THE UNIVERSITY *of* EDINBURGH

This thesis has been submitted in fulfilment of the requirements for a postgraduate degree (e. g. PhD, MPhil, DClinPsychol) at the University of Edinburgh. Please note the following terms and conditions of use:

- This work is protected by copyright and other intellectual property rights, which are retained by the thesis author, unless otherwise stated.
- A copy can be downloaded for personal non-commercial research or study, without prior permission or charge.
- This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author.
- The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.
- When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.

Computational Modelling and Optimal Control of Interacting Particle Systems

Connecting Dynamic Density Functional Theory

and PDE-Constrained Optimization

Jonna C. Roden

Doctor of Philosophy
University of Edinburgh
2022

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Jonna C. Roden*)

Lay Summary

From animal flocking, the spread of diseases, and formation of opinions to nano-filtration, brewing, and printing – there are plenty of processes in nature, society, and industry that can be thought of as systems of interacting particles. Considering flocking of birds, we can (mathematically) think of each bird as a particle that interacts with the other birds (particles) around them in the sky. Their actions are influenced by the wind current, each others' signals, and environmental factors, such as climate conditions or predators. In an industrial setting, yeast particles are suspended in the beer during the brewing process. These yeast particles interact by clumping together, while being affected by temperature and by gravity, so that they sediment to the bottom of the brewing vessel over time.

We can build mathematical models of such interacting particle systems and use them to describe all the processes mentioned above. Such models are called partial differential equation (PDE) models, and they capture how the particles move under the influence of diffusion and different forces, such as gravity and currents (as in the birds example). Moreover, such models can describe how particles interact with each other, for example whether they form clusters (as the yeast does in brewing).

Once such a PDE model is found, we can ask further questions: How can the sedimentation of yeast be improved, so that the brewing process is sped up? How can we support birds to find the best route for migration? Mathematically, these are questions about optimization, or more specifically, about optimal control. The key question in optimal control is: What is the optimal problem setup (the aspect we can control) that will result in a situation close to some desired outcome at a minimal investment of energy? In the case of brewing we may add more clumping agents (control) to the brewing vessel to cause the formation of larger, heavier clumps of yeast that sediment quicker (the desired outcome).

This thesis is concerned with such questions, i.e., the optimal control of the PDEs that describe particle dynamics. Since the models, as well as the optimization problems, are not solvable analytically, a key topic of this work is the development of fast and accurate computational solvers for such problems. Moreover, several extensions of the method are introduced. Since most real-world examples are not confined to a box-like (i.e., easy to compute) shape, one extension is to adapt the numerical methods so that problems can be computed on more complicated shapes, such as a brewing vessel.

Moreover, depending on the application of interest, more physical effects have to be included into the PDE model. First, we extend the model to describe different kinds of particles and how they may interact. For example, we could describe how seagulls and pigeons interact with each other, as well as among themselves, in the same model. In the original model, we assume each particle to be soft, so that we could squeeze many particles into each other to fit into a small box. In a second extension of the model, each particle is like a marble, so that fewer particles fit in the same box, since they have a hard surface and cannot overlap. The methods developed in this thesis can be applied to the modelling and optimization of various real-world processes.

Abstract

Processes that can be described by systems of interacting particles are ubiquitous in nature, society, and industry, ranging from animal flocking, the spread of diseases, and formation of opinions to nano-filtration, brewing, and printing. In real-world applications it is often relevant to not only model a process of interest, but to also optimize it in order to achieve a desired outcome with minimal resources, such as time, money, or energy.

Mathematically, the dynamics of interacting particle systems can be described using Dynamic Density Functional Theory (DDFT). The resulting models are nonlinear, nonlocal partial differential equations (PDEs) that include convolution integral terms. Such terms also enter the naturally arising no-flux boundary conditions. Due to the nonlocal, nonlinear nature of such problems they are challenging both to analyse and solve numerically.

In order to optimize processes that are modelled by PDEs, one can apply tools from PDE-constrained optimization. The aim here is to drive a quantity of interest towards a target state by varying a control variable. This is constrained by a PDE describing the process of interest, in which the control enters as a model parameter. Such problems can be tackled by deriving and solving the (first-order) optimality system, which couples the PDE model with a second PDE and an algebraic equation. Solving such a system numerically is challenging, since large matrices arise in its discretization, for which efficient solution strategies have to be found. Most work in PDE-constrained optimization addresses problems in which the control is applied linearly, and which are constrained by local, often linear PDEs, since introducing nonlinearity significantly increases the complexity in both the analysis and numerical solution of the optimization problem.

However, in order to optimize real-world processes described by nonlinear, nonlocal DDFT models, one has to develop an optimal control framework for such models. The aim is to drive the particles to some desired distribution by applying control either linearly, through a particle source, or bilinearly, though an advective field. The optimization process is constrained by the DDFT model that describes how the particles move under the influence of advection, diffusion, external forces, and particle–particle interactions. In order to tackle this, the (first-order) optimality system is derived, which, since it involves nonlinear (integro-)PDEs that are coupled nonlocally in space and time, is significantly harder than in the standard case. Novel numerical methods are developed, effectively combining pseudospectral methods and iterative solvers, to efficiently and accurately solve such a system.

In a next step this framework is extended so that it can capture and optimize industrially relevant processes, such as brewing and nano-filtration. In order to do so, extensions to both the DDFT model and the numerical method are made. Firstly, since industrial processes often involve tubes, funnels, channels, or tanks of various shapes, the PDE model itself, as well as the optimization problem, need to be solved on complicated domains. This is achieved by developing a novel spectral element approach that is compatible with both the PDE solver and the optimal control framework. Secondly, many industrial processes, such as nano-filtration, involve more than one type of particle. Therefore, the DDFT model is extended to describe multiple particle species. Finally, depending on the application of interest, additional physical effects need to be included in the model. In this thesis, to model sedimentation processes in brewing, the model is modified to capture volume exclusion effects.

Acknowledgements

I would like to be able to thank my supervisors, Ben Goddard and John Pearson. For better or worse, PhD supervisors have a huge influence on a PhD, and the two of them have done everything to make this a rewarding experience. The level of support I have received during the past few years has been truly extraordinary and there is not enough space on this page to acknowledge it properly. They have both found incredible amounts of time in their (already way too busy) schedules to mentor me and trusted that it would be a good investment. This time was used for explaining the most complicated mathematical concepts patiently and in great detail, for showing me how to tackle interesting yet challenging research problems, for proofreading my work in incredible detail, and for teaching me how to convey the value of my work to others (If anyone asks: I am now exclusively working on sophisticated, novel, state-of-the-art frameworks). I always felt guided in this project and encouraged to ask any and all questions – repeatedly. I can't recall ever leaving a meeting without having learned something new, a plan of the next steps to take, and the knowledge that I can ask for help if I get stuck halfway through the week. At the same time, they were always interested in my thoughts and opinions, and gave me great freedom in choosing both how I work and what I work on. While not quite able to, I would like to thank both of my supervisors for role modelling what it means to be a great researcher, teacher, and mentor.

While my supervisors guided me throughout my PhD, its completion was supported by my examiners Matthias Schmidt and Aretha Teckentrup. I really appreciate the time they spent carefully reading my thesis and that they made my viva a rewarding experience. I very much enjoyed the interesting, yet challenging, in-depth discussion about my research.

In relation to academic mentors I would also like to thank Penny Davies and David Pritchard from the University of Strathclyde. Throughout my undergraduate degree David Pritchard's advice and encouragement were invaluable to me, enabling me to choose the most interesting, yet challenging, classes and career paths. At the end of my degree Penny Davies reached out to me, told me she thought I would make a good PhD candidate, and encouraged me to give it a shot. I am very grateful she did – I would not have considered applying to MIGSAA without her intervention.

I would like to thank MIGSAA[1] for giving me the opportunity, time, and financial support to do the work in this thesis, and for offering interesting and challenging classes that taught me a lot and connected me to other PhD students at the Maxwell Institute. Through working together on assignments, projects, and presentations, as well as interacting at the PG Colloquium and in the SIAM Chapter, a community was created that I am very grateful for. In particular, my MIGSAA cohort, Michael, Ivona, Santolo, and Kreshimir made me feel like a part of this group from the very beginning, while Nagisa, Nestor, Magda, and Yvonne became friends that I (almost) forgot are colleagues as well. I would like to thank Kieran for a memorable first year project (and Tom for his patience during it), for the afternoons 'collaborating' over doughnuts and tea at Room and Rumours, and for representing the second 'A' in MIGSAA with me. Bella, thank you for being my lockdown tea time partner and for making our lunch dates an integral part of my PhD. Andrés, thanks for being such a fun academic 'little brother' and for

[1]I would like to thank The Maxwell Institute Graduate School in Analysis and its Applications (EPSRC grant EP/L016508/01), the Scottish Funding Council, Heriot-Watt University, and The University of Edinburgh for supporting this PhD.

giving honest feedback on my work (especially on my drawings in Word). I think I have actually already learned more from you than the other way around, so I apologise and thank you for that!

