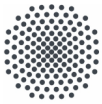

University of Stuttgart
Germany

Bachelor's Thesis

A Multi-grid method

on the Cahn-Hilliard equation and its relaxed variation.

Jonathan Ulmer
Matriculation Number: 3545737

Supervisor: Prof. Dr. Christian Rohde

Institute of Applied Analysis and Numerical Simulation

Completed 21.08.2024

Abstract

This Thesis gives a short overview and derivation for the Cahn-Hilliard Equation. It uses a discretization by the authors [2] as baseline, and expands upon this dicretisation with an elliptical relaxation approach. It introduces evaluation metrics regarding stability in time, space and during sub-iteration. And compares the elliptical approach against the baseline. Furthermore, it shows a qualitative success of the elliptical solver, however it also highlights challenges in numerical stability.

Contents

1 Introduction 1

2 The Cahn-Hilliard equation 3
- 2.1 Physical derivation of the CH equation (2.1) . . . 3
  - 2.1.1 The free energy . . . 3
  - 2.1.2 Derivation of the CH equation from mass balance . . . 4
- 2.2 initial value-boundary problem . . . 6

3 Discretization of the CH equation 7
- 3.1 The discretization of functions and derivative operators . . . 7
- 3.2 Initial data . . . 10
- 3.3 Discretization into a linear system . . . 12

4 Numerical solver 15
- 4.1 Gauss-Seidel smoothing . . . 15
- 4.2 Two-grid method . . . 16
  - 4.2.1 Pre Smoothing . . . 17
  - 4.2.2 Restriction . . . 17
  - 4.2.3 Course grid solution . . . 17
  - 4.2.4 Prolongation . . . 17
  - 4.2.5 Post Smoothing . . . 18
  - 4.2.6 additional considerations . . . 18

5 Numerical experiments 21
- 5.1 Energy evaluations . . . 21
- 5.2 Numerical mass conservation . . . 22
- 5.3 Stability of a two-grid sub-iteration . . . 23
- 5.4 Stability in time . . . 25

6 Relaxed Cahn-Hilliard equation 27
- 6.1 Relaxed initial value-boundary problem . . . 28

6.2 Relaxed energy functional 28
6.3 Relaxed mass conservation 29

7 Discretization of the relaxed problem 31
7.1 Elliptical PDE 31
7.2 Relaxed system 32

8 Relaxed numerical solver 35
8.1 Gauss Seidel solver for the elliptical system 35
8.2 Relaxed Gauss-Seidel iteration 36
8.3 The relaxed two-grid method 36

9 Discrete mass conservation 39

10 Numerical experiments for the relaxed system 43
10.1 explicit and implicit solution of the elliptical problem 43
10.2 optimizer for alpha 45
10.3 effect of α on the Gauss-Seidel iteration 47
10.4 Direct comparison of the baseline solver with the relaxed solver . . . 48
10.5 Relaxed energy evaluations 49
10.6 Stability of a relaxed two-grid sub-iteration 50
10.7 Relaxed numerical mass balance 51
10.8 Relaxed stability in time 52
10.9 additional considerations 53

11 Conclusion 55
11.1 Outlook 56

Bibliography 57

1 Introduction

The Cahn-Hilliard (CH) equation is a well known fourth order partial differential equation (PDE) used in multi-phase flow. It is used to couple different phases in a fluid with a diffuse-interface. It does this using a continuous transition between two phases. The CH equation, beeing fourth order, is difficult to solve numerically. However it is still used since it is able to guarantee conservation of mass. In this thesis we implement numerical solvers for the Cahn-Hilliard equation in the Julia programming language. We begin by giving an overview and a derivation for the analytical CH equation in Chapter 2. We then show mass conservation and a decrease of total energy in time. The Chapter 3 introduces our finite difference discretization. We explain the necessary functions, and give their implementation. Additionally we introduce the initial conditions we used in this thesis. Chapter 4 describes the relevant steps of our numerical implementation and the two-grid method we used. In Chapter 5 we evaluate this method's stability,discrete mass conservation and discrete energy decrease that we have shown continuously for the analytical CH equation. Our thesis introduces a elliptical relaxation approach to the classical CH equation, where instead of solving a fourth order PDE [1], we solver a second order relaxed PDE and an additional elliptical PDE. In the chapter 6 we introduce this approach, and then derive a numerical solver using the method described in chapter 4.2. Hereupon we derive and implement the necessary functions for the discretized relaxed equation, and we introduce a simple solver for the elliptical PDE. Subsequently, in chapter 10, we evaluate our relaxed method against the baseline with the same measures, as introduced in chapter 5.

We began writing this thesis with a reproducible research philosophy in mind. To do so we provide a single source of truth, i.e. a file (`Thesis_jl.org`) that includes explanations and mathematical formulae, accompanied with their numerical implementation, the implementation for our plots as well as the code for our numerical experiments. For convenience we also provide our Thesis in PDF format with some

[1] This solver uses a two dimensional version with 2 second order terms instead of the full fourth order equation.

of the irrelevant code removed. Furthermore we provide our implementation and experiments in separate source directories for ease of execution. Those where directly exported from `Thesis_jl.org`. All shown code is therefore the code that is run on our machine. Since not all parts of the code are relevant for understanding, unimportant sections are implemented elsewhere. Didactically they are replaced with a comment of form `<<unimportant-code-section>>`. Their implementation can be found in `Thesis_jl.org` in a code block of the same name. We did experiment with additional tools such as org-mode that allow for scientific note-taking and literate programming. Sadly this format relies heavily on the Emacs text editor and requires some setup to get working. Therefore, we fall short of making the entire Thesis as accessible as possible. `Thesis_jl.org` is available on our github repository at `https://github.com/ProceduralTree/CahnHilliardJulia.git`. The code to our numerical experiments we provide in the appendix as well as under `experiments/src/` in the repository. We provide the code for the plots next to them in `Thesis_jl.org`. The code for the numerical solvers themselves is in both the PDF and the ORG Document, and gets exported into the `src/` directory of the github repository.

2 The Cahn-Hilliard equation

The Cahn-Hilliard (CH) equation is a partial differential equation (PDE) that describes the dynamics of a two-phase fluid [3]. For this thesis we consider the following formulation of the CH equation in a given domain $\Omega \times (0,T)\,, \Omega \subset \mathbb{R}^d\,, d \in \mathbb{N}\,, T > 0$,

$$\begin{aligned} \partial_t \phi(x,t) &= \nabla \cdot (M(\phi)\nabla\mu), \\ \mu &= -\varepsilon^2 \Delta\phi + W'(\phi), \end{aligned} \tag{2.1}$$

where respectively, the variables $\phi, \mu : \Omega \times (0,T) \to \mathbb{R}^d$ are phase-field variable and chemical potential, ε is a positive constant correlated with interface thickness, $W(\phi)$ is a double well potential and $M(\phi) > 0$ is a mobility coefficient [3]. ϕ is defined in an interval $I = [-1,1]$ and represent the different phases.

$$\phi = \begin{cases} 1 & ,\phi \in \text{phase 1} \\ -1 & ,\phi \in \text{phase 2} \end{cases}$$

In this thesis we assume $M(\phi) \equiv 1$, simplifying the CH equation.

The advantages of the CH approach, as compared to traditional boundary coupling, are for example: "explicit tracking of the interface" [3], as well as "evolution of complex geometries and topological changes [...] in a natural way" [3].

2.1 Physical derivation of the CH equation (2.1)

2.1.1 The free energy

The authors in [3] define the CH equation using the **Ginzburg-Landau** free energy equation:

$$E^{\text{bulk}}[\phi] = \int_\Omega \frac{\varepsilon^2}{2}|\nabla\phi|^2 + W(\phi)\,dx, \tag{2.2}$$

where $W(\phi)$ denotes the Helmholtz free energy density of mixing [3] that we approximate it in further calculations with $W(\phi) = \frac{(1-\phi^2)^2}{4}$ as in [2] shown in Fig. 2.1.

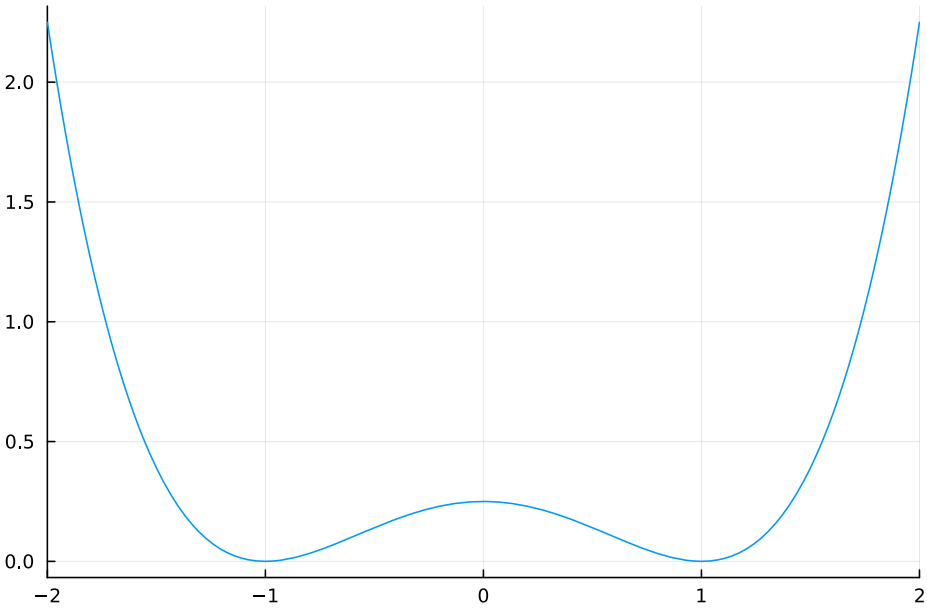

Figure 2.1: Double well potential $W(\phi)$

The chemical potential, μ, then follows as the variational derivation of the free energy in (2.2).

$$\mu = \frac{\delta E_{bulk}(\phi)}{\delta \phi} = -\varepsilon^2 \Delta \phi + W'(\phi) \tag{2.3}$$

2.1.2 Derivation of the CH equation from mass balance

The paper [3] states that the observable phase separation is driven by a diffusion resulting from the gradient in chemical potential μ. The emergent conservative dynamics motivate the following diffusion equation

$$\partial_t \phi + \nabla \cdot \mathbf{J} = 0, \tag{2.4}$$

where $\mathbf{J} = -\nabla\mu$ represents mass-flux. We follow the authors [3] in deriving the CH equation by combining (2.3) and (2.4).

$$\begin{aligned} \implies \partial_t\phi &= -\nabla\cdot\mathbf{J} = \Delta\mu, \\ \mu &= -\varepsilon^2\Delta\phi + W'(\phi)\,, \end{aligned} \tag{2.5}$$

Furthermore the CH equation is mass conservative under homogeneous Neumann boundary conditions, defined as:

$$\begin{aligned} \mathbf{J}\cdot\mathbf{n} &= 0 \quad \text{on}\,\partial\Omega\times(0,T), \\ \partial_n\phi &= 0 \quad \text{on}\,\partial\Omega\times(0,T), \end{aligned} \tag{2.6}$$

where $\mathbf{n}$ is the outward normal on $\partial\Omega$. To show the conservation of mass we analyze the change in total mass in the domain Ω over time.

$$\begin{aligned} \frac{d}{dt}\int_\Omega \phi\; d\mathbf{x} &= \int_\Omega \frac{\partial\phi}{\partial t}\; d\mathbf{x} \\ &= -\int_\Omega \nabla\cdot\mathbf{J}\; d\mathbf{x} \\ &= \int_{\partial\Omega} \mathbf{J}\cdot\mathbf{n}\; d\mathbf{s} \\ &= 0 \qquad\qquad \forall t\in(0,T)\,, \end{aligned} \tag{2.7}$$

In order to show thermodynamic consistency of the CH equation, we take the time derivation of the free energy functional (2.2).

$$\begin{aligned} \frac{d}{dt}E^{bulk}[\phi(t)] &= \int_\Omega (\varepsilon^2\nabla\phi\cdot\nabla\partial_t\phi + W'(\phi)\partial_t\phi)\; d\mathbf{x} \\ &= \int_\Omega (\varepsilon^2\nabla\phi + W'(\phi))\partial_t\phi\; d\mathbf{x} \\ &= \int_\Omega \mu\partial_t\phi\; d\mathbf{x} \\ &= \int_\Omega \mu\cdot\Delta\mu\; d\mathbf{x} \\ &= -\int_\Omega \nabla\mu\cdot\nabla\mu\; dx + \int_{\partial\Omega}\mu\nabla\phi_t\cdot\mathbf{n}\; dS \\ &\overset{\partial_n\phi=0}{=} -\int_\Omega |\nabla\mu|^2\; d\mathbf{x}, \qquad\qquad \forall t\in(0,T) \end{aligned}$$

This a bounded L_2 norm on $\Omega\times(0,T)$ of $\nabla\mu$.

2.2 INITIAL VALUE-BOUNDARY PROBLEM

The aim of the CH equation is then to find solutions $\phi(\vec{x},t), \mu(\vec{x},t) : \Omega \times (0,T) \to \mathbb{R}$ such that they satisfy

$$\begin{aligned}
\partial_t \phi(x,t) &= \nabla \cdot (M(\phi)\nabla\mu), \\
\mu &= -\varepsilon^2 \Delta\phi + W'(\phi), \qquad \text{in } \Omega \times (0,T), \\
-\nabla\mu \cdot \mathbf{n} &= 0 \\
\nabla\phi \cdot \mathbf{n} &= 0 \qquad\qquad\qquad\qquad \text{on } \partial\Omega \times (0,T), \\
\phi(x,0) &= \phi^0(x)\,, \qquad\qquad\qquad \text{in } \Omega
\end{aligned} \tag{2.8}$$

3 Discretization of the CH equation

This thesis uses the methods described by [2] to discretize (2.8). The method used by them is semi-implicit in time and implicit in space.

3.1 The discretization of functions and derivative operators

As baseline for numerical experiments we use a two-grid method based on the finite difference method defined in [2]. We discretize our domain Ω to be a Cartesian-grid Ω_d on a square with side-length $N \cdot h$, where N is the number of grid-points in one direction, and h is the distance between grid-points. In all our initial data h is $3 \cdot 10^{-3}$ and $N = 64$. However, for stability tests we change h and N. The discrete version or our domain is

$$\Omega_d = \{i, j \mid i, j \in \mathbb{N}, i, j \in [2, N+1]\}, \tag{3.1}$$

as shown in 3.1

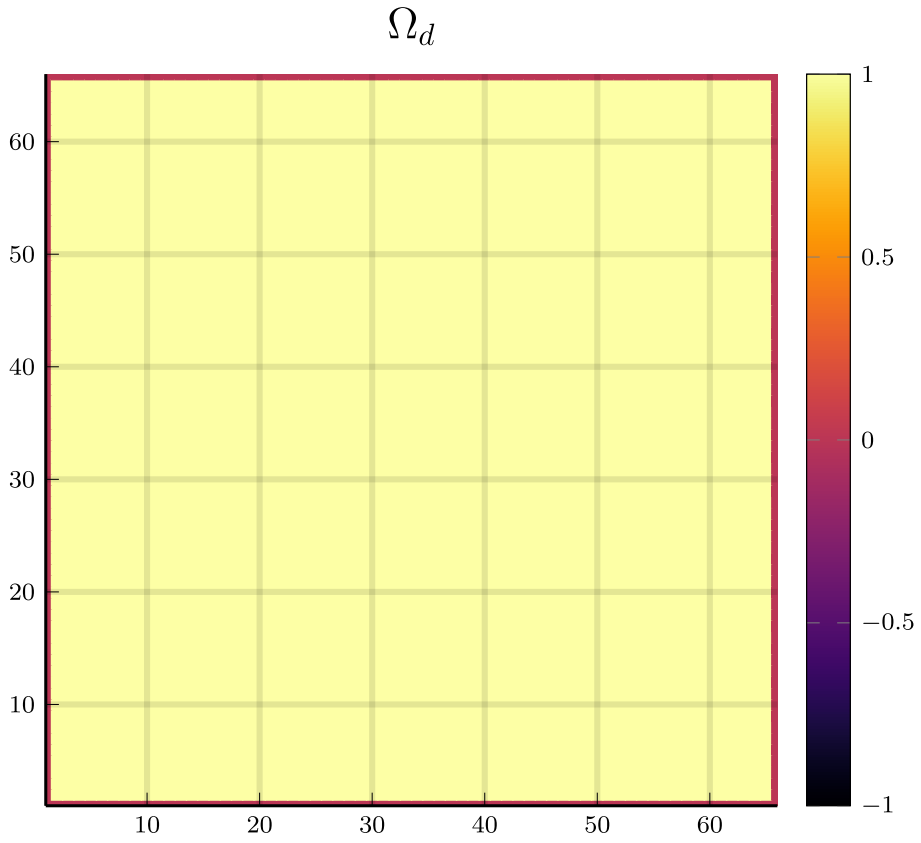

Figure 3.1: Discrete Domain used in this Thesis. 1 is inside and 0 outside of the Domain.

