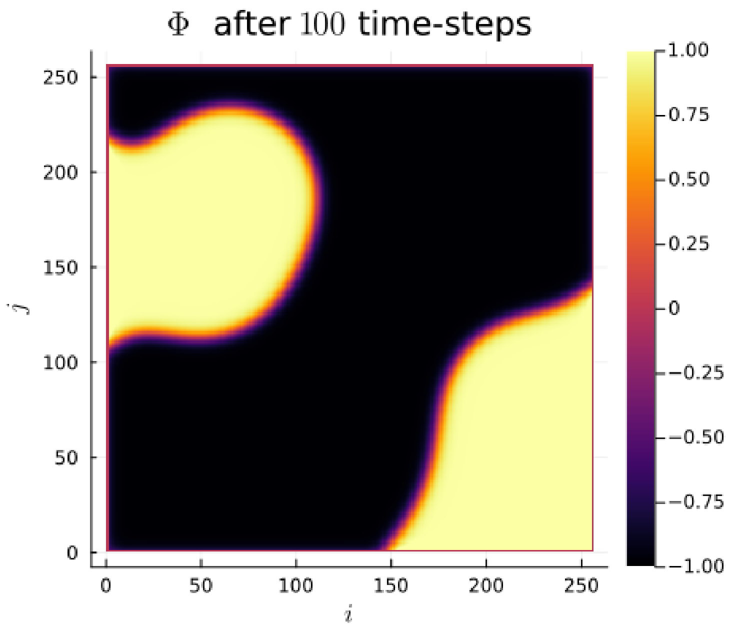
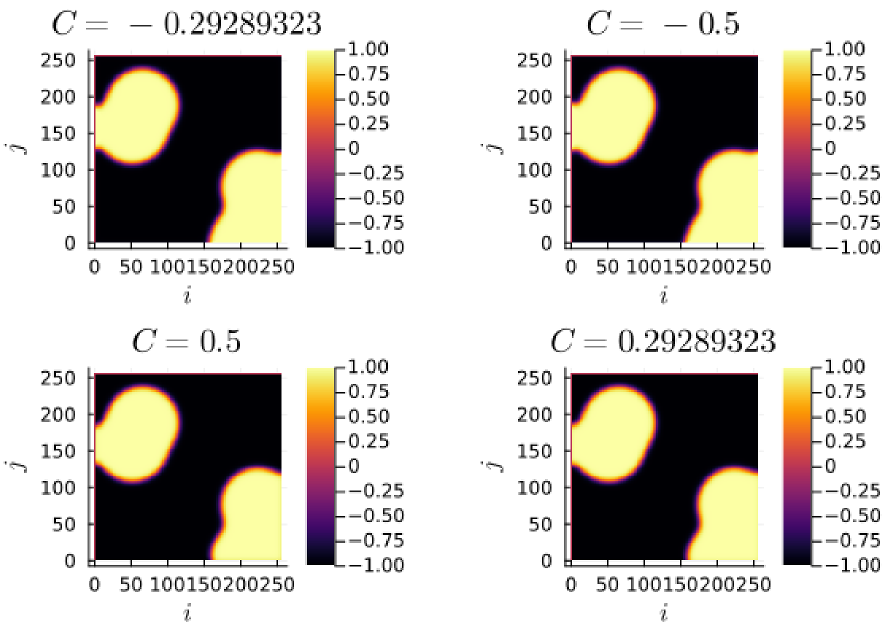
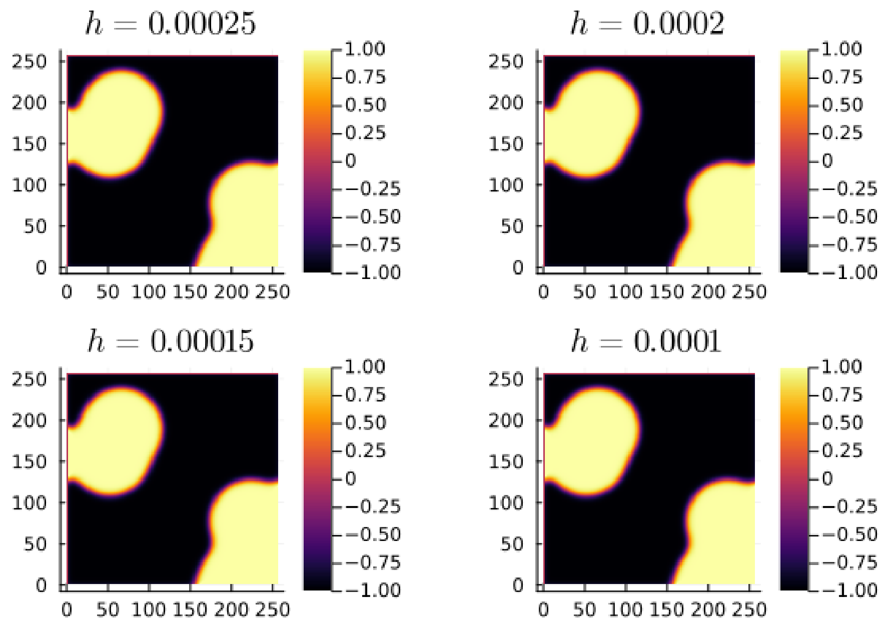