Last but not least, my family and friends outside of academia have contributed greatly to me keeping my sanity in the past few years. My best friend Eszter and my partner Santiago shared this experience most closely, and I am very grateful for their patience and that they told me regularly to *finally* take a break and watch a 'ski-fi' movie with them. Eszter, thank you for taking me to the beach, mountains, and lakes to get a proper break. Santiago, thank you for supporting my choice to do a PhD from the beginning (possibly even before me) and for 'forcing' me to take your office for the write-up period. I also owe a lot to Renate and Linda for all their support during the months I spent in Germany.

Finally, I would like to thank my mum. I have a lot of things to thank her for, but here I will just mention one: Her certainty that I could do hard things and that I would choose the path that's right for me. I trust her on that because she is, as per job description, *always* right. I know how proud she is of me and how little weight she places on academic achievements at the same time – I am immensely grateful for both.

Contents

Chapter 1

Introduction

Particles are the building blocks for countless phenomena in nature, society, and industry. Their size can range from nanometres to kilometres, with some of the smallest such particles being atoms, while some of the largest are planets and stars. Examples of 'medium' sized particles are molecules, grains of sand, or tennis balls. We can also regard birds, cars, or even humans as particles, depending on the phenomenon we aim to describe.

One of the most fundamental particles is the molecule. Molecules arrange themselves in gas, liquid, and solid phases, depending on how close, on average, the molecules are to one another. With these building blocks we can describe classical fluids, but also complex fluids. The latter is an area of physics and engineering concerned with materials that are mixtures of two or more components, such as emulsions, liquid crystals, polymers, foams, and colloidal fluids. Such materials are highly relevant for manufacturing, food processing, or medical applications, with examples covering a broad spectrum: from inks and paints, through makeup, creams and gels, to foodstuffs, such as chocolate and beer. In the case of colloidal fluids, solid particles are suspended in a liquid, such as in ink, consisting of pigment particles in water. The colloid molecules have sizes of the orders of nanometres to micrometers and are much larger than the molecules of the liquid bath.

While modelling such complex fluids, and in particular colloidal fluids, is a key motivation for the work in this thesis, the models for particle dynamics that we apply are more general and can describe other applications, such as crowd dynamics, animal flocking, and opinion formation. Many further industrial, biological, and medical processes can be described by systems of particles. This includes models of cancer growth, drug delivery, or milling processes in pharmaceutical industries.

In order to describe the dynamics of a collection, or system, of particles, some information about the particles is required: Examples are their mass, the forces they experience, whether and how they interact with each other, and if they are suspended in a bath. Examples of external forces are gravitational, electrical, and magnetic fields. A bath can be, for example, water, oil, or air. Interactions between particles could be electrostatic, magnetic, or even an exchange of information.

While there are many different approaches to modelling a system of interacting particles, some of which will be discussed below, the method applied in this thesis is Dynamic Density Functional Theory (DDFT). In systems with a large number of particles, such as in complex fluids applications, the motion of individual particles is often less important than knowing their average dynamics. Moreover, due to the system's size, the individual particle paths are also difficult to track experimentally. DDFT describes the particles' activity on average by considering a particle density, ρ. This describes the probability of finding any particle at a point in space, at a given time. They key idea is to reduce the (at least) $6N$-dimensional problem that describes the positions and momenta of N individual particles to a three-dimensional problem. DDFT enables the inclusion of particle level phenomena in a continuum model, and is therefore an

example of multiscale modelling.

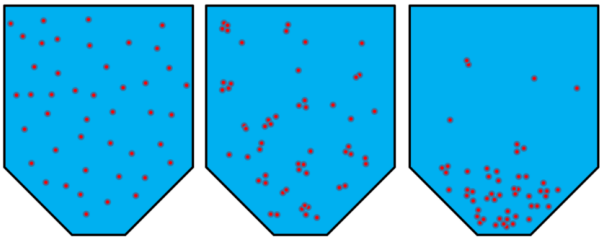

Figure 1.1: Particles are suspended in a fluid. They form clusters and sediment to the bottom of the vessel over time (time running from left to right).

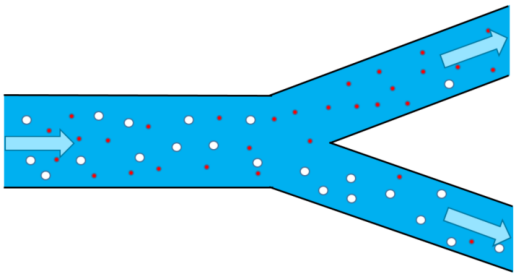

Figure 1.2: A mixture of particles is carried through a channel by an advective field, and separated by the channel geometry.

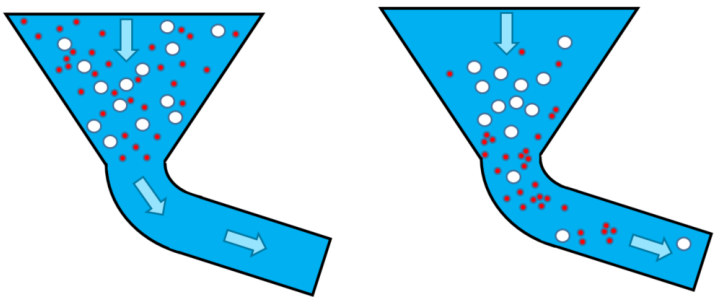

Figure 1.3: A mixture of particles is passing through a funnel. The larger particles form clusters that become too large to pass, exacerbated by the small particles blocking the channel entry (time running from left to right).

Many particle dynamics processes do not take place in a square shape, which is a rather typical choice for a domain in mathematical modelling, but in more complicated domains. Figures 1.1, 1.2, and 1.3 illustrate this point. In Figure 1.1, a vessel contains particles suspended in a bath. The particles form clusters and sediment to the bottom of the vessel over time. Such a situation arises in brewing, where yeast particles are suspended in beer and sediment to the bottom of the brewing vessel before the beer production can be completed. In the second figure (Figure 1.2), a nano-filtration device is sketched. Here, two types of particles are suspended in a fluid, are advected by a flow field, and separated at the end of the channel. In Figure 1.3, a mixture of two particle types is inserted into a funnel and passes into a narrower channel. Both types

of particles form clusters. The small particles enter the channel first and therefore block the larger particles from doing so.

There is not only interest in modelling the kind of processes described up until now, but also in optimizing them. Especially in industry, social science, and medicine, answering optimization questions can be crucial. For example, if we model the process described in Figure 1.3, the natural question to ask is: How can we prevent the white particles from getting stuck in the wider part of the funnel? Assuming the red particles to be magnetic and the white particles not, we could apply a magnetic field to the red particles to keep them in the top part of the funnel until the white particles have passed through. The element that we can *control* is the magnetic field. We can vary this control, so that we get closer to achieving a *desired outcome*, i.e., avoiding obstructions in the funnel. Finding the optimal magnetic field, resulting in minimal obstruction of the funnel, is an example of an *optimal control problem*. Considering the example presented in Figure 1.2, the desired outcome may be the optimal separation of the two types of particles at the right end of the device. The aspect we are able to control could be the rate at which the particle mixture enters the channel, or the strength of the advective field that carries the particles through the device. The aim of the optimal control problem is then to find the optimal inflow rate, or the optimal advective field, that provides the best separation of the two particle species. The desired outcome and the aspect of a process that we can control to closely replicate this outcome are highly problem specific. PDE-constrained optimization is a research field that provides a framework for solving such optimal control problems, in particular problems in which the dynamics of the quantity of interest (e.g., the particle distribution) is described by a PDE model. In this thesis, we aim to solve optimal control problems that are constrained by the integro-PDE models arising from DDFT.

In summary, this thesis is concerned with modelling different kinds of interacting particle systems using Dynamic Density Functional Theory and solving them numerically on complicated domains such as the ones presented in Figures 1.1, 1.2, and 1.3. Moreover, we are interested in numerically solving optimal control problems that involve such particle dynamics models on various domains, as exemplified above.