We discretize the phase-field ,ϕ, and chemical potential ,μ, into grid-wise functions ϕ_{ij}, μ_{ij} such that

$$\begin{aligned} \phi_{ij}^n &: \Omega_d \times \{0, \dots\} \to \mathbb{R}\,, \\ \mu_{ij}^n &: \Omega_d \times \{0, \dots\} \to \mathbb{R}\,, \end{aligned} \tag{3.2}$$

Here n denotes the n^{th} time-step, and (i, j) are Cartesian indices on the discrete domain Ω_d. The authors in [2] then use the characteristic function G of the domain Ω to enforce no-flux boundary conditions (2.6).

$$G(x, y) = \begin{cases} 1, & (x, y) \in \Omega \\ 0, & (x, y) \notin \Omega \end{cases}$$

We implement the discrete version of G on Ω_d as follows:

$$G_{ij} = \begin{cases} 1, & i,j \in [2, N+1] \\ 0, & \text{else} \end{cases}$$

The definition of G_{ij} with $i,j \in [2, N+1]$ enables us to evaluate G_{ij} with non integer values.

```
function G(i, j, len, width)
    if 2 <= i <= len + 1 && 2 <= j <= width + 1
        return 1.0
    else
        return 0.0
    end
end
```

We then define the discrete derivatives $D_x\phi_{ij},\ D_y\phi_{ij}$ using finite differences:

$$D_x\phi^{n+1,m}_{i+\frac{1}{2}j} = \frac{\phi^{n+1,m}_{i+1j} - \phi^{n+1,m}_{ij}}{h} \qquad D_y\phi^{n+1,m}_{ij+\frac{1}{2}} = \frac{\phi^{n+1,m}_{ij+1} - \phi^{n+1,m}_{ij}}{h} \tag{3.3}$$

We define $D_x\mu^{n+\frac{1}{2},m}_{ij}, D_y\mu^{n+\frac{1}{2},m}_{ij}$ in the same way. Next we define the discrete gradient $\nabla_d\phi^{n+1,m}_{ij}$, as well as a modified Laplacian $\nabla_d \cdot (G_{ij}\nabla_d\phi^{n+1,m}_{ij})$:

$$\begin{aligned} \nabla_d\phi^{n+1,m}_{ij} &= \left(D_x\phi^{n+1,m}_{i+1j},\ D_y\phi^{n+1,m}_{ij+1}\right), \\ \nabla_d \cdot (G_{ij}\nabla_d\phi^{n+1,m}_{ij}) &= \frac{G_{i+\frac{1}{2}j}D_x\phi^{n+1,m}_{i+\frac{1}{2}j} - G_{i-\frac{1}{2}}D_x\phi^{n+1,m}_{i-\frac{1}{2}j} + D_y\phi^{n+1,m}_{ij+\frac{1}{2}} - D_y\phi^{n+1,m}_{ij-\frac{1}{2}}}{h} \\ &= \frac{G_{i+\frac{1}{2}j}\phi^{n+1,m}_{i+1j} + G_{i-\frac{1}{2}j}\phi^{n+,m}_{i-1j} + G_{ij+\frac{1}{2}}\phi^{n+,m}_{ij+1} + G_{ij-\frac{1}{2}}\phi^{n+,m}_{ij-1}}{h^2} \\ &\quad - \frac{\left(G_{i+\frac{1}{2}j} + G_{i-\frac{1}{2}j} + G_{ij+\frac{1}{2}} + G_{ij-\frac{1}{2}}\right)\cdot\phi_{ij}}{h^2}, \end{aligned} \tag{3.4}$$

The discretization for $\nabla_d\mu^{n+\frac{1}{2},m}_{ij}, \nabla_d \cdot (G_{ij}\nabla_d\mu^{n+\frac{1}{2},m}_{ij})$ are done the same as for ϕ^{n+1}_{ij}. We define $\nabla_d \cdot (G_{ij}\nabla_d\phi_{ij})$ instead of a discrete Laplacian Δ_d to ensure a discrete version of boundary conditions (2.6). The authors in [2] show this to be the case by expanding $\nabla_d \cdot (G_{ij}\nabla_d\phi_{ij})$. Notably, when one point lies outside the domain, e.g. $G_{i+\frac{1}{2}} = 0$ then the corresponding discrete gradient $\frac{\phi^{n+1}_{i+1} - \phi_i}{h}$ is weighted by 0. This corresponds the discrete version of $\partial_n\phi = 0$ [2].

To simplify the notation for discretized derivatives we use the following abbreviations:

- $\Sigma_G \phi_{ij} = G_{i+\frac{1}{2}j}\phi^{n+1,m}_{i+1j} + G_{i-\frac{1}{2}j}\phi^{n+1,m}_{i-1j} + G_{ij+\frac{1}{2}}\phi^{n+1,m}_{ij+1} + G_{ij-\frac{1}{2}}\phi^{n+1,m}_{ij-1}$
- $\Sigma_{Gij} = G_{i+\frac{1}{2}j} + G_{i-\frac{1}{2}j} + G_{ij+\frac{1}{2}} + G_{ij-\frac{1}{2}}$

The Code for those abreviations is:

```
function neighbours_in_domain(i, j, G, len, width)
    (
        G(i + 0.5, j, len, width)
        + G(i - 0.5, j, len, width)
        + G(i, j + 0.5, len, width)
        + G(i, j - 0.5, len, width)
    )

end
function discrete_G_weigted_neigbour_sum(i, j, arr, G, len, width)
    (
        G(i + 0.5, j, len, width) * arr[i+1, j]
        + G(i - 0.5, j, len, width) * arr[i-1, j]
        + G(i, j + 0.5, len, width) * arr[i, j+1]
        + G(i, j - 0.5, len, width) * arr[i, j-1]
    )
end
```

We can then write the modified Laplacian $\nabla_d \cdot (G\nabla_d \phi^{n+1}_{ij})$ as:

$$\nabla_d \cdot (G\nabla_d \phi^{n+1}_{ij}) = \frac{\Sigma_G \phi^{n+1}_{ij} - \Sigma_{Gij} \cdot \phi^{n+1}_{ij}}{h^2} \tag{3.5}$$

We use this modified Laplacian to deal with boundary conditions. Our abbreviations simplify separating implicit and explicit terms in the discretization.

3.2 Initial data

For testing of our numerical solver for (2.8) we use initial discrete phase-fields defined by the following equations:

$$
\begin{aligned}
\phi_{ij}^0 &= \begin{cases} 1 & ,\|(i,j) - (\frac{N}{2},\frac{N}{2})\|_p < \frac{N}{3} \\ -1 & ,else \end{cases} && \text{where } p \in \{2,\infty\} \\
\phi_{ij}^0 &= \begin{cases} 1 & ,i < \frac{N}{2} \\ -1 & ,else \end{cases} \\
\phi_{ij}^0 &= \begin{cases} 1 & ,\|(i,j) - (\frac{N}{2},2)\|_2 < \frac{N}{3} \\ -1 & ,else \end{cases} \\
\phi_{ij}^0 &= \begin{cases} 1 & ,\|(i,j) - q_k\|_p < \frac{N}{5} \\ -1 & ,else \end{cases} && p \in \{1,2,\infty\}, q_k \in Q
\end{aligned}
\tag{3.6}
$$

where q_k are random points the given domain. We generate those using the following RNG setup in Julia

```julia
using Random
rng = MersenneTwister(42)
gridsize = 64
radius = gridsize /5
blobs = gridsize ÷ 5
rngpoints = rand(rng,1:gridsize, 2, blobs)
```

```
MersenneTwister(42)
64
12.8
12
2×12 Matrix{Int64}:
 48  40  20   1  63  49   8  60  26  58  26  11
 17  13  56  52  15   9  30  14  40   9  40  25
```

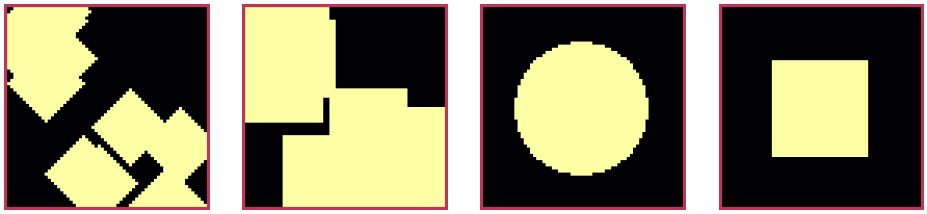

Figure 3.2: Examples of different phase-fields used as the initial condition.

3.3 Discretization into a linear system

The authors in [2] then define the discrete CH equation adapted for the domain as:

$$\begin{aligned} \frac{\phi_{ij}^{n+1} - \phi_{ij}^{n}}{\Delta t} &= \nabla_d \cdot (G_{ij} \nabla_d \mu_{ij}^{n+\frac{1}{2}}), \\ \mu_{ij}^{n+\frac{1}{2}} &= 2\phi_{ij}^{n+1} - \varepsilon^2 \nabla_d \cdot (G_{ij} \nabla_d \phi_{ij}^{n+1}) + W'(\phi_{ij}^{n}) - 2\phi_{ij}^{n}, \end{aligned} \tag{3.7}$$

and derive a numerical scheme from this equation. This method is semi-implicit in time, and consists of a centered difference in space. The authors in [2] derive their method by separating (3.7) into implicit and linear terms, and explicit non-linear terms. We write the implicit terms in form of a function $L : \mathbb{R}^2 \to \mathbb{R}^2$ and the explicit terms in $(\zeta_{ij}^n, \psi_{ij}^n)^T$. We define L as:

$$L\begin{pmatrix} \phi_{ij}^{n+1} \\ \mu_{ij}^{n+\frac{1}{2}} \end{pmatrix} := \begin{pmatrix} \frac{\phi_{ij}^{n+1}}{\Delta t} - \nabla_d \cdot (G_{ij} \nabla_d \mu_{ij}^{n+\frac{1}{2}}) \\ \varepsilon^2 \nabla_d \cdot (G \nabla_d \phi_{ij}^{n+1}) - 2\phi_{ij}^{n+1} + \mu_{ij}^{n+\frac{1}{2}} \end{pmatrix} \quad \forall i, j \in \{2, \dots, N+1\}.$$

```
function L(solver::multi_solver,i,j , phi , mu)
    xi = solver.phase[i, j] / solver.dt -
         (discrete_G_weigted_neigbour_sum(i, j, solver.potential, G, solver.len,
         ↪ solver.width)
          -
          neighbours_in_domain(i, j, G, solver.len, solver.width) * mu
          ↪ )/solver.h^2
    psi = solver.epsilon^2/solver.h^2 *
          (discrete_G_weigted_neigbour_sum(i, j, solver.phase, G, solver.len,
          ↪ solver.width)
           -
           neighbours_in_domain(i, j, G, solver.len, solver.width) * phi) - 2 *
           ↪ phi + mu
    return [xi, psi]
end
```

This function follows from (3.7) and is linear in the unknowns $\left(\phi_{ij}^{n+1}, \mu_{ij}^{n+\frac{1}{2}}\right)$. The non-linear terms of (3.7) are aggregated in $\left(\zeta_{ij}^n, \psi_{ij}^n\right)$, which we define as

$$\begin{pmatrix} \zeta_{ij}^n \\ \psi_{ij}^n \end{pmatrix} := \begin{pmatrix} \frac{\phi_{ij}^n}{\Delta t} \\ W'(\phi_{ij}^n) - 2\phi_{ij}^n \end{pmatrix} \qquad \forall i, j \in \{2, \dots, N+1\}. \tag{3.8}$$

```
function set_xi_and_psi!(solver::T) where T <: Union{multi_solver ,
↪  relaxed_multi_solver}
    xi_init(x) = x / solver.dt
    psi_init(x) = solver.W_prime(x) - 2 * x
    solver.xi[2:end-1, 2:end-1] = xi_init.(solver.phase[2:end-1,2:end-1])
    solver.psi[2:end-1, 2:end-1] = psi_init.(solver.phase[2:end-1,2:end-1])
    return nothing
end
```

The authors [2] defined a numerical method where all non linear terms are evaluated explicitly. Therefore , we know everything needed to calculate $(\zeta_{ij}^n, \psi_{ij}^n)^T$ at the beginning of each time step. We compute those values once and store them in the solver. Using $\left(\zeta_{ij}^n, \psi_{ij}^n\right)$ and $L\left(\phi_{ij}^{n+1}, \mu_{ij}^{n+\frac{1}{2}}\right)$, we can rewrite (3.7) as

$$L\begin{pmatrix}\phi_{ij}^{n+1}\\ \mu_{ij}^{n+\frac{1}{2}}\end{pmatrix} = \begin{pmatrix}\zeta_{ij}^n\\ \psi_{ij}^n\end{pmatrix}. \quad \forall i,j \in \{1,\dots,N\} \tag{3.9}$$

This Linear system consists of NxN, 2 dimensional linear equations. Each equation in the linear system (3.9) can be rewriten in the form $\mathbf{DL}_{ij} \cdot \left(\phi_{ij}^{n+1}, \mu_{ij}^{n+\frac{1}{2}}\right)^T = b_{ij}$: Where $\mathbf{DL}_{ij}$ is

$$\mathbf{DL}_{ij} = \begin{pmatrix}\frac{1}{\Delta t} & \frac{1}{h^2}\Sigma_{Gij}\\ -\frac{\varepsilon^2}{h^2}\Sigma_{Gij} - 2 & 1\end{pmatrix}$$

and where $\Sigma_{Gij} = G_{i+\frac{1}{2}j} + G_{i-\frac{1}{2}j} + G_{ij+\frac{1}{2}} + G_{ij-\frac{1}{2}}$

```
function dL(solver::multi_solver , i , j)
    return [ (1/solver.dt) (1/solver.h^2*neighbours_in_domain(i,j,G,solver.len ,
    ↪  solver.width));
            (-1*solver.epsilon^2/solver.h^2 *
            ↪  neighbours_in_domain(i,j,G,solver.len , solver.width) - 2) 1]
    end
```