Figure 5.3: phase-field ϕ after 100 time-steps with $C = -1$



```
include("src/solvers.jl")
θ = -5f-1
n = 100
arr = _init()
```

```
d = domain(get_backend(arr) , 256 , size(arr))
d(arr)
h = 25e-5
solution = solve(arr , n , θ=θ)
h1 = heatmap(Array(solution) , aspect_ratio=:equal , clims=(-1,1),
↪  lims=(0,size(solution,1)), widen=1.06 , title=L"h=%$h")
h = 20e-5
solution = solve(arr , n , θ=θ)
h2 = heatmap(Array(solution) , aspect_ratio=:equal , clims=(-1,1),
↪  lims=(0,size(solution,1)), widen=1.06 , title=L"h=%$h")
h = 15e-5
solution = solve(arr , n , θ=θ)
h3 = heatmap(Array(solution) , aspect_ratio=:equal , clims=(-1,1),
↪  lims=(0,size(solution,1)), widen=1.06 , title=L"h=%$h")
h = 10e-5
solution = solve(arr , n , θ=θ)
h4 = heatmap(Array(solution) , aspect_ratio=:equal , clims=(-1,1),
↪  lims=(0,size(solution,1)), widen=1.06 , title=L"h=%$h")
plot(h1,h2,h3,h4)
```



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 6.1, we first employ no-flux boundary conditions on a circular domain. We observe the interface perpendicular on the boundary, as we expect.

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

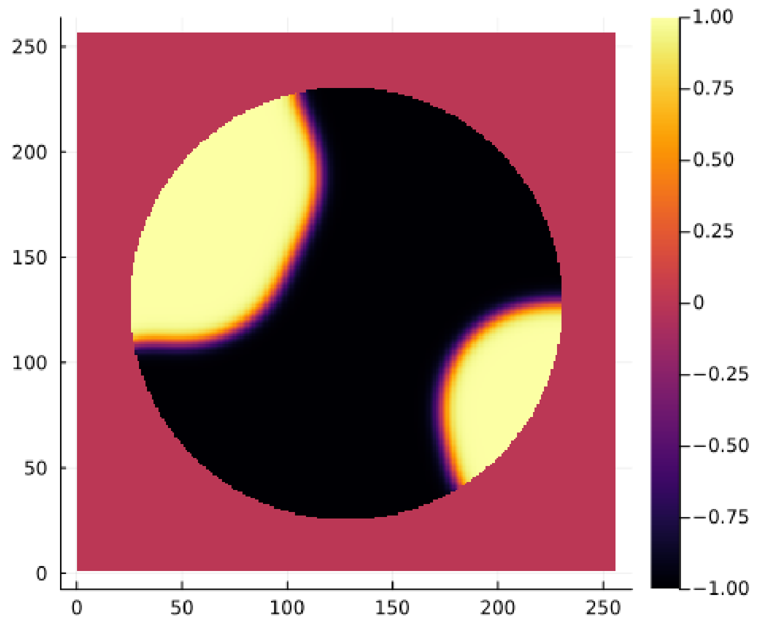


Figure 6.1: ϕ after 100 time steps on a circular domain with no-flux boundary-conditions after 100 time steps on a circular domain with no-flux