1.1 Modelling Particle Dynamics

As discussed above, instead of modelling each individual particle and its motion, we consider a particle density ρ. This can be thought of as a probability density, indicating the probability of finding a particle at a certain position $\vec{x}$ in a spatial domain Ω, at a time $t \in (0, T)$. Modelling such particle densities and their dynamics using DDFT is discussed in detail in Chapter 2. Here we briefly present such a model, which is a partial differential equation (PDE) of the following form

$$\begin{aligned} \frac{\partial \rho}{\partial t} &= \nabla \cdot \left(\rho \nabla \frac{\delta \mathcal{F}[\rho]}{\delta \rho} \right) && \text{in } (0,T) \times \Omega, \\ 0 &= \rho \nabla \frac{\delta \mathcal{F}[\rho]}{\delta \rho} \cdot \vec{n} && \text{on } (0,T) \times \partial\Omega, \\ \rho(0, \vec{x}) &= \rho_0(\vec{x}) && \text{at } \{t = 0\} \times \Omega, \end{aligned} \tag{1.1}$$

where $\vec{n}$ is the outward pointing normal and $\partial\Omega$ denotes the boundary of Ω. The functional $\mathcal{F}$ describes the *free energy* of the system in equilibrium, which will also be discussed in detail in Chapter 2. For now we just note that this free energy encodes the physics of the problem. One explicit form of $\mathcal{F}$ is given by

$$\mathcal{F}[\rho] = \int_\Omega \left[\rho(\vec{x}) \left(\ln \rho(\vec{x}) - 1 \right) + \rho(\vec{x}) V_1(\vec{x}) + \frac{1}{2} \rho(\vec{x}) \int_\Omega \rho(\vec{x}') V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \right] d\vec{x}, \tag{1.2}$$

which leads to the following PDE model that describes how a particle density evolves under the influence of diffusive forces, external forces, advective fields, and pairwise particle–particle inter-

actions, such as attraction or repulsion between particles. Note that higher order interactions are omitted in this form of $\mathcal{F}$. The model is an integro-PDE of the form

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= \nabla^2 \rho + \nabla \cdot (\rho \nabla V_1) + \nabla \cdot \int_\Omega \rho(t,\vec{x}) \rho(t,\vec{x}') \nabla V_2(|\vec{x}-\vec{x}'|) d\vec{x}' && \text{in } (0,T)\times\Omega, \qquad (1.3)\\
0 &= \frac{\partial \rho}{\partial n} + \rho \frac{\partial V_1}{\partial n} + \int_\Omega \rho(t,\vec{x}) \rho(t,\vec{x}') \frac{\partial V_2(|\vec{x}-\vec{x}'|)}{\partial n} d\vec{x}' && \text{on } (0,T)\times\partial\Omega,\\
\rho(0,\vec{x}) &= \rho_0(\vec{x}) && \text{at } \{t=0\}\times\Omega,
\end{aligned}$$

where V_1 is an external potential and V_2 is a particle–particle interaction potential. The term $\frac{\partial}{\partial n}$ denotes a derivative in the direction of the outward unit normal. The boundary conditions are no-flux conditions, which ensure conservation of mass. This is relevant for many real-world applications in order to model processes in which particles cannot escape through the domain boundaries. Solving this DDFT model is numerically challenging, since it is a nonlinear integro-PDE model, in which the nonlocal term also enters the boundary conditions. This is the model we consider throughout this thesis, but extensions to it are made in later chapters to incorporate further physical effects, such as different particle types or volume exclusion effects.

1.2 PDE-Constrained Optimization

In this thesis we are interested in the optimal control of processes that can be described by such particle dynamics models. This requires tools from PDE-constrained optimization, which is an area of optimal control research, and is introduced in detail in Chapter 4. Here we simply introduce the problem of interest and outline some of the challenges associated to it. We consider the problem

$$\begin{aligned}
&\min_{\rho,\vec{w}} \mathcal{J}(\rho,\vec{w}) := \frac{1}{2}\|\rho-\widehat{\rho}\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2}\|\vec{w}\|^2_{L^2((0,T)\times\Omega)}\\
&\text{subject to}\\
&\frac{\partial \rho}{\partial t} = \nabla\cdot\left(\rho\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\right) - \nabla\cdot(\rho\vec{w}) && \text{in } (0,T)\times\Omega, \qquad (1.4)\\
&0 = \rho\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\cdot\vec{n} - \rho\vec{w}\cdot\vec{n} && \text{on } (0,T)\times\partial\Omega,\\
&\rho(0,\vec{x}) = \rho_0(\vec{x}) && \text{at } \{t=0\}\times\Omega.
\end{aligned}$$

The aim is to drive the *state* ρ, i.e., the particle density, to a prescribed *desired state* $\widehat{\rho}$, by varying the *control variable* $\vec{w}$. Here, we control the advective field that the particles are experiencing. The level of exerted control is determined by the *regularization parameter* β. Problem (1.4) is a *flow control* problem, in which the control is applied via an advective field in the PDE model. Additionally, in this thesis, *source control* problems are considered, in which a source term in the PDE model acts as control. Other parameters in the PDE could in principle serve as a control variable instead, depending on the application at hand. The PDE constraint is given by (1.4), which is (1.1) with an additional term involving $\vec{w}$. Since ρ is dictated by the solution to this model, we can change ρ by solving the model with different instances of the control $\vec{w}$. Here, the model includes no-flux boundary conditions, but problems with Dirichlet boundary conditions are also considered in this thesis.

Problems of the form (1.4) are often solved using the *(first-order) optimality system*, a set of necessary conditions that the solution to the optimization problem must satisfy. These conditions consist of the model (1.4), a second PDE for the *adjoint state*, and the *gradient equation*, an algebraic (in time) condition connecting the control, adjoint, and (possibly) state variables. This system of nonlinear, nonlocal PDEs and algebraic equations is coupled nonlocally in space and time. Therefore, a key question addressed in this thesis is how to efficiently solve such systems.

1.3 Outline of the Thesis

This thesis is structured as follows: In Chapter 2, we discuss how to model interacting particle systems. First, foundations in statistical mechanics are discussed, then equilibrium Density Functional Theory is introduced. The main part of Chapter 2 is concerned with the dynamic theory, its derivation, and some of its extensions. In Chapter 3, theoretical results and numerical methods for (integro-)PDEs are introduced. In particular, spectral methods are explained and the implementation of one such method in MATLAB is discussed in detail, including numerical demonstrations of solving (1.1). Chapter 4 introduces the field of PDE-constrained optimization. Again, theoretical aspects and numerical methods are discussed, before focusing on Krylov subspace methods, such as the conjugate gradient, MINRES, and GMRES methods, in view of their relevance to the work in this thesis.

In Chapter 5, a method for solving optimal control problems of the form (1.4) on simple (e.g., rectangular) domains is introduced. Different optimal control setups are discussed and associated optimality conditions are derived. Then, a novel numerical method for solving such problems is described and numerical solutions to some example problems are presented. The results contained in Chapter 5 are published in [1]. As outlined above, we aim to solve DDFT models and optimal control problems on complicated domains. This is addressed in Chapter 6, in which a novel spectral element method is introduced. The implementation is explained in detail and numerical examples are presented. Then, in Chapter 7, the first model generalization is introduced. The DDFT model is extended to describe mixtures of different species of particles, necessary for solving problems such as those illustrated in Figures 1.2 and 1.3. The DDFT model and corresponding optimal control problem are introduced, and optimality conditions are derived. Numerical solutions to such problems on complicated domains are presented. The results contained in Chapters 6 and 7 are summarized in the preprint [2].

A further extension to the DDFT model is discussed in Chapter 8. The DDFT model is extended to capture volume exclusion effects, so that sedimentation processes, such as in Figure 1.1, can be modelled more accurately. The corresponding optimal control problem is introduced and some example problems are presented. Chapter 9 provides a summary of the work in this thesis, highlighting the contributions to the scientific community, before discussing further possible directions of research.

Chapter 2

Modelling Particle Dynamics – Dynamic Density Functional Theory

As discussed in Chapter 1, we are interested in modelling interacting particles by considering the dynamics of a particle density ρ. This gives rise to a PDE model of the form (1.1), which depends on a free energy functional $\mathcal{F}$. This is an example of a Dynamic Density Functional Theory (DDFT) model.

What looks like a relatively straightforward formulation has in fact a rich background, which is the topic of this chapter. The roots of DDFT are found in statistical mechanics, which captures the thermodynamic properties of a system in equilibrium, such as energy, entropy, or temperature, in an empirical sense. From this, equilibrium Density Funtional Theory (DFT) was conceived, which describes the equilibrium properties of such a system in terms of a single unknown variable: the one-body particle density ρ. The free energy $\mathcal{F}$ of a system is now a functional of this ρ. This theory was extended to systems out of equilibrium, which introduces DDFT. DDFT leverages results from its static counterpart, and will be discussed in the coming sections. DDFT can be derived from microscopic particle models, such as Newton's equations of motion and overdamped Langevin dynamics, which can therefore be used to validate DDFT models.

In this chapter, the underlying concepts in statistical physics are introduced in Section 2.1, then Density Functional Theory is derived in Section 2.2, before Dynamic Density Functional Theory is introduced in Section 2.3, in which modelling assumptions, derivations, and extensions of DDFT are discussed. In Figure 2.1, the connections between different DDFT models and their derivations are illustrated.

2.1 Statistical Mechanics

In this section, some general statistical mechanics concepts are introduced, that are needed to explain and derive Density Functional Theory. These include the laws of thermodynamics, the definition of a Hamiltonian, and the concept of statistical ensembles. Most of this section is based on [3, 4, 5].

