THE UNIVERSITY OF CHICAGO

ON THE DYNAMICS OF ACTIVE AND MULTI-COMPONENT LIQUID CRYSTALS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE PRITZKER SCHOOL OF MOLECULAR ENGINEERING
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

BY
JONATHAN SALMERÓN-HERNÁNDEZ

CHICAGO, ILLINOIS
DECEMBER 2024

Copyright 2024 by Jonathan Salmerón-Hernández

All Rights Reserved

ABSTRACT

Controlling Active lyotropic Liquid Crystal (LC) experiments, typically an active nematic coexisting with an isotropic fluid and molecular motors providing activity, remains a challenge due to unclear relationships between their multiple component concentrations. This prompts questions about the general conditions and effective parameters transforming their chaotic dynamics into coherent motion. As a result, continuum theory and numerical simulations become essential tools for predicting and guiding such experiments.

In this thesis, we utilize the out-of-equilibrium GENERIC framework to derive a thermodynamically consistent and novel theory for lyotropic liquid crystals. As a first step, we focus on a two-component system, an LC and an isotropic fluid, treating the time-evolution equations as the sum of energetic and entropic contributions. This approach couples the momentum and energy balances with concentration dependency and the liquid crystalline ordering.

Next, we introduce `LiquidCrystalGLBT.jl`, an open-source Julia-based solver designed for managing these concentration-dependent nematohydrodynamic equations. This hybrid solver combines an upwind finite difference scheme with the Galerkin Lattice Boltzmann method. In our acronym, 'G' indicates the use of the out-of-equilibrium GENERIC framework to derive our equations, while 'LB' signifies the implementation of the Lattice Boltzmann Method through the `Trixi.jl` package, the 'T' in our name.

We showcase three different scenarios resulting from this methodology, each with increasing complexity and incorporating insights from its predecessor:

1) A study of a binary mixture in 2D at equilibrium, resembling chromonic LC data with +/- 1/2 topological defects. 2) The introduction of hydrodynamics by incorporating a controlled velocity profile (Couette or parabolic) and tracking the shape transition of an axial LC droplet immersed in an isotropic environment. 3) The addition of biochemical activity, resulting in turbulent simulations that flow naturally, similar to microtubule or actin systems.

Finally, we discuss ongoing efforts to extend our model into a multi-component system. This extension, to the best of our knowledge, has not been published before. This work concludes with the hope that the insights gained from the systematic study of the binary system will inform the application of the multi-component equations to more complex scenarios. Importantly, this theoretical study, though brief, would not have been possible without the foundational analysis of the binary system.

In short, this work addresses the challenge of achieving stable and efficient simulations for lyotropic liquid crystals, contributing significantly to our understanding of their dynamic behavior. Furthermore, this project has the potential to advance predictions of lyotropic LCs in 3D, potentially elucidating the functionality of tissues in a liquid crystalline state, guiding the artificial design of biological tissues, and paving the way for new LC applications in bioengineering, sensing, and other emerging fields.

"If the goal of chemical engineers is to satisfy human needs, chemical engineers must have some understanding of human nature, of psychology and international relations, social organizations and the clash of cultures"

- John M. Prausnitz (1998), emeritus professor of chemical engineering at the University of California, Berkeley

"The idea that science concerns only scientists is as unscientific as it is antipoetic to claim that poetry concerns only poets"

- Gabriel García Márquez (1985), Literature Nobel Prize

"Science, arts, and culture uniquely represent one big thing: the use of human intelligence for the benefit of society"

-Jonathan Salmerón-Hernández (2020)

TABLE OF CONTENTS

ABSTRACT iii

LIST OF FIGURES viii

ACKNOWLEDGMENTS x

1 INTRODUCTION 1
1.1 Motivation and Thesis Structure 1
1.2 Theoretical Overview for Single-Phase Liquid Crystals 4

2 METHODOLOGY: THE GENERIC THEORY AND NUMERICAL METHODS 8
2.1 Introduction to the GENERIC Framework 8
2.2 Numerical Methods 10
2.2.1 Upwind Difference Scheme 11
2.2.2 Galerkin Lattice Boltzmann Method 14

3 THE MODEL FOR A TWO-COMPONENT LIQUID CRYSTAL MIXTURE . . 19
3.1 Variables 19
3.2 Energy 20
3.3 Entropy 24
3.4 Poisson Operator 26
3.5 Friction Matrix 29
3.6 Time-Evolution Equations 32

4 LIQUIDCRYSTALGLBT.JL: A SOLVER FOR THE CONCENTRATION-DEPENDENT NEMATOHYDRODYNAMIC EQUATIONS 36
4.1 Structs and constructors for hydrodynamic variables 36
4.2 Differential Operators 41
4.3 Time integration with DifferentialEquations.jl 49
4.4 Incoporating the Trixi.jl framework 51

5 SYSTEMS AT EQUILIBRIUM: TOPOLOGICAL DEFECT CORES 58
5.1 Equilibrium Points of the Generalized Cahn-Hilliard and Ginzburg-Landau Equations 58
5.2 Equilibrium Results in 2D: Comparison with Experimental Data 61

6 CONTROLLED FLOWING SYSTEMS: AN AXIAL DROPLET UNDER TWO VELOCITY PROFILES 66
6.1 The Convection-Diffusion-Relaxation System 66
6.2 A Liquid Crystal Droplet under (Semi-)Couette Flow 69
6.3 A Liquid Crystal Droplet under a Poisieuille Flow 71

7 NEMATIC TURBULENCE AND A MODEL FOR MULTI-COMPONENT LIQUID CRYSTALS 74
7.1 A single-component active Liquid Crystal 74
7.2 An active lyotropic Liquid Crystal 78
7.3 The model for Multi-component Liquid Crystals 79

8 CONCLUSIONS 84

9 APPENDIX 86

REFERENCES 89

LIST OF FIGURES

1.1 a) Molecular representation of the 4-Cyano-4'-pentylbiphenyl (5CB) and its corresponding molecular director $\boldsymbol{u}$. The 5CB molecule can be represented by a rodlike ellipsoid. The average of the molecular orientations results into the director vector $\mathbf{n}$. b) Average representation of the molecules when in isotropic phase (left) and in a crystalline phase (right) and its scalar order parameter values. Image adapted from [27] 5

5.1 Plots of Eq. 5.5 (in red) and Eq. 5.6 (in blue) using parameters $T = 1$, $c_1 = c_2 = 1$, $\chi = 3$, $A_0 = 1$, and $U = 20$. x-axis denotes the concentration range, while y-axis indicates the order parameter. One of the intersection points is indicated with a blue dot 60

5.2 Birefringence map of 15% DSCG mixture in a thin cell with tangential anchoring condition. Scale bar is 20 μm.. The white lines represent the director vector while the colors indicate the optical retardance. Image adapted from [66]. 62

5.3 Simulation of two water droplets immersed in a nematic environment. On the left, the director vectors (white lines, scaled according to the values of the concentration profile) and the scalar order parameter (color map ranging from 0 to 0.75) are shown. On the right, the concentration profile is indicated by a color map ranging from 0 to 1. Subfigure (a) presents the ICs, while (b)-(d) depict the successive time evolution, with (d) showing the final time step. 64

6.1 Simulation of an axial droplet immersed in an isotropic fluid. On the left, the director vectors (gray lines, scaled according to the order parameter values) and the scalar order parameter (color map ranging from 0 to 0.8) are shown. The top right shows the concentration profile, indicated by a color map ranging from 0 to 1, while the bottom right presents the velocity profile, colored by its x-component value (from -0.07 to 0.07). Subfigure (a) shows the initial conditions (ICs), while subfigures (b)-(d) illustrate the successive time evolution, with (d) representing the final time step. 70

6.2 Time evolution of a Poly(isobutylene) (PIB, parabol 1300) droplet deformation and orientation under shear flow. Image adapted from [72]. 71

6.3 Simulation of an axial droplet immersed in an isotropic fluid. On the top left, the director vectors (gray lines, scaled according to the order parameter values) and the scalar order parameter (color map ranging from 0 to 0.8) are shown. The bottom left shows the concentration profile, indicated by a color map ranging from 0 to 1, while the right presents the velocity profile, colored by its magnitude (from 0.045 to 0.062). Subfigure (a) shows the initial conditions (ICs), while subfigures (b)-(d) illustrate the successive time evolution, with (d) representing the final time step. 72

6.4 Still image of the deformation of a nematic 5CB droplet in a pressure-driven microfluidic flow control system under a polarized microscope. The size of the droplet is approximately 20μm. Image adapted from [73]. 73

7.1 Simulation of a one-component active nematic. On the left, the director vectors (black lines) and the scalar order parameter (color map ranging from 0 to 0.7) are shown. On the right, the velocity profile (arrows) is depicted, colored according to its magnitude (ranging from 0 to 1). Subfigure (a) shows the initial conditions (ICs), while subfigures (b)-(d) illustrate the successive time evolution, with (d) representing the final time step. Notably, (c) shows the nucleation of defects, while (d) captures defect annihilation. 77
7.2 Still image of one-component active nematics, consisting of polymerized tubulin microtubules in contact with kinesin molecular motors, at a water-oil interface under a confocal microscope. A schematic representation of defects is shown. Courtesy of Dr. Antonio Tavera-Vázquez. 78
7.3 Simulation of an active lyotropic liquid crystal. On the left, the director vectors (black lines) and the scalar order parameter (color map ranging from 0 to 0.75) are shown. In the middle, we present the concentration profile, colored from 0.92697 to 0.927. On the right, the velocity profile (arrows) is depicted, colored according to its magnitude (ranging from 0 to 7.86). Subfigure (a) shows the initial conditions (ICs), while subfigures (b)-(d) illustrate the successive time evolution, with (d) representing the final time step. 80

ACKNOWLEDGMENTS

I have been told this is the part where we can be prosaic, so that is what I am going to do. I actually wish I could write this in Spanish, as my writing talents might shine more in that language. In the conclusion, I was a bit bolder in using a more poetic tone rather than a scientific one; here, I will try to maintain a balance between fun, nostalgia, sadness, and gratitude throughout the acknowledgments.

First and foremost, I want to thank my mom, Claudia Hernández H.Luz. She always believed in me, even going so far as to lend me all her savings so I could travel to Chicago and start my PhD studies. Without that support, I would not be writing this today. She never expected to get that money back, as she confessed, and I have to admit that I get emotional every time I think of it. I also want to thank my family, including my dad, Germán Salmerón, and my sister, Itzel Salmerón. They may not fully understand what I'm doing, but I hope they do by now. Nevertheless, they have always been proud of me.

Next, I want to thank my Ph.D. advisor Juan de Pablo, who once called me *su loquito, gracias Juan.* You were not the first role model I had, but you definitely became one for a long time until our paths, as expected, began to diverge. Thank you for understanding my frequent travels to Cancún, for letting me stay in Mexico for almost two years during the most uncertain times of COVID, for supporting my other interests, and for challenging me. Most of all, thank you for inviting me to your 60th birthday in Cancún. That was a blast.

There are so many people who helped me complete my PhD that I hope I do not miss anyone. My gratitude goes to my PhD thesis committee, with whom I have a special connection.

Hans Öttinger, you taught me a new level of understanding the universe and hardcore math that I will probably use for the rest of my life.

Andrew, Andy, Ferguson, thank you for being a great ally and for becoming a second informal mentor. Thank you for agreeing to be my "pro-forma" PI for my LGBT project.

By the way, if you didn't notice, the name `LiquidCrystalGLBT` is a nod to one of my many identities.

Meghana Ranganathan, you are probably the newest addition to this family, but I hope our paths continue to cross.

Now, I want to acknowledge the third most important person in completing this project (after Juan and Hans): Dr. Pablo Zubieta. Pablo, I have already written a four-page letter for you, for which you won the 2023 PME Maria Lastra award in mentoring. I believe it's clear how impactful your involvement was. Before you joined my project, I was honestly afraid of computers. Now, that has changed, and I will certainly use all the tools you provided in the future. *No tienes idea de lo mucho que agradezco el que me hayas aceptado como tu aprendiz.*

Friends are next on my list. A huge round of applause for Fabian Byléhn. You are the best PhD friend I could have asked for, though I didn't actually ask for one, but it definitely helped. For some reason, we never quite completed our group of three, but I want to acknowledge the positive impact of Vasilis Papasimakopoulos, Víctor Mendoza, and Fabián Hernández.

Gustavo Pérez-Lemus, thank you for being the 'uncle' in this family, for providing insightful scientific discussions, and for listening to my wild stories.

In chronological order, I want to thank the following people:

The cohort above me: Aria Coraor, you are amazing, girl.

My original PhD cohort: Ande Hesser, thanks for teaching me patience and assertive communication skills; Anil Bilgin; Walter Alvarado, you're not officially a PME student, but it always felt like you were.

The cohort after me, my favorite: Reginaldo Gomes and Carlos Medina, my recurring Latin squad; Calixto Mateo Salles; Susan Okrah, I'm sorry for being difficult with math methods, but my heart is with you.

The cohort after that: María Ley, you helped start a big project, and for that, I am still in debt; Eric Schultz, I don't quite understand how you entered this list, but I'm glad you did.

The next cohort: Dasha Safarian, I initially thought you would be different, but I'm glad you brought some joyful moments to my life; Andrea Díaz, you are also here.
The following cohort: Cesar Castro, Jireh García, *mis bebesitos*. I'm going to miss you a lot. For the newest cohort, I sincerely hope that Suraj Sudhakar will be able to handle all the equations, numerical methods, and computational implementations here. He has a big challenge ahead, but I truly believe in him.

To the de Pablo group: my first LC mentors—Rui Zhang, Ali Mozaffari, Kim Weirich, and Noé Atzin—thank you so much for all you taught me at the beginning of my journey. Antonio Tavera-Vázquez, Carina Martínez, Abhishek Sharma, thanks for the great experiments, just kidding.

Other UChicago people and affiliates: Regina Sánchez, Dey Martínez, Rovana Popoff, Laura Rico, David Taylor, Lisa Vonesh; Amin Heari, thank you for initially joining the group, which gave me hope and made me feel less alone.

To the *Clubes de Ciencia* team: Prof. Sonia Hernández, thank you for being my first unofficial mentor at UChicago; Claudia Gil, our adventure in Morelia was incredible; Loria Hijar, thanks for helping me push my research into the biology realm I wanted—the liver is now my most important introductory slide. To the gay project people: Thanks to my co-PI, Erick Adrián Paz, and Jessica Chávez for navigating our challenges; Dr. Aída Gutiérrez, for connecting us to UNAM.

During my years in Cancún and Mexico at the worst times of the COVID pandemic, and my first years in Chicago many new people entered my life, giving me more reasons to stay. All these people are important to this journey. My Rotaract clubs —Moi Vazco, Laura González, Ricardo Rodríguez, Ana Sánchez, Rafael Urbina, Daniel Figueroa—; Marisol Torres, Hilario Magaña, and Yesica Catzin, my official support, and bullying team. Iván Cruz-Méndez, Tania García, Crystel Celis, Diego Orozco, you are the best iqleros. .... ok, I think I should stop here...

CHAPTER 1

INTRODUCTION

1.1 Motivation and Thesis Structure

Active matter refers to dynamic systems whose constituents continuously convert energy into mechanical work [1, 2]. Due to their out-of-equilibrium nature, these systems can spontaneously organize, change shape, and collectively move and flow (be active) on a scale larger than their components' size [3, 4]. Multiple examples span both non-living and living realms, encompassing organs and tissues [5, 6, 7], bacterial suspensions [8], as well as animal groups [9]. Non-living instances, inspired by their alive counterparts, include layers of vibrated granular rods [10], self-propelled Janus particles through thermal effects [11], and reconstituted protein systems featuring elements of the cell cytoskeleton: filamentous structures, like actin and microtubules, along with their associated motor proteins [12, 13].

Many of the systems cited above consist of anisotropic units, that could lead to a liquid crystalline state [2], and hence enabling the application of concepts from liquid crystal (LC) physics to describe them [14]. These partially ordered states, now referred to as active liquid crystals, exhibit a great variety of intriguing phenomena. For example, they showcase spontaneous generation, motion, and annihilation of topological defects [1] as well as phase transitions (e.g., from isotropic to nematic) [15]. If these transitions are solely induced by thermal processes, the LCs are classified as thermotropic. Furthermore, if the transition driving force arises from gradients in their components' concentration, then these LCs are categorized as lyotropic.

As we continue to expand our knowledge of lyotropic LCs, recent investigations, typically involving an active nematic in contact with an isotropic fluid [16, 17, 18], have unveiled complex patterns that remain unexplained [19, 20]. These studies prompt questions about the effective parameters necessary for transforming their chaotic behavior into coherent

motion. Thus, simulations arising from reliable continuum theory appear as a convenient tool to predict and guide these experiments [2]. Through both computational research and experimentation, researchers not only aim to comprehend and replicate the behavior of LC materials, but also to discover and control the full range of collective organization of both living and non-living entities [19]. This, in turn, will pave the way for future applications, to name some, in biomedical devices, artificial tissue design, and nanotechnology.

Early computational efforts, particularly those employing thermotropic LC-based models with the Beris-Edwards equation, have provided valuable insights into experiments involving active lyotropic LCs [13, 21]. However, understanding the factors influencing the ratio between components and incorporating such relationships into these equations remains a challenging task, especially considering that, in their original derivation, concentration was not included as a dynamical variable. This hampers our comprehension, for instance, of the effect of local concentration within defect cores of nematic films [16, 17, 18]. Previous and ongoing research in this area, aiming to integrate concentration dynamics, has made remarkable contributions to the theory of lyotropic LCs [22, 23, 24, 25], but has also sparked debates concerning the thermodynamic consistency of their formulations, numerical stability and good agreement with experimental data.

To tackle that challenging problem, in this thesis, we introduce a systematic novel approach to model the behavior of lyotropic LCs by applying the GENERIC (General Equation for the Non-Equilibrium Reversible-Irreversible Coupling) formalism and state-of-the-art numerical methods. We methodically incorporate physical mechanisms while ensuring generality and avoiding over-specification of the system. This approach is firmly grounded in the first principles of out-of-equilibrium thermodynamics. The thesis is divided as follows:

In next section, we provide a brief overview of the current theoretical description for single-phase or one-component LCs, against which we compare our proposed model in this work.

The next chapter presents our methods to address the problem, covering: 1) an introduction to the GENERIC formalism for deriving the system of differential equations and 2) the numerical methods employed to solve these equations.

In Chapter 3, we present the GENERIC building blocks: the total energy and entropy, along with the Poisson and friction matrices. These are intended to describe, as an initial step, a two-component nematic-isotropic emulsion with diffuse interfaces. Following that, we outline the resulting set of equations for this mixture.

Chapter 4 provides an overview of the computational techniques used to manage the numerics. Specifically, we detail our implementation written in the Julia programming language. This effort leads to the development of an open-source library named `LiquidCrystalGLBT.jl`, which is intended to facilitate further research in this area.

We divide our results into three chapters, each with increasing complexity and incorporating insights from the previous one:

In Chapter 5, we reproduce experimental data under equilibrium conditions (no flow), for which we examine the stable points within our differential equations. The next chapter shows the evolution of axial droplets under two different velocity profiles —Couette and Poiseuille (parabolic)— and compares these with experimental data. This involves using a upwind scheme finite difference for accurate flowing simulations [26]. The subsequent chapter presents systems with nematic turbulence. Its final section includes our equations for multi-component LCs.

This multi-component model represents an extension that, to the best of our knowledge, has not been published before. This work concludes with the hope that the insights gained from the systematic study of the binary system will inform the application of the multi-component equations to more complex scenarios.

Chapter 8 summarizes our findings, discusses future directions for incorporating additional physics (e.g. activity, sharp interfaces), and provides a final reflection on the thesis.

1.2 Theoretical Overview for Single-Phase Liquid Crystals

Many theoretical models have been developed over the years to study the motion of single-phase active LCs. Throughout this thesis, we use a continuum approach where the active anisotropic LC constituents can be simplified as a group of filamentous rod-shaped particles. These anisotropic particles, known as mesogens, have a preferred alignment along one spatial direction. Examples include the 4-Cyano-4'-pentylbiphenyl (5CB) molecule and the F-actin protein. When an LC has a high degree of long-range orientational order, but not positional, it is called nematic phase and the direction of the average molecular ordering is called the director vector $\boldsymbol{n}$ [15], see schematic representation in Fig. 1.1a).

The simplest case of nematic ordering is uniaxial. We represent this uniaxial order with a traceless and symmetric second-order tensor

$$Q_{ij} = q\left(n_i n_j - \frac{I_{ij}}{d}\right), \tag{1.1}$$

known as the alignment tensor, a well-defined structure containing all the information about the average orientation of the rod-like particles. In Eq. (1.1), the director vector is represented with the unit vector satisfying the head-to-tail symmetry $n_i \equiv -n_i$. I_{ij} denotes the identity matrix, d indicates the dimensionality of the system (e.g., 2D or 3D space) and q is the scalar order parameter ($0 \leq q \leq 1$). When $q = 1$, the particles are perfectly aligned along $\boldsymbol{n}$. In a disordered isotropic phase, where the particles are randomly distributed, the order parameter $q = 0$ (Fig. 1.1). As usual, $\boldsymbol{Q}$ is symmetric ($Q_{ij} = Q_{ji}$) and traceless ($Q_{ll} = 0$). Herein, the Latin subscripts (i, j, k, l, m) indicate the basis components, and we assume the Einstein summation convention over repeated indices (unless otherwise stated).

Moving forward, we proceed with the description of the one-component LC hydrodynamics. The tensor $\boldsymbol{Q}$ is strongly coupled to the fluid velocity field $\boldsymbol{v}$, as the rod-like particles highly respond to the velocity gradients ($W_{jk} = \partial_k v_j$), induced by various stresses in the system[28].



(a) (b)

Figure 1.1: a) Molecular representation of the 4-Cyano-4'-pentylbiphenyl (5CB) and its corresponding molecular director $\boldsymbol{u}$. The 5CB molecule can be represented by a rodlike ellipsoid. The average of the molecular orientations results into the director vector $\mathbf{n}$. b) Average representation of the molecules when in isotropic phase (left) and in a crystalline phase (right) and its scalar order parameter values. Image adapted from [27]

This complex relationship is encapsulated in the following Beris-Edwards (BE) equation, published in 1994 and derived by means of the Poisson bracket formalism [29]:

$$\left(\partial_t + v_k\partial_k\right) Q_{ij} - Z_{ij}\left(W_{ij}, Q_{ij}\right) = \Gamma H_{ij}, \tag{1.2}$$

where the first term of the left-hand side (LHS) represents the convective mechanism, t hereafter denotes time, and the second term functions are the generalized nonlinear advective term:

$$\begin{aligned} Z_{ij} &= \left(\lambda\Xi_{ik} + \Omega_{ik}\right)\left(Q_{kj} + \frac{I_{kj}}{d}\right) + \left(Q_{ik} + \frac{I_{ik}}{d}\right)\left(\lambda\Xi_{kj} - \Omega_{kj}\right) \\ &- 2\lambda\left(Q_{ij} + \frac{I_{ij}}{d}\right)\left(Q_{kl}\partial_k v_l\right), \end{aligned} \tag{1.3}$$

with $\Xi_{jk} = (\partial_j v_k + \partial_k v_j)/2$, the strain rate tensor, and $\Omega_{jk} = (\partial_k v_j - \partial_j v_k)/2$, the vorticity tensor. λ is the alignment parameter controlling the degree of coupling between the director field and velocity gradients [15, 28], and, as usual, ∂ indicates partial derivative.

Furthermore, the term on the right-hand side (RHS) of Eq. (1.2) corresponds to a gradient descent term, where Γ is the relaxation parameter with dimensions of [volume/(time·energy)],

which seeks to minimize the free energy of the system. The molecular field

$$H_{ij} = -\left(\frac{\delta E_{free}}{\delta Q_{ij}} - \frac{I_{ij}}{d} Tr \frac{\delta E_{free}}{\delta Q_{kl}}\right), \tag{1.4}$$

transforms the functional derivative ($\delta/\delta Q_{ij}$) of the free energy functional (E_{free}) into a symmetric and traceless (ST) term. Its second term on the RHS acts as a Lagrange multiplier to ensure the traceless condition, and Tr is used to denote the computation of the trace.

Next, the velocity field obeys a generalized Navier-Stokes (NS) equation. Assuming a constant total mass density ρ, the nematic momentum equation can be written as [22]

$$\rho\left(\partial_t + v_k \partial_k\right) v_i = \partial_j \Pi_{ij}, \tag{1.5}$$

where Π_{ij} is a general stress tensor consisting of three parts: (i) the viscous stress $\Pi_{ij}^{viscous} = 2\nu\Xi_{ij}$, where ν is the Newtonian viscosity of the LC, with dimensions of [pressure $\cdot$ time]; (ii) the orientational elastic stress of a passive LC,

$$\begin{aligned}\Pi_{ij}^{passive} &= -pI_{ij} + 2\lambda\left(Q_{ij} + \frac{I_{ij}}{d}\right)\left(Q_{kl}H_{lk}\right) - \lambda H_{ik}\left(Q_{kj} + \frac{I_{kj}}{d}\right) - \lambda\left(Q_{ik} + \frac{I_{ik}}{d}\right)H_{kj} \\ &- \partial_i Q_{kl}\frac{\partial \mathcal{E}_{free}}{\partial\left(\partial_j Q_{lk}\right)} + Q_{ik}H_{kj} - H_{ik}Q_{kj},\end{aligned} \tag{1.6}$$

where p is the hydrostatic pressure, $\mathcal{E}_{free}$ is the free energy density; and (iii) the active stress which reads [30]

$$\Pi_{ij}^{active} = -\zeta Q_{ij}, \tag{1.7}$$

representing the leading contribution of activity to the stress as a dipole force density [31].

In Eq. (1.7), ζ represents the activity coefficient, which is positive if the forces are of extensile origin and negative if they are contractile ones. Furthermore, this coefficient has dimensions or [pressure] or [force/area]. To envision this activity, we refer to the molecular protein motors present almost ubiquitously in muscle cells. These molecular motors, such as

kinesin or myosin, generate the forces to conduct movement along filamentous molecules and drive the systems away from thermodynamic equilibrium. The machinery of the molecular motors obtains energy from the hydrolysis chemical reaction of their surrounding adenosine triphosphate molecules (ATP), known as the energy currency of the cell. In short, this continuous conversion of energy can then be interpreted as the activity, which is the source of hydrodynamic instability or, equivalently, nematic turbulence [2].