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

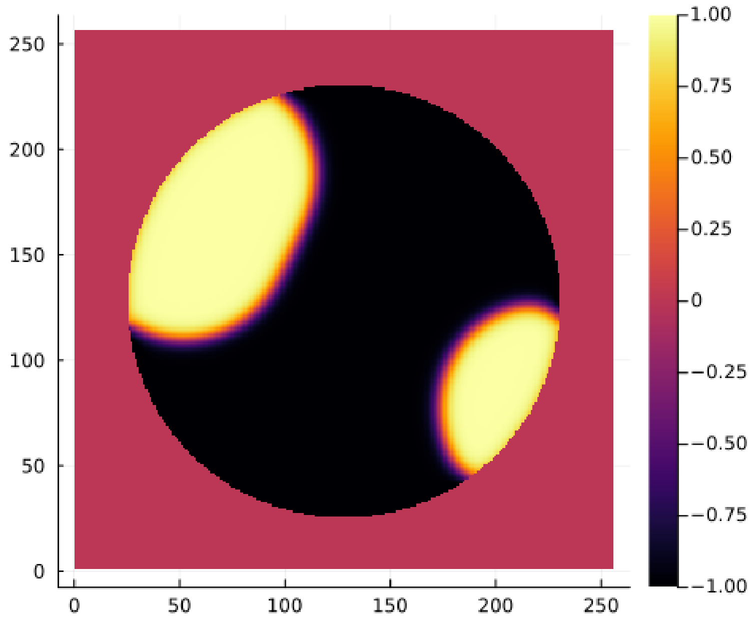


Figure 6.2: phase-field ϕ after 100 time-steps with $C = 1$

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

When evaluating intermediate contact angles in 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.

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.