In this chapter, we describe different approaches for modelling systems of interacting particles. A model that describes the behaviour of each individual particle is called a *microscopic model* and is the subject of study in classical mechanics. Such a model describes a *microstate* of the system, i.e., the position and momentum of each particle in the system. An example of such a model is Newton's equations of motion, describing how the positions $\{\vec{x}_i\}$ and momenta $\{\vec{p}_i\}$ of N particles change under the influence of different forces, where $\{\vec{x}_i\}$ denotes the set

$\{\vec{x}_i\}_{i=1}^N$. This system has a phase-space of $6N$ dimensions. The total energy of such a system is described by the *Hamiltonian*

$$\mathcal{H}(\{\vec{x}_i\},\{\vec{p}_i\}) = \sum_{i=1}^{N} \frac{\vec{p}_i^{\,2}}{2\mathfrak{m}} + \sum_{i=1}^{N} V_1(\vec{x}_i) + \sum_{i<j} V_2(|\vec{x}_i - \vec{x}_j|) + ... \tag{2.1}$$

where the '...' denote higher than two-body interactions. The first term in $\mathcal{H}$ is the kinetic energy of the system, with particle mass $\mathfrak{m}$, while the subsequent terms describe the potential energy of the system, involving an external potential V_1, and a two-body interaction potential V_2, see [4]. Note that V_2 only depends on the distance between two particles and not on each position $\vec{x}$. Since the energy of a system is invariant under translation of the particle position, we can regard the particles as indistinguishable from one another. Moreover, we have $|\vec{x}_i - \vec{x}_j| = |\vec{x}_j - \vec{x}_i|$, for all $i, j = 1, ..., N$. We define $\mathbf{V}_1(\{\vec{x}_i\}) := \sum_{i=1}^N V_1(\vec{x}_i)$, $\mathbf{V}_2(\{\vec{x}_i\}) := \sum_{i<j} V_2(|\vec{x}_i - \vec{x}_j|)$, and omit higher order interaction terms in the following discussion for simplicity.

In real-world systems, the number of particles is often too large to solve the microscopic model, and the particle size too small to track them individually in experiments. Physical quantities often contain the order of 10^{23} particles per cubic metre. Since the dimension of the system of Newton's equations for N particles is $6N$, this becomes entirely unfeasible to solve. Moreover, in real-world applications it is often not of interest how each of the particles moves, but how a system behaves *on average*. Instead of investigating different realizations of the microscopic model – different microstates – we instead consider the *macrostate* of a system. This macrostate is an empirical description, or average, of all possible microstates of the system. Such a macrostate is called a *statistical ensemble* and is the subject of interest in statistical mechanics. A macrostate can be described by a few thermodynamic quantities, such as energy, temperature, pressure, volume, or particle number. Therefore, there exists a close link between thermodynamics and statistical mechanics. In particular, the fundamental laws of thermodynamics hold.

2.1.1 Laws of Thermodynamics

There are four laws of thermodynamics which are the underlying laws of statistical mechanics and consequently Density Functional Theory. They are provided for completeness and are discussed in, e.g., [6, Chapter 4] and [7, Chapter 1].

- Zeroth Law: If systems A and B are in thermodynamic equilibrium with each other and systems A and C are in thermodynamic equilibrium with each other, then so are systems B and C.
- First Law: The change in internal energy $\Delta E := E_2 - E_1$ from state E_1 to E_2 of a system depends on the total work done within the system, denoted by W, and the heat flux, Q, and not on the path taken to get from E_1 to E_2. This is

  $$\Delta E = Q - W.$$

- Second Law: The change of heat ΔQ in the system is proportional to the temperature T of the system and the change in entropy ΔS. This implies, for example, that in a closed system heat will be transferred from a hotter body to a colder body, and not the other way around. We have

  $$\Delta Q = T\Delta S,$$

  for reversible dynamics, with inequality $\Delta Q < T\Delta S$ for irreversible processes.
- Third Law: The minimum entropy of the system $S = 0$ is reached at $T = 0$.

2.1.2 Statistical Ensembles

In statistical mechanics, we can distinguish three types of systems: isolated, closed, and open systems. These are characterized by the thermodynamic quantities that remain constant within

the system. The empirical descriptions of such systems are called the micro-canonical, canonical, and grand-canonical ensembles, respectively.

In isolated systems, we fix the number of particles N, volume V, and energy E of the system, and no exchange with the surrounding is permitted. The statistical description of the system's state is called the *micro-canonical ensemble*. A practical issue with this description is that measurements of the system's energy are not readily obtained in experiments. Instead we can consider a closed system, in which we allow the energy to vary and in which we hold the temperature T constant instead. Here, energy is exchanged with what can be thought of as a heat bath. This surrounding bath can absorb any excess heat of (or supply heat to) the closed system, so that T can in fact remain constant. Here, the *canonical ensemble* describes the macrostate of this system. While this is a step in the right direction for many realistic systems, we may not have a way of measuring the exact number of particles N in a given system. The number of particles in physical systems is often too large and their size too small to do so, as discussed above. Therefore, we can consider an open system, in which not only a heat bath surrounds the system, but the system is also exchanging particles with its surrounding. The quantity that is now fixed is called the *chemical potential*, denoted by μ. This is a measure of how much energy is needed to add or remove a particle from the open system. If μ is a constant, the average number of particles in the system remains constant. The statistical description of this system is called the *grand-canonical ensemble*.

Discussions on statistical ensembles can be found in, e.g., [3, 8, 9]. In the following, we will discuss the canonical and grand-canonical ensemble, and omit micro-canonical considerations, since they are not as relevant for the derivation of DFT. We will follow [3] in this section, unless stated otherwise, but similar discussions can be found in, e.g., [4, 5].

The Canonical Ensemble

As mentioned, the energy E in the canonical ensemble is not fixed. Therefore, it can be informative to consider the probability that the system of N particles is in a particular energy state, or microscopic configuration. This probability is defined by the *Boltzmann distribution*

$$f_0(\{\vec{x}_i\},\{\vec{p}_i\}) = \frac{1}{N!h^{3N}} Z_N^{-1} \exp\left(-\beta \mathcal{H}(\{\vec{x}_i\},\{\vec{p}_i\})\right),$$

where $\mathcal{H}$ is given by (2.1), h is the Planck constant, $\beta = \frac{1}{k_B T}$, T a constant temperature, and k_B the Boltzmann constant. The factor Z_N ensures that f_0 is normalized to one. It is called the partition function and is defined by

$$Z_N = \frac{1}{N!h^{3N}} \int\int \exp\left(-\beta \mathcal{H}(\{\vec{x}_i\},\{\vec{p}_i\})\right) d\{\vec{x}_i\} d\{\vec{p}_i\},$$

where $d\{\vec{x}_i\}$ denotes $d\vec{x}_1 d\vec{x}_2 ... d\vec{x}_N$. Noticing that only the kinetic energy depends on momenta $\{\vec{p}_i\}$, Z_N can be simplified, by using a Gaussian integral, to

$$Z_N = \frac{\Lambda^{-3N}}{N!} \int \exp\Big(-\beta \left(\mathbf{V}_1(\{\vec{x}_i\}) + \mathbf{V}_2(\{\vec{x}_i\})\right)\Big) d\{\vec{x}_i\},$$

where Λ is the thermal de Broglie wavelength (wavelength of a particle) defined as $\Lambda = \sqrt{\frac{h^2\beta}{2\pi \mathfrak{m}}}$, with mass $\mathfrak{m}$. Note that the computation of Z_N is prohibitively expensive, due to the 'curse of dimensionality': since the dimension of Z_N is $3N$ for N particles, and for m computational points in each dimension, the number of computation points would be m^{3N}, which is unfeasible for any reasonable computational resolution and number of particles.

The Helmholtz free energy of the system can be defined in terms of the partition function as

$$F = -k_B T \ln\left(Z_N\right).$$

In terms of the thermodynamic quantities, we can define $F = E - TS$, where $E = -\frac{\partial}{\partial \beta} \ln(Z_N)$, and $S = \frac{\partial F}{\partial T}$, see [3].

The grand-canonical ensemble

In the grand-canonical potential, neither the energy nor the particle number is fixed. Therefore, the probability of the system being in a particular state needs to be extended to take the variable number of particles into account. Therefore, the probability function for the grand-canonical ensemble is

$$f_0(\{\vec{x}_i\}, \{\vec{p}_i\}) = \Xi^{-1} \exp\Big(-\beta\left(\mathcal{H}(\{\vec{x}_i\}, \{\vec{p}_i\}) - \mu N\right)\Big), \tag{2.2}$$

where Ξ is the grand-canonical partition function

$$\Xi = \sum_{N=0}^{\infty} \frac{1}{N! h^{3N}} \int\int \exp\Big(-\beta\left(\mathcal{H}(\{\vec{x}_i\}, \{\vec{p}_i\}) - \mu N\right)\Big) d\{\vec{x}_i\} d\{\vec{p}_i\}.$$

This average taken over the number of particles is used in the derivation of DFT, and is therefore denoted by

$$\operatorname{Tr} f = \sum_{N=0}^{\infty} \frac{1}{N! h^{3N}} \int\int f(\{\vec{x}_i\}, \{\vec{p}_i\}) d\{\vec{x}_i\} d\{\vec{p}_i\}.$$

Note that we have $\operatorname{Tr} f_0 = 1$, since f_0 is normalized by Ξ, and hence f_0 satisfies the definition of a probability distribution.