$\mathbf{DL}_{ij}$ is invertible, since its determinant is positive. Therefore the system (3.9) is solvable

$$\det(\mathbf{DL}_{ij}) = \frac{1}{\Delta t} + \frac{1}{h^2}\Sigma_{Gij}\left(+\frac{\varepsilon^2}{h^2}\Sigma_{Gij} + 2\right) > 0 \tag{3.10}$$

as $\Sigma_{Gij} \in \{0,1,2,3,4\}$ Using The abbreviation for $\nabla_d \cdot (G_{ij} \nabla_d \mu_{ij}^{n+\frac{1}{2}})$ introduced in (3.5) , and we rewrite (3.9) in terms of $\mathbf{DL}_{ij}$

$$
\begin{aligned}
& L \begin{pmatrix} \phi_{ij}^{n+1} \\ \mu_{ij}^{n+\frac{1}{2}} \end{pmatrix} = \begin{pmatrix} \zeta_{ij}^{n} \\ \psi_{ij}^{n} \end{pmatrix} \\
\implies \quad & \mathbf{DL}_{ij} \cdot \begin{pmatrix} \phi_{ij}^{n+1} \\ \mu_{ij}^{n+\frac{1}{2}} \end{pmatrix} + \begin{pmatrix} -\frac{1}{h^2} \Sigma_{Gij} \mu_{ij}^{n+\frac{1}{2}} \\ +\frac{\varepsilon^2}{h^2} \Sigma_{Gij} \phi_{ij}^{n+1} \end{pmatrix} = \begin{pmatrix} \zeta_{ij}^{n} \\ \psi_{ij}^{n} \end{pmatrix}, \qquad (3.11) \\
\implies \quad & \mathbf{DL}_{ij} \cdot \begin{pmatrix} \phi_{ij}^{n+1} \\ \mu_{ij}^{n+\frac{1}{2}} \end{pmatrix} = \begin{pmatrix} \zeta_{ij}^{n} \\ \psi_{ij}^{n} \end{pmatrix} - \begin{pmatrix} -\frac{1}{h^2} \Sigma_{Gij} \mu_{ij}^{n+\frac{1}{2}} \\ +\frac{\varepsilon^2}{h^2} \Sigma_{Gij} \phi_{ij}^{n+1} \end{pmatrix},
\end{aligned}
$$

where

- $\Sigma_G \phi_{ij}^{n+1} = G_{i+\frac{1}{2}j} \phi_{i+1j}^{n+1,m} + G_{i-\frac{1}{2}j} \phi_{i-1j}^{n+1,m} + G_{ij+\frac{1}{2}} \phi_{ij+1}^{n+1,m} + G_{ij-\frac{1}{2}} \phi_{ij-1}^{n+1,m}$, $\Sigma_G \mu_{ij}^{n+\frac{1}{2}} = G_{i+\frac{1}{2}j} \mu_{i+1j}^{n+\frac{1}{2},m} + G_{i-\frac{1}{2}j} \mu_{i-1j}^{n+\frac{1}{2},m} + G_{ij+\frac{1}{2}} \mu_{ij+1}^{n+\frac{1}{2},m} + G_{ij-\frac{1}{2}} \mu_{ij-1}^{n+\frac{1}{2},m}$,

4 Numerical solver

The multi-grid method consists of a linear Gauss-Seidel solver, restriction, and prolongation methods, to move course and fine grids.

4.1 Gauss-Seidel smoothing

The authors [2]derived Gauss-Seidel Smoothing from (3.9) : Smoothing denoted as a SMOOTH operator consists of a Gauss-Seidel method, by solving (3.11) for all i, j with the initial guess for $\zeta_{ij}^n, \psi_{ij}^n$. We define an iterative Gauss-Seidel method. After having solved equation (3.9) for $(i-1, j), (i, j-1)$ we define the Gaus-Seidel iteration in s for (i, j) as follows:

$$\mathbf{DL}_{ij} \cdot \begin{pmatrix} \phi_{ij}^{n+1,s+1} \\ \mu_{ij}^{n+\frac{1}{2},s+1} \end{pmatrix} = \begin{pmatrix} \zeta_{ij}^n \\ \psi_{ij}^n \end{pmatrix} - \begin{pmatrix} -\frac{1}{h^2}\Sigma_{Gij}\mu_{ij}^{n+\frac{1}{2},s+\frac{1}{2}} \\ +\frac{\varepsilon^2}{h^2}\Sigma_{Gij}\phi_{ij}^{n+1,s+\frac{1}{2}} \end{pmatrix}, \tag{4.1}$$

where

- $\Sigma_G \phi_{ij}^{n+1,s+\frac{1}{2}} = G_{i+\frac{1}{2}j}\phi_{i+1j}^{n+1,s} + G_{i-\frac{1}{2}j}\phi_{i-1j}^{n+1,s+1} + G_{ij+\frac{1}{2}}\phi_{ij+1}^{n+1,s} + G_{ij-\frac{1}{2}}\phi_{ij-1}^{n+1,s+1}$,
- $\Sigma_G \mu_{ij}^{n+\frac{1}{2},s+\frac{1}{2}} = G_{i+\frac{1}{2}j}\mu_{i+1j}^{n+\frac{1}{2},s} + G_{i-\frac{1}{2}j}\mu_{i-1j}^{n+\frac{1}{2},s+1} + G_{ij+\frac{1}{2}}\mu_{ij+1}^{n+\frac{1}{2},s} + G_{ij-\frac{1}{2}}\mu_{ij-1}^{n+\frac{1}{2},s+1}$,

This constitutes a Gaus-Seidel method in its element based formula.

```
function SMOOTH!(
    solver::T,
    iterations,
    adaptive
) where T <: Union{multi_solver, adapted_multi_solver , gradient_boundary_solver}
    for s = 1:iterations
        # old_phase = copy(solver.phase)
        for I in CartesianIndices(solver.phase)[2:end-1, 2:end-1]
            i, j = I.I

            <<calculate-left-hand-side>>
```

```
            res = dL(solver, i,j ) \ b
            solver.phase[i, j] = res[1]
            solver.potential[i, j] = res[2]
        end
    end
end
```

We denote the approximations for $\left(\phi_{ij}^{n+1}, \mu_{ij}^{n+\frac{1}{2}}\right)$ after smoothing, as $\left(\bar{\phi}_{ij}^{n+1}, \bar{\mu}_{ij}^{n+\frac{1}{2}}\right)$. In Fig.4.1 we show 4 of the 7 initial data after one 200 iterations of smoothing. It is apparent that the sharp interface from the initial Data has diffused.

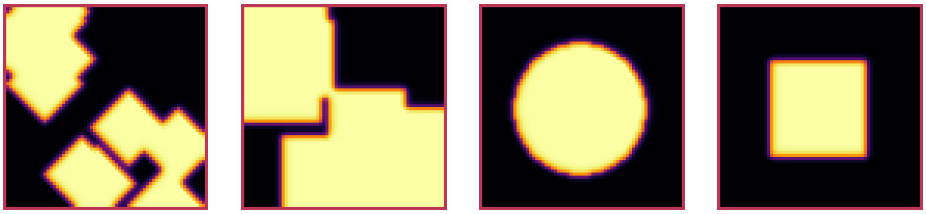

Figure 4.1: Inputs from 3.2 after SMOOTH.

4.2 Two-grid method

The numerical method proposed in [2] consists of repeated sub-iterations of a multi-grid V-cycle. Specifically we use a two-grid implementation with a fixed number of sub-iterations. Defined as:

```
for j in 1:timesteps

    set_xi_and_psi!(solvers[1])

    for i = 1:subiterations

        v_cycle!(solvers, 1)
    end
end
```

The approximations for $\phi_{ij}^{n+1}, \mu_{ij}^{n+\frac{1}{2}}$ after the m$^{\text{th}}$ V-cycle sub-iteration are denoted with $\phi_{ij}^{n+1,m+1}, \mu_{ij}^{n+\frac{1}{2},m+1}$ where m denotes the current sub-iteration Furthermore the V-cycle consists of the following steps:

4.2.1 Pre Smoothing

Pre smoothing consists of a fixed number of Gauss-Seidel iterations, in our case **40**, on the fine grid h, as described in Chapter 4.1. Afterwards we calculate the residual error $\left(d_{ij,H}^{n+1,m}, r_{ij,H}^{n+1,m}\right) := L\left(\phi_{ij}^{n+1}, \mu_{ij}^{n+\frac{1}{2}}\right) - (\zeta_{ij}^{n}, \psi_{ij}^{n})$ for the course grid H correction.

4.2.2 Restriction

Restriction from the fine grid to the course grid $h \to H$ for a variable eg. ϕ_{ij} is done as follows:

$$\phi_{ij}^{H} = \frac{1}{\Sigma_{Gij}}\left(G_{2i,2j}\phi_{2i,2j}^{h} + G_{2i-1,2j}\phi_{2i-1,2j}^{h} + G_{2i,2j-1}\phi_{2i,2j-1}^{h} + G_{2i-1,2j-1}\phi_{2i-1,2j-1}^{h}\right) \tag{4.2}$$

4.2.3 Course grid solution

On the course grid we use a Gauss-Seidel iteration to solve $L(\hat{\phi}_{ij,H}^{n+1,m}, \hat{\mu}_{ij,H}^{n+\frac{1}{2},m})_H = L(\bar{\phi}_{ij,H}^{n+1,m}, \bar{\mu}_{ij,H}^{n+\frac{1}{2},m}) + (d_{ij,H}^{n+1,m}, r_{ij,H}^{n+1,m})$. We solve for $\left(\hat{\phi}_{ij,H}^{n+1,m}, \hat{\mu}_{ij,H}^{n+\frac{1}{2},m}\right)$ using the same iteration as in Chapter 4.1 however we replace $(\zeta_{ij}^{n}, \psi_{ij}^{n})$ with $L(\bar{\phi}_{ij,H}^{n+1,m}, \bar{\mu}_{ij,H}^{n+\frac{1}{2},m})+$ $(d_{ij,H}^{n+1,m}, r_{ij,H}^{n+1,m})$. In the iteration, where $\left(\bar{\phi}_{ij,H}^{n+1,m}, \bar{\mu}_{ij,H}^{n+\frac{1}{2},m}\right)$ are the values after the smooth restricted to the coarser grid and $\left(d_{ij,H}^{n+1,m}, r_{ij,H}^{n+1,m}\right)$ is the residual from the smooth iteration on the fine grid restricted onto the coarse grid.

4.2.4 Prolongation

We prolong the solution from the course grid. Prolongation of a variable eg. ϕ_{ij} from the course grid to the fine grid $H \to h$ we do by using the nearest neighbour weighed by G.

$$\begin{pmatrix} \phi_{2i,2j}^{h} \\ \phi_{2i-1,2j}^{h} \\ \phi_{2i,2j-1}^{h} \\ \phi_{2i-1,2j-1}^{h} \end{pmatrix} = \begin{pmatrix} G_{2i,2j}^{h}\phi_{ij}^{H} \\ G_{2i-1,2j}^{h}\phi_{ij}^{H} \\ G_{2i,2j-1}^{h}\phi_{ij}^{H} \\ G_{2i-1,2j-1}^{h}\phi_{ij}^{H} \end{pmatrix} \tag{4.3}$$

4.2.5 Post Smoothing

After prolongation of the course grid solution we perform a post smoothing step using **80** Gauss-Seidel steps. Post smoothing is otherwise identical to pre smoothing

4.2.6 additional considerations

We Do Gauss-Seidel smoothing with fixed iterations. As well as a fixed number of sub-iterations.

The V-cycle of a two-grid method using pre- and post-smoothing is then stated by:

```
function alt_v_cycle!(grid::Array{T}, level) where T <: solver
    finegrid_solver = grid[level]
    #pre SMOOTHing
    SMOOTH!(solver, 40, false)

    d = zeros(size(finegrid_solver.phase))
    r = zeros(size(finegrid_solver.phase))

    # calculate error between L and expected values
    for I in CartesianIndices(finegrid_solver.phase)[2:end-1, 2:end-1]
        d[I], r[I] = [finegrid_solver.xi[I], finegrid_solver.psi[I]]
        .- L(finegrid_solver, I.I..., finegrid_solver.phase[I],
        ↪  finegrid_solver.potential[I])
    end

    restrict_solver!(grid[level], grid[level+1])
    coursegrid_solver = grid[level+1]
    solution = deepcopy(coursegrid_solver)

    d_large = restrict(d, G)
    r_large = restrict(r, G)


    u_large = zeros(size(d_large))
    v_large = zeros(size(d_large))

    for I in CartesianIndices(coursegrid_solver.phase)[2:end-1, 2:end-1]
        coursegrid_solver.xi[I]   , coursegrid_solver.psi[I] = L(coursegrid_solver
        ↪  , I.I... , coursegrid_solver.phase[I] , coursegrid_solver.potential[I]
        ↪  ) .+ [d_large[I],r_large[I]]
    end
```

```
    SMOOTH!(coursegrid_solver, 40 , false)

    u_large = coursegrid_solver.phase .- solution.phase
    v_large = coursegrid_solver.potential .- solution.potential

    finegrid_solver = grid[level]
    finegrid_solver.phase .+= prolong(u_large , G)
    finegrid_solver.potential .+= prolong(v_large, G)


    SMOOTH!(finegrid_solver, 80, false)
end
```

```
function v_cycle!(grid::Array{T}, level) where T <: solver
    solver = grid[level]
    #pre SMOOTHing:
    SMOOTH!(solver, 400, false)

    d = zeros(size(solver.phase))
    r = zeros(size(solver.phase))

    # calculate error between L and expected values
    for I in CartesianIndices(solver.phase)[2:end-1, 2:end-1]
        d[I], r[I] = [solver.xi[I], solver.psi[I]] .- L(solver, I.I...,
        ↪  solver.phase[I], solver.potential[I])
    end

    <<restrict-to-coarse-grid>>

    #Newton Iteration for solving smallgrid
    for i = 1:300
        for I in CartesianIndices(solver.phase)[2:end-1, 2:end-1]

            diffrence = L(solution, I.I..., solution.phase[I],
            ↪  solution.potential[I])
                        .- [d_large[I], r_large[I]]
                        .- L(solver, I.I..., solver.phase[I], solver.potential[I])

            local ret = dL(solution, I.I...) \ diffrence

            u_large[I] = ret[1]
            v_large[I] = ret[2]
        end
```

```
        solution.phase .-= u_large
        solution.potential .-= v_large
    end

    <<prolong-to-fine-grid>>

    SMOOTH!(solver, 800, false)
end
```

After a few iterations, V-cycle exhibits the following behavior:

```
<<init>>
using JLD2
using DataFrames
results = jldopen("experiments/iteration.jld2")["result"]
anim = @animate for res in eachrow(results)
    heatmap(res.solver.phase , title="phase field" , legend=:none ,
    ↪  aspectratio=:equal , showaxis=false , grid=false , size=(400 ,400))
end
gif(anim , "images/iteration.gif" , fps = 10)
```

images/iteration.gif

5 Numerical experiments

In the previous Chapter we discretized the CH equation based on the two-grid method described by the authors in [2] and we obtained a numerical scheme for ϕ, μ. In this chapter we analyze the change in mass, change in total energy E^{bulk}, the stability in time, and during sub-iterations. Since we do not have exact solutions for the initial values tested we evaluate our solvers with a Cauchy criterion. The initial values we use, if not mentioned otherwise, where:

Variable:	ε	Δ t	h
Value:	8 * 10^{-3}	10^{-3}	3*10^{-3}

5.1 Energy evaluations

Since the continuous total energy (2.2) decreases over time, we expect it's discrete counterpart to exhibit the same behaviour. We implement a discrete version of the energy, and evaluate our solutions on it.

$$\begin{aligned} E_d^{\text{bulk}}(\phi^n) &= \sum_{i,j\in\Omega} \frac{\varepsilon^2}{2} |G\nabla_d \phi_{ij}|^2 + W(\phi_{ij}) \\ &= \sum_{i,j\in\Omega} \frac{\varepsilon^2}{2} G_{i+\frac{1}{2}j}(D_x\phi_{i+\frac{1}{2}j})^2 + G_{ij+\frac{1}{2}}(D_y\phi_{ij+\frac{1}{2}})^2 + W(\phi_{ij}). \end{aligned} \tag{5.1}$$

In Fig.5.1 we observe the discrete total energy going down with increasing number of time-steps, as we expect from a CH based solver. Visually we observe the energy decrease as reduced surface curvature.