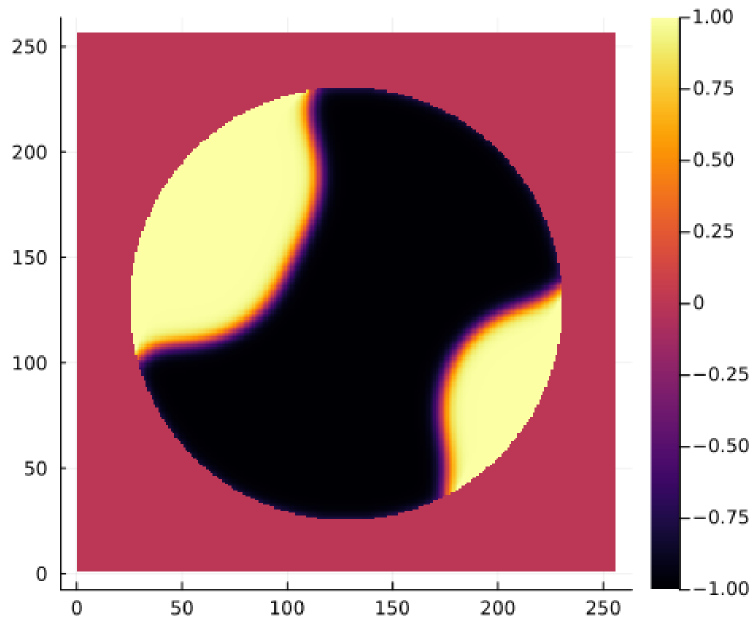


Figure 6.3: phase-field ϕ after 100 time-steps with $C = -1$

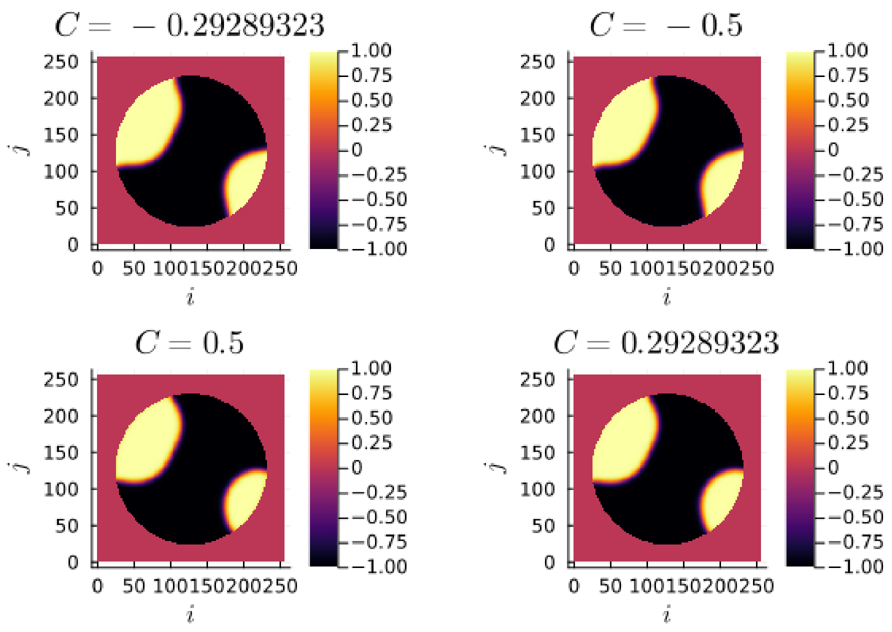


Figure 6.4: phase-field ϕ after 500 time-steps with $C \in \{-1 + \frac{\sqrt{2}}{2}, -0.5, 0.5, 1 - \frac{\sqrt{2}}{2}\}$ on a circular domain.

7 Angle

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.

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

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

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

```
angle (generic function with 1 method)
```

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 7.1 we present the results of a calculated angle, together with the normals and the point it is calculated from.

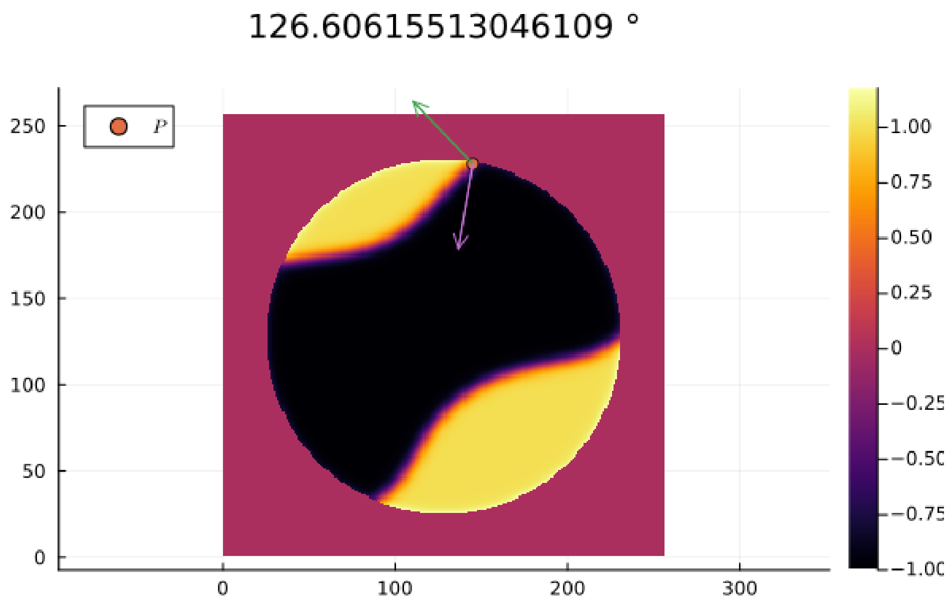


Table 7.1: value for θ and corresponding angle α after 200 time-steps

-0.1	173.49096591056502
-0.095	173.10715739345923
-0.09	172.18364087939332
-0.085	171.54740091859054
-0.08	171.3054040677464
-0.075	171.1455632002332
-0.07	171.02869693204397
-0.065	170.3901810227686
-0.06	170.0449796355949
-0.055	173.27274052589075
-0.05	170.3373892767722
-0.045	168.11953739721892
-0.04	167.41386769034298
-0.035	166.62088559081457
-0.03	164.9014365935728
-0.025	162.8061312020723
-0.02	159.92337650959868
-0.015	155.82320048245077
-0.01	147.4707481361878

-0.005	129.77836444929315
0.0	91.28977210940522
0.005	47.27538237804684
0.01	26.60911004838421
0.015	6.306468865037136
0.02	11.495581754132852
0.025	8.059259459078769
0.03	2.997826637980469
0.035	2.442790881259583
0.04	2.314200756133827
0.045	1.883610279597664
0.05	1.3567468712125557
0.055	0.8024311153759808
0.06	0.5869880299417852
0.065	0.4356076759230446
0.07	0.32719257485287145
0.075	0.03099970458170946
0.08	0.37685133141547533
0.085	0.4151229191583983
0.09	0.7049376111739059
0.095	0.8671639875701463
0.1	1.0282690721714873