CHAPTER 2

METHODOLOGY: THE GENERIC THEORY AND NUMERICAL METHODS

2.1 Introduction to the GENERIC Framework

We employ the GENERIC (General Equation for the Non-Equilibrium Reversible-Irreversible Coupling) framework to establish a thermodynamically-consistent model for active lyotropic LCs. In this framework, the dynamics of out-of-equilibrium systems are governed by the time-evolution equation towards equilibrium, of the form [32, 33, 34, 35]:

$$\frac{\partial \boldsymbol{x}}{\partial t} = \boldsymbol{L}(\boldsymbol{x}) \cdot \frac{\delta E(\boldsymbol{x})}{\delta \boldsymbol{x}} + \boldsymbol{M}(\boldsymbol{x}) \cdot \frac{\delta S(\boldsymbol{x})}{\delta \boldsymbol{x}}, \tag{2.1}$$

where $\boldsymbol{x}$ represents a set of variables, like density and momentum, used to describe the situation of interest, and t, as in the previous chapter, signifies time. Note that $\boldsymbol{x}$ is in bold to distinguish it from x which denotes one of the usual Cartesian coordinates. This convention, boldface or additional indices, is followed throughout the thesis to differentiate between variables. The linear operators, $\boldsymbol{L}$ and $\boldsymbol{M}$, also known as Poisson and friction matrices, play a crucial role in incorporating and effectively coupling the physical mechanisms under consideration (e.g. convection and diffusion). The total energy (E) and the entropy (S) are real-valued functionals written in terms of $\boldsymbol{x}$, while the symbol $(\cdot)$ denotes matrix multiplication, or dot product. As in the previous chapter, the symbol $\delta_{\boldsymbol{x}}$ implies a functional derivative rather than a partial one ($\partial/\partial t$).

In elucidating our choice of the GENERIC formalism, we must briefly highlight its significant robustness and wide-ranging applications. The GENERIC Equation (2.1) is not only capable of reproducing esteemed equations across diverse areas of physics, such as the Navier-Stokes and Maxwell equations [35, 36], but also extend its use to relativistic

hydrodynamics, aligning well with the principles of general relativity [32]. At its core, the GENERIC framework embodies a thoughtful synthesis of thermodynamics, drawing upon concepts from statistical physics and linear irreversible thermodynamics, which is well supported by mathematical geometrical structures along with their necessary constraints. Further explanation of these mathematical aspects will follow as needed. Crucially, GENERIC has also demonstrated to be compatible with the Poisson brackets formalism [37], or the theoretical methodology utilized by Beris-Edwards to formulate their equation, thereby confirming its applicability for extending their theory to more complex systems like lyotropic liquid crystals.

In simple terms, by respectively associating each derivative with reversible (energy-conserving) and irreversible (entropy-producing) processes, Eq. (2.1) elegantly conveys the evolution of any system as the sum of both reversible and irreversible contributions. To illustrate this concept, we draw an analogy with the first and second laws of *equilibrium* thermodynamics. The first law states that the exchange of work and heat results in a change in the total internal energy of a system. The second law further indicates a linear relationship (temperature) between heat transfer and entropy, with entropy determining the (ir)reversibility of a process.

Thus, the reversible contribution can be understood as work performed 'under mechanistic control' within the system, while the irreversible contribution includes processes 'not under mechanistic control', such as heat transfer. In the framework of GENERIC, both reversible and irreversible processes govern the time evolution of our variable of interest $\boldsymbol{x}$, analogous to how the first law of thermodynamics describes the change in internal energy as the sum of work and heat.

In addition, as introduced earlier, both reversible and irreversible contributions must fulfill some mathematical constraints for a complete description of the nonequilibrium thermodynamics. In classical mechanics, the reversible contributions have very-well established

geometric structures that are associated with Hamiltonian dynamical systems, such as sympletic, Poisson and Dirac structures. These structures relate the Hamiltonian dynamics with the energy of the system. For the GENERIC framework, we build this relationship by means of the Poisson bracket structures, similar to Beris-Edwards, allowing us to describe reversible contributions in terms of energy derivatives. Similarly, the irreversible contribution can then be described by "dissipative brackets" and hence be expressed in terms of entropy gradients. Both Poisson and dissipative brackets can be generalized as matrices $\boldsymbol{L}$ and $\boldsymbol{M}$, which should satisfy two degeneracy conditions

$$\boldsymbol{L}(\boldsymbol{x}) \cdot \frac{\delta S(\boldsymbol{x})}{\delta \boldsymbol{x}} = 0, \tag{2.2}$$

$$M(\boldsymbol{x}) \cdot \frac{\delta E(\boldsymbol{x})}{\delta \boldsymbol{x}} = 0, \tag{2.3}$$

and the Jacobi identity [32]. Furthermore, the Poison $\boldsymbol{L}$ matrix should be antisymmetric, while the dissipative $\boldsymbol{M}$ matrix should be Onsager-Casimir symmetric and positive semidefinite, also known as a non-negative condition. This non-negative constraint along with the two degeneracy properties guarantee that the entropy is always an increasing function in time, the second law of equilibrium thermodynamics. In simple terms, these constraints also ensure total energy conservation. For a more detailed discussion referred to [32].

As a final note, the $\boldsymbol{L}$ matrix is typically heavily restricted, making it difficult to incorporate new physical mechanisms, whereas the $\boldsymbol{M}$ matrix is more flexible and can thus incorporate additional physics.

2.2 Numerical Methods

In general, solving analytically the resulting Partial Differential Equations (PDEs) from the GENERIC framework is unfeasible. Therefore, we turn to numerical approaches, focusing primarily on two methods: 1) an Upwind Finite Difference (FD) scheme for spatial derivatives

combined with higher-order methods, including a forward Euler's method, for time derivatives, and 2) a Galerkin Lattice Boltzmann Method (LBM) for solving the generalized NS equations as well as the continuity equation. In short, we use the method of lines, which separates the full space-time discretization into two steps: first, discretizing the spatial domain to reduce the PDEs to a potentially large system of ordinary differential equations, and then solving this system numerically using a time integration method.

From now on, we use Cartesian coordinates unless otherwise stated. The reference plane is standard: the horizontal axis represents the x-direction, while the vertical axis corresponds to the y-direction in 2D (if not indicated).

2.2.1 Upwind Difference Scheme

In this work, we use two distinct FD approaches for spatial discretization based on the nature of the equations being solved. For non-hydrodynamic components (where flow velocity is not involved), such as first-order derivatives (e.g., gradient and divergence) and second-order derivatives (e.g., the Laplacian operator), we employ a second-order central difference scheme. This approach provides accurate approximations by considering values at neighboring grid points in both positive and negative directions along each axis [26].

For illustrative purposes, consider an arbitrary one-dimensional function $\psi(x)$ dependent solely on the x direction, discretized on a structured mesh with a constant spacing step Δx. In this context, the second-order central difference scheme approximates spatial derivatives as follows:

$$\frac{\partial \psi}{\partial x} \approx \frac{\psi(x+\Delta x) - \psi(x-\Delta x)}{2\Delta x}, \tag{2.4}$$

$$\frac{\partial^2 \psi}{\partial x^2} \approx \frac{\psi(x+\Delta x) - 2\psi(x) + \psi(x-\Delta x)}{\Delta x^2}, \tag{2.5}$$

where the function values at the grid points immediately in the positive and negative directions from the central point $\psi(x)$ are denoted by $\psi(x+\Delta x)$ and $\psi(x-\Delta x)$.

Furthermore, time integration, in the absence of hydrodynamics, employs higher-order numerical methods, not explicitly described here, from the `DifferentialEquations.jl` package [38] in the Julia language (see Chapter 4), such as an adaptive Runge-Kutta scheme (e.g., Tsit5) or an explicit method (e.g., CarpenterKennedy2N54). The choice of integrator is based on empirical testing, prioritizing performance and numerical stability.

Conversely, for hydrodynamic simulations, where flow velocity is involved, we use a backward and forward FD scheme. This approach improves accuracy and stability by accounting for the direction of the flow [26].

In a one-dimensional exemplary setting, the backward and forward scheme discretizes the first-order derivative of $\psi(x)$ (typically the only term affected by convective effects) based on the sign of the flow velocity v that is moving the system.

For a positive flow velocity ($v > 0$):

$$\frac{\partial \psi}{\partial x} \approx \frac{\psi(x) - \psi(x - \Delta x)}{\Delta x}, \tag{2.6}$$

and for a negative flow velocity ($v < 0$):

$$\frac{\partial \psi}{\partial x} \approx \frac{\psi(x + \Delta x) - \psi(x)}{\Delta x}, \tag{2.7}$$

here, only one neighboring point is considered, unlike the two points in a second-order central difference scheme. For a more compact form, equations (2.6) and (2.7) are typically combined with the velocity as

$$v \cdot \frac{\partial \psi}{\partial x} = v^{+} \frac{\psi(x) - \psi(x - \Delta x)}{\Delta x} + v^{-} \frac{\psi(x + \Delta x) - \psi(x)}{\Delta x} \equiv \mathrm{Op_x}(\psi, v), \tag{2.8}$$

where

$$v^{+} = \max(v, 0), \qquad v^{-} = \min(v, 0), \tag{2.9}$$

$\max(v, 0)$ representing the maximum of v and 0 (which is v if $v > 0$ and 0 otherwise), and $\min(v, 0)$ denoting the minimum of v and 0 (v if $v < 0$, and 0 otherwise). This compact form combines backward and forward approximations for positive and negative velocities into a single expression, defined as the $\mathrm{Op_i}$ operator, with its subscript indicating both the velocity component and the discretized axis.

For time integration, we only employ an iterative forward Euler's method when using the backward and forward FD scheme, because, in our tests, this seemingly less sophisticated method appears more stable compared to those found in the `DifferentialEquations.jl` package. As before, the forward Euler's method is illustrated with a simple example.

Given a differential equation of the form $\frac{\partial \psi(t)}{\partial t} = \Psi(t, \psi(t))$, one step of the Euler method from t_a to $t_{a+1} = t_a + \Delta t$ is given by

$$\psi_{a+1} = \psi_a + \Delta t \cdot \Psi(t_a, \psi_a), \tag{2.10}$$

where Δt represents the time step between two iterations, the subscript a denotes the time step index and Ψ is a second auxiliary function.

Furthermore, for multidimensional systems, the backward and forward FD scheme requires modifications known as the dimensional splitting method [26]. This method involves approximating the derivatives, along one dimension (e.g., x-direction), using equation (2.8) and then updating the values of the function at each grid point based on this computation. Specifically, the old values are replaced with new ones through the Euler integrator. After the solution is updated, the procedure is then applied along the next dimension (e.g., y-direction) using the updated values from the previous step. In a 3D case, this procedure is followed sequentially for all three dimensions. The updated values from each dimension form the final solution, representing one single time step.

In mathematical terms, the dimensional splitting method can be understood through the

discretization of the 2D advection equation

$$\frac{\partial \psi}{\partial t} = -v_x \frac{\partial \psi}{\partial x} - v_y \frac{\partial \psi}{\partial y}, \tag{2.11}$$

with the update step for ψ given by:

$$\psi_{a+1} = \psi_a - \Delta t \left[\mathrm{Op_x}(\psi_a, v_x) + \mathrm{Op_y}\left(\psi_a + \Delta t \cdot \mathrm{Op_x}(\psi_a, v_x), v_y\right) \right], \tag{2.12}$$

where the subscripts in the velocity components denote the usual Cartesian directions. In equation (2.12), the advection (or movement) is first applied in the x-direction and then in the y-direction (see the arguments in $\mathrm{Op_y}$), with the subscript $a+1$ indicating the completion of one time step as the sum of both movements. This equation serves as the basis when convection plays a significant role in our numerical results.

In summary, the approximations in equations (2.4), (2.5), (2.8) and (2.12), as well as the higher numerical methods for time integration, form the foundation of our upwind difference scheme.

2.2.2 Galerkin Lattice Boltzmann Method

In this subsection, we briefly review the LBM, explain how to incorporate additional stress tensors to meet our needs, and provide a brief overview of its Galerkin variation to illustrate how we couple it with our upwind difference scheme.

In recent years, the LBM has grown as an alternative and promising numerical scheme for simulating fluid flows and modeling physics in fluids [39]. This method was first developed from the lattice gas automata [40] and later on established as an independent numerical method [41].

Unlike conventional numerical schemes for fluid dynamics, which are based on discretizations of macroscopic continuum equations like the NS equations, the LBM is based on

microscopic models through a mesoscopic kinetic equation. This approach assumes that the macroscopic dynamics of a fluid result from the collective behavior of many microscopic particles in the system [42]. This mesoscopic kinetic equation was derived by Ludwig Boltzmann in 1872 and has the form:

$$\left(\partial_t + \mathbf{v}_b^\circ \cdot \partial_{\boldsymbol{r}} + \mathbf{g} \cdot \partial_{\mathbf{v}_b^\circ}\right) f_b(\boldsymbol{r}, t) = \Phi(f). \tag{2.13}$$

In eq. 2.13, f_b represents the main variable of interest of the LBM, known as the single distribution function. This functions is associated with the probability density of different particles moving in various directions and has dimensions of [mass/volume]. $\mathbf{v}_b^\circ$ denotes the velocity of such microscopic particles (overall constant), $\mathbf{g}$ represents an external specific body force (such as gravity) with units of [force/mass], and $\boldsymbol{r}$ indicates the vector of spatial coordinates. The term on the RHS, known as the Bhatnagar–Gross–Krook (BGK) operator, describes the effect of collisions between particles and reads

$$\Phi(f) = -\frac{1}{\tau}\left(f_b - f_b^\circ\right) + \mathcal{F}_b, \tag{2.14}$$

in this operator, τ is a relaxation time related to the viscosity of the fluid, f_b° is the the Maxwell-Boltzmann distribution function in equilibrium, and $\mathcal{F}_b$ is a forcing term. This forcing term incorporates complex molecular interactions into the operator (if applicable) and is not provided directly by the LBM itself. While some authors treat the forcing term separately, we have chosen to merge it into Φ. Additionally, the equilibrium distribution, using the same nomenclature, is expressed as:

$$f_b^\circ = \rho w_b \left[1 + \frac{\mathbf{v}_b^\circ \cdot \boldsymbol{v}}{c_s^2} + \frac{(\mathbf{v}_b^\circ \cdot \boldsymbol{v})^2}{2c_s^4} - \frac{\boldsymbol{v} \cdot \boldsymbol{v}}{2c_s^2}\right], \tag{2.15}$$

where c_s is the speed of sound, and ρ and $\boldsymbol{v}$ are, once again, the classical macroscopic total

mass density and velocity field, respectively. The term w_b represents the dimensionless weights used to normalize the equilibrium distribution function. Each weight corresponds to the relative probability of the microscopic particles to move in a particular direction. They are predetermined and constant throughout a given simulation, but vary with the dimensionality (e.g., 2D or 3D) and the number of allowed traveling directions at each grid point of the discretized mesh. For instance, in 2D, the typical choice for allowed directions is 9. The subscript b denotes one of these discrete directions, which should not be confused with Einstein's summation convention, using indices i, j, k, l, m.

Importantly, the analytical integration of the Boltzmann-BGK equations yields both the continuity equation and the NS equations with an ideal Newtonian stress (when $\mathcal{F} = 0$). However, additional interactions or forces beyond this Newtonian stress may be necessary. For instance, forces in complex fluids are typically described by intricate stress tensor expressions. Below is a brief description of how to incorporate such complex tensors and their corresponding forces into the Boltzmann-BGK equations.

First, the force density (F_i) on the fluid due to complex fluid effects is computed from the divergence of the non-Newtonian stress tensor,

$$F_i = \partial_m \pi_{mi}^{\text{complex}} \tag{2.16}$$

generally using an FD method, such as those described in Section 2.2. $\pi_{mi}^{\text{complex}}$ is typically formulated separately, and in Chapter 3, we show its form for our lyotropic LC. Next, the forcing term ($\mathcal{F}_b$), using the Guo forcing form [42], is computed as

$$\mathcal{F}_b = \left(1 - \frac{\Delta t}{2\tau}\right) w_b \left[\frac{\mathbf{v}_b^\circ - \boldsymbol{v}}{c_s^2} + \frac{(\mathbf{v}_b^\circ \cdot \boldsymbol{v})}{c_s^4}\mathbf{v}_b^\circ\right] \cdot \boldsymbol{F} \tag{2.17}$$

where Δt represents the chosen time step (as in the upwind scheme). Finally, we express the macroscopic variables as sums of the distribution function over the discrete traveling

directions (b), reverting to hydrodynamics, using the following expressions:

$$\rho = \sum_b f_b, \tag{2.18}$$

$$\boldsymbol{v} = \frac{1}{\rho}\left(\sum_b f_b \mathbf{v}_b^\circ + \frac{\boldsymbol{F}\Delta t}{2}\right). \tag{2.19}$$

Furthermore, the numerical integration of the above Boltzmann–BGK equations indeed leads to the LBM and its discretized version is usually written as

$$f_b(\boldsymbol{r} + \Delta t \mathbf{v}_b^\circ, t + \Delta t) - f_b(\boldsymbol{r}, t) = \Delta t \cdot \Phi(f). \tag{2.20}$$

For details on the basic numerical steps, we refer to [42]. While LBM is usually spatially discretized with FD schemes, this work employs a Galerkin scheme found in the `Trixi.jl` package [43, 44, 45] from the Julia language (see Chapter 4), where the integration is primarily based on the evolution of the collision operator Φ. Therefore, we provide a concise introduction to the Galerkin spatial discretization to explain how we combine this Galerkin scheme with our upwind difference scheme, introduced in the previous subsection

We heavily rely on the description of Galerkin schemes found at the Trixi.jl documentation[1]. In this package, the Galerkin method is spatially discretized in elements, with the number of elements being a free parameter. Each element contains $\mathcal{N}$ solutions at $\mathcal{N}$ Gauss-Lobatto nodes. These solutions are identical across the element, or in its nodes, due to the Lax-Friedrichs flux method, which we do not describe here. The number of Gauss-Lobatto nodes, and thus the number of solutions, in each cell is calculated (for a structured Cartesian domain) as $\mathcal{N} = (P_{deg} + 1)^d$, where P_{deg} is the degree of the polynomial Lagrange basis functions, and we recall that d is the dimensionality. The number of nodes is multiplied, or repeated, along each axis, except at the boundaries. Then, for boundary cells, $\mathcal{N} = (P_{deg} + 1)^{d-1}$, as the

1. `docs.sciml.ai/Trixi/stable/tutorials/scalar_linear_advection_1d/`

corner node is not repeated. Polynomial Lagrange basis functions are used in this method to approximate the solution of the differential equation, transforming it into an algebraic problem. Generally, a higher polynomial degree enhances overall performance and is also a free parameter.

In practical terms, this computational Galerkin grid resembles a structured mesh, although these elements are more like cells rather than grid points, as in an FD scheme. To address this difference, we conceptually align the corners of these elements with our FD grid points. Thus, we map one of the $\mathcal{N}$ solutions (typically the first Gauss-Lobatto node) in each element, here corresponding to ρ or $\boldsymbol{v}$, to its respective grid point in the upwind scheme, thereby coupling both methods. Note that due to the element, or cell-like, at the corner, the number of grid points in the upwind scheme needs to be one more than the number of Galerkin elements. For example, if there are two Galerkin elements along an axis, three upwind grid points are required along that axis for the solutions to coincide.

To conclude, the time integration of the Galerkin LBM is performed similarly to the methods described using `DifferentialEquations.jl` in the previous subsection, but more details are given in Chapter 4.

CHAPTER 3

THE MODEL FOR A TWO-COMPONENT LIQUID CRYSTAL MIXTURE

In this chapter, we introduce the hydrodynamical model of a binary mixture consisting of an isotropic liquid and a lyotropic liquid crystal. We first define the dynamical variables $\boldsymbol{x}$, followed by the total energy and entropy densities, as well as their corresponding Poisson and friction matrices. The hydrodynamics are dictated by Eq. (2.1), and we close by presenting the final form of the time-evolution equations.

3.1 Variables

To define our system, we specify the set of variables $\boldsymbol{x}$, encompassing the total mass density ρ, the total momentum density $\boldsymbol{m}$, and the internal energy density ϵ —the three classical hydrodynamic fields. We also include the mass fraction of the liquid crystal ϕ ($0 \leq \phi \leq 1$), hereafter referred to as the LC concentration. Note that the concentration of the second component, an isotropic liquid, can alternatively be expressed as $1 - \phi$. Furthermore, we choose the second-order tensor $\boldsymbol{Q}$ to describe the liquid crystalline order, consistent with the properties detailed in Section 1.2. In short, the vector $\boldsymbol{x} = (\rho, \boldsymbol{m}, \epsilon, \phi, \boldsymbol{Q})$ comprises five different fields, each of them position (or space, $\boldsymbol{r}$) dependent.

As usual, we use the variable, $\boldsymbol{Q}$, to describe the LC ordering because it remains well-defined even around topological defects, where the director $\boldsymbol{n}$ usually diverges [15]. Of course, we can recover both the order parameter and the director vector as they are related to $\boldsymbol{Q}$'s highest eigenvalue and its associated eigenvector, respectively [46]. For illustrative purposes, we present the simulation results in terms of $\boldsymbol{n}$ and q

Importantly, while we primarily consider the uniaxial case for simplicity in our calculations, our model can readily extend to scenarios involving significant biaxiality without

major modifications. This extension might, however, require computing an additional order parameter and its associated director vector [24].

3.2 Energy

Next, we define the total energy $E(\boldsymbol{x})$ by integrating the total energy density $\mathcal{E}(\boldsymbol{x})$ over the entire constant volume (V) occupied by our system

$$\begin{aligned} E(\boldsymbol{x}) &= \int \mathcal{E}(\boldsymbol{x})\, dV = \int \left[\frac{1}{2} \frac{{m_k}^2}{\rho} + \epsilon + \frac{1}{2}\kappa_E \left(\frac{\partial \phi}{\partial r_k} \right)^2 + \frac{1}{2}\kappa' \left(\frac{\partial \phi}{\partial r_i} \right) \left(Q_{ij} + \frac{I_{ij}}{d} \right) \left(\frac{\partial \phi}{\partial r_j} \right) \right. \\ &+ \left. \frac{1}{2} K(\phi) \frac{\partial Q_{ij}}{\partial r_k} \frac{\partial Q_{ij}}{\partial r_k} + \rho\chi\phi(1-\phi) + \epsilon_{\mathrm{LdG,E}}(\rho, \phi, \boldsymbol{Q}) \right] dV, \end{aligned} \tag{3.1}$$

this energy density includes contributions from kinetic, internal, interfacial, and anchoring energy densities, as well as terms from the elastic Frank-Oseen (FO), Flory-Huggins (FH), and Landau-de Gennes (LdG) theories. In the following paragraphs, we briefly discuss the physical reasoning behind these contributions.

The first term in Eq.(3.1) denotes the total kinetic energy density, while the second term adds the internal energy density (ϵ) to the integral.

Next, the third term incorporates diffuse interfaces into our model, commonly found in the phase field and liquid crystal literature [22, 47]. As the concentration profile changes both spatially and temporally, this term defines the phase interface as the region with the largest variation in ϕ. Here, such variations are smooth, or diffuse, resulting from the derivatives that ensure a continuous concentration function and prevent non-differentiable sharp changes. Furthermore, the coefficient κ_E controls the size or thickness of this interfacial region [48].

While we choose the diffuse interface method for its simple computational implementation [49], we recognize its constraints in computing interfacial properties [50]. These constraints often stem from assuming that bulk thermodynamic relationships can readily apply to interfacial variables. This assumption is not always valid, especially at equilibrium (see pp. 56

and 59 of [51]), and limits our spatial resolution to the interface thickness. To address these issues, we recommend incorporating a sharp interface description [49], through GENERIC and small system thermodynamics [52], to consider the interface as an additional independent dynamical variable. As a comprehensive comparison between sharp and diffuse interfaces is beyond our scope, we briefly mention that we numerically explored how our calculations of total energy, entropy, and interfacial thickness are impacted by discretization size. These quantities appear to remain unaffected even with varying the spacing step (Δx). However, these findings are ultimately not incorporated into the final thesis due to time constraints.

Now, the anchoring energy density, or fourth term in Eq.(3.1), serves to couple the LC orientation with the concentration gradient [53]. Here, we introduce an additional isotropic term [54] ($\boldsymbol{I}/d$) compared to Eq. (9) of [22]. Other terms depending on gradients of ϕ, such as $(\partial\phi/\partial r_i)(\partial Q_{ij}/\partial r_j)$, may also be considered [55]. We however restrict our model to one coupling term as we better understand how its coefficient κ' controls the orientation of the director vector at the interface (see below).

The fifth term represents the classical FO energy density, under the approximation of a single elastic constant (K), penalizing gradients of $\boldsymbol{Q}$ and capturing the orientational elasticity on the length scale of the liquid crystal [56].

The coefficients κ_E, κ' and K have dimensions of [force], or [energy/length]. κ_E and K are expected to be positive, while κ' can be of either sign, depending on whether the preferred orientation of the liquid crystal is perpendicular ($\kappa' < 0$) or parallel ($\kappa' > 0$) to the interface [22]. To avoid further complicating our calculations, we ignore the dependency on ϕ of both κ_E and κ'. This includes omitting the Lifshitz–de Gennes inversely proportional relation ($\kappa_E \propto 1/\phi$), as this decision essentially leads to the same results [57]. We however maintain the ϕ dependency of K.

The second term in the second line of Eq.(3.1) represents the conventional FH energy density, describing the repulsion interaction between the two liquids or the tendency towards

phase separation. Its positive constant parameter χ has dimensions of [energy/mass], contrasting with [energy/lattice site] in the FH lattice theory. To convert, the mass associated with each lattice site needs to be known. This term reaches its minimum when ϕ approaches either zero or one, corresponding to cases where each phase (isotropic or nematic) is predominantly composed of a unique component. Since this scenario is atypical for most lyotropic LCs, with ordered phases having a particular component ratio different from zero or one [58], additional terms are below introduced.

The final energetic contribution $\epsilon_{\mathrm{LdG,E}}$ in Eq. (3.1) models the isotropic-nematic transition. To align with the GENERIC framework, we split the conventional LdG free energy into its energetic and entropic contributions, following the classical analysis of Landau [59]. Here, we introduce only its energetic part, as represented by the quartic form [60]

$$\epsilon_{\mathrm{LdG,E}}(\rho, \phi, \boldsymbol{Q}) = -\rho B(\phi) \left[\frac{1}{6} Q_{ij} Q_{ji} + \frac{1}{3} Q_{ij} Q_{jk} Q_{ki} - \frac{1}{4} \left(Q_{ij} Q_{ji} \right)^2 \right], \tag{3.2}$$

where the nonnegative parameter $B(\phi)$ has dimensions of [energy/mass]. Additionally, $B(\phi)$ should be proportional to ϕ^2, as $\epsilon_{\mathrm{LdG,E}}$ arises from the pairwise interactions between the LC units [60], consistent with current literature stating that solely LC constituents contribute to this term [22].