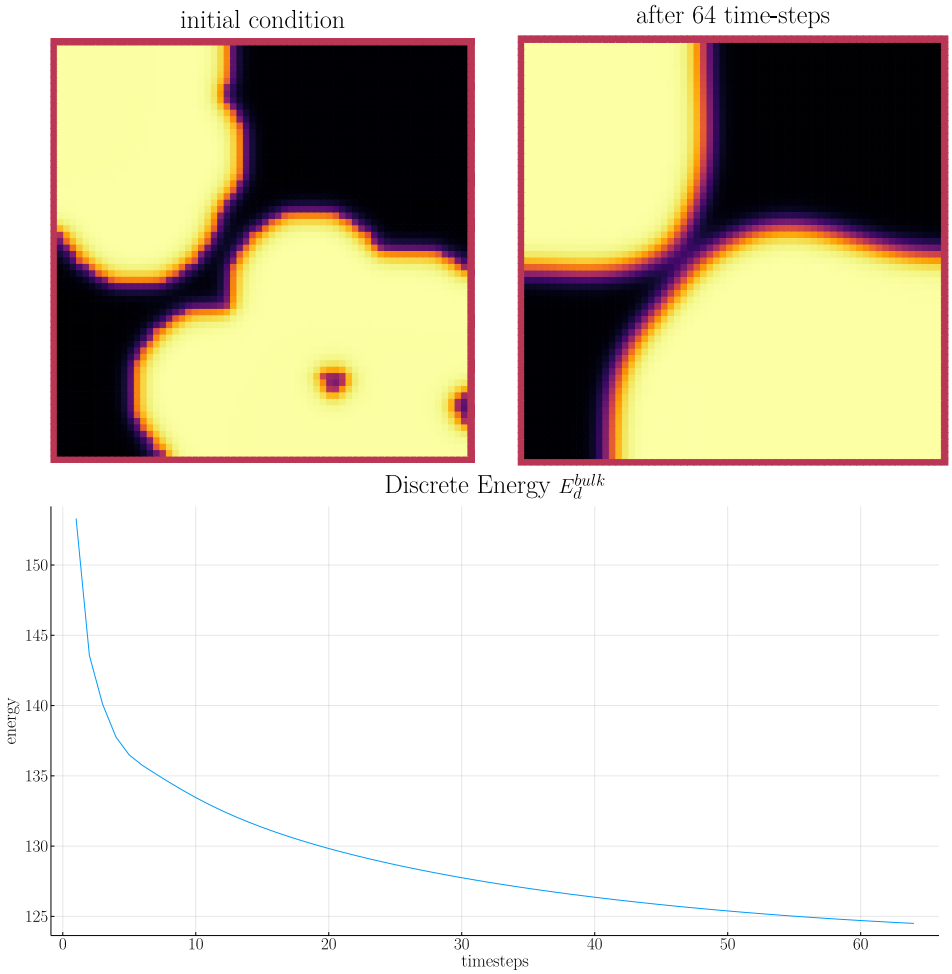

Figure 5.1: Behaviour of energy E_{bulk} over time for one initial condition ϕ_0.

5.2 Numerical mass conservation

The analytical CH equation in (2.1) is mass conservative as shown in (2.7). For numerical experiments we observe the average value of ϕ_{ij}^n on Ω_d.

$$\frac{\sum_{i,j\in\Omega_d} \phi_{ij}^n}{N^2} \qquad\qquad n \in \{0,\dots,64\}$$

Analytical mass conservation tells us

$$\frac{d}{dt}\int_\Omega \phi \, d\mathbf{x} = 0 \quad \text{in}\, \Omega \times (0,T) \tag{5.2}$$

Therefore the average value of $\phi(\vec{x}, t)\,, \vec{x} \in \Omega$ is constant in time. Hence, the average of our numerical solution should stay constant as well. In practice we observe slight fluctuations in Figure 5.2. Those however are close to machine precision and can therefore be ignored.

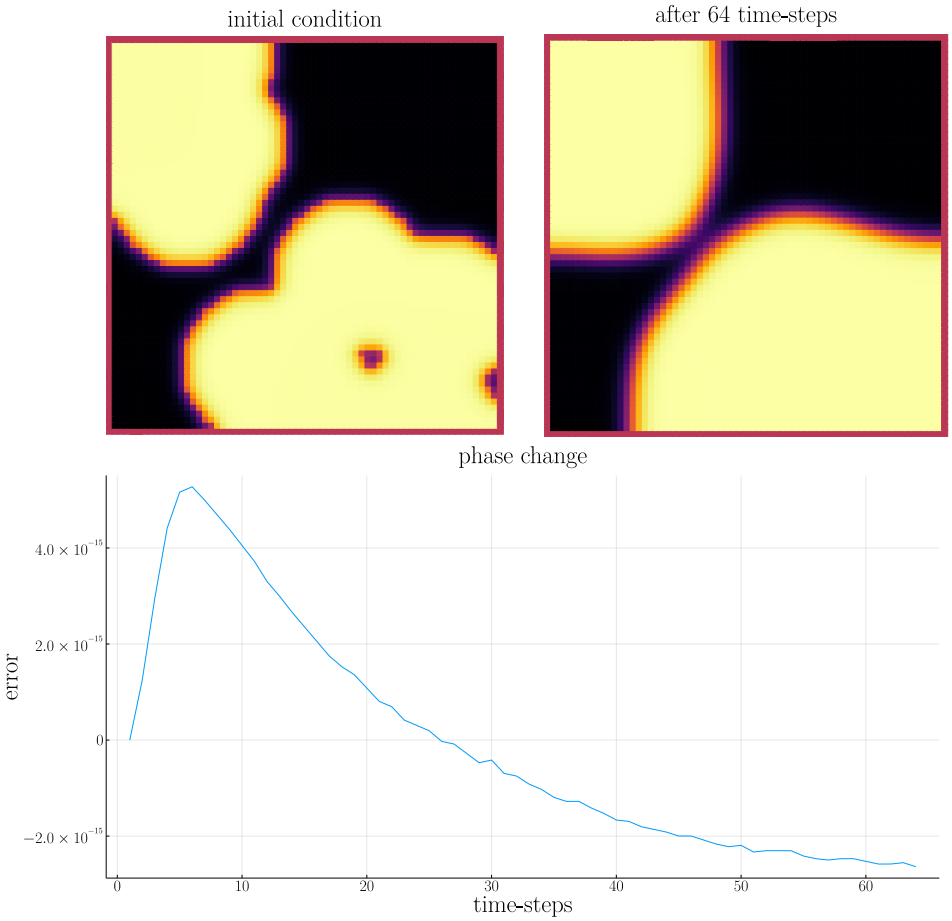

Figure 5.2: Behaviour of phase change over time for one initial condition ϕ_0.

5.3 Stability of a two-grid sub-iteration

We expect our solver to stay stable when increasing the number of two-grid sub-iterations. To validate this assumption we show convergence with the following Cauchy criterion.

$$\|\phi^{n+1,m-1} - \phi^{n+1,m}\|_{Fr} := \sqrt{\sum_{i,j \in \Omega_d} \left|\phi_{ij}^{n+1,m-1} - \phi_{ij}^{n+1,m}\right|^2} \tag{5.3}$$

We use similar criteria in the following sub chapters to show convergence for different hyperparameters. We expect sub-iterations to show Cauchy convergence, which is what we observe in Figure 5.3.

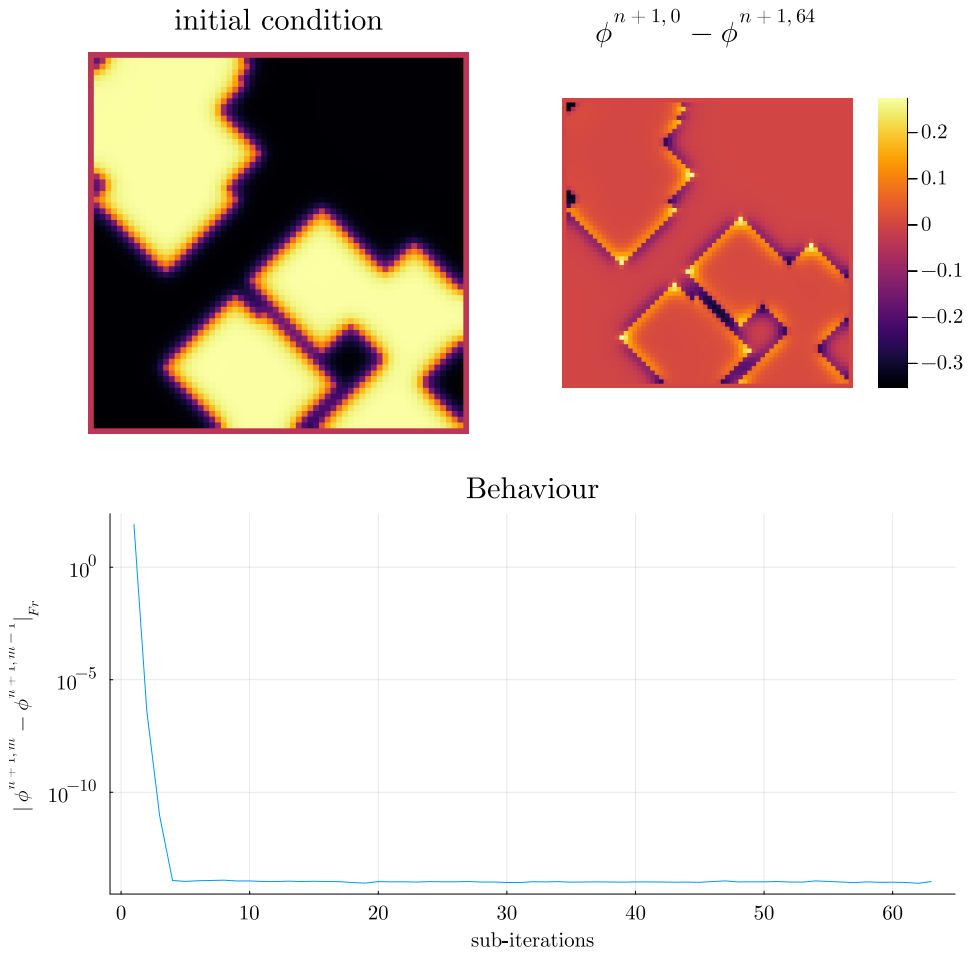

Figure 5.3: Stability of the original CH solver for increasing sub-iterations

During sub-iterations the convergence is exponential , and is reached after about 16 sub-iterations. The bend is only observed in the first time-step, and is likely due to the sharp interface in the initial values which is diffused during the first few sub-iterations. Looking at the difference before, and after one time step, it is apparent , that change is largest in areas with high curvature, which are mainly corners in the interface. Testing showed, that the number of sub-iterations required for convergence is dependant on the number of Gauss-Seidel iterations on each two-grid cycle. Though the general exponential behaviour stayed the same.

5.4 Stability in time

We expect our numerical error to decrease when calculating with smaller time steps. To test this, we successively subdivide the original time interval $[0, T]$ in finer parts. We fix $\Delta t \cdot n = T$ for $T = 10^{-2}$ and test different values of n. In Figure 5.4, as before, we employ a Cauchy criterion to compare the solution at $T = 10^{-2}$. We employ $\|\phi^{n,64} - \phi^{n-1,64}\|_{Fr}$ as measure.

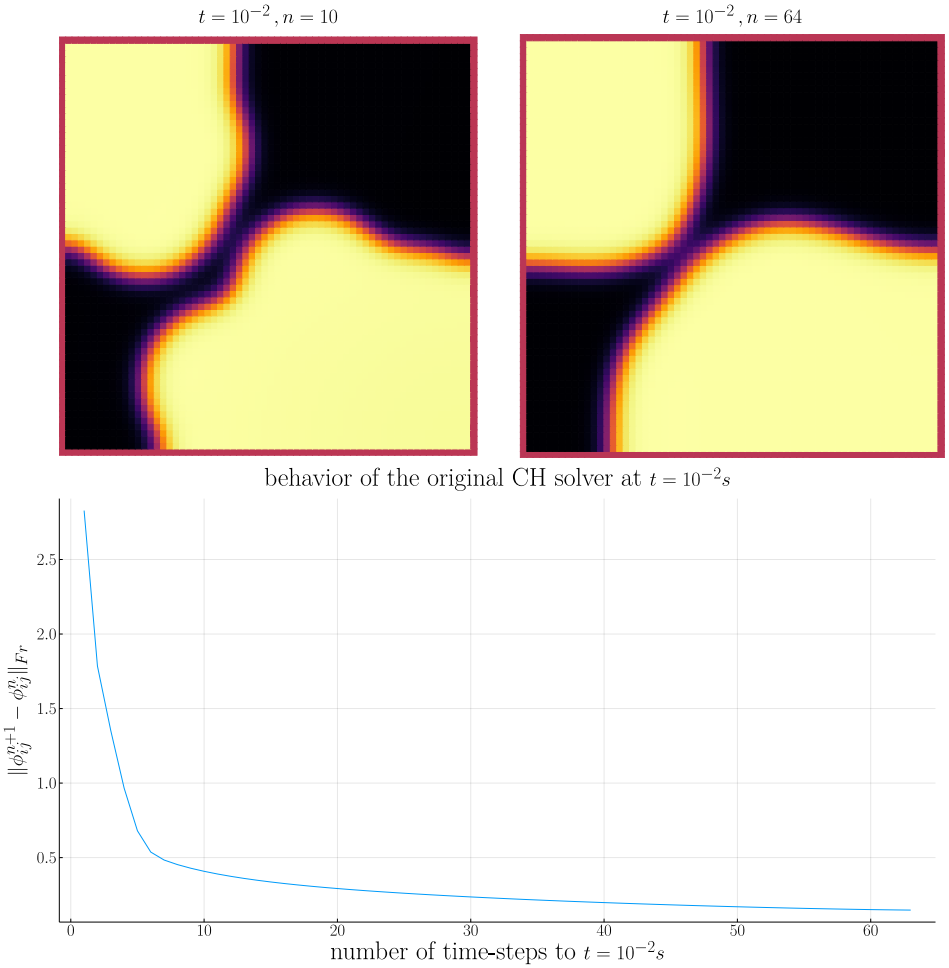

Figure 5.4: Behavior of the baseline solver while solving the time interval $T = [0, 10^{-2}]$ with increasing number of time-steps.

6 Relaxed Cahn-Hilliard equation

In effort to decrease the order of complexity, from fourth order derivative to second order, we propose an elliptical relaxation approach, where the relaxation variable c is the solution of the following elliptical PDE:

$$-\Delta c^\alpha + \alpha c^a = \alpha \phi^\alpha, \qquad \text{in} \quad \Omega \tag{6.1}$$

where α is a relaxation parameter. We expect to approach the original solution of the CH equation (2.1) as $\alpha \to \infty$. This results in the following relaxation for the classical CH equation

$$\begin{aligned} \partial_t \phi^\alpha &= \Delta \mu\,, && \text{in} \quad \Omega \times (0,T) \\ \mu &= -\varepsilon^2 \alpha (c^\alpha - \phi^\alpha) + W'(\phi). && \text{in} \quad \Omega \times (0,T) \end{aligned} \tag{6.2}$$

It requires solving the elliptical PDE for each time-step to calculate c.