Instead of the Helmholtz free energy, in this ensemble we are interested in the grand potential energy defined as

$$\Omega_0 = -k_B T \ln(\Xi), \tag{2.3}$$

which relates to the Helmholtz free energy by the relation $\Omega_0 = F - \mu N$.

2.2 Equilibrium Density Functional Theory (DFT)

Equipped with these basics of statistical mechanics, we can begin to discuss Density Functional Theory. The key idea is that the free energy of a thermodynamic system in equilibrium is uniquely determined by the equilibrium one-body particle density, denoted by ρ. This implies that knowing the probability of finding a single particle at a position $\vec{x}$ in the system is enough to describe the intrinsic properties of said system. This is a rather remarkable result, reducing the dimension of the considered problem from $3N$ (or even $6N$ when momentum is considered), as seen in the previous section, to only three. This idea originates from the famous quantum mechanical equivalent, which was published in the 60s, see [10, 11]. It was first applied to classical fluids in the mid-70s [12, 13]. Some standard references for the content of this section are [14] and [15].

2.2.1 Derivation of DFT

The derivation of Density Functional Theory for fluids presented in the following is based on [3, 4, 16]. The starting point for the derivation is the functional

$$\Omega[f] = \operatorname{Tr}\left[f(\mathcal{H} - \mu N + k_B T \ln f)\right], \tag{2.4}$$

defined in [17]. Here, f is a probability distribution, $\mathcal{H}$ is defined by (2.1), μ is the chemical potential, and $k_B T f \ln f$ is the entropy term, relating to the ideal gas contribution in the free energy. If we have $f = f_0$, as defined in (2.2), then $\Omega[f_0]$ is the grand potential energy for the equilibrium system, defined in (2.3). It can furthermore be shown that f_0 is the global

minimizer for Ω.

We can define the ensemble average of some operator $\mathcal{O}$ as

$$\langle \mathcal{O} \rangle = \operatorname{Tr} f_0 \mathcal{O}, \tag{2.5}$$

using the equilibrium distribution f_0. If we define the density operator as

$$\hat{\rho}(\vec{x}) = \sum_{i=1}^{N} \delta(\vec{x} - \vec{x}_i),$$

then the average density of particles is

$$\rho(\vec{x}) = \langle \hat{\rho}(\vec{x}) \rangle,$$

and we denote the equilibrium density as $\rho_0(\vec{x})$. The notation $\hat{\rho}$ in this chapter should not be confused with $\widehat{\rho}$, the desired state in optimal control problems such as (1.4).

The key step in deriving DFT is establishing a relationship between the equilibrium N-body probability f_0 and the one-body density ρ_0. One can show that: (i) f_0 is a functional of V_1, (ii) V_1 is uniquely determined by $\rho_0(\vec{x})$, see [17]. Moreover, each positive choice of density $\rho_0(\vec{x})$ gives rise to a system influenced by an external potential V_1, see [18]. This results in the probability function (2.2) being a functional of the particle density ρ, denoted by $f_0[\rho_0]$. Then, since Ω is a functional of f, we have that $\Omega[f_0] = \Omega[\rho_0] := \Omega_0$.

In particular, an equilibrium one-body density ρ_0 minimizes Ω, so that for any other density ρ, in a system under the influence of the same V_1, we have the condition

$$\Omega[\rho] \geq \Omega[\rho_0].$$

Note that uniqueness of the minimizer ρ_0 is not given, since a system can have coexisting phases, see, e.g., [19]. Moreover, if ρ_0 is in fact unique, approximations to Ω and computational inaccuracies can introduce errors, so that the approximation to the one-body density may not unique when it should be.

Combining these insights with (2.4), we have

$$\begin{aligned}
\Omega[\rho] &= \operatorname{Tr}\left[f(\mathcal{H} - \mu N + k_B T \ln f)\right] \\
&= \operatorname{Tr}\left[f(\mathcal{H} + k_B T \ln f)\right] - \langle \mu N \rangle \\
&= \underbrace{\operatorname{Tr}\left[f(\mathcal{H} + k_B T \ln f)\right]}_{=:\mathcal{F}[\rho]} - \mu \int \rho(\vec{x}) d\vec{x},
\end{aligned}$$

where $\mathcal{F}$ is the Helmholtz free energy of the system. Note that the trace operator (2.5) takes an average over all position, momenta, and number of particles in the system. We can split $\mathcal{F}$ into three parts; the ideal gas contribution, the external contribution, and the excess free energy, i.e.,

$$\mathcal{F} = \mathcal{F}_{\text{id}} + \mathcal{F}_{\text{ext}} + \mathcal{F}_{\text{exc}}.$$

The ideal gas contribution is of the form

$$\mathcal{F}_{\text{id}}[\rho] = k_B T \int \rho(\vec{x}) \left(\ln \Lambda^3 \rho(\vec{x}) - 1\right) d\vec{x}, \tag{2.6}$$

the external contribution is

$$\mathcal{F}_{\text{ext}}[\rho] = k_B T \int \rho(\vec{x}) V_1 d\vec{x},$$

while the excess free energy $\mathcal{F}_{\text{exc}}$ is generally not known and needs to be approximated. Finally, knowing that ρ_0 minimizes the grand potential, this can be expressed by the variational principle (Euler–Lagrange equation)

$$\left.\frac{\delta\Omega}{\delta\rho(\vec{x})}\right|_{\rho=\rho_0} = 0.$$

Note that variational principles and Euler–Lagrange equations will be discussed in a more theoretical setting in Chapter 3. We can use this principle to find

$$\left.\frac{\delta\Omega}{\delta\rho(\vec{x})}\right|_{\rho=\rho_0} = k_B T \ln \Lambda^3 \rho(\vec{x}) + \frac{\delta\mathcal{F}_{\text{exc}}}{\delta\rho} + V_1(\vec{x}) - \mu\Big|_{\rho=\rho_0} = 0. \tag{2.7}$$

Furthermore, we can split up μ into an ideal gas and excess contribution so that we have $\mu = \mu_{\text{id}} + \mu_{\text{exc}} = k_B T \ln \Lambda^3 \rho_{\text{bulk}} + \mu_{\text{exc}}$, with ρ_{bulk} a constant density. This can be found by computing the minimization for an ideal gas, i.e., with $\mathcal{F}_{\text{exc}} = 0$ and $V_1 = 0$. Equation (2.7) can be rearranged to get

$$\rho(\vec{x}) = \rho_{\text{bulk}} \exp\left(-\beta\left(\frac{\delta\mathcal{F}_{\text{exc}}}{\delta\rho} + V_1(\vec{x}) - \mu_{\text{exc}}\right)\right).$$

While both sides of this equation depend on ρ, and $\mathcal{F}_{\text{exc}}$ is not known exactly, for a given approximate excess free energy $\mathcal{F}_{\text{exc}}$, this equation can be used to iteratively solve for ρ_0, the equilibrium density, as described in [20]. For an ideal gas with $V_1 \neq 0$, the exact solution is given by

$$\rho(\vec{x}) = \rho_{\text{bulk}} \exp\big(-\beta V_1(\vec{x})\big).$$

2.2.2 Correlation Functions and the Ornstein–Zernicke Equation

Following [4], we can define several quantities in terms of $\rho(\vec{x})$. Given an excess free energy $\mathcal{F}_{\text{exc}}$, a hierarchy of direct correlation functions can be defined, which describe the correlation between N particles in the system in terms of the densities at positions $\vec{x}_1, \vec{x}_2, ..., \vec{x}_N$ and $\mathcal{F}_{\text{exc}}$. The N-th order correlation functions can be defined by

$$c^{(N)}(\vec{x}_1, ..., \vec{x}_N) = -\frac{1}{k_B T} \frac{\delta^n \mathcal{F}_{\text{exc}}[\rho]}{\delta\rho(\vec{x}_1)...\delta\rho(\vec{x}_N)}.$$

One way of determining one of the most relevant direct correlation function, i.e., the one for $N = 2$, is via the Ornstein–Zernicke equation. It is defined as

$$h^{(2)}(\vec{x}_1, \vec{x}_2) = c^{(2)}(\vec{x}_1, \vec{x}_2) + \int c^{(2)}(\vec{x}_1, \vec{x}_3)\rho(\vec{x}_3)h^{(2)}(\vec{x}_3, \vec{x}_2)d\vec{x}_3,$$

where

$$h^{(2)}(\vec{x}_1, \vec{x}_2) = \frac{\rho^{(2)}(\vec{x}_1, \vec{x}_2)}{\rho(\vec{x}_1)\rho(\vec{x}_2)} - 1,$$

is the total pair correlation. Using these definitions, the Ornstein–Zernike equation can be rewritten to give [14, 21]

$$\rho^{(2)}(\vec{x}_1, \vec{x}_2) = (1 + c^{(2)}(\vec{x}_1, \vec{x}_2))\rho(\vec{x}_1)\rho(\vec{x}_2) + \rho(\vec{x}_2) \int \left((\rho^{(2)}(\vec{x}_1, \vec{x}_3) - \rho(\vec{x}_1)\rho(\vec{x}_3))c^{(2)}(\vec{x}_3, \vec{x}_2)\right),$$

which relates the second-order correlation function $c^{(2)}$ to the pair correlation $\rho^{(2)}$. Then, if we know or can approximate the total correlation $c^{(2)}$, we can find the pair correlation function $\rho^{(2)}$, demonstrating that knowledge of $\mathcal{F}_{\text{exc}}$ and $\rho^{(2)}$ provide equivalent information at the level $n = 2$. Some recent work that utilize this method can be found in [22, 23, 24].