To clarify between the entropic and energetic origins of each term, we consider that, generally speaking, energetic contributions tend to favor ordering or alignment (e.g. the nematic phase), whereas entropic effects are associated with randomness. For the LdG free energy, the distinction is as follow: we designate $\epsilon_{\mathrm{LdG,E}}$ as the energetic part because its coefficient does not explicitly depend on temperature or vary slowly within the temperature range of interest. In contrast, its entropic counterpart exhibits a greater temperature dependency, thus implying a higher degree of disorder [59].

Although we acknowledge that incorporating terms into either entropic or energetic functionals can be reconsidered (as in Eq. (2.4b) of [61], where the FH energy has mistakenly

been included in the entropy), we strongly recommend proceeding with caution, as improper assignments can prevent the entropy from maximizing, resulting in a loss of physical meaning, even if other variables appear to behave correctly.

Note that both the FH and LdG energy terms are proportional to ρ to provide an extensive density characteristic and their opposing effects are discussed in detail in the succeeding entropy section.

Moving forward, we take functional derivatives with respect to the field variables $\boldsymbol{x}$, as depicted in Eq. (2.1), to obtain

$$\frac{\delta E}{\delta \boldsymbol{x}} = \begin{pmatrix} \frac{\delta E}{\delta \rho} \\ \frac{\delta E}{\delta \boldsymbol{m}} \\ \frac{\delta E}{\delta \epsilon} \\ \frac{\delta E}{\delta \phi} \\ \frac{\delta E}{\delta \boldsymbol{Q}}^{\mathtt{ST}} \end{pmatrix} = \begin{pmatrix} -\frac{1}{2}{v_k}^2 + \chi\phi(1-\phi) + \frac{\epsilon_{\mathrm{LdG,E}}}{\rho} \\ v_k \\ 1 \\ \mu_1 + \mu_2 + \mu_3 \\ \boldsymbol{H}_{an} + \boldsymbol{H}_{el} + \boldsymbol{H}_{LdG} \end{pmatrix}, \tag{3.3}$$

where

$$\mu_1 = -\kappa_E \frac{\partial}{\partial r_k}\frac{\partial \phi}{\partial r_k} - \kappa' \frac{\partial}{\partial r_k}\left[\left(Q_{kl} + \frac{I_{kl}}{d}\right)\frac{\partial \phi}{\partial r_l}\right], \tag{3.4}$$

$$\mu_2 = \frac{1}{2}\frac{dK}{d\phi}\left(\frac{\partial Q_{ij}}{\partial r_k}\right)^2 \tag{3.5}$$

$$\mu_3 = \rho\chi(1-2\phi) + \frac{1}{B}\frac{dB}{d\phi}\,\epsilon_{\mathrm{LdG,E}} \tag{3.6}$$

$$(H_{an})_{ij} = \frac{1}{2}\kappa'\left[\left(\frac{\partial \phi}{\partial r_i}\right)\left(\frac{\partial \phi}{\partial r_j}\right) - \frac{I_{ij}}{d}\left(\frac{\partial \phi}{\partial r_l}\right)^2\right], \tag{3.7}$$

$$(H_{el})_{ij} = -K\frac{\partial}{\partial r_k}\frac{\partial Q_{ij}}{\partial r_k}, \tag{3.8}$$

$$(H_{LdG})_{ij} = -\rho B(\phi)\left[\frac{1}{3}Q_{ij} + Q_{ik}Q_{kj} - \left(Q_{ij} + \frac{I_{ij}}{d}\right)Q_{lk}Q_{kl}\right], \tag{3.9}$$

and we further introduce the velocity field $\boldsymbol{v} = \boldsymbol{m}/\rho$. In accordance with standard conventions,

we denote the expressions appearing in the derivatives of ϕ and $\boldsymbol{Q}$ with μ and $\boldsymbol{H}$, commonly related to the chemical potential and molecular field, respectively. As usual, the symmetric and traceless (ST) properties of $\boldsymbol{Q}$ are also reflected in its associated functional derivative

$$\left(\frac{\delta E}{\delta Q_{ij}}\right)^{\mathtt{ST}} = \frac{\delta E}{\delta Q_{ij}} - \frac{I_{ij}}{d}\left(\frac{\delta E}{\delta \boldsymbol{Q}}\right)_{ll}, \tag{3.10}$$

where the second term on the right-hand side (RHS) acts as a Lagrange multiplier, ensuring the traceless condition, and adds terms proportional to $\boldsymbol{I}/d$ in equations (3.7) and (3.9). Hereafter, we omit the (ST) superscripts when referring to the functional derivatives of $\boldsymbol{Q}$ to simplify notation.

3.3 Entropy

Similar to the previous subsection, the entropy S is also written as an integral over V. We split this entropy into two terms: 1) the background fluid entropy density $s_0(\rho, \epsilon)$, used to later define the background pressure and the system temperature, and 2) any additional entropic contribution $\mathcal{S}$, expressed in terms of $\boldsymbol{x}$ and its spatial derivatives,

$$\begin{aligned} S(x) &= \int \Big[s_0(\rho,\epsilon) + \mathcal{S}(x) \Big] dV \\ &= \int \left[s_0(\rho,\epsilon) + \frac{1}{2}\kappa_S \left(\frac{\partial \phi}{\partial r_k}\right)^2 + s_{\mathrm{mix}}(\rho,\phi) + \epsilon_{\mathrm{LdG,S}}(\rho,\phi,\boldsymbol{Q}) \right] dV. \end{aligned} \tag{3.11}$$

Here, $\mathcal{S}$ encompasses the interfacial and mixing entropy densities, as well as the LdG entropic effect. We incorporate the $(\partial\phi/\partial r_k)^2$ term in both equations (3.1) and (3.11) since its resulting interfacial tension could contain both energetic and entropic properties. The constant parameter κ_S has dimensions of [entropy/length] or [force/temperature], with its subscript S denoting its entropic effect. Correspondingly, in Eq. (3.1), the subscript E of κ_E denotes an energetic contribution.

The entropy of mixing, which counteracts the FH energetic term introduced in Eq. (3.2), takes the conventional form

$$s_{\text{mix}}(\rho,\phi) = -\rho\Big[C_1\,\phi\ln\phi + C_2(1-\phi)\ln(1-\phi)\Big], \tag{3.12}$$

consistent with the statistical mechanics formulation for mixtures [62]. Here, the coefficients C_1 (C_2) represent Boltzmann's constant divided by the mass of one liquid crystal (or isotropic liquid) molecule. Furthermore, as ϕ approaches the pure species limit ($\phi = 0, 1$), this term tends towards zero with an infinite slope, implying a diverging chemical potential. In other words, beyond these two limits, our mixing entropy yields imaginary values, confining the concentration to the physical range $\phi \in [0, 1]$. Such a restriction cannot be enforced by the frequently used quartic free energy $(1-\phi^2)^2$ term [22, 23, 48], which potentially leads to unrealistic concentration estimates.

The last term in Eq. (3.11), or the entropic part of the LdG free energy, serves as the opposite effect of $\epsilon_{\text{LdG,E}}$ in Eq. (3.1), and reads

$$\epsilon_{\text{LdG,S}}(\rho,\phi,\boldsymbol{Q}) = -\frac{1}{2}\rho A(\phi)\,Q_{ij}Q_{ji}, \tag{3.13}$$

where the parameter A has dimensions of [entropy/mass]. We highlight that the LdG parameters A and B depend on concentration and are inherently material-dependent. A common practice in the LC literature [22] introduces a common factor of ϕ^2 for both functions $A(\phi)$ and $B(\phi)$. While we understand the rationale behind $B(\phi)$ (stemming from pair-wise interactions), this reasoning might not extend well for the entropic contribution $A(\phi)$, which is associated with Brownian forces and expected to be proportional to ϕ. The ratio of $B(\phi)/A(\phi) \propto \phi$ aligns well with the theory developed in [60]. Furthermore, this quadratic orientational entropy contribution may suffice, but additional terms, with a logarithm of the determinant of $\boldsymbol{Q}$, can also be considered. A more explicit form for both $A(\phi)$ and $B(\phi)$ is

later provided when solving the resulting time-evolution equations.

Similar to their opposing effects, the overall factor ρ provides the character of an extensive density for both $\epsilon_{\mathrm{LdG,S}}$ and s_{mix}.

Finally, we compute the derivatives of the entropy functional with respect to the independent field $\boldsymbol{x}$, as dictated in Eq. (2.1), to get

$$\frac{\delta S}{\delta \boldsymbol{x}} = \begin{pmatrix} \frac{\partial s_0}{\partial \rho} + \frac{s_{\mathrm{mix}}}{\rho} - \frac{1}{2}A(\phi)Q_{ij}Q_{ji} \\ 0_k \\ 1/T \\ -\kappa_S \frac{\partial}{\partial r_k}\frac{\partial \phi}{\partial r_k} - \rho\Big[C_1 \ln\phi - C_2 \ln(1-\phi) + C_1 - C_2\Big] - \frac{1}{2}\frac{dA}{d\phi}\,\rho\, Q_{ij}Q_{ji} \\ -A(\phi)\rho\, Q_{ij} \end{pmatrix}, \tag{3.14}$$

where the third component of this gradient allows us to define the background fluid temperature T by $1/T = \partial s_0/\partial \epsilon$.

3.4 Poisson Operator

Based on equations (2.60) and (4.44) of [32] and (11.5-37) of [29], we introduce the following $\boldsymbol{L}$ matrix, or Poisson Operator, with indices i, j and k, l referring to the left side [rows] and the right side [columns], respectively

$$L = -\begin{pmatrix} 0 & \frac{\partial}{\partial r_k}\rho & 0 & 0 & 0 \\ \rho\frac{\partial}{\partial r_i} & \frac{\partial}{\partial r_k}m_i + m_k\frac{\partial}{\partial r_i} & \epsilon\frac{\partial}{\partial r_i} + \frac{\partial}{\partial r_m}\pi^S_{mi} & -\frac{\partial \phi}{\partial r_i} & \boldsymbol{L}_{25} \\ 0 & \frac{\partial}{\partial r_k}\epsilon + \pi^S_{km}\frac{\partial}{\partial r_m} & 0 & 0 & 0 \\ 0 & \frac{\partial \phi}{\partial r_k} & 0 & 0 & 0 \\ 0 & \boldsymbol{L}_{52} & 0 & 0 & 0 \end{pmatrix}, \tag{3.15}$$

with

$$\begin{aligned}\boldsymbol{L}_{25} &= -\frac{\partial Q_{kl}}{\partial r_i} - \frac{\partial}{\partial r_m}\left[I_{ki}\left(Q_{ml}+\frac{I_{ml}}{d}\right)+\left(Q_{km}+\frac{I_{km}}{d}\right)I_{il}\right] \\ &+ 2\frac{\partial}{\partial r_m}\left(Q_{mi}+\frac{I_{mi}}{d}\right)\left(Q_{kl}+\frac{I_{kl}}{d}\right), \end{aligned} \tag{3.16}$$

and

$$\begin{aligned}\boldsymbol{L}_{52} &= \frac{\partial Q_{ij}}{\partial r_k} - \left[I_{ik}\left(Q_{mj}+\frac{I_{mj}}{d}\right)+\left(Q_{im}+\frac{I_{im}}{d}\right)I_{kj}\right]\frac{\partial}{\partial r_m} \\ &+ 2\left(Q_{ij}+\frac{I_{ij}}{d}\right)\left(Q_{km}+\frac{I_{km}}{d}\right)\frac{\partial}{\partial r_m}. \end{aligned} \tag{3.17}$$

In equation (3.15) and onwards, we follow the GENERIC conventions, where derivatives of the form $\frac{\partial \psi}{\partial r_m}$ act *only* on ψ, an auxiliary function in this context, or on what appears in the numerator. In contrast, $\frac{\partial}{\partial r_m}$, with nothing in the numerator, indicates that differentiation applies to all subsequent terms, particularly after matrix multiplication. We also clarify that the columns and rows in the Poisson brackets follow the order of $\boldsymbol{x}$. For instance, the (2,4)-elements represents the terms associated with the second and fourth variable ($\boldsymbol{m}$ and ϕ).

Since most of the terms shown in equation (3.15) have been extensively described in [32] and [35], we summarize the basic ideas behind the construction of this operator. Firstly, when convection is the only reversible effect considered in the model, as is common in hydrodynamics, all non-zero entries are found in the row and column associated with the momentum density. Furthermore, we use the antisymmetric property of the $\boldsymbol{L}$ matrix to determine these non-zero entries. We recognize that this antisymmetry may not be immediately apparent to those unfamiliar with the GENERIC framework. In particular, it becomes less evident after certain calculations, typically arising from integration by parts, which, however, simplify the application of the Poisson operator. The antisymmetry can be

thought of as a flip between some operators and signs, such as $\frac{\partial}{\partial \boldsymbol{r}}\rho$ versus $\rho\frac{\partial}{\partial \boldsymbol{r}}$. Specifically, the (1,2)-element gives rise to the continuity equation, after multiplication with the energy gradient (equation 3.3), and then (2,1)-element appears due to the antisymmetric restriction.

Secondly, in GENERIC, hydrodynamic variables have well-established entries in both their momentum column and row, based on their tensorial orders, physical properties, and the type of convective phenomenon driving their behavior [32, 33]. With that in mind, the (2,2)-element is chosen for the "lower-convected" vector density $\boldsymbol{m}$, and the (2,4)- and (4,2)-elements are used to describe scalars, such as ϕ. Additionally, the (2,5)- and (5,2)-elements are based on (11.5-37) of [29], referring to the LC $\mathbf{Q}$-tensor Poisson bracket, used by Beris-Edwards, in integral form. Lastly, the (2,3)- and (3,2)-elements appear when describing the scalar density ϵ, which has a spatial derivative similar to other scalar densities, such as ρ in the (1,2)-element, but also includes the pressure tensor's entropic contribution π^S.

Thirdly, the occurrence of a pressure tensor in the Poisson bracket is an important result of the GENERIC framework. To understand its physical interpretation, consider that in simple fluids, volume changes, such as compression, affect not only the pressure but also the internal energy, which, in turn, impacts the momentum balance, as energy and momentum are intrinsically connected. These volume changes are typically described by velocity gradients and represented by a Newtonian stress tensor. However, in complex fluids, the (scalar) pressure and the Newtonian stress tensor are insufficient due to the presence of configurational variables like $\boldsymbol{Q}$. Therefore, an additional pressure tensor's entropic contribution, in the elements associated with ϵ, is needed to accurately represent the fluid's internal structure.

Mathematically speaking, the entropic pressure tensor is determined by the first degeneracy condition, equation (2.2), involving the $\boldsymbol{L}$ matrix and the entropy gradient in equation (3.14).

Its general form is given as

$$\begin{aligned}
\boldsymbol{\pi}^S_{mi} &= T\Bigg\{ \left[\rho\frac{\partial}{\partial r_k}\left(\frac{\partial \mathcal{S}}{\partial(\partial_k\rho)}\right) + \epsilon\frac{\partial}{\partial r_k}\left(\frac{\partial \mathcal{S}}{\partial(\partial_k\epsilon)}\right) - \rho\frac{\partial \mathcal{S}}{\partial\rho} - \epsilon\frac{\partial \mathcal{S}}{\partial\epsilon} + \mathcal{S} + \frac{p_0}{T}\right] I_{mi} \\
&- \left[\left(\frac{\partial\rho}{\partial r_i}\right)\left(\frac{\partial \mathcal{S}}{\partial(\partial_m\rho)}\right) + \left(\frac{\partial\epsilon}{\partial r_i}\right)\left(\frac{\partial \mathcal{S}}{\partial(\partial_m\epsilon)}\right)\right. \\
&+ \left.\left(\frac{\partial\phi}{\partial r_i}\right)\left(\frac{\partial \mathcal{S}}{\partial(\partial_m\phi)}\right) + \left(\frac{\partial Q_{kl}}{\partial r_i}\right)\left(\frac{\partial \mathcal{S}}{\partial(\partial_m Q_{kl})}\right)\right] \\
&- 2\left[\left(Q_{mi} + \frac{I_{mi}}{d}\right)\left(Q_{kl} + \frac{I_{kl}}{d}\right)\frac{\delta S}{\delta Q_{kl}} - \left(Q_{ml} + \frac{I_{ml}}{d}\right)\frac{\delta S}{\delta Q_{li}}\right]\Bigg\}. \qquad (3.18)
\end{aligned}$$

The detailed calculation of the entropic stress tensor can be found in Chapter 9. We define the background pressure p_0 using the local-equilibrium Euler equation:

$$\frac{p_0}{T} = s_0 - \rho\frac{\partial s_0}{\partial\rho} - \epsilon\frac{\partial s_0}{\partial\epsilon}, \qquad (3.19)$$

where T, the background fluid temperature, appears naturally.

As an example, by combining equations (3.18) and (3.11), the entropic tensor contribution in 3D ($d = 3$) is given by:

$$\begin{aligned}
\boldsymbol{\pi}^S_{mi} &= I_{mi}\, p_0 - 2\rho T A(\phi)\left(Q_{ml} + \frac{1}{3}I_{ml}\right)\left(Q_{li} - I_{li}\, Q_{jk}Q_{kj}\right) \\
&- \kappa_S T\left[\frac{\partial\phi}{\partial r_m}\frac{\partial\phi}{\partial r_i} - \frac{1}{2}\left(\frac{\partial\phi}{\partial r_k}\right)^2 I_{mi}\right], \qquad (3.20)
\end{aligned}$$

3.5 Friction Matrix

To construct our dissipative matrix, we first set the entries associated with the variable ρ to zero by ignoring the irreversible processes involved in mass balance. Next, following GENERIC conventions, we assume that the friction matrix $\boldsymbol{M}$ is the sum of multiple matrices, each fulfilling the necessary conditions: symmetric, positive semidefinite, and satisfying the degeneracy equation (2.3). Importantly, these matrices are used to incorporate physical

mechanisms specific to each variable into our model. We introduce the following dissipative processes:

$$\boldsymbol{M}^{=}\boldsymbol{M}_{\boldsymbol{Q}}^{\text{relax}} + \boldsymbol{M}_{\phi}^{\text{diff}} + \boldsymbol{M}_{\boldsymbol{Q}}^{\text{slip}}, \tag{3.21}$$

with

$$\boldsymbol{M}_{\boldsymbol{Q}}^{\text{relax}} = \begin{pmatrix} \frac{\delta E}{\delta \boldsymbol{Q}} : \boldsymbol{R} : \frac{\delta E}{\delta \boldsymbol{Q}} & 0 & -\frac{\delta E}{\delta \boldsymbol{Q}} : \boldsymbol{R} \\ 0 & 0 & 0 \\ -\mathbf{R} : \frac{\delta E}{\delta \mathbf{Q}} & 0 & \mathbf{R} \end{pmatrix}, \tag{3.22}$$

and

$$\boldsymbol{M}_{\phi}^{\text{diff}} = \begin{pmatrix} -\frac{\delta E}{\delta \phi}\frac{1}{\rho}\frac{\partial}{\partial \boldsymbol{r}} \cdot \rho \boldsymbol{D} \cdot \frac{\partial}{\partial \boldsymbol{r}}\frac{1}{\rho}\frac{\delta E}{\delta \phi} & \frac{\delta E}{\delta \phi}\frac{1}{\rho}\frac{\partial}{\partial \boldsymbol{r}} \cdot \rho \boldsymbol{D} \cdot \frac{\partial}{\partial \boldsymbol{r}}\frac{1}{\rho} & 0 \\ \frac{1}{\rho}\frac{\partial}{\partial \boldsymbol{r}} \cdot \rho \boldsymbol{D} \cdot \frac{\partial}{\partial \boldsymbol{r}}\frac{1}{\rho}\frac{\delta E}{\delta \phi} & -\frac{1}{\rho}\frac{\partial}{\partial \boldsymbol{r}} \cdot \rho \boldsymbol{D} \cdot \frac{\partial}{\partial \boldsymbol{r}}\frac{1}{\rho} & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.23}$$

Both $\boldsymbol{M}_{\phi}^{\text{diff}}$ and $\boldsymbol{M}_{\boldsymbol{Q}}^{\text{relax}}$ represent the 3x3 block of the friction matrix associated with the components ϵ, ϕ and $\boldsymbol{Q}$, while the other entries are zero. $\boldsymbol{M}_{\phi}^{\text{diff}}$ indicates a diffusion process for ϕ, similar to Eq. (4.65) of [32], whereas $\boldsymbol{M}_{\boldsymbol{Q}}^{\text{relax}}$ describes a relaxation mechanism for the tensor $\boldsymbol{Q}$ according to Eq. (4.54) of [32].

To envision the construction of these blocks, we first explain the mathematical formulation of their physical mechanisms. For modeling diffusion processes, we choose to add spatial derivatives, similar to other diffusion's law like Fick's. The positive semidefinite tensor $\boldsymbol{D} = \boldsymbol{D}(\rho, \epsilon, \phi, \boldsymbol{Q})$, appearing in between such derivatives, describes an anisotropic diffusion behavior. In contrast, because a relaxation process implies the absence of transport phenomena, the occurrence of spatial derivatives is not needed, just a relaxation parameter $\boldsymbol{R}$ to drive the system towards a minimum or equilibrium.

Furthermore, a concrete form for the symmetric and positive semidefinite fourth-rank

tensor $\boldsymbol{R}$ has been proposed in Eq. (11.5-40) of [29] (see also Eqs. (4.88) and (4.107) of [32]),

$$R_{ijkl} = R(\rho, \epsilon, \phi, \boldsymbol{Q}) \left[\frac{1}{2} (I_{ik}\, I_{jl} + I_{il}\, I_{jk}) \right], \tag{3.24}$$

where we drop the $3Q_{ij}Q_{kl}$ term, in the original equation, to recover the Ginzburg-Landau equation in the absence of flow (e.g., we are working in a non-inertial framework) with a positive relaxation parameter $R(\rho, \epsilon, \phi, \boldsymbol{Q})$ that has dimensions of [volume/(time·entropy)]. The work of Doi [60] suggests $R \propto \phi^{-3}$, and we expect further dependence on $\boldsymbol{Q}$ and ρ (perhaps $1/\rho$).

Second, to determine the non-vanishing elements in $\boldsymbol{M}_Q^{\text{relax}}$, we proceed as follows: 1) Place the tensor $\boldsymbol{R}$ in the $\boldsymbol{M}_Q^{\text{relax}}$ (3,3)-element, related to the driving force $\frac{\delta S}{\delta \boldsymbol{Q}}$; 2) satisfy the degeneracy condition (Eq. 2.3) in the third row with the (1,3)-element; 3) complete the (3,1)-element by means of the symmetry condition; and 4) fulfill the degeneracy requirement again in the first row with the (1,1)-element. Similarly, we determine the (1,1)-, (2,1)-, and (1,2)-elements of the $\boldsymbol{M}_\phi^{\text{diff}}$ matrix by first placing the spatial derivatives in its center and then following an analogous procedure.

Next, we introduce the slip coefficient λ, as discussed in Section 1.2, by incorporating Schowalter derivatives that model a more realistic viscoelastic behavior. This accounts for the tunable response of $\boldsymbol{Q}$ to velocity gradients and allows us to recover the BE equation (1.2). We base our $\boldsymbol{M}_{\boldsymbol{Q}}^{\text{slip}}$ on the BE dissipative bracket (Eqs. 11.4-38 and 11.5-41 of [29]) in integral form. To translate it into the language of matrices, we follow the approach outlined in Section V.B of [35] and incorporate ideas from equation (4.79) of [32], including symmetry and degeneracy conditions, as above, to complete all the non-zero entries. The final ingredient

of our model is the resulting $\boldsymbol{M}_{\boldsymbol{Q}}^{\text{slip}}$, which reads

$$\boldsymbol{M}_{\boldsymbol{Q}}^{\text{slip}} = \begin{pmatrix} 0 & \boldsymbol{M}_{12} & \boldsymbol{M}_{13} \\ \boldsymbol{M}_{21} & 0 & \boldsymbol{M}_{23} \\ \boldsymbol{M}_{31} & \boldsymbol{M}_{32} & 0 \end{pmatrix}, \tag{3.25}$$

with

$$(M_{12})_i = \frac{\partial}{\partial r_m} L^Q_{mikl} \frac{\delta E}{\delta Q_{kl}} T - \frac{\partial}{\partial r_m} L^Q_{mijj} \left(Q_{kl} + \frac{I_{kl}}{d} \right) \frac{\delta E}{\delta Q_{kl}} T, \tag{3.26}$$

$$(M_{13})_{ikl} = -\frac{\partial}{\partial r_m} L^Q_{mikl} T + \frac{\partial}{\partial r_m} L^Q_{mijj} \left(Q_{kl} + \frac{I_{kl}}{d} \right) T, \tag{3.27}$$

$$(M_{21})_k = T \frac{\delta E}{\delta Q_{ij}} L^Q_{ijkl} \frac{\partial}{\partial r_l} - T \frac{\delta E}{\delta Q_{ij}} \left(Q_{ij} + \frac{I_{ij}}{d} \right) L^Q_{kmll} \frac{\partial}{\partial r_m}, \tag{3.28}$$

$$(M_{23})_{ij} = -T \left[\frac{\partial}{\partial r_k} \frac{\delta E}{\delta m_l} \right] L^Q_{klij} + T \left[\frac{\partial}{\partial r_k} \frac{\delta E}{\delta m_l} \right] \left(Q_{lk} + \frac{I_{lk}}{d} \right) L^Q_{ijmm}, \tag{3.29}$$

$$(M_{31})_{mij} = -T L^Q_{mijk} \frac{\partial}{\partial r_k} + T L^Q_{mill} \left(Q_{kj} + \frac{I_{kj}}{d} \right) \frac{\partial}{\partial r_k}, \tag{3.30}$$

$$(M_{32})_{mi} = T L^Q_{mijk} \left[\frac{\partial}{\partial r_k} \frac{\delta E}{\delta m_j} \right] - T L^Q_{mill} \left(Q_{kj} + \frac{I_{kj}}{d} \right) \left[\frac{\partial}{\partial r_k} \frac{\delta E}{\delta m_j} \right], \tag{3.31}$$

and

$$L^Q_{ijkl} = \frac{(\lambda - 1)}{2} \left[I_{il} \left(Q_{jk} + \frac{I_{jk}}{d} \right) + I_{jl} \left(Q_{ik} + \frac{I_{ik}}{d} \right) + I_{ik} \left(Q_{jl} + \frac{I_{jl}}{d} \right) + I_{jk} \left(Q_{il} + \frac{I_{il}}{d} \right) \right] \tag{3.32}$$

3.6 Time-Evolution Equations

By inserting all the building blocks developed in the previous sections into the GENERIC equation (2.1), we derive the final time-evolution equations. The continuity equation takes

the familiar form,

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial r_k}(\rho\, v_k). \tag{3.33}$$