6.1 Relaxed initial value-boundary problem

The aim of the relaxed CH equation is then to find solutions $\phi(\vec{x},t), \mu(\vec{x},t) : \Omega \times (0,T) \to \mathbb{R}$ such that they satisfy

$$\begin{aligned}
\partial_t \phi(x,t) &= \Delta \mu^\alpha \,, \\
\mu^\alpha &= -\varepsilon^2 \alpha (c^\alpha - \phi^\alpha) + W'(\phi) \,, \\
-\Delta c^\alpha + \alpha c^a &= \alpha \phi^\alpha, && \text{in} \quad \Omega \times (0,T \\
-\nabla \mu^\alpha \cdot \mathbf{n} &= 0 \,, \\
\nabla \phi^\alpha \cdot \mathbf{n} &= 0 \,, \\
\nabla c^\alpha \cdot \mathbf{n} &= 0 && \text{on} \, \partial\Omega \times (0,T) \,, \\
\phi^\alpha(\vec{x},0) &= \phi^0(\vec{x}) \,, \\
c^\alpha(\vec{x},0) &= \phi^0(\vec{x}) \,, && \text{in} \quad \Omega
\end{aligned} \tag{6.3}$$

6.2 Relaxed energy functional

Since the relaxed CH equation does not trivialy satisfy the same energy decay for (2.2), we show the existence of a similar equation for the relaxed problem. Motivated by the energy functional used in [1], we let $\phi_0 \in H^4(\Omega)$ and $T > 0$ be fixed. We assume, there exists a classical solution $\phi^\alpha, c^\alpha : \Omega \times (0,1) \to \mathbb{R}$ of (6.3). Then the energy functional for the relaxed CH equation for $\forall t \in (0,1)$ can be written as:

$$\frac{d}{dt} E_{rel}(\phi^\alpha, c^\alpha) := \frac{d}{dt} \int_\Omega \frac{1}{2} \varepsilon^2 \alpha (c^\alpha - \phi^\alpha)^2 + W(x) \, d\mathbf{x} \tag{6.4}$$

Which is derived by a L_2 inner product of (6.2) with μ^α.

$$\langle \phi_t^\alpha, \mu^\alpha \rangle = \langle \Delta \mu^\alpha, \mu^\alpha \rangle \tag{6.5}$$

it then follows for the left hand side:

$$\begin{aligned}
\langle \phi_t^\alpha, \mu^\alpha \rangle &= \langle \phi_t^\alpha, -\varepsilon^2 \alpha (c^\alpha - \phi^\alpha) + W'(\phi^\alpha) \rangle \\
&= \int_\Omega -\phi_t^\alpha \varepsilon^2 \alpha (c^\alpha - \phi^\alpha) \, d\mathbf{x} + \int_\Omega \phi_t^\alpha W'(\phi^\alpha) \, d\mathbf{x} \\
&= \frac{d}{dt} \int_\Omega \frac{1}{2} \varepsilon^2 \alpha (c^\alpha - \phi^\alpha)^2 \, d\mathbf{x} + \frac{d}{dt} \int_\Omega W'(\phi^\alpha) \, d\mathbf{x} \\
&= \frac{d}{dt} \int_\Omega \frac{1}{2} \varepsilon^2 \alpha (c^\alpha - \phi^\alpha)^2 + W(x) \, d\mathbf{x} = \frac{d}{dt} E_{rel}(\phi^\alpha, c^\alpha)
\end{aligned} \tag{6.6}$$

and using the boundary condition $(\nabla\mu \cdot \vec{n}) = 0$ on the right hand side:

$$\begin{aligned}\langle \Delta\mu^\alpha, \mu^\alpha \rangle &= \int_\Omega \mu^\alpha \Delta\mu^\alpha d\,\mathbf{x} \\ &= -\int_\Omega |\nabla\mu^\alpha| d\,\mathbf{x} + \int_{\partial\Omega} \mu^\alpha (\nabla\mu^\alpha \cdot \vec{n}) d\,\mathbf{A} \\ &= -\|\nabla\mu^\alpha\| \leq 0\end{aligned} \tag{6.7}$$

it therefore holds for a relaxed energy:

$$\frac{d}{dt} E_{rel}(\phi^\alpha, c^\alpha) = \frac{d}{dt} \int_\Omega \frac{1}{2}\varepsilon^2 \alpha (c-\phi)^2 + W(x)\; d\,\mathbf{x} \leq 0 \tag{6.8}$$

which gives a L_2 bound for $\Delta c = \alpha(c-\phi)$ and $\nabla\mu^\alpha$, similar to the estimate for $\nabla\mu$ given in the original CH equation.

6.3 Relaxed mass conservation

We use the same aproach as in (2.7) to show that the CH equation (6.2) is mass conservative.

$$\int_\Omega \partial_t \phi^\alpha \; d\,\mathbf{x} = \int_\Omega \Delta\mu^\alpha \; d\,\mathbf{x} = \int_{\partial\Omega} \nabla\mu^\alpha \cdot n \; d\,\mathbf{x} = 0 \qquad \forall t \in (0,T) \tag{6.9}$$

7 Discretization of the relaxed problem

As approach for the numerical solver for the CH equation we propose:

$$\begin{aligned}\frac{\phi_{ij}^{n+1,\alpha} - \phi_{ij}^{n,\alpha}}{\Delta t} &= \nabla_d \cdot (G_{ij} \nabla_d \mu_{ij}^{n+\frac{1}{2},\alpha}), \\ \mu_{ij}^{n+\frac{1}{2},\alpha} &= 2\phi_{ij}^{n+1,\alpha} - \varepsilon^2 a(c_{ij}^{n+1,\alpha} - \phi_{ij}^{n+1,\alpha}) + W'(\phi_{ij}^{n,\alpha}) - 2\phi_{ij}^{n,\alpha}. \quad i,j \in \{2,\dots,N+1\}\end{aligned} \tag{7.1}$$

This approach is inspired by (3.7) and adapted to the relaxed CH equation (7.1). We then apply the multi-grid method proposed in 4.2 to the relaxed problem by replacing the differential operators with their discrete counterparts, as defined in (3.4), and expand them. To solve the additional elliptical system, we propose a simple implicit finite difference scheme similar to what we use for the baseline solver.

$$-\nabla_d \cdot (G_{ij} \nabla_d c_{ij}^{n+1,\alpha}) + \alpha c_{ij}^{n+1,\alpha} = \alpha \phi_{ij}^{n+1,\alpha}, \qquad i,j \in \{2,\dots,N+1\}$$

7.1 Elliptical PDE

We then use the finite differences defined in (3.4) to derive the corresponding linear system.

$$\begin{aligned}-\frac{1}{h^2}(&G_{i+\frac{1}{2}j}(c_{i+1j}^{n+1,\alpha} - c_{ij}^{n+1,\alpha}) \\ &+G_{ij+\frac{1}{2}}(c_{ij+1}^{n+1,\alpha} - c_{ij}^{n+1,\alpha}) \\ &+G_{i-\frac{1}{2}j}(c_{i-1j}^{n+1,\alpha} - c_{ij}^{n+1,\alpha}) \\ +G_{ij-\frac{1}{2}}(&c_{ij-1}^{n+1,\alpha} - c_{ij}^{n+1,\alpha})) + \alpha c_{ij}^{n+1\alpha} = \alpha \phi_{ij}^{n+1,\alpha}, \qquad i,j \in \{2,\dots,N+1\}\end{aligned}$$

We abbreviate $\Sigma_G c_{ij}^{n+1,\alpha} = G_{i+\frac{1}{2}j} c_{i+1j}^{n+1,\alpha} + G_{i-\frac{1}{2}j} c_{i-1j}^{n+1,\alpha} + G_{ij+\frac{1}{2}} c_{ij+1}^{n+1,\alpha} + G_{ij-\frac{1}{2}} c_{ij-1}^{n+1,\alpha}$ and $\Sigma_{Gij} = G_{i+\frac{1}{2}j} + G_{i-\frac{1}{2}j} + G_{ij+\frac{1}{2}} + G_{ij-\frac{1}{2}}$. Then the discrete elliptical PDE can be stated as:

$$-\frac{\Sigma_G c_{ij}^{n+1,\alpha}}{h^2} + \frac{\Sigma_G}{h^2} c_{ij}^{n+1,\alpha} + \alpha c_{ij}^{n+1,\alpha} = \alpha \phi_{ij}^{n+1,\alpha}. \quad i,j \in \{2,\dots,N+1\}, \alpha \in \mathbb{R}^+ \tag{7.2}$$

this constitutes a linear system with $N \times N$ equations

7.2 Relaxed system

#+end$_{\text{src}}$ We use the same discretization approach, as for the baseline system. We reformulate the discretization (7.1) in terms of the relaxed function L_r as follows:

$$L_r \begin{pmatrix} \phi_{ij}^{n+1,\alpha} \\ \mu_{ij}^{n+\frac{1}{2},\alpha} \end{pmatrix} = \begin{pmatrix} \frac{\phi_{ij}^{n+1,m,\alpha}}{\Delta t} - \nabla_d \cdot (G_{ji} \nabla_d \mu_{ji}^{n+\frac{1}{2},m,\alpha}) \\ \varepsilon^2 \alpha (c_{ij}^{\alpha} - \phi_{ij}^{n+1,m,\alpha}) - 2\phi_{ij}^{n+1,m,\alpha} - \mu_{ji}^{n+\frac{1}{2},m,\alpha} \end{pmatrix} \quad \forall i,j \in \{2,\dots,N+1\}$$

```
function L(solver::relaxed_multi_solver,i,j , phi , mu)
    xi = solver.phase[i, j] / solver.dt -
         (discrete_G_weigted_neigbour_sum(i, j, solver.potential, G, solver.len,
         ↪  solver.width)
          -
          neighbours_in_domain(i, j, G, solver.len, solver.width) * mu
          ↪  )/solver.h^2
    psi = solver.epsilon^2 * solver.alpha*(solver.c[i,j] - phi) -
    ↪  solver.potential[i,j] - 2 * solver.phase[i,j]
    return [xi, psi]
end
```

and its Jacobian:

$$DL_r \begin{pmatrix} \phi_{ij}^{n+1,\alpha,m} \\ \mu_{ij}^{n+\frac{1}{2},m,\alpha} \end{pmatrix} = \begin{pmatrix} \frac{1}{\Delta t} & \frac{1}{h^2}\Sigma_G \\ -\varepsilon^2\alpha - 2 & 1 \end{pmatrix} \qquad \forall i,j \in \{2,\dots,N+1\}$$

```
function dL(solver::relaxed_multi_solver , i , j)
    return [ (1/solver.dt) (1/solver.h^2*neighbours_in_domain(i,j,G,solver.len ,
    ↪  solver.width));
             (-1*solver.epsilon^2 * solver.alpha  - 2) 1]
    end
```

Much like in the original solver in (3.11) now write (7.4) in terms of the function above.

$$L_r \begin{pmatrix} \phi_{ij}^{n+1,\alpha} \\ \mu_{ij}^{n+\frac{1}{2},\alpha} \end{pmatrix} = \begin{pmatrix} \zeta_{ij}^{n} \\ \psi_{ij}^{n} \end{pmatrix}, \qquad \forall i,j \in \{2,\dots,N+1\} \tag{7.3}$$

where $\left(\zeta_{ij}^{n}, \psi_{ij}^{n}\right)$ are the same in the original and relaxed solvers. Since the relaxed CH equation is no longer second order in both directions the resulting LES is simpler. To take advantage of this, we resolve the system algebraically for each grid-point $(i.j) \in \{2,\dots,N+1\}$.

$$-\frac{\Sigma_{Gij}}{h^2}\mu_{ji}^{n+\frac{1}{2},m,\alpha} = \frac{\phi_{ij}^{n+1,m,\alpha}}{\Delta t} - \zeta_{ij}^{n,\alpha} - \frac{\Sigma_G \mu_{ij}}{h^2}\,, \tag{7.4}$$

$$\varepsilon^2\alpha\phi_{ij}^{n+1,m,\alpha} + 2\phi_{ij}^{n+1,m,\alpha} = \varepsilon^2\alpha c_{ij}^{n,\alpha} - \mu_{ji}^{n+\frac{1}{2},m,\alpha} - \psi_{ij}^{n,\alpha}\,, \tag{7.5}$$

where

- $\Sigma_G\mu_{ij} = G_{i+\frac{1}{2}j}\mu_{i+1j}^{n+\frac{1}{2},m} + G_{i-\frac{1}{2}j}\mu_{i-1j}^{n+\frac{1}{2},m} + G_{ij+\frac{1}{2}}\mu_{ij+1}^{n+\frac{1}{2},m} + G_{ij-\frac{1}{2}}\mu_{ij-1}^{n+\frac{1}{2},m}$,

We simplify (7.4) by substituting $\mu_{ij}^{n+1,\alpha}$ from the first line into the second.

$$\varepsilon^2\alpha(\phi_{ij}^{n+1,m,\alpha}) + 2\phi_{ij}^{n+1,m,\alpha} = \varepsilon^2\alpha c^{\alpha} - \frac{h^2}{\Sigma_G}\left(\frac{\phi_{ij}^{n+1,m,\alpha}}{\Delta t} - \zeta_{ij}^{n} - \frac{1}{h^2}\Sigma_G\mu_{ij}\right) - \psi_{ij}$$

We solve this system for $\phi_{ij}^{n+1,m,\alpha}$. This results in the following system

$$\begin{aligned}
\phi_{ij}^{n+1,m,\alpha} &= \left(\varepsilon^2\alpha c^{\alpha} - \frac{h^2}{\Sigma_G}\left(-\zeta_{ij}^{n} - \frac{\Sigma_G\mu_{ij}}{h^2}\right) - \psi_{ij}\right)\left(\varepsilon^2\alpha + 2 + \frac{h^2}{\Sigma_G\Delta t}\right)^{-1} \\
\mu_{ij}^{n+\frac{1}{2},m,\alpha} &= \frac{h^2}{\Sigma_G}\left(\frac{\phi_{ij}^{n+1,m,\alpha}}{\Delta t} - \zeta_{ij}^{n} - \frac{1}{h^2}\Sigma_G\mu_{ij}\right) \qquad \forall i,j \in \{2,\dots,N+1\}
\end{aligned} \tag{7.6}$$

8 Relaxed numerical solver

8.1 Gauss Seidel solver for the elliptical system

To solve the elliptical system, we introduce a Gauss-Seidel solver similar to the Gauss-Seidel Solver used for the smoothing step in the two-grid method. We define this iteration in terms of s. For the Gauss-Seidel Iterative solver, we define the abbreviations

$$\Sigma_G c_{ij}^{n+1,\alpha,s+\frac{1}{2}} = G_{i+\frac{1}{2}j} c_{i+1j}^{n+1,\alpha,s} + G_{i-\frac{1}{2}j} c_{i-1j}^{n+1,\alpha,s+1} + G_{ij+\frac{1}{2}} c_{ij+1}^{n+1,\alpha,s} + G_{ij-\frac{1}{2}} c_{ij-1}^{n+1,\alpha,s+1}$$