2.2.3 Approximations to the Excess Free Energy

As mentioned above, the excess free energy $\mathcal{F}_{\text{exc}}$ is generally not known exactly and needs to be approximated. There are many different approximations to $\mathcal{F}_{\text{exc}}$, depending on the application at hand, which capture the particle interactions in a target system. One of the most commonly used approximations is Fundamental Measure Theory (FMT), which describes hard particle mixtures. This theory is based on the only known exact form of $\mathcal{F}_{\text{exc}}$, which is the excess free energy for a system of hard rods in one dimension.

Exact Result in 1D

The exact solution for hard-rods in one dimension has first been formulated by Percus in 1976 [25], and has later been extended to n_s species of particles [26]. Following the FMT framework introduced by Rosenfeld [27], the excess free energy can be expressed as

$$\mathcal{F}_{\text{exc}}^{\text{hr}}[\{\rho_i\}] = \beta \int \Phi(\{n_\alpha\})dx, \tag{2.8}$$

where Φ is the excess free energy density. The weighted densities n_α are defined by

$$n_\alpha(x) = \sum_{i=1}^{n_s} \int \rho_i(x')w_i^{(\alpha)}(x - x')dx',$$

with weights

$$w_i^{(0)}(x) = \frac{1}{2}\left(\delta(x - R_i) + \delta(r + R_i)\right),$$

and

$$w_i^{(1)}(x) = \Theta(R_i - |x|),$$

where R_i is the radius of the excluded volume, δ is the Dirac delta function, and Θ is the Heaviside step function. The weight functions correspond to the geometry of the one-dimensional particles of species i, with $w_i^{(0)}$ representing a surface measure, and $w_i^{(1)}$ describing the volume. These quantities can be combined to give the exact function

$$\Phi(\{n_\alpha\}) = -n_0 \ln(1 - n_1)$$

in (2.8).

Fundamental Measure Theory

Fundamental Measure Theory (FMT) [27] is a Density Functional Theory for hard sphere mixtures. The basis of this theory is that the excess free energy functional is of the form

$$\beta F_{exc}[\rho_i] = \int \Phi(\{n_\alpha(\vec{x}')\})d\vec{x}',$$

compare with (2.8) in one dimension. Here, i is the species count and Φ is a function of the weighted densities n_α. By now, many versions of Φ have been developed, yielding approximations of $\mathcal{F}_{exc}$ with different limitations, see [20]. Rosenfeld's original version [27] is defined

as

$$\Phi = -n_0 \ln(1-n_3) + \frac{n_1 n_2 - \vec{n}_1 \cdot \vec{n}_2}{1-n_3} + \frac{n_2^3 - 3 n_2 \vec{n}_2 \cdot \vec{n}_2}{24\pi(1-n_3)^2}.$$

The weighted densities for n_s species are

$$n_\alpha(\vec{x}) = \sum_{i=1}^{n_s} \int \rho_i(\vec{x}') \omega_i^\alpha(\vec{x} - \vec{x}') d\vec{x}', \tag{2.9}$$

with weight functions given by

$$\omega_i^3(\vec{x}) = \Theta(R_i - \|\vec{x}\|), \quad \omega_i^2(\vec{x}) = \delta(R_i - \|\vec{x}\|), \quad \vec{\omega}_i^2(\vec{x}) = \frac{\vec{x}}{\|\vec{x}\|}\delta(R_i - \|\vec{x}\|),$$
$$\omega_i^1(\vec{x}) = \omega_i^2(\vec{x})/(4\pi R_i), \quad \omega_i^0(\vec{x}) = \omega_i^2(\vec{x})/(4\pi R_i^2), \quad \vec{\omega}_i^1(\vec{x}) = \vec{\omega}_i^2(\vec{x})/(4\pi R_i),$$

where R_i is the radius of the excluded volume, Θ is the Heaviside function, and δ is the delta function. Integrating over ω_α, with $\alpha = 0, 1, 2, 3$, we obtain the fundamental measures of a sphere: volume, surface area, radius and the Euler characteristic [27, 20].

Based on this theory for three-dimensional spheres and the fact that the theory for hard rods is known exactly [25], Rosenfeld derived a version of this approach for two-dimensional hard disks [28]. However, some additional approximations have to be made when choosing the weighted densities, which is not necessary in one and three dimensions. The resulting equation is

$$\Phi = -n_0 \ln(1-n_3) + \frac{1}{4\pi}\frac{n_2 n_2}{1-n_3} + \frac{1}{4\pi}\frac{\vec{n_2} \cdot \vec{n_2}}{1-n_3}.$$

In the uniform limit, for one particle species, we get that

$$n_0 = \rho, \quad n_2 = 2\pi R\rho, \quad n_3 = \pi R^2 \rho,$$

by solving the integrals in (2.9), using spherical polar coordinates, with $\rho = \rho_{\text{bulk}}$ a constant.

Substituting this in the 2D version of Φ gives

$$\Phi = -\rho \ln(1 - \pi R^2 \rho) + \frac{1}{4\pi}\frac{4\pi^2 R^2 \rho^2}{1 - \pi R^2 \rho} + \frac{1}{4\pi}\frac{\vec{n}_2 \cdot \vec{n}_2}{1 - \pi R^2 \rho},$$

where $\vec{n}_2 = \vec{0}$ in the uniform limit, since the corresponding equation in (2.9) is an integral over an odd function. Noting that $R = \sigma/2$ and $\eta = \pi\sigma^2\rho/4$, we get that

$$\Phi = -\rho \ln(1-\eta) + \frac{\rho\eta}{1-\eta} = \rho\left(-\ln(1-\eta) + \frac{1}{1-\eta} - 1\right). \tag{2.10}$$

This expression for the free energy for the bulk fluid is the same as derived by scaled particle theory (SPT) [29, 30, 31], which also coincides with the Percus–Yevic compressibility equation [32], as detailed in [33]. While the SPT approximation (2.10) and its three-dimensional equivalent are used in classical DFT, see [34, 35, 36, 37, 38, 39], and other statistical mechanics approaches, see [40, 41, 42, 43], in dynamical DFT it is not commonly applied and only the work of Archer et al. [44, 45, 46, 47] is known to us in this context.

Mean Field Approximation

Originating from a truncated functional Taylor expansion of $\mathcal{F}_{\text{exc}}$, the Ramakrishnan–Youssouff approximation [48] is

$$\mathcal{F}_{\text{exc}}[\rho] = \frac{k_B T}{2} \int\int c^{(2)}(\vec{x} - \vec{x}') \Delta\rho(\vec{x}) \Delta\rho(\vec{x}') d\vec{x} d\vec{x}',$$

where $\Delta\rho = \rho(\vec{x}) - \rho_0$, ρ_0 a homogeneous reference density. Choosing the random phase approximation

$$c^{(2)}(\vec{x} - \vec{x}') = \frac{V_2(|\vec{x} - \vec{x}'|)}{k_B T}$$

and replacing $\Delta\rho$ by ρ, results in the *mean field approximation*

$$\mathcal{F}_{\text{exc}}[\rho] = \frac{1}{2} \int \int \rho(\vec{x})\rho(\vec{x}')V_2(|\vec{x} - \vec{x}'|)d\vec{x}d\vec{x}'. \tag{2.11}$$

This approximation stipulates that $\rho^{(2)}(\vec{x}, \vec{x}') \approx \rho(\vec{x})\rho(\vec{x}')$, which is an assumption that the particle densities are independent of each other. This is a good approximation for soft particles in high densities and is used frequently as an approximation to $\mathcal{F}_{\text{exc}}$, see [5]. Moreover, it can be used in combination with other approximations of $\mathcal{F}_{\text{exc}}$, such as FMT, see [49], or other volume exclusion approximations, see [44].