Furthermore, in the momentum-balance equation, or Generalized Navier-Stokes equation, a number of additional pressure-tensor contributions appear

$$\frac{\partial m_i}{\partial t} = -\frac{\partial}{r_m}\Big[v_m m_i + \pi^S_{mi} + \pi^E_{mi} - \pi^{slip}_{mi}\Big], \tag{3.34}$$

where we use the definition of the entropic tensor in (3.18). The energetic contribution, $\boldsymbol{\pi}^E$, arises after condensing all terms from the matrix multiplication of $L(\boldsymbol{x}) \cdot \frac{\delta E(\boldsymbol{x})}{\delta \boldsymbol{x}}$. Its calculation is left as an exercise for interested readers, as it is similar in spirit to the entropic pressure tensor derivation (see Chapter 9). The explicit form of the energetic pressure tensor reads

$$\begin{aligned}
\boldsymbol{\pi}^E_{mi} &= -\Bigg\{\left[\rho\frac{\partial}{\partial r_k}\left(\frac{\partial \mathcal{E}}{\partial(\partial_k \rho)}\right) + \epsilon\frac{\partial}{\partial r_k}\left(\frac{\partial \mathcal{E}}{\partial(\partial_k \epsilon)}\right) - \rho\frac{\partial \mathcal{E}}{\partial \rho} - \epsilon\frac{\partial \mathcal{E}}{\partial \epsilon} + \mathcal{E}\right] I_{mi} \\
&- \left[\left(\frac{\partial \rho}{\partial r_i}\right)\left(\frac{\partial \mathcal{E}}{\partial(\partial_m \rho)}\right) + \left(\frac{\partial \epsilon}{\partial r_i}\right)\left(\frac{\partial \mathcal{E}}{\partial(\partial_m \epsilon)}\right)\right. \\
&+ \left.\left(\frac{\partial \phi}{\partial r_i}\right)\left(\frac{\partial \mathcal{E}}{\partial(\partial_m \phi)}\right) + \left(\frac{\partial Q_{kl}}{\partial r_i}\right)\left(\frac{\partial \mathcal{E}}{\partial(\partial_m Q_{kl})}\right)\right] \\
&- 2\Bigg[\left(Q_{mi} + \frac{I_{mi}}{d}\right)\left(Q_{kl} + \frac{I_{kl}}{d}\right)\frac{\delta E}{\delta Q_{kl}} - \left(Q_{ml} + \frac{I_{ml}}{d}\right)\frac{\delta E}{\delta Q_{li}}\Bigg]\Bigg\}.
\end{aligned} \tag{3.35}$$

Note the similarities between $\boldsymbol{\pi}^E$ and $\boldsymbol{\pi}^S$ which can be used for simplification, especially considering that $E_{free} = E - T\ S$. In 3D, the energetic tensor, using our total energy definition in Eq. (3.1), can be written as the sum of various contributions,

$$\boldsymbol{\pi}^E = \boldsymbol{\pi}^{E(1)} + \boldsymbol{\pi}^{E(2)} + \boldsymbol{\pi}^{E(3)}, \tag{3.36}$$

given by

$$\pi_{mi}^{E(1)} = \kappa_E \left[\frac{\partial \phi}{\partial r_m} \frac{\partial \phi}{\partial r_i} - \frac{1}{2} \frac{\partial \phi}{\partial r_k} \frac{\partial \phi}{\partial r_k} I_{mi} \right] + \kappa' \left[\frac{\partial \phi}{\partial r_k} \left(Q_{kl} + \frac{1}{3} I_{kl} \right) \frac{\partial \phi}{\partial r_l} \left(Q_{mi} - \frac{1}{6} I_{mi} \right) \right], \quad (3.37)$$

$$\boldsymbol{\pi}^{E(2)} = 2\rho B(\phi) \left[\boldsymbol{I} \cdot - \left(\boldsymbol{Q} + \frac{1}{3} \boldsymbol{I} \right) \boldsymbol{I} : \right] \left(\boldsymbol{Q} + \frac{1}{3} \boldsymbol{I} \right)^2 \cdot \left(\boldsymbol{Q} - \boldsymbol{I} \operatorname{tr} \boldsymbol{Q}^2 \right), \quad (3.38)$$

and

$$\begin{aligned} \pi_{mi}^{E(3)} &= K \Bigg[\frac{\partial Q_{kl}}{\partial r_m} \frac{\partial Q_{kl}}{\partial r_i} - \frac{1}{2} \frac{\partial Q_{kl}}{\partial r_j} \frac{\partial Q_{kl}}{\partial r_j} I_{mi} + 2 \left(Q_{ml} + \frac{1}{3} I_{ml} \right) \frac{\partial^2 Q_{li}}{\partial {r_k}^2} \\ &- 2 \left(Q_{mi} + \frac{1}{3} I_{mi} \right) Q_{kl} \frac{\partial^2 Q_{kl}}{\partial {r_j}^2} \Bigg]. \end{aligned} \quad (3.39)$$

In numerically solving the time-evolution equations, we use either these explicit versions, equations (3.37), (3.38), and (3.39), or the more general form, given by Eq. (3.35). Additionally, $\boldsymbol{\pi}_{mi}^{slip}$ is the contribution from $\boldsymbol{M}_{\boldsymbol{Q}}^{\text{slip}}$ and reads

$$\begin{aligned} \boldsymbol{\pi}_{mi}^{slip} &= \Bigg[\left(Q_{ml} + \frac{I_{ml}}{d} \right) \frac{\delta E}{\delta Q_{li}} + \left(Q_{il} + \frac{I_{il}}{d} \right) \frac{\delta E}{\delta Q_{lm}} \\ &- 2 \left(Q_{mi} + \frac{I_{mi}}{d} \right) \left(Q_{kl} + \frac{I_{kl}}{d} \right) \frac{\delta E}{\delta Q_{kl}} \\ &- \left(Q_{ml} + \frac{I_{ml}}{d} \right) \frac{\delta S}{\delta Q_{li}} T - \left(Q_{il} + \frac{I_{il}}{d} \right) \frac{\delta S}{\delta Q_{lm}} T \\ &+ 2 \left(Q_{mi} + \frac{I_{mi}}{d} \right) \left(Q_{kl} + \frac{I_{kl}}{d} \right) \frac{\delta S}{\delta Q_{kl}} T \Bigg] \left(\lambda - 1 \right) \end{aligned} \quad (3.40)$$

Equations (3.18), (3.35), and (3.40) can be combined to obtain the same terms, and more, to those in Equation (1.6). We stress that in any of these pressure contribution a term of the form $\phi \frac{\delta E}{\delta \phi}$, such as that found in Eq. (13) of [22], does not appear and should not be expected, a verification achieved by applying the GENERIC formalism.

In addition, the internal energy balance is expressed as

$$\begin{aligned}\frac{\partial \epsilon}{\partial t} &= -\frac{\partial}{\partial r_k}\left(v_k\,\epsilon\right) - \pi^S_{km}\,W_{km} + \frac{\delta E}{\delta Q_{kl}} R\left(\frac{1}{T}\frac{\delta E}{\delta Q_{kl}} - \frac{\delta S}{\delta Q_{kl}}\right) \\ &+ T\left(\lambda - 1\right)\left[2\left(Q_{ij} + \frac{I_{ij}}{d}\right)\frac{\delta S}{\delta Q_{ij}}\left(Q_{lk} + \frac{I_{lk}}{d}\right)W_{lk} - \left(Q_{ik} + \frac{I_{ik}}{d}\right)W_{kj}\,\frac{\delta S}{\delta Q_{ij}}\right. \\ &- \left.\left(Q_{ik} + \frac{I_{ik}}{d}\right)W_{jk}\,\frac{\delta S}{\delta Q_{ij}}\right] - \frac{\delta E}{\delta \phi}\frac{1}{\rho}\frac{\partial}{\partial r_k}\rho D_{kl}\,\frac{\partial}{\partial r_l}\frac{1}{\rho}\left(\frac{1}{T}\frac{\delta E}{\delta \phi} - \frac{\delta S}{\delta \phi}\right),\end{aligned} \tag{3.41}$$

the convection-diffusion, or a Generalized Cahn-Hilliard equation for the composition field is given by

$$\frac{\partial \phi}{\partial t} = -v_k\,\frac{\partial \phi}{\partial r_k} + \frac{1}{\rho}\frac{\partial}{\partial r_k}\rho D_{kl}\,\frac{\partial}{\partial r_l}\frac{1}{\rho}\left(\frac{1}{T}\frac{\delta E}{\delta \phi} - \frac{\delta S}{\delta \phi}\right), \tag{3.42}$$

and a convection-relaxation, or a Generalized Beris-Edwards equation for the orientation tensor prays

$$\begin{aligned}\frac{\partial Q_{ij}}{\partial t} &= -v_k\,\frac{\partial Q_{ij}}{\partial r_k} + \left(\lambda \Xi_{ik} + \Omega_{ik}\right)\left(Q_{kj} + \frac{I_{kj}}{d}\right) + \left(Q_{ik} + \frac{I_{ik}}{d}\right)\left(\lambda \Xi_{kj} - \Omega_{kj}\right) \\ &- 2\lambda\left(Q_{ij} + \frac{I_{ij}}{d}\right)\left(Q_{kl} + \frac{I_{kl}}{d}\right)W_{lk} - R\left(\frac{1}{T}\frac{\delta E}{\delta Q_{ij}} - \frac{\delta S}{\delta Q_{ij}}\right),\end{aligned} \tag{3.43}$$

where we recall that $W_{ij} = \partial v_i/\partial r_j$, $\Xi_{jk} = (\partial_j v_k + \partial_k v_j)/2$ and, $\Omega_{jk} = (\partial_k v_j - \partial_j v_k)/2$.

To conclude, we remark that the structure of the time-evolution equations remains unchanged as long as the number of $\boldsymbol{x}$ variables remains the same, even with the addition of new energetic and entropy density contributions in Equations (3.1) and (3.11), if the physics of the system requires it—such as in cases involving flexoelectric or piezoelectric effects [63] or additional elastic terms.

CHAPTER 4

LIQUIDCRYSTALGLBT.JL: A SOLVER FOR THE CONCENTRATION-DEPENDENT NEMATOHYDRODYNAMIC EQUATIONS

To handle the substantial number of variables involved in the intricate concentration-dependent nematohydrodynamic equations presented in Chapter 3, as well as the complex numerical methods described in Section 2.2 used to solve these equations, we develop the `LiquidCrystalGLBT.jl` library, written in the Julia language. In this acronym, 'G' indicates the use of the out-of-equilibrium GENERIC framework to derive our equations, while 'LB' signifies the implementation of the Lattice Boltzmann Method through the `Trixi.jl` package, the 'T' in our name.

Although a comprehensive and detailed description of the code implementation is fairly involved, in this chapter, we highlight the most important features, hoping to broadly clarify our approach. We have designed this code to be modular, allowing for easy integration of new components and enabling the simulation of more complex scenarios. We follow the conventions described in Section 2.2, including Cartesian coordinates.

The working examples to reproduce the results in the subsequent chapters can be found in our repository. If they are not available, interested individuals can contact the author directly.

4.1 Structs and constructors for hydrodynamic variables

We first present the most important structs and constructors of our `LiquidCrystalGLBT.jl` package to handle variables. A *struct* in Julia is a composite data type that groups related variables, or fields, under a single entity, allowing for organized and efficient data management. *Constructors*, on the other hand, are functions that initialize instances of structs and provide

methods to handle specific kinds of data, including mapping between different types or representations.

Our first example is the struct named `phiQFLocal`, designed to encapsulate three key components: the concentration (ϕ), the LC tensor ($\boldsymbol{Q}$), and the distribution function from the LBM (f). Below is a snippet of the code:

```
struct phiQFLocal{N, T, M, L}
    phi::T
    Q::QLocal{N, T, M}
    f::SVector{L, T}
end
```

The `phiQFLocal` struct is parameterized by four types: `N`, `T`, `M`, and `L`. In this computational context, these letters represent generic types and should not be confused with specific variables, such as T or M, that might be used elsewhere in the theoretical description of this thesis. To the best of our capabilities, we use typographically distinct letters to avoid confusion.[1]

- `N` represents the spatial dimension (e.g., 2 or 3).
- `T` denotes the data type (e.g., `Float64`).
- `M` specifies the size of the LC tensor data (e.g., a 2x2 or 3x3 matrix), which varies depending on the spatial dimension.
- `L` indicates the length or number of allowed traveling directions of the distribution function. For example, in 2D, we use 9 allowed directions (see subsection 2.2.2).

The fields within the `phiQFLocal` struct are:

- `phi::T`: A scalar representing the concentration.

1. We apologize if any confusion still arises despite these efforts.

- `Q::QLocal{N, T, M}`: The LC tensor stored in a `QLocal` object, which is defined below.
- `f::SVector{L, T}`: The distribution function is stored as a static vector `SVector`, which is a common data type in Julia similar to an array.

We choose this as the opening example because it includes all of our variables and serves as an easy introduction for readers who are not familiar with structs in Julia. This nomenclature is followed throughout the rest of the chapter. Within this struct, however, there is an underlying structure that we explain as follows. Specifically, in `phiQFLocal`, there is another struct named `QLocal`, which is designed to handle data related to LC tensors in different dimensions and serves as the basis for most snippets. Here is its breakdown:

```
struct QLocal{N, T, M} <: StaticMatrix{N, N, T}
    data::SVector{M, T}

    QLocal{3, T, 6}(data::SVector{6, T}) where {T} = new{3, T,
       6}(data)
    QLocal{2, T, 3}(data::SVector{3, T}) where {T} = new{2, T,
       3}(data)
end
```

This `QLocal` contains three type parameters: `N` for dimension, `T` for data type, and `M` for matrix size. It inherits from `StaticMatrix{N, N, T}`, another Julia data type for storing matrix-type arrays. This indicates that `QLocal` is also a static matrix type with dimensions `N x N`. Furthermore, the `data` field is an `SVector` of length `M` and type `T`, used to store the LC tensor data.

The struct includes two specific constructors depending on the dimensionality of the system: The constructor `QLocal{3, T, 6}` instantiates a `QLocal` object for 3D tensors with 6 elements, taking an `SVector{6, T}` as input. Similarly, the constructor `QLocal{2, T, 3}`

instantiates a `QLocal` object for 2D tensors with 3 elements. In both cases, the symmetric properties of $\boldsymbol{Q}$ are utilized to store the data more efficiently.

Aliases for two and three dimensions are provided to simplify the use of `QLocal`:

```
const QLocal3{T} = QLocal{3, T, 6}
const QLocal2{T} = QLocal{2, T, 3}
```

Here, `QLocal3{T}` is an alias for `QLocal{3, T, 6}`, making it easier to refer to the 3D case of `QLocal`. Similarly, `QLocal2{T}` is an alias for `QLocal{2, T, 3}`, simplifying the reference to the 2D case.

Furthermore, we create constructors for `QLocal` to handle different types of input:

```
QLocal(data::SVector) = qtype(data)(data)
QLocal(data::MVector) = QLocal(SVector(data))
```

The first constructor processes an input `data` of type `SVector`. It uses the `qtype` function, which is not shown here but is analogous to the `phiqtype` function exemplified below, to determine the appropriate `QLocal` type based on the length of the `SVector`. It then creates a new instance of `QLocal` with this type. The second constructor converts an `MVector` into an `SVector` and then creates a `QLocal` instance using the `SVector` constructor. This approach facilitates handling various representations of `data` such as `SVector`, `MVector`, which can be used for visualizing or accessing the data in different ways, as needed.

Next, we provide illustrative snippets of constructors that involve adding a new variale `phi` to create instances of `phiQLocal`:

1. Constructor for 2D Q-Tensor and phi: This constructor provides a convenient way to create an instance of `phiQLocal2` using three lower triangular elements of $\boldsymbol{Q}$. It groups these elements into an `SVector` and passes them to the general `phiQLocal` constructor, using again the symmetric properties of the 2D LC tensor.

```
phiQLocal(phi, q1, q2, q3) = phiQLocal(phi, SVector(q1, q2, q3))
```

2. Constructor for 3D Q-Tensor and phi: This constructor creates an instance of `phiQLocal3` from six elements, which are the non-repeated elements in 3D $\boldsymbol{Q}$. It converts these elements into an `SVector` and then uses the general `phiQLocal` constructor.

```
phiQLocal(phi, q1, q2, q3, q4, q5, q6) = phiQLocal(phi, SVector(q1,
    q2, q3, q4, q5, q6))
```

Lastly, to map these methods between `SVector` types and `phiQLocal`, we define the following constructors:

3. Mapping `SVector` to `phiQLocal`: The function `phiqtype` maps an `SVector` type to the appropriate `phiQLocal` container type based on its length, or more accurately dimension.

```
function phiqtype end
phiqtype(::Type{SVector{3, T}}) where {T} = phiQLocal2{T}
phiqtype(::Type{SVector{4, T}}) where {T} = phiQLocal2{T}
phiqtype(::Type{SVector{6, T}}) where {T} = phiQLocal3{T}
phiqtype(::Type{SVector{7, T}}) where {T} = phiQLocal3{T}
phiqtype(::S) where {S <: SVector} = phiqtype(S)
```

From the last three snippets, we observe that the procedure for adding new variables becomes systematic. We also have different ways to store information based on the dimensionality, which is indicated by the last number in `phiQLocal`, and by increasing the number of variables. The constructors are recursive and are also applicable when the distribution function (f) is involved or incorporated, which gives rise to the constructors and structs of `phiQFLocal`. These details are not shown here for the sake of brevity, but the generalization is straightforward. Importantly, these constructors allow us to perform mathematical operations (such as dot products or double dot products between tensors and vectors) more efficiently, which is a significant computational advantage.

4.2 Differential Operators

When handling the algebraic versions of the derivatives, such as equations (2.5) and (2.6) in our upwind scheme, we design operators inspired by `DiffEqOperators.jl` Julia package. Unfortunately, this package does not support our specific data type, necessitating the development of a custom architecture. We follow a similar procedure as in the previous subsection where we introduce structs and constructors. As the code structure includes several key components, each playing a specific role in handling numerical operations and boundary conditions, below, we explain the most important components and provide corresponding code snippets. It is assumed that the matrices presented below store the discretized values of the hydrodynamic variable on the computational grid.

CenteredDifference: We begin by introducing the `CenteredDifference` struct, representing a centered difference operator parameterized by `N` for dimensions, `T` for data type, and `Tag` for the type of diffential operator used. It has two primary fields: 1) `dr` represents the spacing steps `dr` $= (\Delta x, \Delta y, \Delta z)$ for each dimension. If only one dimension is specified, it assumes that the other spacing steps are equal. 2) `coeffs` stores the coefficients for the centered difference scheme. These coefficients, which are based on the order of the derivative (first or second) expressed in the operator tag, are computed by dividing the regular finite difference coefficients, see equation (2.5), by $(\texttt{dr})^{\texttt{N}}$. Note that versions for `ForwardDifference` and `BackwardDifference` follow a similar structure but are not shown here.

```
struct CenteredDifference{N, T, Tag}
    dr::NTuple{N, T}
    coeffs::NTuple{N, SVector{3, T}}
end

function CenteredDifference{Tag}(Dr::NTuple{N, T}) where {N, T, Tag}
    n = derivative_order(Tag)
```

```
    v = if n == 1
        SVector{3,T}(-0.5, 0, 0.5)
    elseif n == 2
        SVector{3,T}(1, -2, 1)
    end
    coeffs = Tuple(v / Dri^n for Dri in Dr)
    return CenteredDifference{N, T, Tag}(Dr, coeffs)
end

function CenteredDifference{N, Tag}(Dr::T) where {N, T, Tag}
    dr = ntuple(i -> Dr, Val(N))
    return CenteredDifference{Tag}(dr)
end
```

OperatorTag: Next, we describe the `OperatorTag` struct, which includes specific tags like `Laplacian`, `Divergence`, and `Gradient` to specify the type of differential operator to use. The function `derivative_order` links these tags to specific derivative orders, which are used in `CenteredDifference`. Their corresponding functions handle both 2D and 3D cases and accommodate the tensorial rank (e.g., scalar, matrix, vector) of the variable to which the operator is applied, performing the operation with provided coefficients `coeffs`. For brevity, we provide the function for `Laplacian`, a second-order derivative in 3D, as an example. It is parameterized by `A`, the matrix containing the variable values, and `I`, which denotes the index where the operator is applied (see more information about indexes below).

```
abstract type OperatorTag end

struct Laplacian <: OperatorTag end
struct Divergence <: OperatorTag end
struct Gradient <: OperatorTag end
```

```
6
7  function derivative_order(tag::Laplacian) return 2 end
8  function derivative_order(tag::Divergence) return 1 end
9  function derivative_order(tag::Gradient) return 1 end
10
11 function laplacian(A, I, coeffs::NTuple{3})
12     Iz = CartesianIndex(0,0,1)
13     Iy = CartesianIndex(0,1,0)
14     Ix = CartesianIndex(1,0,0)
15
16     Cx, Cy, Cz = coeffs
17
18     return (
19         Cx[1] * A[I - Ix] + Cx[2] * A[I] + Cx[3] * A[I + Ix] +
20         Cy[1] * A[I - Iy] + Cy[2] * A[I] + Cy[3] * A[I + Iy] +
21         Cz[1] * A[I - Iz] + Cz[2] * A[I] + Cz[3] * A[I + Iz]
22     )
23 end
```

BoundaryCondition: Furthermore, to manage Boundary Conditions (BCs) more easily, abstract types such as `BoundaryCondition`, `AxisBC`, and `Axis`, as well as specific BC types like `DirichletBC`, `PeriodicBC`, etc., are defined. BCs are handled separately for each axis using these structs (e.g.,`DirichletAxisBCA`) and can be applied in a multi-dimensional context using `BoxBC`, which is the container for each BC at each side of the selected axis, where `A` is used to parameterize the axis. For brevity, we only show the implementation, struct and construct, for Dirichlet BCs (`DirichletBC`), which stores the values `c_l` and `c_h` as the values at the boundaries. However, Neumann, Robin and Periodic BCs are already implemented and are very similar in spirit.

```
abstract type BoundaryCondition end
abstract type AxisBC <: BoundaryCondition end
abstract type Axis <: BoundaryCondition end

struct DirichletBC{A} <: AxisBC end
struct NeumannBC{A} <: AxisBC end
struct PeriodicBC{A} <: AxisBC end

abstract type Axis end
struct XAxis <: Axis end
struct YAxis <: Axis end
struct ZAxis <: Axis end

struct BoxBC{A} <: BoundaryCondition end
struct BoxBC{N, T <: NTuple{N, AxisBC}} <: BoundaryCondition
    bcs::T
end

struct DirichletBC{N, T <: Real} <: BoundaryCondition
    c_l::NTuple{N, T}
    c_h::NTuple{N, T}
end

function DirichletBC{N}(c::T) where {N, T}
    c_ = ntuple(i -> c, Val(N))
    return DirichletBC(c_, c_)
end
```

```
28
29 BoxBC(bcs::AxisBC...) = BoxBC(bcs)
```

With these definitions, we can integrate the BCs into our architecture. Functions like `build_caches`, `ghost_ranges`, and `boundary_ranges` manage the extension of data arrays to account for BCs.

These extensions are incorporated into an `extended` matrix, using the `build_caches` function, which is parameterized by the differential operator `Op`. This `extended` matrix simplifies handling BCs and includes the original grid, whose values are copied, and 'ghost' points. *Ghost grid points* are extra elements added around the boundary of the computational domain (they are not part of the original problem domain but are used to allocate the BCs). For example, a matrix of size `m, n`, where `m` and `n` correspond to the number of points on the discretized grid, is extended to `m + 2, n + 2` to include ghost cells along the edges. Below, we provide an snippet of the function for `DirichletAxisBCA`, where the BCs are constant (`c_l` and `c_h`) and applied at the "ghost" edges of the extended matrix by means of `CartesianIndices`. We use the `CartesianIndices` function multiple times, including in this example, to iterate over the indices of the array in a Cartesian coordinate system. Other BCs follow a similar approach.

```
1 function build_caches(op, bc::DirichletBC{2}, u0)
2     m, n = size(u0) .+ 2
3     extended = zeros(m, n)
4     result = copy(extended)
5
6     CI_l1 = CartesianIndices((2:m-1, 1:1))
7     extended[CI_l1] .= bc.c_l[1]
8
9     CI_h1 = CartesianIndices((2:m-1, n:n))
```

```
    extended[CI_h1] .= bc.c_h[1]

    CI_l2 = CartesianIndices((1:1, 2:n-1))
    extended[CI_l2] .= bc.c_l[2]

    CI_h2 = CartesianIndices((m:m, 2:n-1))
    extended[CI_h2] .= bc.c_h[2]

    return extended, result
end
```

Next, the `boundary_ranges` function determines the indices for the boundary points in the original grid domain. It provides indices for both the low and high (`lo` and `hi`) boundary points (i.e., both edges of the axis) based on the size `sz` or dimension of the matrix. These indices are important for updating BCs in the ghost elements while the simulation runs. Here, we only show examples for the X-axis in 2D and 3D.

```
function boundary_ranges(sz::NTuple{2}, ::DerivativeAxisBC{XAxis})
    m, n = sz
    lo = CartesianIndices((2:2, 1:n-2))
    hi = CartesianIndices((m-2-1:m-2-1, 1:n-2))
    return lo, hi
end

function boundary_ranges(sz::NTuple{3}, ::DerivativeAxisBC{XAxis})
    l, m, n = sz
    lo = CartesianIndices((2:2,1:m-2, 1:n-2))
    hi = CartesianIndices((l-2-1:l-2-1,1:m-2, 1:n-2))
    return lo, hi
```

```
end
```

Furthermore, the `ghost_ranges` function calculates the indices of the ghost points of the `extended` matrix in a grid of size `sz`. These indices are important for both performance and understanding of the code. They are needed because in FD methods, an extra auxiliary neighboring point is sometimes needed to account for the boundary. The function differs based on the axis type (`XAxis`, `YAxis`, `ZAxis`) and whether the grid is 2D or 3D. Here, we only show the example for the X-axis in both 2D and 3D:

```
function ghost_ranges(sz::NTuple{2}, ::Type{XAxis})
    m, n = sz
    return (1:1, 2:n-1), (m:m, 2:n-1)
end

function ghost_ranges(sz::NTuple{3}, ::Type{XAxis})
    l, m, n = sz
    return (1:1, 2:m-1, 2:n-1), (l:l, 2:m-1, 2:n-1)
end
```

In short, the `build_caches` function creates an `extended` matrix that includes ghost points for BCs. The `boundary_ranges` function provides the indices where the values of the BCs are located in the original grid, while the `ghost_ranges` function returns the indices where these BC values are copied into this `extended` matrix.