7.2 square

the normal vector on a square domain is a litle bit more complicated than the circle normal. For the following 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 results are presented in 7.2

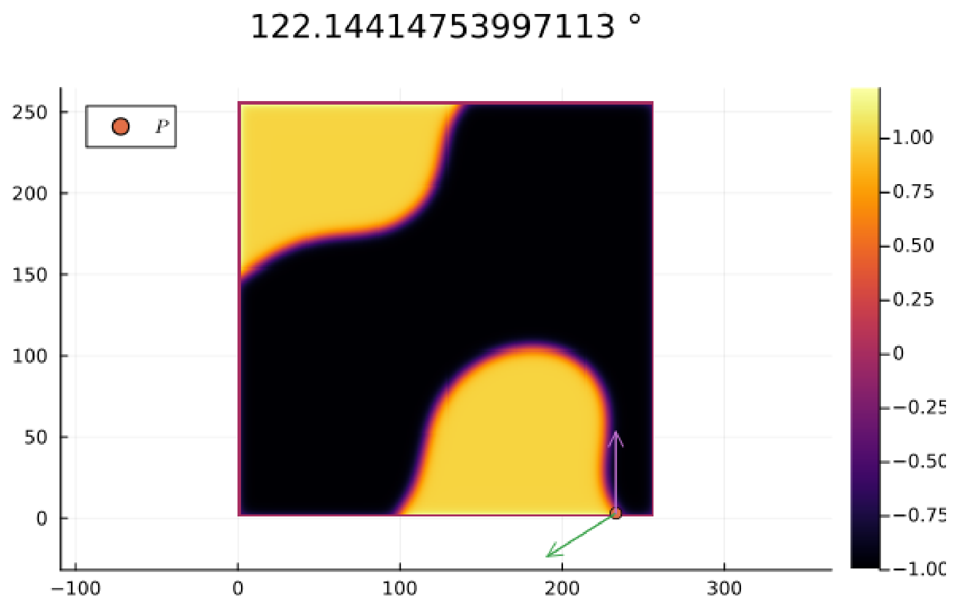


Table 7.2: value for θ and corresponding angle α after 200 time-steps

-0.1	177.95311184304808
-0.099	109.40440982717261
-0.098	113.75621160591214
-0.097	105.41178782179448
-0.096	102.88914136385036
-0.095	100.11964229844625
-0.094	103.81071062215845
-0.093	95.69795700045161
-0.092	95.67336816077285
-0.091	99.40375428329858
-0.09	106.41888954576633
-0.089	178.0077678154438
-0.088	178.3025715727096
-0.087	178.316114694492
-0.086	178.2255733054392
-0.085	95.46277809162602
-0.084	178.3549136743765
-0.083	178.26719963602554
-0.082	178.32251888851366

-0.081	178.3565799403403
-0.08	178.35467577409685
-0.079	178.38020838014717
-0.078	95.20563164308682
-0.077	95.14172673016061
-0.076	95.14296499007257
-0.075	105.44742113736852
-0.074	93.98032429805323
-0.073	93.93639872551873
-0.072	93.90397166303447
-0.071	95.04923946745302
-0.07	178.78211729473668
-0.069	102.99168583129313
-0.068	94.93247349592393
-0.067	94.89895399516479
-0.066	94.88468241056746
-0.065	94.83389070992926
-0.064	178.92443595960236
-0.063	100.16055296439168
-0.062	94.75596984934359
-0.061	101.27470856133
-0.06	179.0930181995219
-0.059	105.64189890451894
-0.058	94.61890118077682
-0.057	105.2681437129132
-0.056	94.55977561405972
-0.055	94.53388712342255
-0.054	93.34691119445903
-0.053	179.33441528850165
-0.052	93.2910844897572
-0.051	93.24648058021675
-0.05	93.23521452164574
-0.049	179.16701750513553
-0.048	179.2966784153456
-0.047	94.23632639847965
-0.046	179.29473359110352