2.3 Dynamic Density Functional Theory (DDFT)

The previous sections provided an overview of statistical mechanics concepts, and static Density Functional Theory. This theory was extended in 1979 by Evans [14] to describe out-of-equilibrium situations for thermodynamic systems. The general description of deterministic DDFT is of the form

$$\frac{\partial\rho(t, \vec{x})}{\partial t} = \Gamma\nabla \cdot \left(\rho(t, \vec{x})\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\right), \tag{2.12}$$

where $\mathcal{F}$ is the equilibrium free energy and Γ is a constant mobility. Furthermore, $\Gamma > 0$, since $\Gamma = \frac{1}{\mathfrak{m}\gamma}$, where γ is the (necessarily positive) friction coefficient and $\mathfrak{m}$ is the particle mass. The free energy functional is, as in DFT, made up from several components

$$\mathcal{F}[\rho] = \mathcal{F}_{\text{id}} + \mathcal{F}_{\text{exc}} + \int V_1\rho(\vec{x})d\vec{x},$$

where the ideal gas contribution $\mathcal{F}_{\text{id}}$ is given by (2.6) and $\mathcal{F}_{\text{exc}}$ needs to be approximated, as discussed in Section 2.2.3. If we choose the mean field approximation (2.11), $\mathcal{F}$ is given by

$$\mathcal{F}[\rho] = \int \left(k_B T\rho(\vec{x})(\ln\Lambda^3\rho(\vec{x}) - 1) + V_1\rho(\vec{x}) + \frac{1}{2}\int \rho(\vec{x})\rho(\vec{x}')V_2(|\vec{x} - \vec{x}'|)d\vec{x}'\right)d\vec{x}. \tag{2.13}$$

2.3.1 Derivation of an Integro-PDE Model from the DDFT Definition

In order to rewrite (2.12) with free energy (2.13) into an explicit PDE model, we first need to take the functional derivative of (2.13). While the first two terms are standard, the two-body interaction term needs to be treated with care. The derivative is

$$\begin{aligned}\frac{\delta}{\delta\rho}\left(\frac{1}{2}\int\int \rho(\vec{x})\rho(\vec{x}')V_2(|\vec{x} - \vec{x}'|)d\vec{x}'d\vec{x}\right) = &\frac{1}{2}\int\int \rho(\vec{x}')V_2(|\vec{x} - \vec{x}'|)d\vec{x}'d\vec{x} \\ &+ \frac{1}{2}\int\int \rho(\vec{x})V_2(|\vec{x} - \vec{x}'|)d\vec{x}'d\vec{x},\end{aligned}$$

where the product rule was applied. Now since the particles are indistinguishable, we can relabel $\vec{x} \to \vec{x}'$ and $\vec{x}' \to \vec{x}$ in the second term, and the derivative becomes

$$\begin{aligned}\frac{\delta}{\delta\rho}\left(\frac{1}{2}\int\int \rho(\vec{x})\rho(\vec{x}')V_2(|\vec{x} - \vec{x}'|)d\vec{x}'d\vec{x}\right) = &\frac{1}{2}\int\int \rho(\vec{x}')V_2(|\vec{x} - \vec{x}'|)d\vec{x}'d\vec{x} \\ &+ \frac{1}{2}\int\int \rho(\vec{x}')V_2(|\vec{x}' - \vec{x}|)d\vec{x}d\vec{x}'.\end{aligned}$$

Now, since V_2 is symmetric, the term becomes

$$\frac{\delta}{\delta\rho}\left(\frac{1}{2}\int\int\rho(\vec{x})\rho(\vec{x}')V_2(|\vec{x}-\vec{x}'|)d\vec{x}'d\vec{x}\right)=\int\int\rho(\vec{x}')V_2(|\vec{x}-\vec{x}'|)d\vec{x}'d\vec{x}.$$

Therefore, the functional derivative of $\mathcal{F}$ is

$$\frac{\delta\mathcal{F}[\rho]}{\delta\rho}=k_BT\ln\Lambda^3+k_BT\ln(\rho(\vec{x}))+V_1+\int\rho(\vec{x})V_2(|\vec{x}-\vec{x}'|)d\vec{x}'.$$

Then, the gradient is taken, giving

$$\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}=k_BT\nabla\ln(\rho(\vec{x}))+\nabla V_1+\nabla\int\rho(\vec{x})V_2(|\vec{x}-\vec{x}'|)d\vec{x}'.$$

Multiplying by ρ and applying the relation $\rho\nabla\ln(\rho(\vec{x}))=\nabla\rho$, we obtain

$$\rho\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}=k_BT\nabla\rho(\vec{x})+\rho(\vec{x})\nabla V_1+\rho(\vec{x})\nabla\int\rho(\vec{x}')V_2(|\vec{x}-\vec{x}'|)d\vec{x}'.$$

Finally, the divergence operator is applied and the DDFT model (2.12) becomes

$$\frac{\partial\rho(t,\vec{x})}{\partial t}=\Gamma\left(k_BT\nabla^2\rho(t,\vec{x})+\nabla\cdot(\rho(t,\vec{x})\nabla V_1)+\nabla\cdot\int\rho(t,\vec{x})\rho(t,\vec{x}')\nabla_{\vec{x}}V_2(|\vec{x}-\vec{x}'|)d\vec{x}'\right).$$

This is the DDFT model that we will encounter throughout this thesis. Note that parameters $k_BT=1$ and $\Gamma=1$ in later chapters, since by rescaling V_1, V_2, and t we can obtain this choice. The explicit parameter-dependence can be reintroduced without much additional work when necessary.

2.3.2 The Decrease in Free Energy

The free energy of a thermodynamic system decreases monotonically in time, which means that the system is moving towards its equilibrium state. This result can be shown for the classic DDFT (2.12) as discussed in [50, 51]. In order to show this result we require that

$$\frac{\partial\mathcal{F}[\rho]}{\partial t}<0.$$

We apply the chain rule for functional derivatives to get

$$\frac{\partial\mathcal{F}[\rho]}{\partial t}=\int\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\frac{\partial\rho}{\partial t}d\vec{x}=\Gamma\int\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\nabla\cdot\left(\rho(t,\vec{x})\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\right)d\vec{x},$$

using the definition for $\frac{\partial\rho}{\partial t}$ in Equation (2.12). Integrating by parts and letting the boundary terms go to zero results in

$$\begin{aligned}\frac{\partial\mathcal{F}[\rho]}{\partial t}&=-\Gamma\int\rho(t,\vec{x})\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\cdot\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}d\vec{x}\\&=-\Gamma\int\rho(t,\vec{x})\left|\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\right|^2d\vec{x}<0,\end{aligned}$$

given that $\rho>0$, since it is a density.

2.3.3 Assumptions and Approximations in DDFT

There are some important assumptions made in standard DDFT, which are highlighted below and some of which are used in deriving DDFT in later sections. As in DFT, it can be shown that the system of interest can be described by a one-body particle density ρ, as opposed to an N-body density, denoted by $f^{(N)}$. For time-dependent systems this has first been established

by Chan and Finken in 2005 [52], who showed that an invertible one-to-one map exists between the time-dependent densities ρ and $f^{(N)}$, given a (time-dependent) external potential V_1. Note, however, that this ρ now depends on the chosen initial configuration $\rho(0, \vec{x}) = \rho_0(\vec{x})$, which is in contrast to equilibrium DFT. Furthermore, equilibrium DFT is set in the grand-canonical ensemble, in which the chemical potential, μ, is held constant, which corresponds to a constant average number of particles. However, DDFT is defined in the canonical ensemble, in which the exact (not average) number of particles is fixed. This arises from the derivation of DDFT from microscopic equations, for which an exact number of particles is defined, see Section 2.3.4. For many systems this difference is negligible, since the two ensembles are equal in the thermodynamic limit. However, for systems with a small number of particles the choice of ensemble can become relevant. This issue is addressed in [53], where a framework for particle conserving dynamics is developed, that can accurately describe small systems in the canonical ensemble.

Local Equilibrium Approximation and High Friction Limit

A further assumption in standard DDFT is that the momentum $\vec{p}$ relaxes quickly, in comparison to the position $\vec{x}$, to an equilibrium distribution, denoted by $g(\vec{p})$. In this case, the density $f^{(1)}(\vec{x}, \vec{p}) = \rho(\vec{x})g(\vec{p})$, with $g(\vec{p})$ known, and inertial effects can be neglected. Often, this momentum distribution is assumed to be a Gaussian, with mean value of the solvent flow and variance dependent on the temperature of the system. Note that if the surrounding solvent is stationary, the mean velocity is zero.

For many systems, where the colloid particles experience high friction, this is a reasonable assumption, since the solvent causes the momenta of the colloid particles to converge to equilibrium. Here, friction forces are dominant in the system, dampening the effects of external forces and interactive forces. Such dynamics are called overdamped dynamics and are modelled by DDFT of the form (2.12). There exist extensions to DDFT, which capture inertial effects [54], as illustrated in Section 2.3.5. However, even in this extension, the so-called local equilibrium approximation is made, prescribing that locally the velocity varies slowly with respect to the average local velocity of particles.