LinearAlgebra.mul! and **apply_BCs!:** Finally, we show how to apply the operators, like `Laplacian`, into the `extended` matrix, which could represent any of our hydrodynamic variables, achieving the final goal. First, the `apply_BCs!` function updates the ghost points of `extended` matrix (with the boundary values) as the simulation progresses. For simplicity, we only show the implementation for Periodic BCs below. Second, `LinearAlgebra.mul!` applies the operator using `Op` to call the desired operator. Specializing this "multiplication"

(`mul!`) function in Julia's `LinearAlgebra` library aligns with Julia's established practices for matrix and array operations. This function creates Cartesian indices for the interior points of the `extended` matrix, excluding boundaries since derivatives are not computed at these auxiliary ghost points. For each interior index, the function applies the operator to `A` and updates `B` with the result. The results of both `apply_BCs!` and `LinearAlgebra.mul!` are stored in new matrices, which we define as the output of application of the BCs and, more importantly, the operator, respectively.

```
function apply_BCs(op, u, bc::PeriodicAxisBC{A}, extended) where {A
   <: Axis}
    sz = size(extended)
    lo_g, hi_g = ghost_ranges(sz, A)
    CI_lb, CI_hb = boundary_ranges(sz, bc)
    CI_lg = CartesianIndices(lo_g)
    extended[CI_lg] .= u[CI_lb]
    CI_hg = CartesianIndices(hi_g)
    extended[CI_hg] .= u[CI_hb]
    return extended
end

function LinearAlgebra.mul!(B::AbstractArray,
   Op::CenteredDifference{N, T, Tag}, A::AbstractArray) where {N, T,
   Tag}
    op = concretize(Tag)
    sz = size(A) .- 1
    CI = CartesianIndices(UnitRange.(2, sz))
    coeffs = Op.coeffs
    for I in CI
        B[I] = op(A, I, coeffs)
```

```
    end
    return B
end
```

4.3 Time integration with DifferentialEquations.jl

As explained in subsection 2.2, we follow the method of lines, where the spatial derivatives are handled by FD and the operators presented in the previous section, reducing the problem to algebraic equations. For time integration, we use the **DifferentialEquations.jl** library [38], and its documentation[2]. Here, we summarize the key features relevant to our purposes:

DifferentialEquations.jl is a versatile Julia library designed for solving differential equations. First, we define the problem using the `ODEProblem` function. This involves specifying its fields: the differential equations, in our case equations (3.42) and (3.43), which can be solved together or separately and should return a `du`, representing the differential to solve $\partial_t \psi$, with ψ indicating a dynamical variable. We also specify the initial conditions, the time span from 0 to t_f, where t_f is the final time, or total number of time steps, of the simulation, along with other parameters such as constants in our equations, caches for efficiently storing information, the differential operators, the box of BCs, and any other required elements

Next, we solve the ODE with the `solve` function, which applies a numerical method to integrate the problem over time. There are many methods listed in the documentation; some popular ones include `Euler()` and `Tsit5()`, a higher-order Runge-Kutta method. Additionally, we specify the time step (`dt`), which controls the intervals at which the solution is computed, unless the method is time-adaptive (where `dt` changes with each iteration). More technical aspects include the `saveat` option, which determines the points in time at

2. `docs.sciml.ai/DiffEqDocs/stable/`

which the solution is saved, and `save_start`, ensuring that the solution at the initial time is saved. Finally, we set relative and absolute tolerances (`reltol`, `abstol`) to control the accuracy of the numerical solution.

Below is a snippet that solves the Cahn-Hilliard equation (3.42) without the LC contribution, incorporating the differential operators discussed in Section 4.2. The implementation is simple and clear. This code yields results depending on the initial conditions `phi_0` and serves as the foundation for the more complex equations developed in this thesis.

```
params = (Laplacian, bc, E_phi, P_phi, E_mu, P_mu);

function cahn_hilliard(du, u, p, t)
    Laplacian, bc, E_phi, P_phi, E_mu, P_mu = p
    RT = R * T
    phi = @view u[:, :]
    dphi = @view du[:, :]

    # Extended matrix with ghosts at the edges
    E_phi = LiquidCrystals.apply_BCs(Laplacian, phi, bc, E_phi)
    # Apply Laplacian Operator on phi
    mul!(P_phi, Laplacian, E_phi)

    # Compute chemical potential
    Lap_phi = @view(P_phi[2:end-1, 2:end-1])
    mu = @. -K * Lap_phi + chi * (1 - 2 * phi) + RT * log(phi / (1 -
       phi))

    E_mu = LiquidCrystals.apply_BCs(Laplacian, mu, bc, E_mu)
    # Apply Laplacian Operator on mu
    mul!(P_mu, Laplacian, E_mu)
```

```
21
22      @. dphi = D * @view(P_mu[2:end-1, 2:end-1])
23
24      return du
25  end
26
27  prob = ODEProblem(cahn_hilliard, phi_0, (0.0, t_f), params);
28  sol = solve(prob, Tsit5(), progress = true, saveat = t_f / Nt,
        save_start = true)
```

4.4 Incoporating the Trixi.jl framework

Sections 4.2, and 4.3 discuss how to solve equations (3.42) and (3.43). However, when incorporating equations (3.33) and (3.34), which require using the LBM (see Section 2.2), we use another library known as `Trixi.jl` [45]. Here, we summarize its key features relevant to our purposes.

`Trixi.jl` is a numerical simulation framework based on Galerkin methods for hyperbolic conservation laws. It combines an extensible design with fast implementation and emphasizes ease of use. To integrate this framework with our FD scheme, we follow its specific guidelines[3] for customized use.

First, `Trixi.jl` already includes a Galerkin LBM, among many other numerical methods, and therefore, we need a way to transfer our differential operators to this framework. We couple our upwind grid with Trixi's cells, as described in Section 2.2, since Trixi uses cells rather than grid points with repeated solutions. We do not delve into the details here but instead present the function `DFtoTrixi!` used to map the computational meshes and explain its fields, utilizing most of the existing Trixi implementation to adapt our upwind scheme.

3. `docs.sciml.ai/Trixi/stable/tutorials/custom_semidiscretization/`

In the code excerpt below, `u` is the output Trixi array with data from the FD representation; `func_mesh` represents the input array containing the FD data to be transferred to the Trixi mesh; `mesh::Trixi.AbstractMesh2` denotes the Trixi mesh where the FD data integrates, with `2` indicating a two-dimensional mesh. The argument `equations` contains the parameters needed for the LBM, which assists in mapping the FD data to the correct locations in the Trixi mesh. The variables `solver::DG`, `cache`, and `DG` are specific to Trixi and are not detailed here. `Cart_Trixi_mesh` represents the Cartesian coordinates or indices used to map between the FD data and the Trixi mesh. The last argument, `nvariables`, specifies the number of variables in the FD data, which for LBM in 2D, are typically 9 distribution functions f.

```
function DFtoTrixi!(u, func_mesh, mesh::Trixi.AbstractMesh{2},
   equations, solver::DG, cache, Cart_Trixi_mesh, nvariables)

    for element in eachelement(solver, cache)
    for j in eachnode(solver), i in eachnode(solver)
    CI_node = GetIndexFromTrixi(Cart_Trixi_mesh, equations, solver,
       i, j, element)
    u_node = func_mesh[CI_node]

    set_node_vars_FD!(u, u_node, nvariables, solver, i, j, element)
    end

    end

end
```

Next, we need the Trixi LBM to read our stress tensors, such as those in equation (3.18), so we can compute the forces in LBM using equation (2.16) along with our FD scheme. To achieve this, we discuss a key implementation in Trixi: the `semidiscretization`.

Semidiscretizations in `Trixi.jl` are designed to create `ODEProblems` compatible with `DifferentialEquations.jl`, introduced earlier, for ordinary differential equations (ODEs). Specifically, a spatial semidiscretization—a specific Trixi struct that contains all necessary components to describe the spatial discretization of a hyperbolic conservation law (such as the Lattice Boltzmann equation in our case)—can be wrapped into an `ODEProblem` using `SemidiscretizedHyperbolic`. This `ODEProblem` includes the initial condition, a right-hand side (RHS) function representing the time evolution of $\partial_t \psi$, the time span, and any additional parameters. For a `SemidiscretizedHyperbolic`, the `ODEProblem` utilizes a `Trixi.rhs!` function as the ODE RHS, here the RHS of Eq. (2.20). We can modify `rhs!` by adding our own specifications (additional stress tensors) and creating a custom `rhs!` function.

Thus, we present our own semidiscretization, including the parameters for our differential operators in the field `t` an the original `Trixi SemidiscretizedHyperbolic` in the field `semi`:

```
struct CustomSemidiscretization{Semi, T} <:
   Trixi.AbstractSemidiscretization
semi::Semi
t::T
end
```

We next introduce the simplest working example of a custom ODE RHS, which is the basis for more complex calculations:

```
function my_rhs_withtensor!(du_coupled, u_coupled, semi_custom, t)
    # extract parts for Trixi.jl
    du_trixi = du_coupled.x[1]
    u_trixi = u_coupled.x[1]

    phiQ = @view u_coupled.x[2][:,:]
    dphiQ = @view du_coupled.x[2][:,:]

```

```
    semi = semi_custom.semi

    # extract parts for Finite_Differences
    grad, E_gradQ, P_gradQ, Laplacian, bc_box, E_LaplacianphiQ,
       P_LaplacianphiQ, div, equations, mesh, solver, node_CI, tlbm,
       ode_lbm, E_divPi, P_divPi, flat_Fx, flat_Fy, nnodes, P_Fx,
       P_Fy, v_cache = semi_custom.t.t

    #Apply BCs
    E_LaplacianphiQ = LiquidCrystals.apply_BCs(Laplacian, phiQ,
       bc_box, E_LaplacianphiQ)
    # Apply Laplacian Operator on phi and Q
    mul!(P_LaplacianphiQ, Laplacian, E_LaplacianphiQ)
    LaplacianQ = @view recover_Q(P_LaplacianphiQ)[2:end-1, 2:end-1]

    functional_Q = - kappa .* LaplacianQ
    Pi = QFunctional_Stress(phiQ, functional_Q, xhi) .- alpha .*
       recover_Q(phiQ) .+ FrankOseen_Stress(phiQ, kappa, grad,
       E_gradQ, P_gradQ)

    # divergence of stress tensor = forces
    E_divPi = LiquidCrystals.apply_BCs(div, Pi, bc_box, E_divPi)
    mul!(P_divPi, div, E_divPi)
    flat_Fx, flat_Fy = flat_forces(flat_Fx, flat_Fy, -1 *
       P_divPi[2:end-1, 2:end-1])

    F_DF =
    toSvecN.(getindex.((arrange_cells(reinterpret(SVector{n_variable,
```

```
        Float64}, reshape(u_trixi, n_variable, :)), (nnodes,
        n_element, Trixi.nvariables(semi_custom.t.semi.equations)),
        mesh)s,), TrixtoDF), semi_custom.t.semi.equations)
30
31      rho_v = reinterpret(Tuple{Float64, SVector{n_dims, Float64},
        Float64}, macroscopic_variables.(F_DF,
        semi_custom.t.semi.equations, flat_Fx, flat_Fy, dt))
32      rho = recover_rho(rho_v)
33      v = recover_V(rho_v)
34      v_cache .= v
35
36      nvariables_force = 1 # to convert force to FD
37
38      DFtoTrixi!(P_Fx, flat_Fx, mesh, equations, solver, semi.cache,
        node_CI, nvariables_force)
39      DFtoTrixi!(P_Fy, flat_Fy, mesh, equations, solver, semi.cache,
        node_CI, nvariables_force)
40
41      semi_custom.t.semi.equations.Fx .= P_Fx
42      semi_custom.t.semi.equations.Fy .= P_Fy
43
44      Trixi.rhs!(du_trixi, u_trixi, semi_custom.t.semi, t)
45      return nothing
46  end
```

The explanation of the previous snippet is as follows: Lines 2–7 unwrap our variables, packed previously, using the extra fields `x[1]` and `x[2]` for Trixi and FD, respectively. Lines 9–12 unpack the standard Trixi semidiscretization from our customized one, as well as the

parameters for our upwind scheme. This first part of the snippet looks very similar to that shown in Section 4.3, when extracting variables and differential operators. Next, lines 15–18 calculate derivatives needed for the stress tensor, named `Pi`. For simplicity, we only show the calculation of the stress from the FO elastic energy contributions and an additional activity parameter proportional to $\boldsymbol{Q}$, in lines 20–21, but the rest of the stresses are already implemented. The functions `recover_Q`, `recover_rho`, and `recover_v` are used to separate $\boldsymbol{Q}$, ρ and $\boldsymbol{v}$ from the array of collective variables, and we hope that any other function or variable is clear based on its name. Lines 24–26 then apply differential operators again to compute `P_divPi`, indicating the divergence (`div`) of a stress tensor, which provides the force densities needed in the LBM. These forces are separated into `flat_Fx` and `flat_Fy` as their x and y components, respectively. In this snippet, probably, the most obscure part is line 28, `F_DF`, and its related functions. We note that these are intermediate steps required to compute ρ and $\boldsymbol{v}$ using equations (2.18) and (2.19) in lines 30–33. The computed forces are stored in our custom semidiscretization (lines 37–41) for the next iteration, using our mapping function `DFtoTrixi!` to transfer data from the FD scheme to the Trixi mesh. We conclude the code with a new `Trixi.rhs!` definition, tailored for our LBM implementation with additional force densities, called with the parameters in the custom semidiscretization.

Some additional modifications to the functions found in the `Trixi.jl` source code[4] are further necessary to account for the forces; however, these modifications retain the original spirit, which is way we omit them here.

The code in `my_rhs_withtensor!` might seem lengthy, but we aim for it to be user-friendly and easy to follow, ensuring that intricate stress tensor calculations are straightforward to perform, which is a significant advantage of our implementation.

Finally, we set up an `ODEProblem`, as in Section 4.3, that yields results depending on the initial conditions `fphiQ_0`:

4. `github.com/trixi-framework/Trixi.jl/blob/7fd4503af9dfdf7022ae75f7398cac8fda811c85/src/equations/lattice_boltzmann_2d.jl`

```
semi_custom_tensor = CustomSemidiscretization(semi_Trixi,
   params_diff_op )
ode_semi_custom_tensor = ODEProblem(my_rhs_withtensor!, fphiQ_0 ,
   (0.0, dt_LBM), semi_custom_tensor)

sol_LbM_tensor = solve(ode_semi_custom_tensor,
   CarpenterKennedy2N54(williamson_condition=false), dt=dt_LBM,
   save_everystep=false, adaptive=false, callback= callbacks_custom);
```

where, `semi_trix` corresponds to the usual Trixi semidiscretization, `params_diff_op` refers to the parameters introduced for our upwind difference scheme, `dt_LBM` is the time spacing Δt for the LBM, and `CarpenterKennedy2N54` is the numerical method for time integration.

CHAPTER 5

SYSTEMS AT EQUILIBRIUM: TOPOLOGICAL DEFECT CORES

In this chapter, we present the numerical results obtained from solving our equations in the absence of flow. We compare these results with experimental data to validate their predictability. To this end, we first discuss the equilibrium solutions of our system of differential equations to ensure a stable steady-state.

5.1 Equilibrium Points of the Generalized Cahn-Hilliard and Ginzburg-Landau Equations

We begin our analysis by solving equations (3.42) and (3.43) in the absence of flow, which is the simplest way to describe a lyotropic liquid crystal at equilibrium. This approach is well-justified by the GENERIC formalism and implies that our variables $\boldsymbol{x}$ are reduced to ϕ and $\boldsymbol{Q}$, with all terms in the Poisson operator L being zero due to the lack of convective mechanisms in the system. Thus, the equations of motion simplify to

$$\frac{\partial \phi}{\partial t} = D_\rho \boldsymbol{\nabla}^2 \left(\frac{1}{T} \frac{\delta E}{\delta \phi} - \frac{\delta S}{\delta \phi} \right), \tag{5.1}$$

a Cahn-Hilliard type, or diffusion, equation. Here, we have assumed the parameter D_ρ to be constant and absorbed the total (constant) density dependence, with units of [length^5/(entropy$\cdot$ time)]. Additionally,

$$\frac{\partial \boldsymbol{Q}}{\partial t} = -R \left(\frac{1}{T} \frac{\delta E}{\delta \boldsymbol{Q}} - \frac{\delta S}{\delta \boldsymbol{Q}} \right), \tag{5.2}$$

represents a Ginzburg-Landau type equation, and, for simplicity, we take the relaxation parameter R to be constant.

To ensure that these equations have stable steady-state solutions and are physically meaningful, we need to find their equilibrium points, e.g. where the time derivatives are zero

($\partial \boldsymbol{Q}/\partial t = \mathbf{0}$ and $\partial\phi/\partial t = 0$) [64]. This leads to the equilibrium conditions of the form:

$$\frac{1}{T}\frac{\delta E}{\delta \phi} - \frac{\delta S}{\delta \phi} = 0, \tag{5.3}$$

and

$$\frac{1}{T}\frac{\delta E}{\delta q} - \frac{\delta S}{\delta q} = 0. \tag{5.4}$$

Note that Eq. (5.4) is written in terms of the scalar order parameter q, rather than the tensor $\boldsymbol{Q}$, due to the expected correlation between the concentration ϕ and q. Specifically, we anticipate that when the concentration exceeds the critical point for phase transition (indicating a 'high' value), the order parameter should reflect a nematic phase, typically chosen to be when $q \geq 0.6$. Conversely, with a low concentration (or below the nematic critical point), the order parameter should be close to zero. More explicitly, these conditions, expressed in terms of our energy and entropy functionals (equations 3.1 and 3.11) read:

$$\chi(1-2\phi)+T\left[c_1 \ln \phi - c_2 \ln(1-\phi) + c_1 - c_2\right] - 2A_0 U \phi \left(\frac{1}{12}q^2 - \frac{1}{16}q^4\right) + \frac{1}{4}A_0 T q^2 = 0, \tag{5.5}$$

and

$$-\frac{1}{2}\rho A_0 \phi q^2 \left[\frac{\phi U}{3} - \frac{\phi U q^2}{2} - T\right] = 0, \tag{5.6}$$

where we assume that, at equilibrium, the system is homogeneous. This implies that the spatial derivatives vanish at the physical boundary condition as $\boldsymbol{r} \to \infty$: $\partial \boldsymbol{Q}/\partial \boldsymbol{r} = \mathbf{0}$ and $\partial\phi/\partial \boldsymbol{r} = 0$. Additionally, to align with the analysis in our previous chapter and the notation used in the literature [13, 22, 65], the LdG parameters are defined as:

$$A(\phi) = A_0 \phi, \tag{5.7}$$

and

$$B(\phi) = A_0 U \phi^2, \tag{5.8}$$

where A_0 is a constant parameter controlling the strength of both the entropic and energetic LdG contributions and U is known as the nematic potential related to the anisotropic particle density [60].

The equilibrium relationships, (5.5) and (5.6), are plotted in the curves shown in Fig. 5.1. To manage the extensive set of degrees of freedom, we set the parameters to $T = 1$, $c_1 = c_2 = 1$, $\chi = 3$, and $A_0 = 1$, which are standard for dimensionless parameters. To ensure that both curves intersect around $\phi = 0.977$ and $q = 0.75$, we choose $U = 20$. Furthermore, we note that expressions (5.5) and (5.6) are valid only for a 2D system, as we have related the LdG terms to q using matrix multiplication relationships. In 2D, these relationships are $Q_{ij}Q_{ji} = \frac{q^2}{2}$ and $Q_{ij}Q_{jk}Q_{ki} = 0$, which differ from those in a 3D configuration.

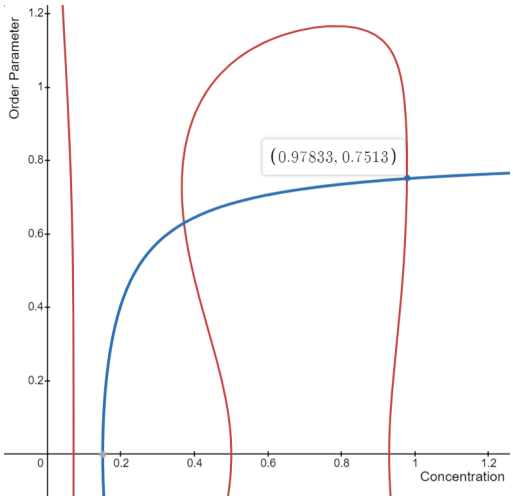


Figure 5.1: Plots of Eq. 5.5 (in red) and Eq. 5.6 (in blue) using parameters $T = 1$, $c_1 = c_2 = 1$, $\chi = 3$, $A_0 = 1$, and $U = 20$. x-axis denotes the concentration range, while y-axis indicates the order parameter. One of the intersection points is indicated with a blue dot

We use this analysis to achieve a nematic phase with a LC concentration different from 1 in our simulations. This contrasts with the common belief in other works, where concentrations of 1 or other non-physical values are often assumed [22, 25]. This initial attempt, using a

concentration value around 0.976, serves as our starting point. Adjusting the equilibrium curve to achieve a lower (or higher) concentration while maintaining a similar q can be achieved by modifying the parameters, specifically c_1 and c_2, which are related to the molecular weight of the species in the mixture. From this point forward, unless otherwise stated, these parameters are used when presenting simulation results.

5.2 Equilibrium Results in 2D: Comparison with Experimental Data

To demonstrate the predictability of our equations, numerical methods, and analysis of equilibrium points, we compare our simulation results with the following experimental data:

Recently, Zhou et al. characterized a binary mixture of lyotropic chromonic liquid crystals (LCLC). The LCLC consists of disodium cromoglycate (DSCG), the liquid crystal, dissolved in water at a weight concentration of 15%, corresponding to the nematic phase at room temperature [66]. The DSCG mixture was confined between two parallel glass plates, imposing a degenerate tangential orientation. The plates were then analyzed using a polarizing microscope, generating birefringence maps (Fig. 5.2). In these maps, the region depleted by DSCG forms two $\pm 1/2$ tactoid defects. Both defects develop sharp peaks, one peak in the case of the $+1/2$ defect and three peaks for the -1/2 defect. The colors represent the optical retardance, which is proposed to linearly correlate to the order parameter [67].

Next, we present the computational parameters used to solve equations (5.1) and (5.2) in 2D, setting $d = 2$, and simulate Fig. 5.2. As previously mentioned, we use the equilibrium physical parameters described in Section 5.1, and we choose the spacing step of our upwind scheme to be $\Delta x = \Delta y = 1.0$.

In selecting the parameters related to spatial gradient terms (diffuse interfaces, FO, and anchoring), we note that these terms are governed, and stabilized, by the order of magnitude of their respective accompanying parameters (such as Δx and their constants). As a first

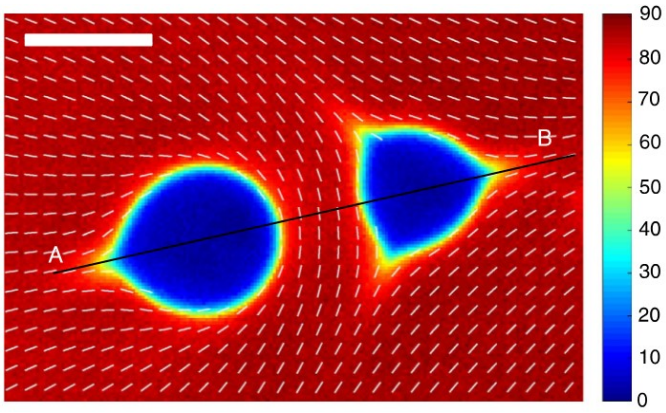


Figure 5.2: Birefringence map of 15% DSCG mixture in a thin cell with tangential anchoring condition. Scale bar is 20 μm.. The white lines represent the director vector while the colors indicate the optical retardance. Image adapted from [66].

approximation, it has been suggested [68] that $\kappa_E \approx \Delta x$, and we maintain this guideline across other parameters. Consequently, after tuning, we settle on $\kappa_E = 1.0$, $K = 1.5$ (if constant), and $\kappa' = 1.5$, which, as a reminder, also enforces planar, or tangential, anchoring in the director field around the interface. For the sake of simplicity, we disregard the entropic effect of diffuse interfaces, setting $\kappa_E = 0$.

The diffusion and relaxation parameters are set to $R = 0.1$ and $D_\rho = 0.0075$, consistent with the order of magnitude from previous studies, but they can be tuned as needed. In particular D_ρ is often changed to keep ϕ in its physical range. Similar to the equilibrium parameters, these are used henceforth unless otherwise specified when presenting simulation results.

Further details in solving our differential equations include the BCs and initial conditions (ICs). The BCs, which are considered periodic, are handled by our `LiquidCrystalGLBT.jl` package (see Section 4.2). In particular, Fig. 5.3a shows the ICs: on the left, we depict both the director field and q fields, while on the right, we present the concentration profile. For the IC of the director field, we propose a profile not far from the equilibrium condition, such that the orientation of $\boldsymbol{n}$ triggers the appearance of topological cores, or follows a similar path as observed in the experiment. The IC for q is however homogeneous ($q \approx 0.75$) at the

equilibrium condition shown in Fig. 5.1. The IC for the concentration profile consists of two rounded droplets with low concentration ($\phi \approx 0.05$), corresponding to a low order parameter, immersed in a higher concentration environment near the equilibrium condition ($\phi \approx 0.97$). In other words, these are two water droplets surrounded by a nematic phase.

The final aspects to discuss are the time step Δt, chosen to be 1.0, and the size of our simulation box $(N_x, N_y) = (200, 200)$. We use the `CarpenterKennedy2N54` time integrator from `DifferentialEquations.jl` and evolve the system over 120000 time steps.

As time progresses, Fig. 5.3 shows the evolution of the three fields ($\boldsymbol{n}$, q and ϕ) at different snapshots. We observe that the $\boldsymbol{n}$ field becomes smoother and initiates the nucleation of $\pm 1/2$ topological cores in the q field; with q nearly zero inside the defects and approaching the expected equilibrium value outside them. Notably, the originally rounded concentration profile changes such that ϕ matches the q profile, causing the droplets to adopt the shape of the tactoid cores, while the concentration profile outside the droplets remains close to the equilibrium value. This deformation appears naturally as a result of using the equilibrium parameters presented in Sec. 5.1. Otherwise, the droplets disappear. Additionally, we observe the presence of an interfacial region around the simulated tactoids, which resembles the experimental observations.