-0.045	179.37255040579183
-0.044	101.54088044034116
-0.043	94.1661313225638
-0.042	94.12197781117199
-0.041	94.12000946956198
-0.04	94.0721925427342
-0.039	108.89374993386689
-0.038	179.63202655555918
-0.037	109.68666272268365
-0.036	108.51834091791333
-0.035	108.72075987854112
-0.034	92.74812555828719
-0.033	92.71114315877244
-0.032	109.35418670763997
-0.031	108.77670875273492
-0.03	92.61596109538934
-0.029	108.40044962449264
-0.028	92.56516258112002
-0.027	109.79996459123538
-0.026	109.65918977762365
-0.025	110.3044265042822
-0.024	83.62357084082397
-0.023	111.54991541493794
-0.022	111.03194196511788
-0.021	114.64047084976055
-0.02	109.05304432663621
-0.019	110.82994921660216
-0.018	92.24653067846398
-0.017	111.37975919462826
-0.016	113.65543027908225
-0.015	95.7731294576787
-0.014	110.03068049610467
-0.013	83.48278675686815
-0.012	114.18712728091695
-0.011	112.06506481164256
-0.01	88.97890310519033

-0.009	88.93437302618706
-0.008	88.92621027458483
-0.007	113.9196518388393
-0.006	88.8748154828598
-0.005	86.97101403557363
-0.004	111.88388458499975
-0.003	114.84915820262506
-0.002	113.98233214194332
-0.001	95.92713614334707
0.0	113.32020946225946
0.001	114.32873475565437
0.002	88.64471414125116
0.003	86.73475050008524
0.004	86.64299294912377
0.005	88.36892220070804
0.006	86.59429493690158
0.007	98.65732746097177
0.008	86.5904172845819
0.009	86.46759774660268
0.01	86.47786889099278
0.011	86.49649245589079
0.012	86.40563784781438
0.013	86.33651818354659
0.014	86.27783343769556
0.015	86.18667749563494
0.016	86.39821801641867
0.017	86.21768848775416
0.018	86.20656128064493
0.019	86.24422054042941
0.02	86.14661931403819
0.021	86.24452599358314
0.022	86.07925485738538
0.023	86.04273807339838
0.024	86.03922751662701
0.025	85.99686516380959
0.026	85.90542702435317

0.027	85.85309029518746
0.028	86.0330891413772
0.029	85.83084772186491
0.03	85.80380070559619
0.031	85.75495062019647
0.032	85.73494449356643
0.033	85.78267050897831
0.034	159.66794160038253
0.035	85.66466805791654
0.036	85.71079741144949
0.037	85.57577052103707
0.038	161.10166145787883
0.039	85.60187813586091
0.04	85.55578961267256
0.041	91.54706243587353
0.042	82.4478259247905
0.043	85.98986247300422
0.044	90.33035601585524
0.045	86.019092020242
0.046	86.09513749097509
0.047	85.92736023132964
0.048	82.61718340337106
0.049	161.86923826388156
0.05	177.4220744567218
0.051	177.40353870713483
0.052	177.36633804831473
0.053	177.31175619981855
0.054	177.27918695960085
0.055	177.20060770371583
0.056	91.09774195008896
0.057	177.18972050909164
0.058	177.24316583597977
0.059	177.12485385461935
0.06	176.10107235482425
0.061	177.05223887985608
0.062	176.07334642303667

0.063	175.99698804768886
0.064	176.9999733404206
0.065	176.94751164680537
0.066	175.95039284274893
0.067	175.9082495880407
0.068	175.8897212411926
0.069	99.731362526186
0.07	105.85378264391653
0.071	175.7949947173414
0.072	175.72222177353814
0.073	175.728407033159
0.074	175.69574900078072
0.075	175.64171193057405
0.076	175.57012583572595
0.077	175.65746918915175
0.078	105.57897708616683
0.079	102.28726139379764
0.08	175.46866309344034
0.081	102.96816792863312
0.082	102.63511936406039
0.083	101.19114810827676
0.084	175.38535145158926
0.085	103.4655632099724
0.086	103.62341228170168
0.087	103.9030792249824
0.088	103.67976184810772
0.089	175.20984823407616
0.09	102.9927434094263
0.091	83.71112467848221
0.092	83.74812231513913
0.093	83.69429256748194
0.094	102.11018607717594
0.095	103.75109938930927
0.096	104.5608960512815
0.097	83.5030462660855
0.098	83.54222287954207

0.099	106.86440916136596
0.1	83.45791163638641

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 prelimminary work to this project, we investigated an analytical relaxation. The solver used therein for the relaxed problem is compatible with the boundary approach introduced herin. Initial tests with the solver for the relaxed system where promissin, and further research may investigate those results.

9 References

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. “Untitled1”. Bachelor Thesis. University of Stuttgart, 3000.

[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`.