We then define the Gaus-Seidel iteration by the following, and solve algebraically for $c_{ij}^{n+1,\alpha,s+1}$

$$\left(\frac{\Sigma_{Gij}}{h^2} + \alpha\right) c_{ij}^{n+1,\alpha,s+1} = \alpha\phi_{ij}^{n+1,\alpha} + \frac{\Sigma_G c_{ij}^{n+1,\alpha,s+\frac{1}{2}}}{h^2}$$

$$c_{ij}^{n+1,\alpha,s+1} = \frac{\alpha\phi_{ij}^{n+1,\alpha} + \frac{\Sigma_G c_{ij}^{n+1,\alpha,s+\frac{1}{2}}}{h^2}}{\frac{\Sigma_G}{h^2} + \alpha}$$

$$c_{ij}^{n+1,\alpha,s+1} = \frac{\alpha h^2 \phi_{ij}^{n+1,\alpha}}{\Sigma_{Gij} + \alpha h^2} + \frac{\Sigma_G c_{ij}^{n+1,\alpha,s+\frac{1}{2}}}{\Sigma_{Gij} + \alpha h^2}$$

We the Gaus-Seidel solver for **1000** iterations to ensure convergence. Furthermore we denote the solution of the iterative solver with $c_{ij}^{n+1,\alpha}$. We implement the corresponding iteration as follows: #+begin$_{\text{src}}$ julia :eval never :tangle src/elypssolver.jl function elyps$_{\text{solver}}$!(solver::T, n) where T <: Union{relaxed$_{\text{multisolver}}$, adapted$_{\text{relaxedmultisolver}}$} for k in 1:n for i = 2:(solver.len+1) for j = 2:(solver.width+1) bordernumber = neighbours$_{\text{indomain}}$(i, j,G, solver.len, solver.width) solver.c[i, j] = (solver.alpha * solver.phase[i, j] + discrete$_{\text{Gweigtedneigboursum}}$(i, j, solver.c, G, solver.len, solver.width) / solver.h^2) / (bordernumber / solver.h^2 + solver.alpha)

end end end end

8.2 Relaxed Gauss-Seidel iteration

Similar to (4.1), we derive a Gauss-Seidel iteration for the relaxed problem from (7.6).

$$\begin{aligned}\phi_{ij}^{n+1,\alpha,s+1} &= \left(\varepsilon^2\alpha c^\alpha - \frac{h^2}{\Sigma_G}(-\zeta_{ij}^n - \frac{\Sigma_G\mu_{ij}^{n+\frac{1}{2},\alpha,s+\frac{1}{2}}}{h^2}) - \psi_{ij}^n\right)\left(\varepsilon^2\alpha + 2 + \frac{h^2}{\Sigma_G\Delta t}\right)^{-1}\\ \mu_{ij}^{n+\frac{1}{2},\alpha,s+1} &= \frac{h^2}{\Sigma_G}(\frac{\phi_{ij}^{n+1,\alpha,s+1}}{\Delta t} - \zeta_{ij}^n - \frac{1}{h^2}\Sigma_G\mu_{ij}^{n+\frac{1}{2},\alpha,s+\frac{1}{2}})\end{aligned} \tag{8.1}$$

where

- $\Sigma_G\mu_{ij}^{n+\frac{1}{2},\alpha,s+\frac{1}{2}} = G_{i+\frac{1}{2}j}\mu_{i+1j}^{n+\frac{1}{2},s} + G_{i-\frac{1}{2}j}\mu_{i-1j}^{n+\frac{1}{2},s+1} + G_{ij+\frac{1}{2}}\mu_{ij+1}^{n+\frac{1}{2},s} + G_{ij-\frac{1}{2}}\mu_{ij-1}^{n+\frac{1}{2},s+1}$,

Contrary to the Gauss-Seidel iteration in the baseline solver, this iteration is significantly cheaper to calculate, since it no longer requires solving a 2x2 LES for each grid-point. We implement the iteration as:

```
function SMOOTH!(
    solver::T,
    iterations,
    adaptive
) where T <: Union{relaxed_multi_solver , adapted_relaxed_multi_solver}
    for k = 1:iterations
        # old_phase = copy(solver.phase)
        for I in CartesianIndices(solver.phase)[2:end-1, 2:end-1]
            i, j = I.I
            <<solve-for-phi>>
            <<update-potential>>
        end

        #if adaptive && LinearAlgebra.norm(old_phase - solver.phase) < 1e-10
            ##println("SMOOTH terminated at $(k) succesfully")
            #break
        #end
    end
end
```

8.3 The relaxed two-grid method

As the difference between both methods is abstracted away in the operators, the relaxed V-cycle replaces the original operators with their relaxed counterparts. Due to

julias multiple dispatch feature this changes nothing in the implementation Therefore we reuse the original V-cycle as defined in the 4.2. In the executions for each time step, we add the elliptic solver inside the sub-iteration. The iterative solver is then defined as:

```
for j in 1:timesteps

    set_xi_and_psi!(solvers[1])

    for i = 1:subiterations

        elyps_solver!(solvers[1] , 1000)
        v_cycle!(solvers, 1)
    end
end
```

9 Discrete mass conservation

Since both the CH equation (2.1) and the baseline solver from Fig.5.2 are mass conservative, the relaxed solver should be as well. Mass conservation for the CH equation is given as

$$\int_{\Omega} \partial_t \phi = 0 \tag{9.1}$$

We show a discrete analogue for both the baseline and the relaxed approach

$$\sum_{i,j \in \Omega_d} \frac{1}{\Delta t}(\phi_{ij}^{n+1} - \phi_{ij}^{n}) = 0. \tag{9.2}$$

We show this for a square domain Ω_d using the first line of (3.7) and (7.1) respectively.

$$\sum_{i=2}^{N+1}\sum_{j=2}^{N+1} \frac{1}{\Delta t}\left(\phi_{ij}^{n+1} - \phi_{ij}^{n}\right) = \sum_{i=2}^{N+1}\sum_{j=2}^{N+1} \nabla_d \cdot (G_{ij} \nabla_d \mu_{ij}^{n+\frac{1}{2}}) \tag{9.3}$$

We split the right double sum into three parts. We consider them separately. The first part consists of the inner sum, where $G_{i+\frac{1}{2}j} = G_{i+\frac{1}{2}j} = G_{ij+\frac{1}{2}} = G_{ij-\frac{1}{2}} = 0$. The inner sum can therefore be written as such:

$$= \sum_{i=3}^{N}\sum_{j=3}^{N} \frac{1}{h^2}\left(\mu_{i+1j}^{n+\frac{1}{2}} + \mu_{i-1j}^{n+\frac{1}{2}} + \mu_{ij+1}^{n+\frac{1}{2}} + \mu_{ij-1}^{n+\frac{1}{2}} - 4\mu_{ij}^{n+\frac{1}{2}}\right) \tag{9.4}$$

The second part consists of the sums over the edges excluding the corners.

$$\begin{aligned}
&+\sum_{i=3}^{N}\frac{\Sigma_G\mu_{i2}^{n+\frac{1}{2}}-\Sigma_{Gi2}\cdot\mu_{i2}^{n+\frac{1}{2}}}{h^2}\\
&+\sum_{i=3}^{N}\frac{\Sigma_G\mu_{iN+1}^{n+\frac{1}{2}}-\Sigma_{GiN+1}\cdot\mu_{iN+1}^{n+\frac{1}{2}}}{h^2}\\
&+\sum_{j=3}^{N}\frac{\Sigma_G\mu_{i2}^{n+\frac{1}{2}}-\Sigma_{Gi2}\cdot\mu_{i2}^{n+\frac{1}{2}}}{h^2}\\
&+\sum_{j=3}^{N}\frac{\Sigma_G\mu_{N+1j}^{n+\frac{1}{2}}-\Sigma_{GN+1j}\cdot\mu_{N+1j}^{n+\frac{1}{2}}}{h^2}
\end{aligned}\tag{9.5}$$

And the third part consists of the corners.

$$\begin{aligned}
&+\frac{\Sigma_G\mu_{N+1N+1}^{n+\frac{1}{2}}-\Sigma_{GN+1,N+1}\cdot\mu_{N+1,N+1}^{n+\frac{1}{2}}}{h^2}\\
&+\frac{\Sigma_G\mu_{N+1,2}^{n+\frac{1}{2}}-\Sigma_{GN+1,2}\cdot\mu_{N+1,2}^{n+\frac{1}{2}}}{h^2}\\
&+\frac{\Sigma_G\mu_{2,N+1}^{n+\frac{1}{2}}-\Sigma_{G2,N+1}\cdot\mu_{2,N+1}^{n+\frac{1}{2}}}{h^2}\\
&+\frac{\Sigma_G\mu_{2,2}^{n+\frac{1}{2}}-\Sigma_{G2,2}\cdot\mu_{2,2}^{n+\frac{1}{2}}}{h^2}
\end{aligned}\tag{9.6}$$

The first double sum is a telescopic sum, and contracts to the following:

$$\begin{aligned}
\sum_{i=3}^{N}\sum_{j=3}^{N}\frac{1}{h^2}\left(\mu_{i+1j}^{n+\frac{1}{2}}+\mu_{i-1j}^{n+\frac{1}{2}}+\mu_{ij+1}^{n+\frac{1}{2}}+\mu_{ij-1}^{n+\frac{1}{2}}-4\mu_{ij}^{n+\frac{1}{2}}\right) &= \sum_{i=3}^{N}\mu_{i2}^{n+\frac{1}{2}}-\mu_{i3}^{n+\frac{1}{2}}\\
&+\sum_{i=3}^{N}\mu_{iN+1}^{n+\frac{1}{2}}-\mu_{iN}^{n+\frac{1}{2}}\\
&+\sum_{j=3}^{N}\mu_{2j}^{n+\frac{1}{2}}-\mu_{3j}^{n+\frac{1}{2}}\\
&+\sum_{j=3}^{N}\mu_{N+1j}^{n+\frac{1}{2}}-\mu_{Nj}^{n+\frac{1}{2}}
\end{aligned}\tag{9.7}$$

Additionally, we simplify each of the sums in the second part, since the values of G are known on the boundary. On the right boundary ,for $2 < i < N+1$, $G_{iN+\frac{3}{2}} = 0$ and $G_{iN+\frac{1}{2}} = G_{i+\frac{1}{2}N+1} = 1$ it therefore follows:

$$\begin{aligned}\sum_{i=3}^{N} \frac{\Sigma_G \mu_{iN+1}^{n+\frac{1}{2}} - \Sigma_{GiN+1} \cdot \mu_{iN+1}^{n+\frac{1}{2}}}{h^2} &= \frac{1}{h^2} \sum_{i=3}^{N} G_{i+\frac{1}{2}N+1} \mu_{i+1N+1}^{n+\frac{1}{2}} + G_{i-\frac{1}{2}N+1} \mu_{i-1N+1}^{n+\frac{1}{2}} \\ &\quad + G_{iN+\frac{3}{2}} \mu_{iN+2}^{n+\frac{1}{2}} + G_{iN-\frac{3}{2}} \mu_{iN}^{n+\frac{1}{2}} \\ &\quad - (G_{iN+\frac{3}{2}} + G_{iN+\frac{1}{2}} + G_{i+\frac{1}{2}N+1} + G_{i-\frac{1}{2}N+1}) \mu_{iN+1}^{n+\frac{1}{2}} \\ &= \frac{1}{h^2} \sum_{i=3}^{N} \mu_{i+1N+1}^{n+\frac{1}{2}} + \mu_{i-1N+1}^{n+\frac{1}{2}} + \mu_{iN}^{n+\frac{1}{2}} - 3\mu_{iN+1}^{n+\frac{1}{2}}\end{aligned} \tag{9.8}$$

this sum, as it it is telescopic simplify further to

$$\begin{aligned}\sum_{i=3}^{N} \frac{\Sigma_G \mu_{iN+1}^{n+\frac{1}{2}} - \Sigma_{GiN+1} \cdot \mu_{iN+1}^{n+\frac{1}{2}}}{h^2} &= (\mu_{NN+1} + \mu_{3N+1}) - (\mu_{N+1N+1} + \mu_{2N+1}) \\ &\quad - \sum_{i=3}^{N} \mu_{i1N}^{n+\frac{1}{2}} - \mu_{iN+1}^{n+\frac{1}{2}}\end{aligned} \tag{9.9}$$

similar the other three sums on the boundary simplify to

$$\begin{aligned}\sum_{i=3}^{N} \frac{\Sigma_G \mu_{i2}^{n+\frac{1}{2}} - \Sigma_{Gi2} \cdot \mu_{i2}^{n+\frac{1}{2}}}{h^2} &= (\mu_{N,2} + \mu_{3,2}) - (\mu_{N+1,2} + \mu_{2,2}) - \sum_{i=3}^{N} \mu_{i1N}^{n+\frac{1}{2}} - \mu_{iN+1}^{n+\frac{1}{2}} \\ \sum_{j=3}^{N} \frac{\Sigma_G \mu_{2j}^{n+\frac{1}{2},\alpha} - \Sigma_{G2j} \cdot \mu_{2j}^{n+\frac{1}{2}}}{h^2} &= (\mu_{2,3} + \mu_{2,N}) - (\mu_{2,N+1} + \mu_{2,2}) - \sum_{j=3}^{N} \mu_{2j}^{n+\frac{1}{2}} - \mu_{3j}^{n+\frac{1}{2}} \\ \sum_{j=3}^{N} \frac{\Sigma_G \mu_{N+1j}^{n+\frac{1}{2}} - \Sigma_{GN+1j} \cdot \mu_{N+1j}^{n+\frac{1}{2}}}{h^2} &= (\mu_{N+1,N} + \mu_{N+1,3}) - (\mu_{N+1,N+1} + \mu_{N+1,2}) - \sum_{j=3}^{N} \mu_{N+1,j}^{n+\frac{1}{2}} - \mu_{N,j}^{n+\frac{1}{2}}\end{aligned} \tag{9.10}$$

we observe that the resulting sums are equal and opposite to the result from the first sum. They therefore cancel each other and we are left with the corner terms. Those terms sum up to

$$\mu_{N+1,N} + \mu_{N+1,3} - 2\mu_{N+1,N+1} + + \mu_{2,3} + \mu_{2,N} - 2\mu_{2,2} + \mu_{N,2} + \mu_{N+1,3} - 2\mu_{2,N+1} + \mu_{N,N+1} + \mu_{3,N+1} - 2\mu_{N+1,2} \tag{9.11}$$

The third sum, on the corners, can be simplified the same way as the two others.

$$\begin{aligned}
\frac{\Sigma_G \mu_{N+1N+1}^{n+\frac{1}{2}} - \Sigma_{GN+1,N+1} \cdot \mu_{N+1,N+1}^{n+\frac{1}{2}}}{h^2} &= \frac{1}{h^2}(\mu_{NN+1}^{n+\frac{1}{2}} + \mu_{N+1N}^{n+\frac{1}{2}} - 2\mu_{N+1N}^{n+\frac{1}{2}}) \\
\frac{\Sigma_G \mu_{2,2}^{n+\frac{1}{2}} - \Sigma_{G2,2} \cdot \mu_{2,2}^{n+\frac{1}{2}}}{h^2} &= \frac{1}{h^2}(\mu_{3,2}^{n+\frac{1}{2}} + \mu_{2,3}^{n+\frac{1}{2}} - 2\mu_{2,2}^{n+\frac{1}{2}}) \\
\frac{\Sigma_G \mu_{2N+1}^{n+\frac{1}{2}} - \Sigma_{G2,N+1} \cdot \mu_{2,N+1}^{n+\frac{1}{2}}}{h^2} &= \frac{1}{h^2}(\mu_{3N+1}^{n+\frac{1}{2}} + \mu_{2N}^{n+\frac{1}{2}} - 2\mu_{2N+1}^{n+\frac{1}{2}}) \\
\frac{\Sigma_G \mu_{N+1,2}^{n+\frac{1}{2}} - \Sigma_{GN+1,2} \cdot \mu_{N+1,2}^{n+\frac{1}{2}}}{h^2} &= \frac{1}{h^2}(\mu_{N+1,3}^{n+\frac{1}{2}} + \mu_{N,2}^{n+\frac{1}{2}} - 2\mu_{N+1,2}^{n+\frac{1}{2}})
\end{aligned} \tag{9.12}$$

Those terms cancel out with what remains in the second sum. We therefore conclude

$$\sum_{i=2}^{N+1} \sum_{j=2}^{N+1} \frac{1}{\Delta t} \left(\phi_{ij}^{n+1} - \phi_{ij}^{n} \right) = 0 \tag{9.13}$$

Therefore, in (9.13), the discretizations for both, the baseline and the relaxed methods, have a discrete equivalent of mass conservation.