For the simulation presented in this chapter, we assume K to be constant. Although not presented here, we performed some tests to explore the implications of considering K as a function of ϕ, which could lead to more realistic scenarios [69]. Overall, if K is modified to be a sigmoidal function dependent on ϕ, mimicking the behaviour of the diffuse interface, then the droplets become more swollen. During our parameter tuning process, we generally observe that κ_E governs the size of the interface in the ϕ field, as expected, while (constant) K similarly affects the interface thickness of the order parameter q (or the isotropic-nematic interface). Additionally, κ' adjusts the deformation level of the topological peaks and influences how quickly ϕ and q become correlated. However, further research is

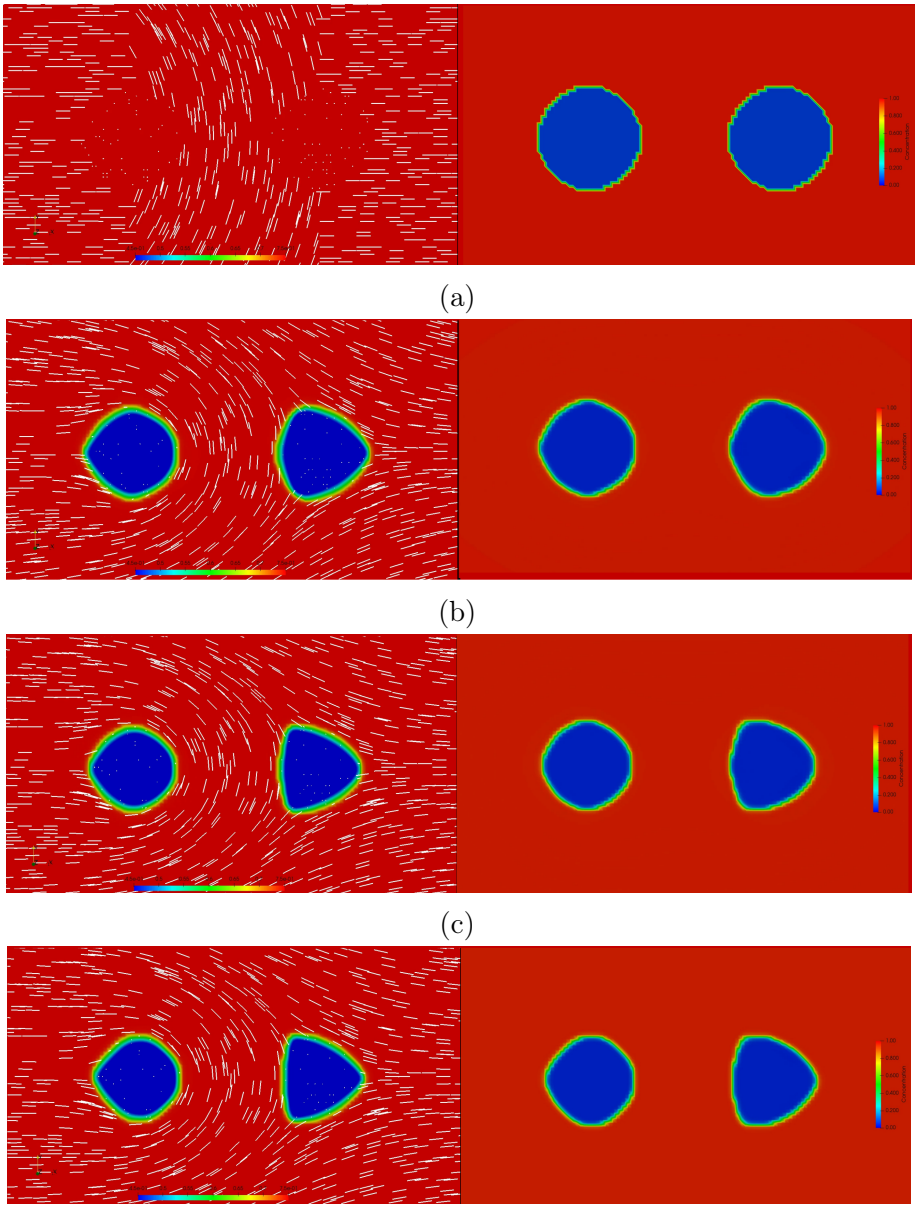


Figure 5.3: Simulation of two water droplets immersed in a nematic environment. On the left, the director vectors (white lines, scaled according to the values of the concentration profile) and the scalar order parameter (color map ranging from 0 to 0.75) are shown. On the right, the concentration profile is indicated by a color map ranging from 0 to 1. Subfigure (a) presents the ICs, while (b)-(d) depict the successive time evolution, with (d) showing the final time step.

recommended to thoroughly investigate the impact of these parameters, both analytically and through simulations.

In summary, these findings demonstrate that our equations and proposed methodology can qualitatively reproduce experimental data. Our simulations are stable and importantly maintain the physical range of ϕ and q between 0 and 1, with the director vector remaining unitary. Looking ahead, it is possible to achieve more quantitative predictions by correlating our selected computational parameters with physical values through dimensionless analysis. This approach could help identify the set of parameters that can adjust the equilibrium concentration to 0.15, the nematic concentration for DSCG. Further validation of the computed free energies with experimental data remains an avenue for future exploration.

CHAPTER 6

CONTROLLED FLOWING SYSTEMS: AN AXIAL DROPLET UNDER TWO VELOCITY PROFILES

In this chapter, we introduce hydrodynamics to our equations, resulting in a convection-relaxation-diffusion system. To achieve this, we first discuss the numerical constraints of our upwind scheme and explain how we address these challenges. Second, we compare our simulation results with experimental data, as in the previous chapter. While the experimental observations do not precisely mirror our simulations, we assert that the underlying physics are similar enough to support the predictive capability of our implementation.

6.1 The Convection-Diffusion-Relaxation System

When hydrodynamics is incorporated, equations (3.42) and (3.43) can be expressed as:

$$\frac{\partial \phi}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \phi = D_\rho \boldsymbol{\nabla}^2 \left(\frac{1}{T} \frac{\delta E}{\delta \phi} - \frac{\delta S}{\delta \phi} \right), \tag{6.1}$$

and

$$\frac{\partial \boldsymbol{Q}}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \boldsymbol{Q} = -R \left(\frac{1}{T} \frac{\delta E}{\delta \boldsymbol{Q}} - \frac{\delta S}{\delta \boldsymbol{Q}} \right), \tag{6.2}$$

which represent a Convection-Diffusion-Relaxation system and we incorporate the assumptions from the previous chapter. Additionally, we neglect the terms dependent on velocity gradients ($\boldsymbol{W}$) in equation (3.43) because this chapter examines two velocity profiles: one with constant velocities, where the velocity gradient vanishes, and another with a parabolic profile. For the parabolic profile, we hypothesize that contributions from $\boldsymbol{W}$ are minimal, primarily due to the smoothness of the chosen profile, compared to the dominant convection mechanism in one direction. Furthermore, these velocity gradients typically appear when the variable of interest is tensorial in nature [70], such as $\boldsymbol{Q}$. In contrast, one of the most significant variables in this

chapter is q, a scalar that does not require these velocity gradients for transport. Thus, we allow the director vectors $\boldsymbol{n}$ to rearrange during the relaxation process. In the next chapter, nevertheless, we include all previously neglected terms in this analysis.

Furthermore, if the diffusion or relaxation mechanisms are absent ($D_\rho = 0$ and $R = 0$, or if their associated $\boldsymbol{M}^{diff}$ and $\boldsymbol{M}^{relax}$ matrices are not included), equations (6.1) and (6.2) simplify to two advection equations. The literature provides numerous methods to solve advection equations [71], often involving sophisticated numerical techniques such as higher-order differentiation and averaging methods. Here, we opt for a simpler approach: using our first-order upwind scheme to compute the advection component and then solving the diffusion-relaxation part, as described in the previous chapter.

The primary aim of this chapter is to gain insights and test our upwind scheme. Therefore, we use two uniform and controlled velocity profiles as ansätze, which do not require solving the momentum balance due to their well-known and simple forms. We assume that the conditions necessary to achieve these flows are imposed externally, such as through external pressure gradients or moving walls.

As before, we use periodic boundary conditions for the spatial derivatives, meaning that the system of interest, which is to be convected, is placed in a channel with translational symmetry with a *fully developed* flow at the onset of the simulation. We presume that this periodicity has been incorporated into the initial ansatz, and thus, it does not affect our results. In simple terms, we aim to ensure that our system flows as intended without any numerical errors.

With the boundary conditions established, the focus shifts to another important consideration for upwind schemes: the Courant–Friedrichs–Lewy (CFL) condition [26, 71]

$$C_{FL} = \Delta t \left(\frac{\hat{v}_x}{\Delta x} + \frac{\hat{v}_y}{\Delta y} + \frac{\hat{v}_z}{\Delta z} \right) \leq C_{max}, \tag{6.3}$$

where C_{FL} is the dimensionless Courant number, $\hat{v}$ represents the absolute value of the

velocity component in the direction indicated by the subscript, and C_{max} is a parameter used to ensure stability in the simulation. For an explicit time-marching solver, such as the Euler scheme used in our upwind approach, $C_{max} = 1$. The other terms have their classical meanings.

Equation (6.3) is commonly used to determine Δt, as setting the velocity profile can be challenging, especially in turbulent systems. However, for a system with a given unidirectional velocity where the Euler method is applied for time integration, the CFL condition simplifies to

$$\hat{v}_x \leq \frac{\Delta x}{\Delta t}, \tag{6.4}$$

which ensures that the system avoids spurious or oscillating numerical instabilities. Despite this, some level of error is inevitable with this approach, as the scalar variables often reach higher (and nonphysical) values with each iteration [26]. From now on, we use equation (6.4) to guide our choice of parameters.

Lastly, to mitigate the impact of these inevitable errors, we use the diffusion-relaxation component of equations (6.1) and (6.2) to restore q and ϕ to their physical (equilibrium) values, as detailed below. This also adjusts the director vectors to their new configuration, imposed by the velocity profile.

We begin by performing one movement step (advection) according to the outlined specifications for our upwind scheme (the backward and forward derivatives). Subsequently, we integrate these transported values into the diffusion-relaxation process described in the previous chapter (or the second-order order central difference combined with higher methods for time integration), representing a single overall time step. We then alternate back to the advection phase, continuing this cycle until reaching the final time step. This setup introduces two distinct time scales: one for the advective process (Δt_{adv}) and another for diffusion-relaxation (Δt_{dr}). In the subsequent sections, we typically set the diffusion-relaxation time scale to be around 2 times longer than the advective time scale. We adjust this scale based

on the expected physics, speculating that it may relate to certain rheological properties of the material, such as the breaking point length or droplet elasticity.

6.2 A Liquid Crystal Droplet under (Semi-)Couette Flow

As a first working example, we examine the evolution of an axial LC droplet immersed in a richer isotropic environment under a semi-Couette flow in 2D. In other words, we observe how a nematic droplet in water responds to a (semi-)shear flow. This represents the simplest example we use for comparison with experiments.

We first sketch the velocity $\boldsymbol{v}$ profile for equations (6.1) and (6.2), as shown in Fig. 6.1a on the bottom right corner. We refer to this as a semi-Couette flow because, unlike a typical Couette flow, which varies linearly in the y-direction, our flow is constant and positive in the upper half of the simulation box and has an equal magnitude but opposite direction in the lower half. This ansatz profile remains 'active' until the droplet breaks, after which the velocity profile drops to zero.

Next, we specify $\Delta t_{adv} = 15$ as the time step for the advective part. As the velocity v is unidirectional, we can use equation (6.4) to determine the constant values of the velocity profile, $\hat{u}_x = 0.0663$, at both the top and bottom of the simulation box.

Furthermore, Fig. 6.1a illustrates the ICs for both ϕ and $\boldsymbol{Q}$, positioned at the top right corner and on the left, depicting an axial LC droplet. These ICs are generated using the methodology detailed in Chapter 5, starting from an initial random configuration of $\boldsymbol{n}$ and a nematic droplet immersed in an isotropic medium. The primary parameter difference is the introduction of perpendicular anchoring ($\kappa' = -0.85$), a crucial characteristic for axial droplets, while all other parameters remain unchanged from those specified in that chapter (including the spacing steps). This further underscores the robustness of our approach in replicating previous results.

The final aspects to discuss include the size of the box, set as $(N_x, N_y) = (300, 50)$ to

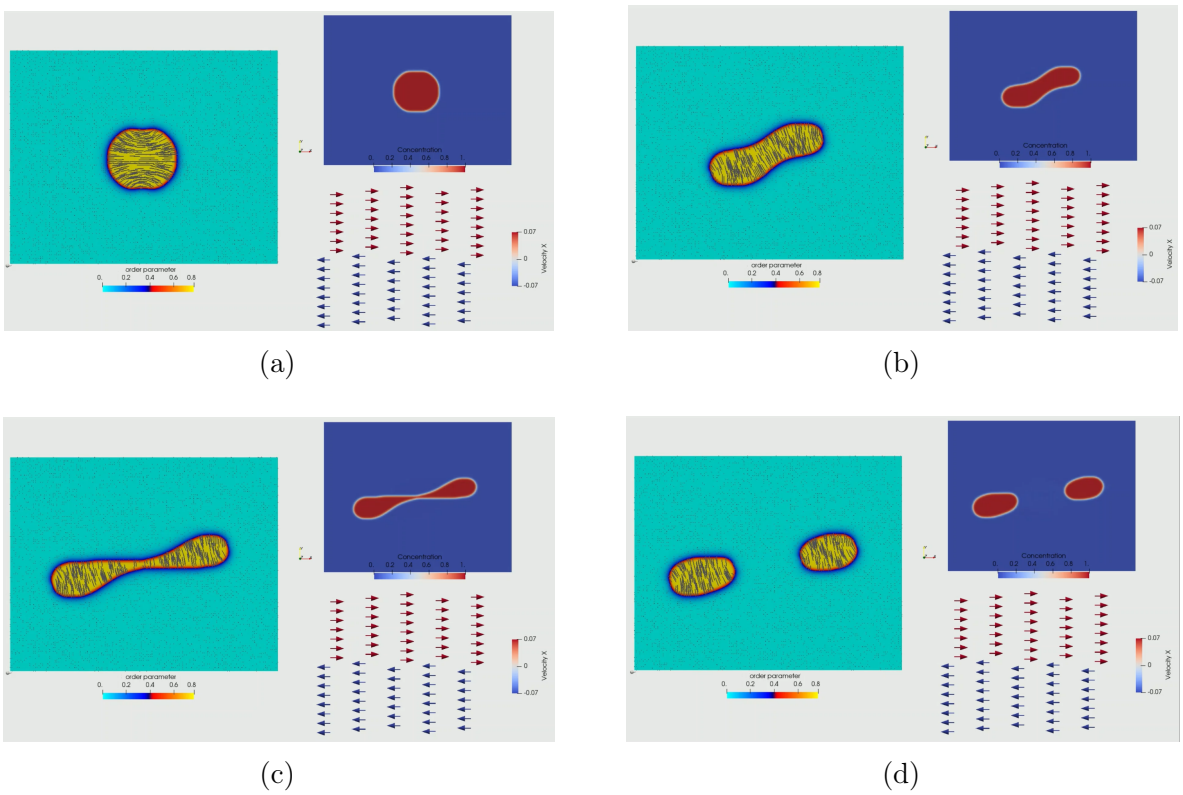


Figure 6.1: Simulation of an axial droplet immersed in an isotropic fluid. On the left, the director vectors (gray lines, scaled according to the order parameter values) and the scalar order parameter (color map ranging from 0 to 0.8) are shown. The top right shows the concentration profile, indicated by a color map ranging from 0 to 1, while the bottom right presents the velocity profile, colored by its x-component value (from -0.07 to 0.07). Subfigure (a) shows the initial conditions (ICs), while subfigures (b)-(d) illustrate the successive time evolution, with (d) representing the final time step.

mimic a flow channel, the total number of advective time steps for the system is 45, and the choice of diffusion-relaxation time steps within each advective time step is 400 (with $\Delta t_{dr} = 0.8$ for this purpose). With this information, we can proceed to run the simulation and present the following results.

As time progresses, Fig. 6.1 shows the evolution of the three fields ($\boldsymbol{n}$, q and ϕ) at different snapshots. We observe that the droplet begins to elongate. The motion continues until the breakup point, where a small neck forms and then two droplets emerge. These two droplets undergo their relaxation process once convection ceases. Now, we also have the

capability to track the evolution of the director vectors, which realign due to the droplet's new configuration, enabled by our stable simulations. To our knowledge, no analogous liquid crystal (LC) experiment exists. However, for completeness, Fig. 6.2 shows an emulsion (not liquid crystalline) under Couette flow, which resembles our simulation results.

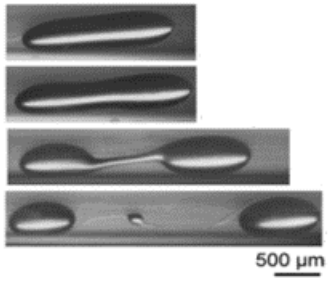
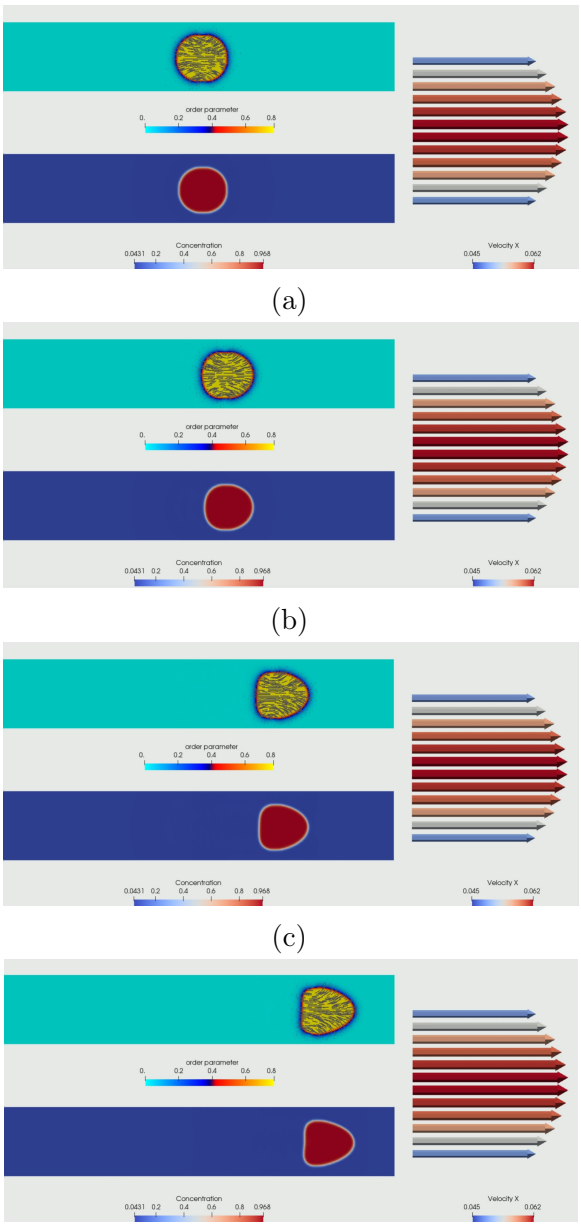


Figure 6.2: Time evolution of a Poly(isobutylene) (PIB, parabol 1300) droplet deformation and orientation under shear flow. Image adapted from [72].

6.3 A Liquid Crystal Droplet under a Poisieuille Flow

As a second example, we examine the evolution of the same axial LC droplet under a Poiseuille, or parabolic, flow in 2D, using the same initial conditions as in the previous Couette flow section. We also keep the parameters listed above, and the time integration remains the same. The difference lies in the velocity $\boldsymbol{v}$ used in this section.

We utilize a parabolic flow profile described by the Plane Poiseuille flow, with the following analytical form $v_x(y) = 4V_{\max}\left(\frac{y}{H}\left(1-\frac{y}{H}\right)\right)$, where $V_{\max}$ is the maximum velocity at the center of the channel and H is the full height of the channel. As we align the center of this auxiliary channel with the center of our simulation box, the flow peaks at $y = Nx/2$, exhibiting a very smooth but still preserving the underlying parabolic profile (see Fig. 6.3a). This subtle parabolic variation is consistent with the assumptions made at the beginning of this section. We set $V_{\max}$ to be 0.066, similar to the velocity used in the previous section. This ensures that the CFL condition, given in Eq. (6.4), is satisfied. Here, the simulation box size remains the same $(N_x, N_y) = (300, 50)$. The total number of advective time steps is



(a)

(b)

(c)

(d)

Figure 6.3: Simulation of an axial droplet immersed in an isotropic fluid. On the top left, the director vectors (gray lines, scaled according to the order parameter values) and the scalar order parameter (color map ranging from 0 to 0.8) are shown. The bottom left shows the concentration profile, indicated by a color map ranging from 0 to 1, while the right presents the velocity profile, colored by its magnitude (from 0.045 to 0.062). Subfigure (a) shows the initial conditions (ICs), while subfigures (b)-(d) illustrate the successive time evolution, with (d) representing the final time step.

99 (with $\Delta t_{adv} = 15$) while the total number of diffusion-relaxation time steps within each advective time step is shortened to 32 ($\Delta t_{dr} = 0.8$).

Fig. 6.3 shows the evolution of the three fields ($\boldsymbol{n}$, q and ϕ) at different snapshots. We observe that the droplet begins to deform into a triangular shape, as dictated by the velocity profile. The director vector realigns and curves around the vertices of the triangle while pointing perpendicular to the edges, as expected due to the imposed perpendicular anchoring. To our knowledge, there is no analogous LC experiment; however, Fig. 6.4 shows a nematic LC droplet induced by temperature gradients (making it a thermotropic LC rather than lyotropic) surrounded by another nematic phase, under a parabolic flow, which resembles our simulation results.

In summary, these findings demonstrate that our equations and proposed methodology can qualitatively resemble experimental data. We achieve this by incorporating the analysis of Chapter 5 and discussing how our upwind scheme ensures accurate flowing simulations. Our results are numerically stable and importantly maintain the physical range of ϕ and q between 0 and 1, with the director vector remaining unitary. Looking ahead, it is possible to achieve more quantitative predictions by incorporating the velocity gradient $\boldsymbol{W}$ for more complex flows, including turbulent ones, as discussed in the next chapter.



Figure 6.4: Still image of the deformation of a nematic 5CB droplet in a pressure-driven microfluidic flow control system under a polarized microscope. The size of the droplet is approximately 20μm. Image adapted from [73].

CHAPTER 7

NEMATIC TURBULENCE AND A MODEL FOR MULTI-COMPONENT LIQUID CRYSTALS

In this chapter, we present preliminary results demonstrating the successful coupling of our upwind scheme with `Trixi.jl`. To illustrate this, we provide examples for an active single-component LC and an active lyotropic LC.

As LBM already integrates NS with a Newtonian stress, the GENERIC $\boldsymbol{M}$ friction matrix corresponding to this mechanism is not included in Chapter 3. However, we recognize that this contribution should be added when solving the energy balance or equation (3.41), where the viscosity of the system plays a significant role. Adding this term nevertheless is straightforward (see [32]).

In the last section, we present a generalization of our model for multi-component LCs. This new research area hopes to integrate all prior knowledge to solve the proposed equations.

7.1 A single-component active Liquid Crystal

In this example, our simulation continues in 2D ($d = 2$), and therefore we use a D2Q9 LBM scheme [42], where Q is used to denote the number of allowed travel dimensions for f (see Section 2.2). This sets our w:

$$w = (1/9,\, 1/9,\, 1/9,\, 1/9,\, 1/36,\, 1/36,\, 1/36,\, 1/36,\, 4/9)\,, \tag{7.1}$$

and $\mathbf{v}^{\circ}$:

$$\mathbf{v}_x^{\circ} = (c_s,\, 0,\, -c_s,\, 0,\, c_s,\, -c_s,\, -c_s,\, c_s,\, 0), \tag{7.2}$$

$$\mathbf{v}_y^{\circ} = (0,\, c_s,\, 0,\, -c_s,\, c_s,\, c_s,\, -c_s,\, -c_s,\, 0). \tag{7.3}$$

and we $c_s = 1$ as is usual for simulations.

The purpose of this section is to demonstrate the successful coupling of our `LiquidCrystalGLBT.jl` library with `Trixi.jl`. To prove this, we simultaneously solve equations (3.34) and (3.43), which describes the LC ordering and the momentum balance. An instance of scientific relevance is simulating an active nematic by adjusting the activity (ζ) in the active stress tensor shown in equation (1.7). Despite its apparent simplicity, we have not yet justified this stress tensor within the GENERIC framework, as the activity coefficient is expected to impact the entropy balance in a very specific fashion [74]. Furthermore, if we assume that this activity coefficient arises from molecular motors that obeys Bell's dynamics [75], where these motors exponentially respond to internal medium stresses, constructing a GENERIC $\boldsymbol{M}$ matrix requires special attention. For the time being, we consider equation (1.7) to be a reasonable approximation.

The basic process consists of first running the Trixi Galerkin LBM, which generates a chaotic velocity profile due to the activity, and then using those velocities for the integration of the convection-relaxation step of $\boldsymbol{Q}$, as described in the previous chapter, to move the director vectors and the scalar order parameter. We base our Trixi step primarily on its working example[1], along with our customized semidiscretization described in Section 4.4, which incorporates the tensors shown in equations (3.18), (3.35), and (3.40).

As done in the previous chapter, where we divide our problem into two time scales, in this section we hypothesize the existence of a third time scale, given by the time integration of the LBM. We then define different time steps, Δt's for the advective, or convective, part (Δt_{adv}), the LBM (Δt_{LBM}), and the relaxation process (Δt_r). In addition, the LBM has a constraint analogous to the CFL condition. Therefore, choosing Δt_{LBM} must be done cautiously; a general rule of thumb is $\tau > \frac{\Delta t_{LBM}}{2}$ [42].

We perform only one iteration, or one global time step, in the LBM and observe that

1. `github.com/trixi-framework/Trixi.jl/blob/7fd4503af9dfdf7022ae75f7398cac8fda811c85/examples/tree_2d_dgsem/elixir_lbm_couette.jl`

the computed stresses (due to activity and other contributions) are sufficient to define the direction and magnitude of the flow for the evolution of $\boldsymbol{Q}$, without requiring additional internal steps. These extra time steps may correct the approximation given by Eq. (2.19). Of course, this may differ in scenarios that demand more numerical accuracy.

Furthermore, we repeat the computational and physical parameters from the previous chapter; however, the nematic potential $U = 2.7$ is adjusted to achieve an equilibrium order parameter of approximately 0.65. This example only requires using one equilibrium curve, rather than two, as there is no contribution from the LC concentration (see Section 5.1), and therefore $\phi = 1$.

In Fig. 7.1a, we show our initial conditions, with $\boldsymbol{v} = \boldsymbol{0}$ everywhere and the $\boldsymbol{Q}$ field is initialized with a preexisting configuration of two topological defects. We hypothesize that the bending director vectors around these defects can induce flow, as activity tends to create flows in regions with twisted directors [31]. We use $\zeta = -48$, after multiple tuning attempts, to induce turbulence while keeping the velocity magnitude below 1. We also set periodic boundary conditions in both fields. We choose $dt_{LBM} = 0.01$, $dt_{adv} = 0.1$, $dt_r = 0.006$, and $\tau = 0.02$, which is related to Newtonian viscosity. The number of Galerkin cells is $(149, 149)$ and the number of grid points is $N_x = N_y = 150$ with $\Delta x = \Delta y \approx 0.03$. We run the simulation for 600 LBM-advective iterations and approximately 100 steps in the relaxation phase (the evolution of $\boldsymbol{Q}$), within the LBM.

Unlike in the previous chapter, hereafter, we incorporate all the terms dependent on velocity gradients ($\boldsymbol{W}$), ignored before, specifically in the convection step of the evolution of $\boldsymbol{Q}$, as these terms produce turbulence. We also observe that these terms help maintain the q parameter within its physical range, making the CFL condition less critical. However, we still attempt to keep the velocity within the constraints imposed by the CFL condition by tuning ζ. We do not fully understand how the value of $\boldsymbol{v}$ or its asymptotic behavior changes with this activity, which represents another interesting area for investigation.

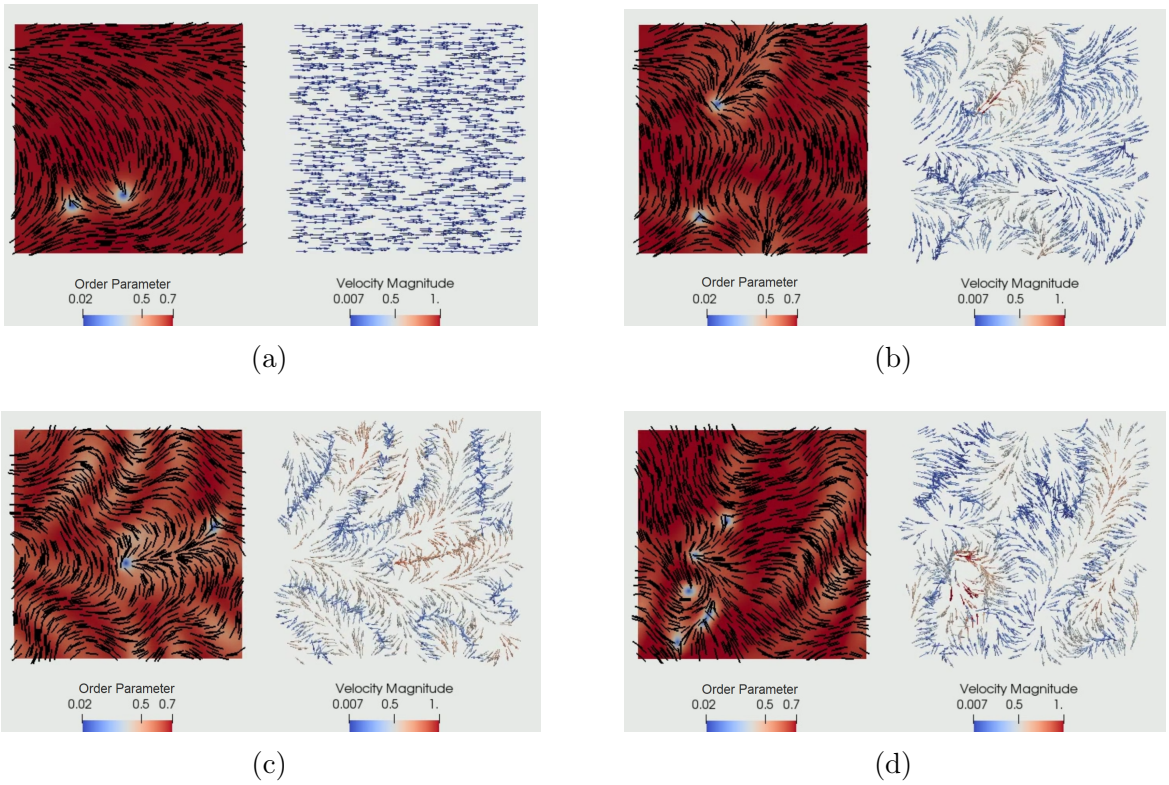


Figure 7.1: Simulation of a one-component active nematic. On the left, the director vectors (black lines) and the scalar order parameter (color map ranging from 0 to 0.7) are shown. On the right, the velocity profile (arrows) is depicted, colored according to its magnitude (ranging from 0 to 1). Subfigure (a) shows the initial conditions (ICs), while subfigures (b)-(d) illustrate the successive time evolution, with (d) representing the final time step. Notably, (c) shows the nucleation of defects, while (d) captures defect annihilation.

Finally, Fig. 7.1 shows the time evolution of our active nematics, resembling Fig. 7.2, an experimental image of an active LC, thereby confirming our implementation. In particular, Figures 7.1c and 7.1d capture defect nucleation and annihilation, respectively. As expected, one of these defects ($+1/2$) travels faster, which is evident from its associated velocity profile, while the $-1/2$ defect exhibits a significantly slower velocity magnitude. Additionally, the director vectors align perpendicular to the direction of the flow, which is consistent with the theory discussed in [31].

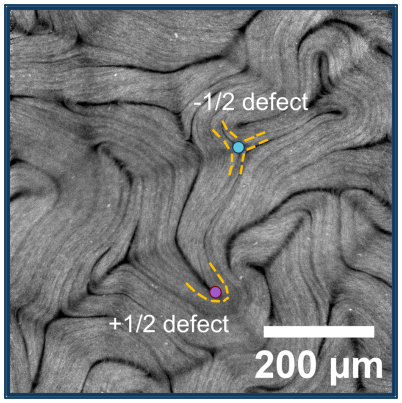


Figure 7.2: Still image of one-component active nematics, consisting of polymerized tubulin microtubules in contact with kinesin molecular motors, at a water-oil interface under a confocal microscope. A schematic representation of defects is shown. Courtesy of Dr. Antonio Tavera-Vázquez.

7.2 An active lyotropic Liquid Crystal

Once we have demonstrated that our implementation can reproduce turbulent results, we use the knowledge gathered in previous sections to simulate an active lyotropic LC, which is the most exciting part for the author of this thesis :). To that end, we follow the specifications, including parameters, from all previous sections and chapters, and solve Equations (3.33), (3.34), (3.42), and (3.43) simultaneously with Periodic BCs. The integration is performed by first solving the Trixi Galerkin LBM and then incorporating the resulting velocities into the convection-diffusion-relaxation step, without neglecting the terms dependent on velocity gradients (see Chapter 6).

In this section, we place special emphasis on tuning the ζ activity parameter, as it can erratically increase the velocity magnitude. If the CFL condition is not satisfied, it can significantly impact ϕ, causing it to become excessively high and pushing it out of its physical range. After tuning, we set $\zeta = -6.5$ and $dt_{LBM} = 0.01$, $dt_{adv} = 0.003$, $dt_{dr} = 0.05$. These are the only computational parameters that differ from those in the previous section. The other parameters are taken from Chapter 6.

Figures 7.3 show the ICs and snapshots at different times of our active concentration-

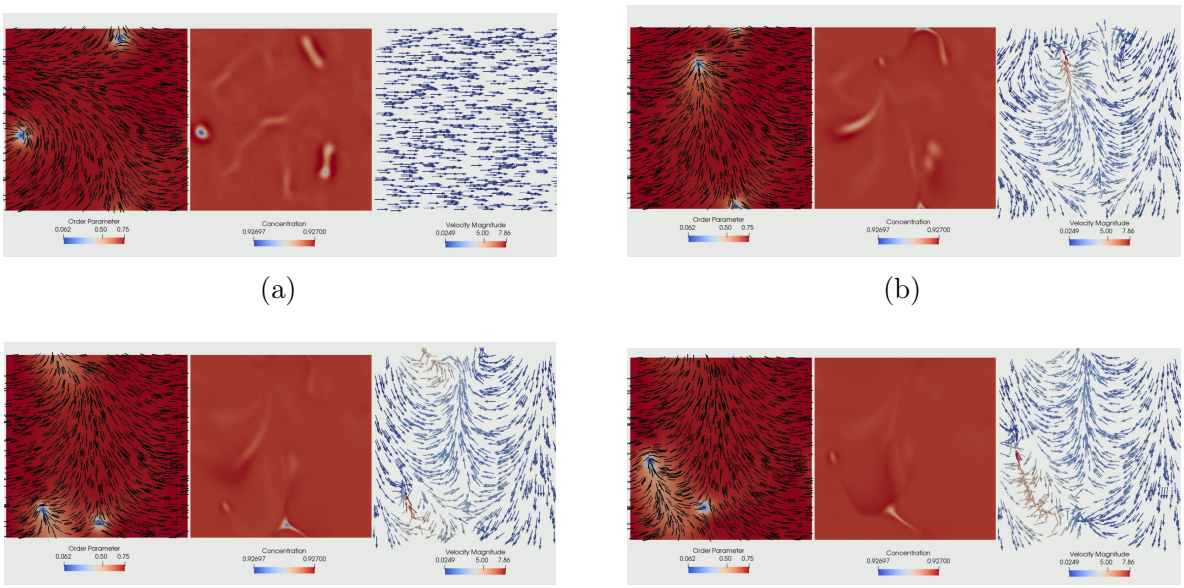
dependent liquid crystal. We continue the simulation until we encounter numerical instability. Although the simulation seem unstable, specifically the evolution of $\boldsymbol{Q}$, to the best of our knowledge, this is the first simulation ever of an active lyotropic LC. To enhance stability, we recommend adjusting the parameters so that the activity does not drive the velocity beyond the CFL condition, or addressing any other potential restrictions that we have not considered or that have been previously discussed. For example, adding a function to maintain $\boldsymbol{Q}$ as traceless. This traceless condition is sometimes not fully achieved with our numerical integrators but can be easily fixed with a `callback` function. Callbacks are a way to modify or influence the behavior of the numerical solver during its execution in `DifferentialEquations.jl` and are used to implement constraints, or modifications to the integration process based on specific criteria. The Trixi library heavily relies on these functions. For now, we are pleased to demonstrate that the four differential equations derived with GENERIC are all coupled through our `LiquidCrystalGLBT.jl` library.

Importantly, it is now possible to track the concentration profile as the defects move, which was not possible before. Although the variation is very small, this represents a significant contribution of this thesis. As observed, the concentration tends to be lower in defect regions, where there is less ordering and more isotropic liquid regions likely persist.

7.3 The model for Multi-component Liquid Crystals

Evidently, lyotropic LC experiments often involve more than two components, such as a secondary LC, oil, surfactant, ions, molecular motors [20, 76, 77], and a solvent. To address this, we present a multi-component theory in this section, assuming that the concentration ϕ can describe both an isotropic liquid and an LC. We derive the multi-component equations, inspired by the work presented throughout this thesis. This approach is justified by our experience and understanding of the GENERIC framework.

We repeat the process presented in Chapter 3 and hope that readers find the generalization



(a) (b)

(c) (d)

Figure 7.3: Simulation of an active lyotropic liquid crystal. On the left, the director vectors (black lines) and the scalar order parameter (color map ranging from 0 to 0.75) are shown. In the middle, we present the concentration profile, colored from 0.92697 to 0.927. On the right, the velocity profile (arrows) is depicted, colored according to its magnitude (ranging from 0 to 7.86). Subfigure (a) shows the initial conditions (ICs), while subfigures (b)-(d) illustrate the successive time evolution, with (d) representing the final time step.

straightforward. Here, we summarize the basic ingredients needed for the model, such as what is required for computing simulations, including the energy and entropy functionals and the time-evolution equations. We omit the introduction of the $\boldsymbol{L}$ and $\boldsymbol{M}$ matrices, as they are extended versions of multiple components introduced in that chapter.

The vector of variables $\boldsymbol{x}$ are the same as in Chapter 3, but we further label ϕ and $\boldsymbol{Q}$ with a subscript Λ to indicate the component number

$$\boldsymbol{x} = (\rho, \boldsymbol{m}, \epsilon, \phi_\Lambda, \boldsymbol{Q}_\Lambda). \tag{7.4}$$

The multi-component total energy reads

$$\begin{aligned} E(\boldsymbol{x}) &= \int \mathcal{E}(\boldsymbol{x})\, d^3\boldsymbol{r} = \int \Bigg\{ \frac{1}{2}\frac{{m_k}^2}{\rho} + \epsilon + \sum_{\Lambda=1}^{\Upsilon} \left[\frac{1}{2}\kappa_E \left(\frac{\partial \phi}{\partial r_k} \right)^2 \right]_\Lambda + \sum_{\Lambda=1}^{\Upsilon} (\epsilon_{\mathrm{LdG,E}})_\Lambda \\ &+ \sum_{\Lambda=1}^{\Upsilon} \left[\frac{1}{2} K(\phi) \frac{\partial Q_{ij}}{\partial r_k} \frac{\partial Q_{ij}}{\partial r_k} \right]_\Lambda + \sum_{\Lambda=1}^{\Upsilon} \left[\frac{1}{2}\kappa' \left(\frac{\partial \phi}{\partial r_i} \right) \left(Q_{ij} + \frac{I_{ij}}{d} \right) \left(\frac{\partial \phi}{\partial r_j} \right) \right]_\Lambda \\ &+ \sum_{\Lambda,\Theta=1}^{\Upsilon} W_{R,\Lambda\Theta} \Big(Q_{ij,\Lambda} - Q_{ij,\Theta} \Big)^2 + \sum_{\Lambda,\Theta=1}^{\Upsilon} \rho \chi_{\Lambda\Theta} \phi_\Lambda \phi_\Theta \Bigg\} d^3\boldsymbol{r}, \end{aligned} \tag{7.5}$$

where Υ represents the total number of components, and the energies associated with $\boldsymbol{Q}$ and ϕ are labeled according to their Λ-th component. Furthermore, the parameters χ and W_R, and their corresponding energies, are also parameterized by a second label Θ, which represents pairwise interactions with a second liquid. We also include an energetic contribution inspired by the Rapini-Papoular potential [78], the first term in the third line of Eq.(7.5) which functions as a harmonic term that penalizes differences between the orientations of different LCs in contact (λ and Θ), aiding their alignment. The parameter W_R serves to adjust this alignment strength. The other energetic contributions are described in Chapter 3.

Next, the multi-component entropy reads

$$\begin{aligned} S(x) &= \int \Big[s_0(\rho, \epsilon) + \mathcal{S}(x) \Big] d^3 r \\ &= \int \left[s_0(\rho, \epsilon) + \sum_{\Lambda=1}^{\Upsilon} \left[\frac{1}{2}\kappa_S \left(\frac{\partial \phi}{\partial r_k} \right)^2 \right]_\Lambda + \sum_{\Lambda=1}^{\Upsilon} c_\Lambda \phi_\Lambda \ln \phi_\Lambda + \sum_{\Lambda=1}^{\Upsilon} (\epsilon_{\mathrm{LdG,S}})_\Lambda \right] d^3 r. \end{aligned} \tag{7.6}$$

and the time-evolution equations are given by

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial r_k}(\rho\, v_k), \tag{7.7}$$

$$\frac{\partial m_i}{\partial t} = -\frac{\partial}{r_m} \left[v_m m_i + \pi^S_{mi} + \pi^E_{mi} - \sum_{\Lambda=1}^{\Upsilon} \pi^{slip}_{mi,\Lambda} \right], \tag{7.8}$$

$$
\begin{aligned}
\frac{\partial \epsilon}{\partial t} &= -\frac{\partial}{\partial r_k}\left(v_k\,\epsilon\right) - \pi^S_{km}\,W_{km} + \sum_{\Lambda=1}^{\Upsilon}\left\{\frac{\delta E}{\delta Q_{kl}} R_\Lambda\left(\frac{1}{T}\frac{\delta E}{\delta Q_{kl}} - \frac{\delta S}{\delta Q_{kl}}\right)\right. \\
&+ T\left(\lambda_\Lambda - 1\right)\left[2\left(Q_{ij} + \frac{I_{ij}}{d}\right)\frac{\delta S}{\delta Q_{ij}}\left(Q_{lk} + \frac{I_{lk}}{d}\right)W_{lk} - \left(Q_{ik} + \frac{I_{ik}}{d}\right)W_{kj}\,\frac{\delta S}{\delta Q_{ij}}\right. \\
&- \left.\left.\left(Q_{ik} + \frac{I_{ik}}{d}\right)W_{jk}\,\frac{\delta S}{\delta Q_{ij}}\right] - \frac{\delta E}{\delta \phi}\frac{1}{\rho}\frac{\partial}{\partial r_k}\rho D_{kl,\Lambda}\,\frac{\partial}{\partial r_l}\frac{1}{\rho}\left(\frac{1}{T}\frac{\delta E}{\delta \phi} - \frac{\delta S}{\delta \phi}\right)\right\}_\Lambda,
\end{aligned}
\tag{7.9}
$$

$$
\frac{\partial \phi_\Lambda}{\partial t} = -v_k\,\frac{\partial \phi_\Lambda}{\partial r_k} + \frac{1}{\rho}\frac{\partial}{\partial r_k}\rho D_{kl,\Lambda}\,\frac{\partial}{\partial r_l}\frac{1}{\rho}\left(\frac{1}{T}\frac{\delta E}{\delta \phi_\Lambda} - \frac{\delta S}{\delta \phi_\Lambda}\right), \tag{7.10}
$$

$$
\begin{aligned}
\frac{\partial Q_{ij,\Lambda}}{\partial t} &= -v_k\,\frac{\partial Q_{ij,\Lambda}}{\partial r_k} + \left(\lambda_\Lambda \Xi_{ik} + \Omega_{ik}\right)\left(Q_{kj,\Lambda} + \frac{I_{kj}}{d}\right) + \left(Q_{ik,\Lambda} + \frac{I_{ik}}{d}\right)\left(\lambda_\Lambda \Xi_{kj} - \Omega_{kj}\right) \\
&- 2\lambda_\Lambda\left(Q_{ij,\Lambda} + \frac{I_{ij}}{d}\right)\left(Q_{kl,\Lambda} + \frac{I_{kl}}{d}\right)W_{lk} - R_\Lambda\left(\frac{1}{T}\frac{\delta E}{\delta Q_{ij,\Lambda}} - \frac{\delta S}{\delta Q_{ij_\Lambda}}\right),
\end{aligned}
\tag{7.11}
$$

with

$$
\begin{aligned}
\boldsymbol{\pi}^E_{mi} &= -\left\{\left[\rho\frac{\partial}{\partial r_l}\left(\frac{\partial \mathcal{E}}{\partial(\partial_l \rho)}\right) + \epsilon\frac{\partial}{\partial r_l}\left(\frac{\partial \mathcal{E}}{\partial(\partial_l \epsilon)}\right) - \rho\frac{\partial \mathcal{E}}{\partial \rho} - \epsilon\frac{\partial \mathcal{E}}{\partial \epsilon} + \mathcal{E}\right] I_{mi}\right. \\
&- \left[\left(\frac{\partial \rho}{\partial r_i}\right)\left(\frac{\partial \mathcal{E}}{\partial(\partial_m \rho)}\right) + \left(\frac{\partial \epsilon}{\partial r_i}\right)\left(\frac{\partial \mathcal{E}}{\partial(\partial_m \epsilon)}\right) + \sum_{\Lambda=1}^{\Upsilon}\left(\frac{\partial \phi_\Lambda}{\partial r_i}\right)\left(\frac{\partial \mathcal{E}}{\partial(\partial_m \phi_\Lambda)}\right)\right] \\
&+ \sum_{\Lambda=1}^{\Upsilon}\left[\left(\frac{\partial Q_{kl}}{\partial r_i}\right)\left(\frac{\partial \mathcal{E}}{\partial(\partial_m Q_{kl})}\right)\right]_\Lambda - 2\sum_{\Lambda=1}^{\Upsilon}\left[\left(Q_{mi} + \frac{I_{mi}}{d}\right)\left(Q_{kl} + \frac{I_{kl}}{d}\right)\frac{\delta E}{\delta Q_{kl}}\right. \\
&- \left.\left.\left(Q_{ml} + \frac{I_{ml}}{d}\right)\frac{\delta E}{\delta Q_{li}}\right]_\Lambda\right\}.
\end{aligned}
\tag{7.12}
$$

For brevity, we only present the form of the energetic pressure contribution. The entropic pressure contribution is similar, but uses the entropy functional instead. It has the opposite sign and is proportional to the temperature. For reference, see equations (3.18) and (3.35). Note that the diffusion $\boldsymbol{D}$ and relaxation R parameters, as well as π^{slip}_{mi} and λ_Λ, are also labeled with the subscript Λ.

To conclude, it is important to note that the Newtonian stress tensor has not been included

in the energy balance yet. Additional physical effects can be incorporated by modifying the energy or entropy functionals, or by changing the $\boldsymbol{M}$ matrix. Importantly, this theoretical study, though brief, would not have been possible without the foundational analysis of the binary system. We acknowledge that the generalization to a multi-component system requires the constraint that the sum of all concentrations equals one, which has not yet been verified. This constraint could be enforced using Lagrange multipliers (an area for future exploration). However, we hope that these equations prove useful for modeling more complex scenarios. Our library and the analysis presented, along with the equations here, are intended to guide further research and facilitate new applications for LCs.

CHAPTER 8

CONCLUSIONS

As a general conclusion, we show how we systematically derive a comprehensive model for lyotropic LCs, that can be active, based on first principles of thermodynamics and their necessary physics. We then analyze, piece by piece, how these differential equations communicate, interact, and ultimately behave under specific conditions. We sincerely hope that this systematic methodology will advance the field and inspire new applications, especially since accurately simulating lyotropic LCs has been a longstanding challenge. Combining computational and experimental efforts can help drive applications—for instance, in identifying optimal conditions for artificial muscles [79], developing new sensing technologies [80, 81], directing bacterial movement [82], and improving drug delivery systems [83]. Furthermore, this study may extend beyond LCs to other areas that are written in the elegant language of differential equations

Despite these positive outcomes, many open questions remain, and we hope they will be addressed in the future, either by us or others. As mentioned in the thesis, some issues of concern include mapping our computational "physical" parameters to real physical quantities. For example, if we set $T = 1$, what are the actual units? Is it 1 K, or is there a different physical value? The same question applies to our other "dimensionless" computational parameters. Due to time constraints, we could not fully address this matching process, but we hope it will be achievable in the future. Another question is how to incorporate sharp versus diffuse interfaces. While sharp interfaces are generally challenging to implement computationally—or at least, that is our current understanding—there are various models for simulating interfaces. Identifying an interface model that is easy to understand, implement, and aligns well with our theoretical foundations is another area of research. Additionally, are the interfaces observed in experiments truly diffuse or sharp? The images suggest diffuse interfaces, but this seems to contradict some thermodynamic principles.

Furthermore, another key issue to explore is the role of activity. While using a parameter proportional to $\boldsymbol{Q}$ to drive the system out of equilibrium is appealing due to its simplicity, is this relationship sufficient, or is it merely a shortcut to produce simulations that resemble experimental results? We recognize that there is a physical interpretation behind it, and some mechanical proof to support it, but we believe that thermodynamics, and experimental, research are needed.

Moreover, and very importantly, what is the relationship between this activity and the velocity profile? Can we develop a more systematic method for controlling the velocity magnitude? This may be possible, but it might require someone with a deep passion for mathematics to delve into the problem.

Regarding the computational aspect, we hope that no errors have occurred in our code. Given the complexity and length of the equations, it is easy to make mistakes. However, the positive results we have achieved suggest that we have succeeded to the best of our capabilities. Although we managed to connect `Trixi.jl` with our `LiquidCrystalGLBT.jl` package, we are uncertain whether the time scales match our conceptual expectations. This uncertainty arises because the time step is managed internally, in `Trixi.jl`, through multiple hidden layers, eventually becoming obscured. Further exploratory research is needed to ensure everything is functioning as intended. In a similar vein, is the approach of dividing the time scales to handle numerical errors correct? Based on the agreement with experimental results and the use of the method of lines and dimensional splitting, it seems so. However, can we prove that the algorithm converges to a unique physical solution? Are there any additional thermodynamic or numerical constraints that we have not considered in solving our equations?

These are just some of the general questions that people might wonder about, as we certainly do. Overall, we are pleased with the final outcome, but as always, new scientific inquiries will lead to more questions as we explore new avenues in science.

CHAPTER 9

APPENDIX

In this section, we present the derivation of equation (3.18). We use the same variables as in the main text, unless stated otherwise. Given a general entropy density for a lyotropic LC emulsion of the form

$$s_{total} = s(\rho, \epsilon, \phi, \boldsymbol{Q}, \partial_{\boldsymbol{r}}\rho, \partial_{\boldsymbol{r}}\epsilon, \partial_{\boldsymbol{r}}\phi, \partial_{\boldsymbol{r}}\boldsymbol{Q}) + s_0(\rho, \epsilon), \tag{9.1}$$

and the corresponding total entropy given by

$$S_{Total} = \int s_{total}(\rho, \epsilon, \phi, \boldsymbol{Q}, \partial_{\boldsymbol{r}}\rho, \partial_{\boldsymbol{r}}\epsilon, \partial_{\boldsymbol{r}}\phi, \partial_{\boldsymbol{r}}\boldsymbol{Q})\ dV, \tag{9.2}$$

where s_0 is again the background fluid entropy and s is any additional entropy in terms of the variables $\boldsymbol{x}$ and their gradients. Here we explicitly state that s_{total} is not a function of $\boldsymbol{m}$ or the gradients of $\boldsymbol{m}$ and depends only on first-order derivatives of $\boldsymbol{x}$, which implies the functional derivative is

$$\frac{\delta S_{Total}}{\delta \boldsymbol{x}} = \frac{\partial s_{total}}{\partial \boldsymbol{x}} - \frac{\partial}{\partial r_m}\left(\frac{\partial s_{total}}{\partial(\partial_m \boldsymbol{x})}\right). \tag{9.3}$$

Furthermore, we can decompose the regular spatial gradient of equation (9.1) in terms of partial derivatives as follows

$$\frac{\partial s_{total}}{\partial r_i} = \frac{\partial s}{\partial r_i} + \frac{\partial s_0}{\partial r_i}. \tag{9.4}$$

The partial derivatives in Eq. (9.4) are computed using the chain rule:

$$\begin{aligned}\frac{\partial s}{\partial r_i} &= \frac{\partial \rho}{\partial r_i}\frac{\partial s}{\partial \rho} + \frac{\partial^2 \rho}{\partial r_i \partial r_m}\frac{\partial s}{\partial(\partial_m \rho)} + \frac{\partial \epsilon}{\partial r_i}\frac{\partial s}{\partial \epsilon} + \frac{\partial^2 \epsilon}{\partial r_i \partial r_m}\frac{\partial s}{\partial(\partial_m \epsilon)} \\ &+ \frac{\partial \phi}{\partial r_i}\frac{\partial s}{\partial \phi} + \frac{\partial^2 \phi}{\partial r_i \partial r_m}\frac{\partial s}{\partial(\partial_m \phi)} + \frac{\partial Q_{kl}}{\partial r_i}\frac{\partial s}{\partial Q_{kl}} + \frac{\partial^2 Q_{kl}}{\partial r_i \partial r_m}\frac{\partial s}{\partial(\partial_m Q_{kl})},\end{aligned} \tag{9.5}$$

and

$$\frac{\partial s_0}{\partial r_i} = \frac{\partial \rho}{\partial r_i}\frac{\partial s_0}{\partial \rho} + \frac{\partial \epsilon}{\partial r_i}\frac{\partial s_0}{\partial \epsilon}. \tag{9.6}$$

As in the main text, we find the total entropy functional derivative

$$\left(\frac{\delta S_{Total}}{\delta \boldsymbol{x}}\right)^{\dagger} = \left(\frac{\delta S_{Total}}{\delta \rho}, \mathbf{0}, \frac{\delta S_{Total}}{\delta \epsilon}, \frac{\delta S_{Total}}{\delta \phi}, \frac{\delta S_{Total}}{\delta Q_{kl}}\right), \tag{9.7}$$

with the superscript † denoting the transpose operation.