10 Numerical experiments for the relaxed system

We expect the relaxed solver to behave the same as the baseline method for all test cases that we have introduced in Chapter 5. Therefore we run the same experiments for our relaxed solver. If not mentioned otherwise, we use the following hyper-parameters

Variable:	ε	Δ t	h
Value:	$8 * 10^{-3}$	10^{-3}	$3*10^{-3}$

10.1 explicit and implicit solution of the elliptical problem

Initially we experimented with solving the elliptical problem explicitly at the beginning of each time-step. This resulted in inconsistent behavior. We show the extend of this in correlation to α and ε in 10.1 where we run 100 elliptical Gauss-Seidel iterations followed by 1000 Gauss-Seidel iterations for the relaxed problem. We observe difficulty in developing the diffuse interface of the CH equation.Furthermore, comparing the results for $\alpha = 10000, \varepsilon \in \{0.05, 0.025\}$ with the result of the original solver for $\varepsilon = 0.01$ might suggest to use a lower value of ε to mitigate this. However this resulted in unpredictable interface with in further time steps.

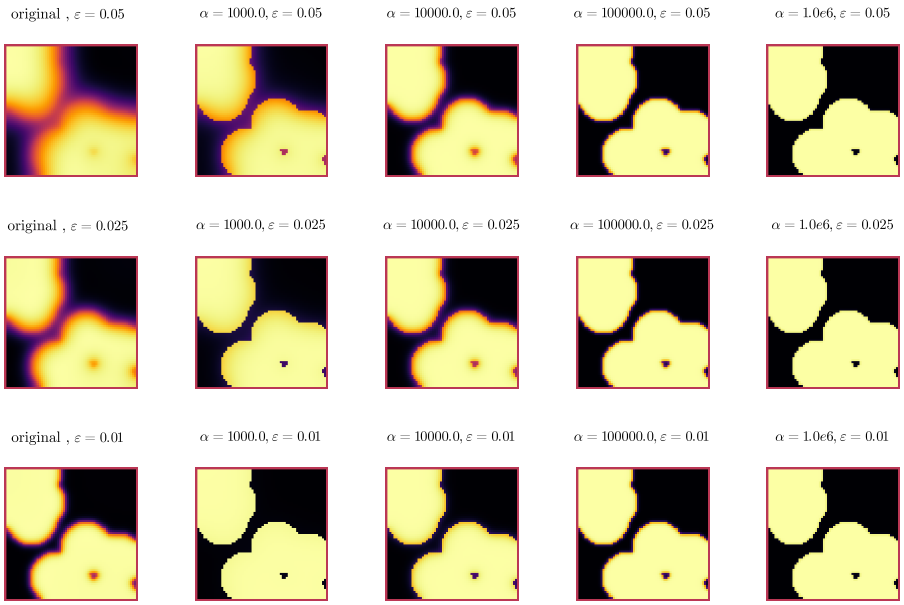

Figure 10.1: Effect of the relaxed SMOOTH operator for different values of α and ε

Since explicit solving (7.2) at the beginning of a time-step did not yield a predictable solver we experiment with solving (7.2) and (7.4) in an alternating manner. For 10.2 we run 100 elliptical then 100 relaxed iterations and repeat both 10 times. For this experiment we observe the relaxed solver to be significantly better in developing the same interface as the baseline solver for a fixed value of ε.

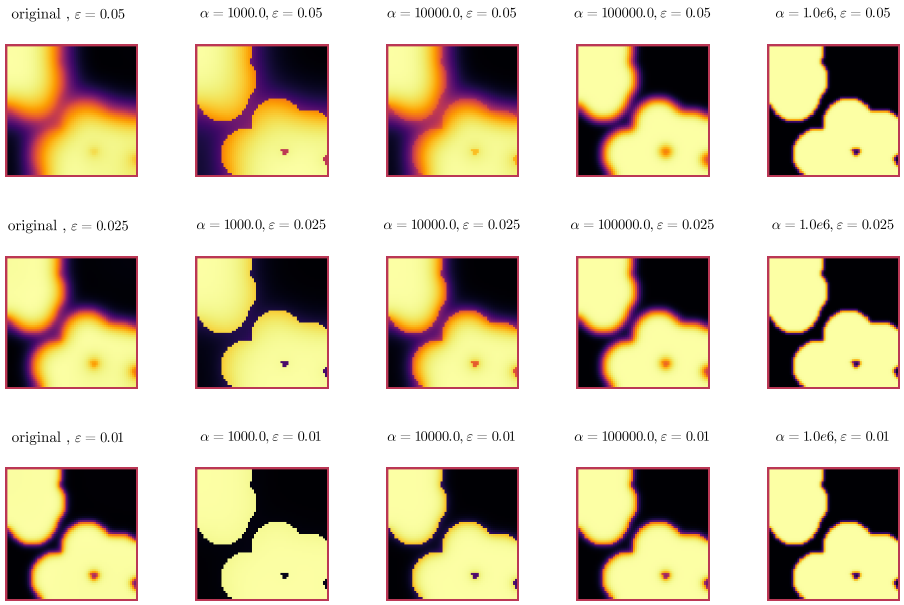

Figure 10.2: Effect of the relaxed SMOOTH operator, and additional solving of the elliptical problem, for different values of α and ε

The experimentation shows that α has an effect similar to ε, where it changes the boundary thickness in the phase-field ϕ. Therefore ε and α might be correlated. To test this hypothesis we use a simple Monte Carlo optimizer for α, ε.

10.2 OPTIMIZER FOR ALPHA

The Monte Carlo optimizer runs both solvers for one time-step and compares their phase-fields ϕ afterwards. It runs the relaxed solver with random values for ε, α in a normal distribution around the current optimum. This results in a optimal ε found that is very close to the ε used in the baseline (9e-3 compared to 8e-3). This gives us confidence that the relaxed method solves the same problem, with the same ε, as the baseline. Optimal values for α varied , however stayed fairly large around $10^5 \rightarrow 10^6$.

```
using Distributions
using DataFrames
using JLD2
include(pwd() * "/src/solvers.jl")
include(pwd() * "/src/adapted_solvers.jl")
```

```
include(pwd() * "/src/utils.jl")
include(pwd() * "/src/multisolver.jl")
include(pwd() * "/src/multi_relaxed.jl")
include(pwd() * "/src/testgrids.jl")
include(pwd() * "/src/elypssolver.jl")
using Plots
using LaTeXStrings
using LinearAlgebra
using Printf
using ProgressBars
default(fontfamily="computer modern" , titlefontsize=32 , guidefontsize=22 ,
↪  tickfontsize = 22 , legendfontsize=22)
pgfplotsx()
layout2x2 = grid(2,2)
layout3x1 = @layout [ b  c ; a]
size3x1 = (1600,1600)
SIZE = 64
M = testdata(SIZE, SIZE ÷ 5, SIZE /5 , 2)


function test_values(alpha_distribution::Distribution ,
↪  epsilon_distribution::Distribution , M)
    alpha = rand(alpha_distribution)
    eps = max(rand(epsilon_distribution)  ,1e-10)
    relaxed_solver = testgrid(relaxed_multi_solver, M, 2; alpha=alpha,
    ↪  epsilon=eps)
    set_xi_and_psi!(relaxed_solver[1])
    #SMOOTH!(relaxed_solver[1], 100, false)
    for j=1:64
    elyps_solver!(relaxed_solver[1], 2000)
    alt_v_cycle!(relaxed_solver , 1)
    end
    error = norm(relaxed_solver[1].phase .- original_solver[1].phase) /
    ↪  *(size(relaxed_solver[1].phase)...)
    return (;alpha=alpha , epsilon=eps , error=error)
end

original_solver = testgrid(multi_solver, M, 2)
set_xi_and_psi!(original_solver[1])
for j=1:64
alt_v_cycle!(original_solver , 1)
end
#SMOOTH!(original_solver[1], 100, false);
eps = 3e-3
```

```
#M = testdata(64, div(64,3), 64/5 , 2)
alpha0 = 10000
epsilon0 = 1e-2
best_alpha = alpha0 / 10
best_epsilon = epsilon0 / 10
best_error  = Inf
results = DataFrame()
for n=1:1000
    searchradius = 1
    alpha_distribution = Normal(best_alpha , searchradius * alpha0)
    epsilon_distribution = Normal(best_epsilon , searchradius * epsilon0)
    result = test_values(alpha_distribution , epsilon_distribution , M)
    if result.error < best_error
        global best_error = result.error
        global best_alpha = result.alpha
        global best_epsilon = result.epsilon
        println(result)
    end
push!(results , result)
end
jldsave("experiments/alpha-epsilon.jld2"; result=results)
println("Best alpha: $best_alpha , Best epsilon: $best_epsilon")
```

10.3 Effect of α on the Gauss-Seidel iteration

To see the impact of α with a fixed value of $\varepsilon = 8 * 10^{-3}$ on our solver, we evaluate both solvers after one time-step , and then calculate the difference between ϕ_{ij}^{n+1} and $\phi_{ij}^{n+1,\alpha}$, for various values of α. Since the solution of the relaxed solver should approach the original solver, we expect

$$||\phi^{n+1} - \phi^{n+1,\alpha}||_{Fr} \to 0. \tag{10.1}$$

In Fig.10.3 we observe the following behaviour where in all cases the difference between the relaxed solver and the original solver is apparent. Furthermore we observe a optimal value of α at approximately $7.5 * 10^5$. We explain this with our observations done for the Smoothing operator, where for small and large values of α the relaxed solver results in restricted behaviour, which we also expect. On the other hand, for large values of α the elliptical equation approaches ϕ, however it does not converge to ϕ for small values of α.

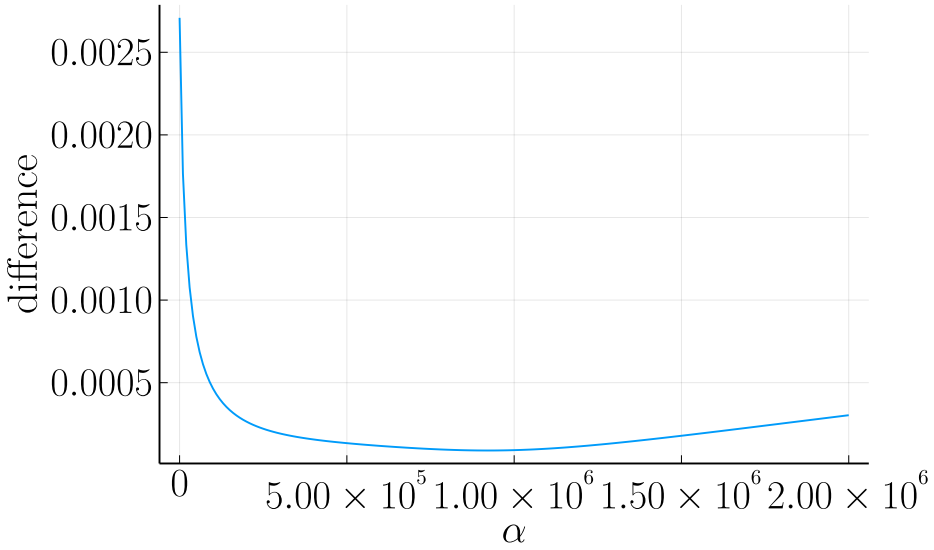

Figure 10.3: Difference between the original solver ϕ_{ij}^1 and the relaxed solver $\phi_{ij}^{1,\alpha}$ for different values of α

10.4 Direct comparison of the baseline solver with the relaxed solver

Then we show a comparison of both solvers by plotting the phase-fields after 64 time-steps, and the difference $\|\phi^{n+1} - \phi^{n+1,\alpha}\|_{Fr}$ over the time-steps $n \in \{0, \ldots, 63\}$. We can observe slight differences between the original solver and the relaxed solver. To quantify those, we run the relaxed solver for a fixed value of $\alpha = 7700$, as it is in the interval where α is minimal in Fig.10.3. We show the numerical difference between ϕ_{ij}^n and $\phi_{ij}^{n,\alpha}$ in Fig.10.4. The observed difference is mainly in areas with high curvature and inclusions of small segments of one phase in the other.

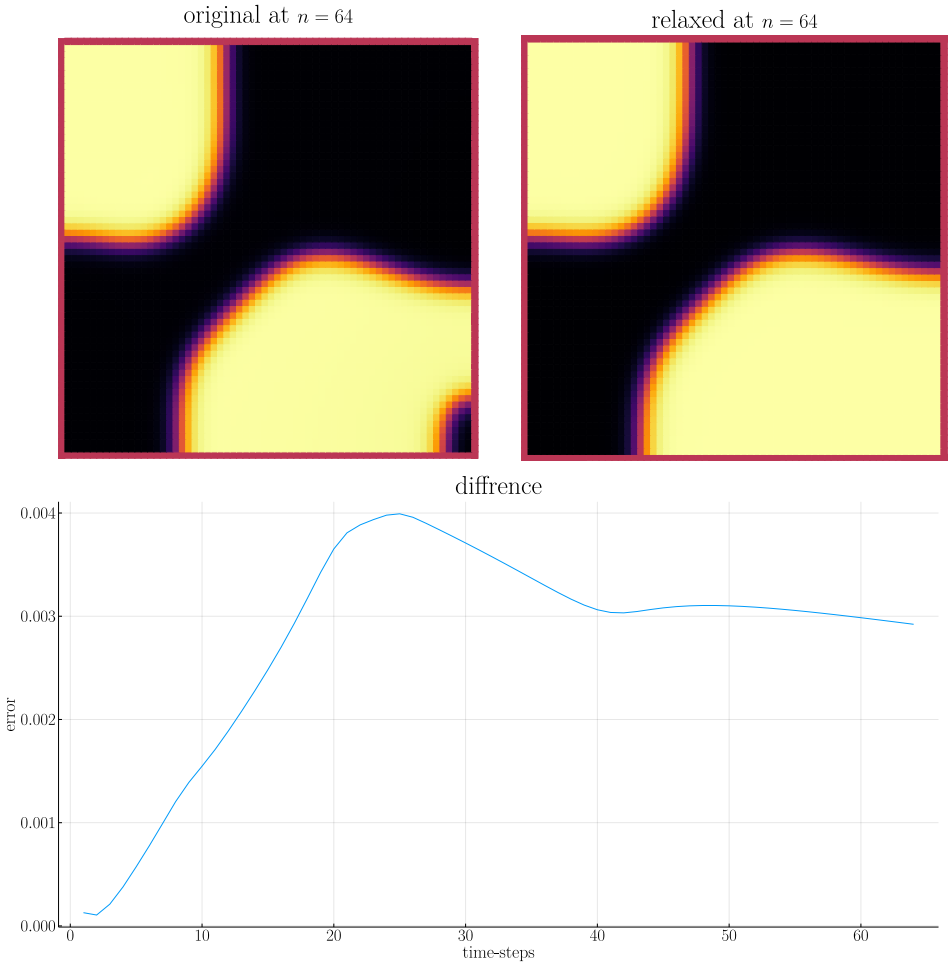

Figure 10.4: Comparison between the original and the relaxed CH solvers.