The relationship useful for finding the entropic pressure contribution is dictated by the degeneracy condition in Eq. (2.2), which, together with Eqs. (3.15) and (9.7), reads

$$\begin{aligned} \boldsymbol{L}\cdot\frac{\delta S_{Total}}{\delta \boldsymbol{x}} &= \rho\frac{\partial}{\partial r_i}\left(\frac{\delta S_{Total}}{\delta \rho}\right) + \epsilon\frac{\partial}{\partial r_i}\left(\frac{\delta S_{Total}}{\delta \epsilon}\right) \\ &+ \frac{\partial}{\partial r_m}\left[\pi^S_{mi}\frac{\delta S_{Total}}{\delta \epsilon}\right] - \frac{\partial \phi}{\partial r_i}\frac{\delta S_{Total}}{\delta \phi} + (L_{25})_{ikl}\frac{\delta S_{Total}}{\delta Q_{kl}} = 0_i, \end{aligned} \tag{9.8}$$

which is the second component of the entropy degeneracy. By combining equations (9.1), (9.3), (3.16), and (9.8), we obtain

$$\begin{aligned} 0_i &= \rho\frac{\partial}{\partial r_i}\left(\frac{\partial s}{\partial \rho}\right) - \rho\frac{\partial}{\partial r_i}\left[\frac{\partial}{\partial r_m}\left(\frac{\partial s}{\partial(\partial_m \rho)}\right)\right] + \rho\frac{\partial}{\partial r_i}\left(\frac{\partial s_0}{\partial \rho}\right) \\ &+ \epsilon\frac{\partial}{\partial r_i}\left(\frac{\partial s}{\partial \epsilon}\right) - \epsilon\frac{\partial}{\partial r_i}\left[\frac{\partial}{\partial r_m}\left(\frac{\partial s}{\partial(\partial_m \epsilon)}\right)\right] + \epsilon\frac{\partial}{\partial r_i}\left(\frac{\partial s_0}{\partial \epsilon}\right) \\ &+ \frac{\partial}{\partial r_m}\left[\pi^S_{mi}\frac{\delta S_{Total}}{\delta \epsilon}\right] - \frac{\partial \phi}{\partial r_i}\frac{\partial s}{\partial \phi} + \frac{\partial \phi}{\partial r_i}\frac{\partial}{\partial r_m}\left(\frac{\partial s}{\partial(\partial_m \phi)}\right) \\ &- \frac{\partial Q_{kl}}{\partial r_i}\frac{\partial s}{\partial Q_{kl}} + \frac{\partial Q_{kl}}{\partial r_i}\frac{\partial}{\partial r_m}\left(\frac{\partial s}{\partial(\partial_m Q_{kl})}\right) + \frac{\partial}{\partial r_m}L^{\mathbf{s}}_{mi}, \end{aligned} \tag{9.9}$$

where

$$L^{\mathbf{s}}_{mi} = 2\left[\left(Q_{mi} + \frac{I_{mi}}{d}\right)\left(Q_{kl} + \frac{I_{kl}}{d}\right)\frac{\delta S_{Total}}{\delta Q_{kl}} - \left(Q_{ml} + \frac{I_{ml}}{d}\right)\frac{\delta S_{Total}}{\delta Q_{li}}\right], \tag{9.10}$$

note that in Eq. (9.9), we, of course, recall $\frac{\partial s_0}{\partial(\partial_m \boldsymbol{x})} = \mathbf{0}$.

To further simplify Eq. (9.9), we need 1) the Leibniz product rule as

$$\boldsymbol{x}\frac{\partial}{\partial r_i}\left(\frac{\partial A_s}{\partial \boldsymbol{x}}\right) = \frac{\partial}{\partial r_i}\left(\boldsymbol{x}\frac{\partial A_s}{\partial \boldsymbol{x}}\right) - \frac{\partial A_s}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial r_i}, \tag{9.11}$$

and

$$\boldsymbol{x}\frac{\partial}{\partial r_i}\left[\frac{\partial}{\partial r_m}\left(\frac{\partial A_s}{\partial(\partial_m \boldsymbol{x})}\right)\right] = \frac{\partial}{\partial r_i}\left[\boldsymbol{x}\frac{\partial}{\partial r_m}\left(\frac{\partial A_s}{\partial(\partial_m \boldsymbol{x})}\right)\right] - \frac{\partial \boldsymbol{x}}{\partial r_i}\frac{\partial}{\partial r_m}\left(\frac{\partial A_s}{\partial(\partial_m \boldsymbol{x})}\right), \tag{9.12}$$

for any arbitrary entropy density A_s; 2) the local-equilibrium Euler equation:

$$\frac{p_0}{T} = -\rho\frac{\partial s_0}{\partial \rho} - \epsilon\frac{\partial s_0}{\partial \epsilon} + s_0, \tag{9.13}$$

which now can be expressed, with Eq. (9.6), as

$$\frac{\partial}{\partial r_m}\left(-\frac{p_0}{T}\ I_{mi}\right) = \frac{\partial}{r_i}\left(\rho\frac{\partial s_0}{\partial \rho}\right) + \frac{\partial}{r_i}\left(\epsilon\frac{\partial s_0}{\partial \epsilon}\right) - \frac{\partial \rho}{\partial r_i}\frac{\partial s_0}{\partial \rho} - \frac{\partial \epsilon}{\partial r_i}\frac{\partial s_0}{\partial \epsilon}; \tag{9.14}$$

and 3) the following relationship:

$$\begin{aligned}
& \frac{\partial}{\partial r_m}\Bigg[- s\ I_{mi} + \left(\frac{\partial \rho}{\partial r_i}\right)\left(\frac{\partial s}{\partial(\partial_m \rho)}\right) + \left(\frac{\partial \epsilon}{\partial r_i}\right)\left(\frac{\partial s}{\partial(\partial_m \epsilon)}\right) \\
+ \quad & \left(\frac{\partial \phi}{\partial r_i}\right)\left(\frac{\partial s}{\partial(\partial_m \phi)}\right) + \left(\frac{\partial Q_{kl}}{\partial r_i}\right)\left(\frac{\partial s}{\partial(\partial_m Q_{kl})}\right)\Bigg] \\
= \quad & -\frac{\partial s}{\partial \rho}\frac{\partial \rho}{\partial r_i} + \frac{\partial \rho}{\partial r_i}\frac{\partial}{\partial r_m}\left(\frac{\partial s}{\partial(\partial_m \rho)}\right) - \frac{\partial s}{\partial \epsilon}\frac{\partial \epsilon}{\partial r_i} + \frac{\partial \epsilon}{\partial r_i}\frac{\partial}{\partial r_m}\left(\frac{\partial s}{\partial(\partial_m \epsilon)}\right) \\
- \quad & \frac{\partial s}{\partial \phi}\frac{\partial \phi}{\partial r_i} + \frac{\partial \phi}{\partial r_i}\frac{\partial}{\partial r_m}\left(\frac{\partial s}{\partial(\partial_m \phi)}\right) - \frac{\partial s}{\partial Q_{kl}}\frac{\partial Q_{kl}}{\partial r_i} + \frac{\partial Q_{kl}}{\partial r_i}\frac{\partial}{\partial r_m}\left(\frac{\partial s}{\partial(\partial_m Q_{kl})}\right)
\end{aligned} \tag{9.15}$$

which can be derived using Eq. (9.5).

Finally, inserting Equations (9.11), (9.12), (9.14), and (9.15), into Eq. (9.9) in that order, so that all terms remain inside derivatives $\frac{\partial}{\partial r_m}$, sometimes requiring a change of index from i

to m using the identity matrix I_{mi}, yields the following equation:

$$\begin{aligned}
& \frac{\partial}{\partial r_m}\left[-\rho\frac{\partial}{\partial r_k}\left(\frac{\partial s}{\partial(\partial_k\rho)}\right)-\epsilon\frac{\partial}{\partial r_k}\left(\frac{\partial s}{\partial(\partial_k\epsilon)}\right)+\rho\frac{\partial s}{\partial\rho}+\epsilon\frac{\partial s}{\partial\epsilon}-s-\frac{p_0}{T}\right]I_{mi} \\
+ \quad & \frac{\partial}{\partial r_m}\left[\left(\frac{\partial\rho}{\partial r_i}\right)\left(\frac{\partial s}{\partial(\partial_m\rho)}\right)+\left(\frac{\partial\epsilon}{\partial r_i}\right)\left(\frac{\partial s}{\partial(\partial_m\epsilon)}\right)\right. \\
+ \quad & \left.\left(\frac{\partial\phi}{\partial r_i}\right)\left(\frac{\partial s}{\partial(\partial_m\phi)}\right)+\left(\frac{\partial Q_{kl}}{\partial r_i}\right)\left(\frac{\partial s}{\partial(\partial_m Q_{kl})}\right)\right] \\
+ \quad & \frac{\partial}{\partial r_m}L^{\mathtt{s}}_{mi}=-\frac{\partial}{\partial r_m}\left[\pi^S_{mi}\frac{\delta S_{Total}}{\delta\epsilon}\right]. \qquad (9.16)
\end{aligned}$$

Equation (9.16) then proves the relationship shown in Eq. (3.18), as desired.

BIBLIOGRAPHY

[1] Apala Majumdar, Marchetti Maria Cristina, and Epifanio G Virga. "Perspectives in active liquid crystals". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 372 (2014).

[2] M. C. Marchetti et al. "Hydrodynamics of soft active matter". In: *Rev. Mod. Phys.* 85 (3 July 2013), pp. 1143–1189. DOI: 10.1103/RevModPhys.85.1143. URL: https://link.aps.org/doi/10.1103/RevModPhys.85.1143.

[3] Tamás Vicsek et al. "Novel Type of Phase Transition in a System of Self-Driven Particles". In: *Phys. Rev. Lett.* 75 (6 Aug. 1995), pp. 1226–1229. DOI: 10.1103/PhysRevLett.75.1226. URL: https://link.aps.org/doi/10.1103/PhysRevLett.75.1226.

[4] Andrey Sokolov et al. "Emergence of Radial Tree of Bend Stripes in Active Nematics". In: *Physical Review. X* 9.3 (July 2019). ISSN: 2160-3308. DOI: 10.1103/PhysRevX.9.031014.

[5] Boris I. Shraiman. "Mechanical feedback as a possible regulator of tissue growth". In: *Proceedings of the National Academy of Sciences* 102.9 (2005), pp. 3318–3323. DOI: 10.1073/pnas.0404782102. eprint: https://www.pnas.org/content/102/9/3318.full.pdf. URL: https://www.pnas.org/content/102/9/3318.

[6] Jonas Ranft et al. "Fluidization of tissues by cell division and apoptosis". In: *Proceedings of the National Academy of Sciences* 107.49 (2010), pp. 20863–20868. ISSN: 0027-8424. DOI: 10.1073/pnas.1011086107. eprint: https://www.pnas.org/content/107/49/20863.full.pdf. URL: https://www.pnas.org/content/107/49/20863.

[7] Hernán Morales-Navarrete et al. "Liquid-crystal organization of liver tissue". In: *eLife* 8 (June 2019). Ed. by Pierre Sens, Arup K Chakraborty, and Benoit Ladoux, e44860. ISSN: 2050-084X. DOI: 10.7554/eLife.44860. URL: https://doi.org/10.7554/eLife.44860.

[8] Christopher Dombrowski et al. "Self-Concentration and Large-Scale Coherence in Bacterial Dynamics". In: *Phys. Rev. Lett.* 93 (9 Aug. 2004), p. 098103. DOI: 10.1103/PhysRevLett.93.098103. URL: https://link.aps.org/doi/10.1103/PhysRevLett.93.098103.

[9] D. J. T Sumpter. *Collective animal behavior*. Princeton University Press, 2010.

[10] Vijay Narayan, Sriram Ramaswamy, and Narayanan Menon. "Long-Lived Giant Number Fluctuations in a Swarming Granular Nematic". In: *Science* 317.5834 (2007), pp. 105–108. ISSN: 0036-8075. DOI: 10.1126/science.1140414. eprint: https://science.sciencemag.org/content/317/5834/105.full.pdf. URL: https://science.sciencemag.org/content/317/5834/105.

[11] Antonio Tavera-Vázquez et al. *Light-activated Janus microswimmers in a thermotropic liquid crystal*. 2023. arXiv: 2312.12866 [cond-mat.soft].

[12] Daniel Needleman and Zvonimir Dogic. "Active matter at the interface between materials science and cell biology". In: *Nature Reviews Materials* 2 (July 2017). Review Article, 17048 EP. URL: http://dx.doi.org/10.1038/natrevmats.2017.48.

[13] Nitin Kumar et al. "Tunable structure and dynamics of active liquid crystals". In: *Science Advances* 4.10 (2018). DOI: 10.1126/sciadv.aat7779. eprint: https://advances.sciencemag.org/content/4/10/eaat7779.full.pdf. URL: https://advances.sciencemag.org/content/4/10/eaat7779.

[14] Sumesh P Thampi, Ramin Golestanian, and Julia M Yeomans. "Vorticity, defects and correlations in active turbulence". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 372.2029 (2014), p. 20130366.

[15] Pierre-Gilles De Gennes and Jacques Prost. *The physics of liquid crystals*. 83. Oxford university press, 1993.

[16] James N Wilking et al. "Biofilms as complex fluids". In: *MRS bulletin* 36.5 (2011), pp. 385–391.

[17] Denis Boyer et al. "Buckling instability in ordered bacterial colonies". In: *Physical biology* 8.2 (2011), p. 026008.

[18] Nicholas Darnton et al. "Moving fluid with bacterial carpets". In: *Biophysical Journal* 86.3 (2004), pp. 1863–1870.

[19] Kun-Ta Wu et al. "Transition from turbulent to coherent flows in confined three-dimensional active fluids". In: *Science* 355.6331 (2017), eaal1979.

[20] Pau Guillamat, Jordi Ignés-Mullol, and Francesc Sagués. "Control of active liquid crystals with a magnetic field". In: *Proceedings of the National Academy of Sciences* 113.20 (2016), pp. 5498–5502.

[21] Louise C Head et al. "Spontaneous self-constraint in active nematic flows". In: *Nature Physics* 20.3 (2024), pp. 492–500.

[22] N Sulaiman, D Marenduzzo, and JM Yeomans. "Lattice Boltzmann algorithm to simulate isotropic-nematic emulsions". In: *Physical Review E* 74.4 (2006), p. 041708.

[23] Livio Nicola Carenza et al. "Lattice Boltzmann methods and active fluids". In: *The European Physical Journal E* 42 (2019), pp. 1–38.

[24] Lan Liu. *Nucleation and Order of a Polymer in a Confined Liquid Crystal Matrix*. University of California, Santa Barbara, 2019.

[25] Renato Assante et al. "Active turbulence and spontaneous phase separation in inhomogeneous extensile active gels". In: *Soft Matter* 19.2 (2023), pp. 189–198.

[26] Dinshaw S. Balsara. *Numerical PDE Techniques for Scientists and Engineers*. Accessed: 2024-08-21. 2013. URL: `https://www3.nd.edu/~dbalsara/Numerical-PDE-Course/index.shtml`.

[27] J. P. Hernández-Ortiz et al. "Modeling flows of confined nematic liquid crystals". In: *J. Chem. Phys.* 134 (2011), p. 134905.

[28] Amin Doostmohammadi et al. "Active nematics". In: *Nature communications* 9.1 (2018), p. 3246.

[29] Antony N Beris and Brian J Edwards. *Thermodynamics of flowing systems: with internal microstructure*. 36. Oxford University Press on Demand, 1994.

[30] R Aditi Simha and Sriram Ramaswamy. "Hydrodynamic fluctuations and instabilities in ordered suspensions of self-propelled particles". In: *Physical review letters* 89.5 (2002), p. 058101.

[31] Julia M Yeomans. "The hydrodynamics of active systems". In: *La Rivista del Nuovo Cimento* 40.1 (2017), pp. 1–31.

[32] Hans Christian Öttinger. *Beyond equilibrium thermodynamics*. John Wiley & Sons, 2005.

[33] M. Pavelka, V. Klika, and M. Grmela. *Multiscale Thermo-Dynamics: Introduction to GENERIC*. Berlin: de Gruyter, 2018.

[34] Miroslav Grmela and Hans Christian Öttinger. "Dynamics and thermodynamics of complex fluids. I. Development of a general formalism". In: *Physical Review E* 56.6 (1997), p. 6620.

[35] Hans Christian Öttinger and Miroslav Grmela. "Dynamics and thermodynamics of complex fluids. II. Illustrations of a general formalism". In: *Physical Review E* 56.6 (1997), p. 6633.

[36] Asja Jelić, Markus Hütter, and Hans Christian Öttinger. "Dissipative electromagnetism from a nonequilibrium thermodynamics perspective". In: *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 74.4 (2006), p. 041126.

[37] Brian J. Edwards. In: *Journal of Non-Equilibrium Thermodynamics* 23.4 (1998), pp. 301–333. DOI: doi:10.1515/jnet.1998.23.4.301. URL: https://doi.org/10.1515/jnet.1998.23.4.301.

[38] Christopher Rackauckas and Qing Nie. "DifferentialEquations.jl–a performant and feature-rich ecosystem for solving differential equations in Julia". In: *Journal of Open Research Software* 5.1 (2017).

[39] Shiyi Chen and Gary D Doolen. "Lattice Boltzmann method for fluid flows". In: *Annual review of fluid mechanics* 30.1 (1998), pp. 329–364.

[40] F J Higuera and Javier Jiménez. "Boltzmann approach to lattice gas simulations". In: *Europhysics letters* 9.7 (1989), p. 663.

[41] Xiaoyi He and Li-Shi Luo. "Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation". In: *Physical review E* 56.6 (1997), p. 6811.

[42] Timm Krüger et al. *The lattice Boltzmann method*. Vol. 10. 978-3. Springer, 2017.

[43] Hendrik Ranocha et al. "Adaptive numerical simulations with Trixi.jl: A case study of Julia for scientific computing". In: *Proceedings of the JuliaCon Conferences* 1.1 (2022), p. 77. DOI: 10.21105/jcon.00077. arXiv: 2108.06476 [cs.MS].

[44] Michael Schlottke-Lakemper et al. "A purely hyperbolic discontinuous Galerkin approach for self-gravitating gas dynamics". In: *Journal of Computational Physics* 442 (June 2021), p. 110467. DOI: 10.1016/j.jcp.2021.110467. arXiv: 2008.10593 [math.NA].

[45] Michael Schlottke-Lakemper et al. *Trixi.jl: Adaptive high-order numerical simulations of hyperbolic PDEs in Julia*. https://github.com/trixi-framework/Trixi.jl. Sept. 2021. DOI: 10.5281/zenodo.3996439.

[46] JM Ball. "The Q-tensor theory of liquid crystals". In: *Lecture slides, Benin Summer School* 28 (2010).

[47] Pierre-Gilles de Gennes. "Dynamics of fluctuations and spinodal decomposition in polymer blends". In: *The Journal of Chemical Physics* 72.9 (1980), pp. 4756–4763.

[48] John W Cahn and John E Hilliard. "Free energy of a nonuniform system. I. Interfacial free energy". In: *The Journal of chemical physics* 28.2 (1958), pp. 258–267.

[49] Nikolas Provatas and Ken Elder. *Phase-field methods in materials science and engineering*. John Wiley & Sons, 2011.

[50] Phillip M Rauscher, Hans Christian Öttinger, and Juan J de Pablo. "Nonequilibrium statistical thermodynamics of multicomponent interfaces". In: *Proceedings of the National Academy of Sciences* 119.24 (2022), e2121405119.

[51] John Shipley Rowlinson and Benjamin Widom. *Molecular theory of capillarity*. Courier Corporation, 2013.

[52] Terrell L Hill. *Thermodynamics of small systems*. Courier Corporation, 1994.

[53] Takeaki Araki and Hajime Tanaka. "Nematohydrodynamic effects on the phase separation of a symmetric mixture of an isotropic liquid and a liquid crystal". In: *Physical review letters* 93.1 (2004), p. 015702.

[54] Koichi Hashiguchi. *Nonlinear continuum mechanics for finite elasticity-plasticity: Multiplicative decomposition with subloading surface model*. Elsevier, 2020.

[55] J Fukuda. "Free energy of semiflexible polymers and structure of interfaces". In: *The European Physical Journal B-Condensed Matter and Complex Systems* 7.4 (1999), pp. 573–583.

[56] Frederick C Frank. "I. Liquid crystals. On the theory of liquid crystals". In: *Discussions of the Faraday Society* 25 (1958), pp. 19–28.

[57] CB Muratov et al. "Theory of phase separation kinetics in polymer–liquid crystal systems". In: *The Journal of chemical physics* 116.11 (2002), pp. 4723–4734.

[58] Natalie Zimmermann et al. “Self-organized assemblies of colloidal particles obtained from an aligned chromonic liquid crystal dispersion”. In: *Soft Matter* 11.8 (2015), pp. 1547–1553.

[59] Ping Sheng and EB Priestley. “The Landau-de Gennes theory of liquid crystal phase transitions”. In: *introduction to Liquid Crystals*. Springer, 1974, pp. 143–201.

[60] Masao Doi. “Molecular dynamics and rheological properties of concentrated solutions of rodlike polymers in isotropic and liquid crystalline phases”. In: *Journal of Polymer Science: Polymer Physics Edition* 19.2 (1981), pp. 229–243.

[61] Asja Jelić, Patrick Ilg, and Hans Christian Öttinger. “Bridging length and time scales in sheared demixing systems: From the Cahn-Hilliard to the Doi-Ohta model”. In: *Physical Review E* 81.1 (2010), p. 011131.

[62] Peter William Atkins, Julio De Paula, and James Keeler. *Atkins' physical chemistry*. Oxford university press, 2023.

[63] Noe Atzin et al. “Minimal Model of Solitons in Nematic Liquid Crystals”. In: *Physical Review Letters* 131.18 (2023), p. 188101.

[64] A Matsuyama, RML Evans, and ME Cates. “Non-uniformities in polymer/liquid crystal mixtures: I. Interfacial tension”. In: *The European Physical Journal E* 9 (2002), pp. 79–87.

[65] R. Zhang et al. “Spatiotemporal control of liquid crystal structure and dynamics through activity patterning”. In: *Nature Materials* 20.6 (2021), pp. 875–882. DOI: 10.1038/s41563-020-00901-4.

[66] S. Zhou, S. Shiyanovskii, H.S. Park, et al. “Fine structure of the topological defect cores studied for disclinations in lyotropic chromonic liquid crystals”. In: *Nature Communications* 8 (2017), p. 14974. DOI: 10.1038/ncomms14974.

[67] Alok Ranjan Tiwary, A Raja Bayanna, and Shibu K Mathew. "Estimation of order parameter of a liquid crystal variable retarder using Haller's approximation". In: *Applied Optics* 56.14 (2017), pp. 4180–4184.

[68] E.V.L. de Mello and Otton Teixeira da Silveira Filho. "Numerical study of the Cahn–Hilliard equation in one, two and three dimensions". In: *Physica A: Statistical Mechanics and its Applications* 347 (2005), pp. 429–443. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2004.08.076. URL: https://www.sciencedirect.com/science/article/pii/S037843710401180X.

[69] G. R. Pérez-Lemus, J. C. Armas-Pérez, and G. A. Chapela. "Numerical evidence of liquid crystalline mesophases of a lollipop shaped model in two dimensions". In: *AIP Advances* 7.12 (2017).

[70] Christopher W Macosko. *Rheology: principles, measurements, and applications*. VCH Publishes, 1994.

[71] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Society for Industrial and Applied Mathematics, 2007.

[72] A. Vananroye, P. Van Puyvelde, and P. Moldenaers. "Effect of confinement on droplet breakup in sheared emulsions". In: *Langmuir* 22.9 (2006), pp. 3972–3974.

[73] Tadej Emeršič et al. "Sculpting stable structures in pure liquids". In: *Science Advances* 5.2 (2019), eaav4283. DOI: 10.1126/sciadv.aav4283. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.aav4283. URL: https://www.science.org/doi/abs/10.1126/sciadv.aav4283.

[74] Stefano S Turzi. "Active nematic gels as active relaxing solids". In: *Physical Review E* 96.5 (2017), p. 052603.

[75] George I. Bell. "Models for the Specific Adhesion of Cells to Cells". In: *Science* 200.4342 (1978), pp. 618–627. DOI: 10.1126/science.347575. eprint: https://www.science.org/doi/pdf/10.1126/science.347575. URL: https://www.science.org/doi/abs/10.1126/science.347575.

[76] F. L. Memarian et al. "Controlling chaos: Periodic defect braiding in active nematics confined to a cardioid". In: *Physical Review Letters* 132.22 (2024), p. 228301. DOI: 10.1103/PhysRevLett.132.228301.

[77] K. L. Weirich et al. "Self-organizing motors divide active liquid droplets". In: *Proceedings of the National Academy of Sciences* 116.23 (2019), pp. 11125–11130. DOI: 10.1073/pnas.1819467116.

[78] A. Rapini and M. Papoular. "Distorsion d'une lamelle nématique sous champ magnétique conditions d'ancrage aux parois". In: *Le Journal de Physique Colloques* 30.C4 (1969), pp. C4–54.

[79] Pierre-Gilles de Gennes. "A semi-fast artificial muscle". In: *Comptes Rendus de l'Academie des Sciences Series IIB Mechanics Physics Chemistry Astronomy* 5.324 (1997), pp. 343–348.

[80] Fiona Mukherjee et al. "Liquid crystals as multifunctional interfaces for trapping and characterizing colloidal microplastics". In: *Small* 19.23 (2023), p. 2207802.

[81] Jiyeon Kim, Mashooq Khan, and Soo-Young Park. "Glucose sensor using liquid-crystal droplets made by microfluidics". In: *ACS applied materials & interfaces* 5.24 (2013), pp. 13135–13139.

[82] Peter C Mushenheim et al. "Dynamic self-assembly of motile bacteria in liquid crystals". In: *Soft Matter* 10.1 (2014), pp. 88–95.

[83] Sandeep Urandur et al. “Theranostic lyotropic liquid crystalline nanostructures for selective breast cancer imaging and therapy”. In: *Acta Biomaterialia* 113 (2020), pp. 522–540.