10.5 Relaxed energy evaluations

We do evaluate our relaxed method using the discrete energy defined in (5.1). On the same initial data, and with the same values for ε, h, dt as in the Chapter.5.1. Since we expect the relaxed approach to solve the same initial value problem (2.8), we expect both solvers to behave the same. In Figure.10.5 we then observe the energy decay we expected. Our relaxed approach closely follows the baseline, although it consistently decays slightly faster. Both solvers decay in a similar manner and both solvers show a slight bend after a few iterations. However the bend in the relaxed solver is noticeably more pronounced. Additionally, in later iterations the relaxed solver shows faster energy decay than the original. We explain these differences with

the observations taken in later experiments, where we observe mass-loss and slower convergence. We suspect the relaxed solver to therefore be more aggressive when minimizing energy.

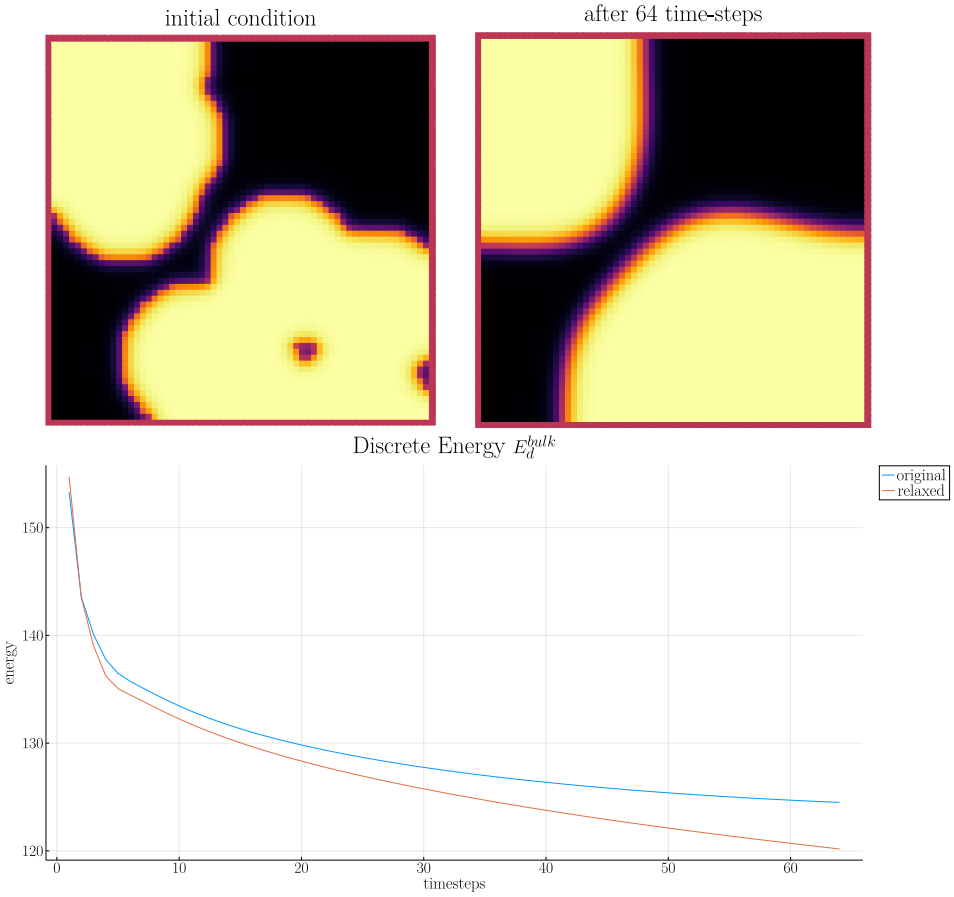

Figure 10.5: Energy decay of the relaxed solver compared to the original solver.

10.6 Stability of a relaxed two-grid sub-iteration

We use the same Cauchy criterion we used for the baseline solver. Furthermore, we compare the sub-iteration behaviour of the relaxed solver to the original we therefore plot $\|\phi_{ij}^{n+1,m} - \phi_{ij}^{n+2,m-1}\|_{Fr}$ against $\|\phi_{ij}^{n+1,m,\alpha} - \phi_{ij}^{n+1,m-1,\alpha}\|$ for $m \in \{2, \ldots, 64\}$. The sub-iterations in Fig.10.6 are stable. However, the relaxed solver shows significantly slower convergence compared to the baseline solver. Which is why without the log-log scale employed, the behavior of both solvers would not be visible in the same plot. This behaviour suggests that the relaxed solver does not converge to-

wards the solution of (7.6). Further experiments on mass conservation confirm this suspicion. During further experiments with different initial conditions and number of Gauss-Seidel steps, we where not able to change the slow convergence behaviour.

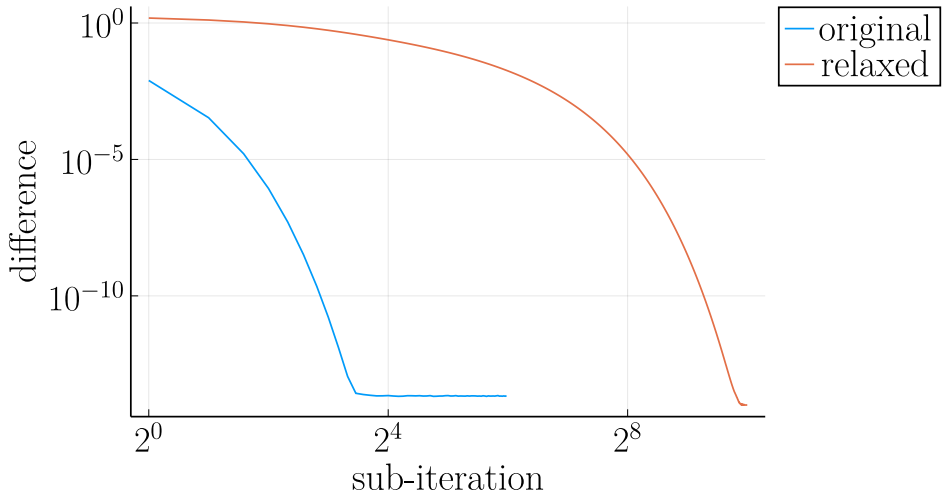

Figure 10.6: Cauchy convergence of the basline and relaxed solver, during sub-iteration V-cycles.

10.7 Relaxed numerical mass balance

As already mentioned the relaxed solver is not mass conservative. Our relaxed solver shows significant mass loss as seen in Fig.10.7, especially when compared to the original solver in Fig.5.2. Both the original approach and the relaxed one exhibit a discrete equivalent of mass conservation, therefore their difference has to be explained by the numerical solver. Which is consistent with the observations made with sub-iteration convergence. We therefore conclude that our relaxed solver does not properly converge. This, most likely, is due to our choice of alternating the solution of c and ϕ. because we intend to solve them both implicitly. Coupling the elliptical and relaxed CH equation might alleviate this, however the resulting system would be of similar complexity to (3.9), which is what we intend to prevent with the relaxation approach. Alternatively, solving c explicitly leads to an unstable solver with the initial conditions used by us. We did not test different initial conditions due to a lack of computational resources and time.

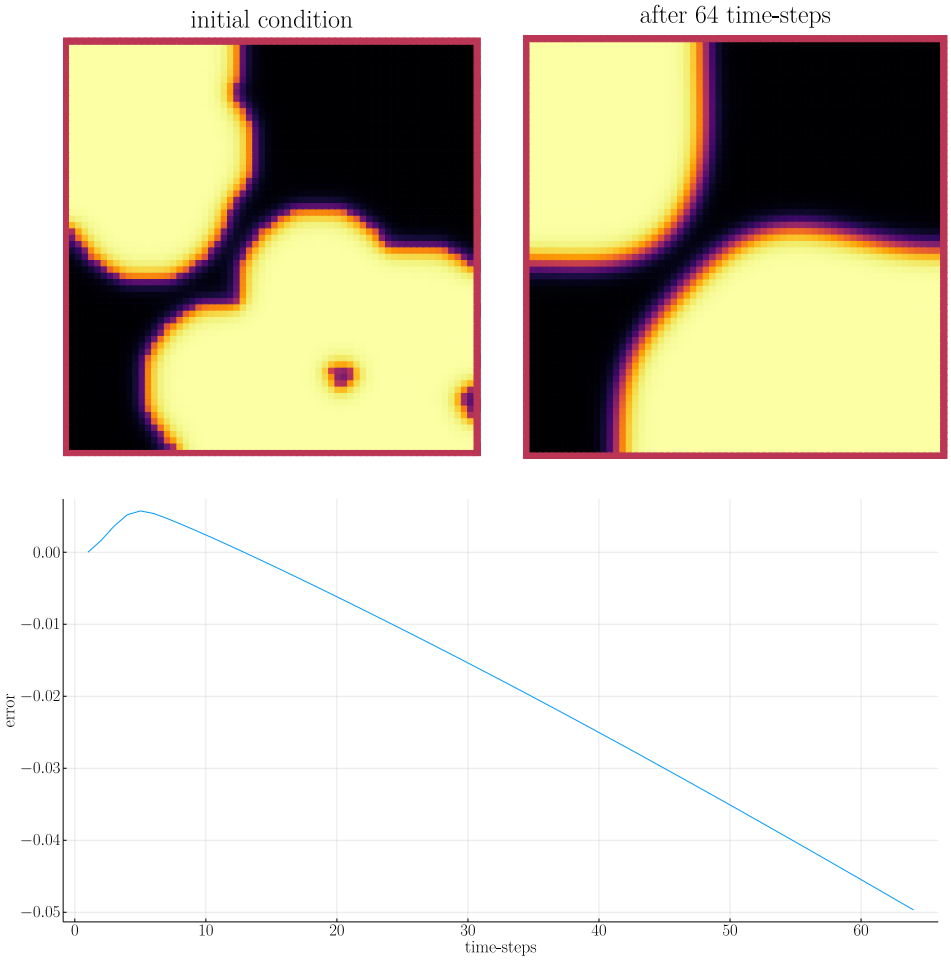

Figure 10.7: Mass loss in the relaxed solver

10.8 Relaxed stability in time

We test the behaviour under refinement in time by successive subdividing of the original time interval $[0, T]$ into finer parts. Then, in **??**, we observe the stability of the relaxed solver in time, wich is similar to the original solver. We compare $\|\phi^{n+1,\alpha}_{\Delta t=\frac{10^{-2}}{n+1}} - \phi^{n,\alpha}_{\Delta t=\frac{10^{-2}}{n}}\|_{Fr}$ at $t = 10^{-2}$, for wich the relaxed solver is consistently lower than the original solver. This might suggest a more consistent method over time. More likely however, this is due to the more aggressive energy decay, and non mass conservation.

10.9 ADDITIONAL CONSIDERATIONS

The approach for the relaxed solver was in our tests significantly faster than the baseline implementation. The run-time in practice is highly dependant on the amount of V-cycle iterations, for the baseline and relaxed solver respectively, as well as the number of Gauss-Seidel iterations for the elliptical problem. When using similar number of V-cycles and sufficiently small number of elliptical iterations, the relaxed solver is several time faster. When we let both solvers iterate until convergence, the relaxed solver still is slightly faster. However, as we will show in our experiments, the relaxed solver has problems with convergence and never converges to the solution of the baseline. Because of this, and due to potential improvements in our implementation, the previous statements are to be taken as a qualitative guide, and not as a quantitative statement, which would require further research.

11 Conclusion

In this thesis we have presented a simple introduction to the CH equation, following the authors in[2] we have shown how to discretize it, and have provided a numerical solver for it. Additionally, we introduced a elliptical relaxation approach for the CH equation that itself is mass conservative. We have presented a solver for the CH equation as a baseline method, and have shown how to derive a Gauss-Seidel based two-grid method from the authors [2] initial approach. For the relaxation approach, we introduced a finite difference discretization using methods that we used for the baseline. Building on this, we derive a Gauss-Seidel based two-grid method, that closely resembles the method we used for our baseline. In hindsight, we wouldn't have used a two-grid method for either solver, because the benefit of a multi-grid method, which is faster convergence, did not contribute to the goal of our Thesis. On the contrary, the added complexity made it harder to distinguish between errors in our numerical implementation, errors in the two-grid implementation, or systematic errors in the relaxation. Since the main goal of our thesis, was to compare the relaxation to our baseline, a Gauss-Seidel Method would have been sufficient for our experiments. For our experiments, we have introduced measures to evaluate the stability of both solvers in regard to time and mass conservation as well as their sub-iteration behaviour. We have observed the baseline to be mass conservative, in a numerical sense, and we have shown it to be stable in all tested measures. On the relaxed solver, we have experimentally tested the effect of the relaxation parameter α. We came to the conclusion, that the solver does not work for to small or to large values of α. We investigated the correlation between α and the interface parameter ε and found no direct correlation after switching to an implicit strategy for the elliptical parameter c. The explicit test was inconclusive. We ran an optimizer for α and ε that strongly suggested that ε is independent of the relaxation. We came to this conclusion since the ε, that best approximated the baseline results was close to the ε in the baseline. However, verifying this hypothesis would require, either numerical or statistical analysis, which we leave open for further research. We show that both discretizations satisfy a discrete version of conservation

of mass. However, since the discretization is implicit, this does not equate to a mass conservative solver. While the basline solver is mass conservative, the relaxed solver is not, which we have shown experimentally. On all other measures, the relaxed solver behaved similar to the baseline, and we explained the small discrepancies through the numerical error that causes mass loss. We intentionally didn't evaluate run-time since numerical experiments have shown both solvers to be dependant on the amount of sub-iterations, hyperparameters such as ε as well as the number off smoothing iterations. It would therefore be unfair to evaluate one solver on a set of parameters tweaked for the other. As example for this dilemma we recall runs where the relaxed solver was around 10x faster than the baseline with the same parameters. The baseline solver was able to run with 10x less smoothing iterations than the relaxed one. A fair comparison would hence require to find the optimal number of smoothing for each solver.

11.1 Outlook

This thesis leaves a lot of room for further research. We have already mentioned run-time evaluations, which require more optimizations, and additional experiments to test the number of smoothing iterations. Here it would be beneficial if both solvers are made adaptive, to ensure fair evaluations. Furthermore, we initially considered a machine learning approach to replace the elliptical system. We didn't follow this idea mostly due to time constraints, as we had already collected trainings data during our numerical experiments. Our choice of programming language would have been of benefit here, as it would enable more advanced techniques, such as integrating the numerical solver in the trainings loop since julia offers automatic differentiation of arbitrary functions, and therefore enables back-propagation (gradient descent) through the entire solver. It would also have been interesting to try different discretizations of the relaxed CH equation, and a different method for solving it, such as a finite volume or finite element method. Those bring the challenge of beeing harder to compare to our baseline.

Bibliography

[1] Andrea Corli, Christian Rohde, and Veronika Schleper. "Parabolic approximations of diffusive–dispersive equations". In: *Journal of Mathematical Analysis and Applications* 414.2 (2014), pp. 773–798. ISSN: 0022-247X. DOI: https://doi.org/10.1016/j.jmaa.2014.01.049. URL: https://www.sciencedirect.com/science/article/pii/S0022247X14000572.

[2] 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.

[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.