The Adiabatic Approximation

The key approximation made in DDFT is called the *adiabatic approximation*. This assumes that the particle interactions in the dynamic system can be approximated by those in the equilibrium system. The particle flux j can then be written in terms of the equilibrium free energy $\mathcal{F}[\rho(\vec{x})]$ via $\mathrm{j} = \rho\nabla\frac{\delta\mathcal{F}}{\delta\rho}$. This is a strong approximation for several reasons. One reason is that DFT is a theory set in the grand-canonical ensemble, while DDFT is defined in the canonical ensemble. The dynamical particle interactions, which are set in the canonical framework are approximated by terms involving the grand-canonical excess free energy $\mathcal{F}_{\mathrm{exc}}[\rho(\vec{x})]$. Moreover, since $\mathcal{F}[\rho(\vec{x})]$ is defined in equilibrium, it is only guaranteed to be a good approximation to the corresponding dynamical term, which we denote by $\mathcal{F}_{\mathrm{dyn}}[\rho(t, \vec{x})]$, when the density ρ varies (infinitely) slowly in time. The following argument is taken from [21]. Consider an equation of the form

$$\frac{\partial\rho(t,\vec{x})}{\partial t} = \Gamma\nabla\cdot\left(k_BT\nabla\rho(t,\vec{x}) + \rho(t,\vec{x})\nabla V_1(\vec{x}) + \int\rho^{(2)}(t,\vec{x},\vec{x}')\nabla V_2(\vec{x},\vec{x}')d\vec{x}' + ...\right),$$

where '...' again denotes higher order interaction terms. In equilibrium DFT, an equilibrium profile $\rho_0(\vec{x})$ is uniquely determined by an external potential Ψ_{ext}. It is therefore possible to find a second potential $u_{\mathrm{ext}}(\vec{x})$, such that at a given time t_i, it holds that

$$\Psi_{\mathrm{ext}}(t_i, \vec{x}) = V_1(t_i, \vec{x}) + u_{\mathrm{ext}}(t_i, \vec{x})$$

gives rise to an equilibrium density $\rho_{t_i}(\vec{x})$.

In equilibrium at t_i, from the variational principle of DFT it is known that

$$\frac{\delta \mathcal{F}[\rho]}{\delta \rho(t_i, \vec{x})} = k_B T \ln\left(\Lambda^3 \rho(t_i, \vec{x})\right) + \Psi_{\text{ext}}(t_i, \vec{x}) + \frac{\delta \mathcal{F}_{exc}[\rho]}{\delta \rho(t_i, \vec{x})} = \mu,$$

where μ is the constant chemical potential, and taking gradients results in a generalized force balance equation [14]. Multiplying by ρ_{t_i} and rearranging gives

$$k_B T \nabla \rho_{t_i}(\vec{x}) + \rho_{t_i}(\vec{x}) \nabla V_1(\vec{x}) + \rho_{t_i}(\vec{x}) \nabla u_{\text{ext}} + \rho_{t_i}(\vec{x}) \nabla \frac{\delta \mathcal{F}_{\text{exc}}[\rho_{t_i}(\vec{x})]}{\delta \rho_{t_i}(\vec{x})} = 0.$$

Replacing the parametric variable t_i by the variable t and the equilibrium μ by its nonequilibrium counterpart, $\mu(t, \vec{x})$, the following relation holds

$$\nabla u_{\text{ext}}(t, \vec{x}) = -\nabla \frac{\delta \mathcal{F}_{\text{dyn}}[\rho(t, \vec{x})]}{\delta \rho(t, \vec{x})} = -\nabla \mu(t, \vec{x}).$$

This equation describes the driving force of the system $\nabla \mu(t, \vec{x})$, the gradient of the non-equilibrium chemical potential, to be related to the additional external potential u_{ext}. This external force ∇u_{ext} would be needed to drive the given system back into equilibrium. Here, $\mathcal{F}_{\text{dyn}}$ denotes

$$\mathcal{F}_{\text{dyn}}[\rho] = \int \left[k_B T \rho(t, \vec{x}) \left(\ln(\Lambda^3 \rho(t, \vec{x})) - 1 \right) + \rho(t, \vec{x}) V_1(t, \vec{x}) \right] d\vec{x} + \mathcal{F}_{\text{exc}}[\rho].$$

However, we approximate the non-equilibrium excess free energy by $\mathcal{F}_{\text{exc}}$, the equilibrium excess free energy, so that

$$\int \rho^{(2)}(t, \vec{x}, \vec{x}') \nabla V_{1,2}(\vec{x}, \vec{x}') d\vec{x}' + \ldots = \rho(t, \vec{x}) \nabla \frac{\delta \mathcal{F}_{\text{exc}}[\rho(t, \vec{x})]}{\delta \rho(t, \vec{x})}.$$

This recovers the standard DDFT (2.12).

2.3.4 Derivations of DDFT

In this section, a few different ways of deriving the standard DDFT (2.12) are presented. In Sections 2.3.5 and 2.3.6 additional derivations are presented, which lead to related theories. In Figure 2.1, a flow chart illustrates how the different derivations are connected. In this section, phenomenological approaches are discussed and, after explaining the derivation of the Langevin equation from Hamilton's equations for the colloid and bath particles, two derivations of DDFT are presented; one which starts from a Langevin equation and another which considers the Smoluchowski equation as a starting point. A third approach, which is omitted here, uses projection operator methods to derive DDFT, and can be found in [5].

Phenomenological Considerations

The first person to derive standard DDFT by phenomenological means was Evans in 1979 [14]. He proposed the extension of DFT to dynamic situations, based on known principles in fluid mechanics. One of the two key ideas was to observe that a particle density ρ, with constant mass, will satisfy a continuity equation of the form

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathrm{j}.$$

The second idea was to relate the current j to the chemical potential μ of the system by a generalized Fick's law

$$\mathrm{j} = -\Gamma \rho(t, \vec{x}) \nabla \mu(t, \vec{x}). \tag{2.14}$$

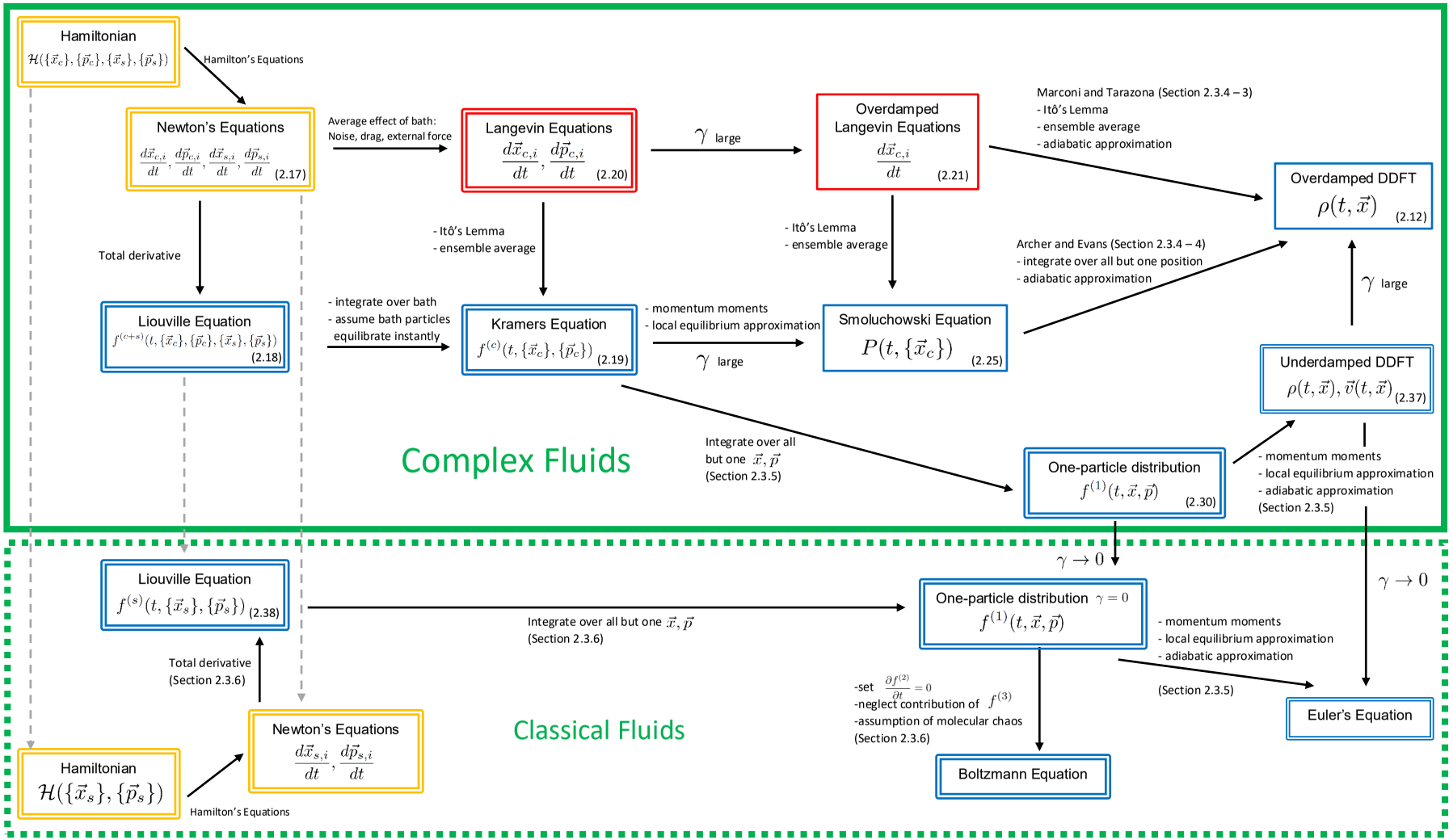

Figure 2.1: Flow chart of the derivations of DDFT and related theories discussed in Sections 2.3.4, 2.3.5, and 2.3.6. Each equation is contained in a rectangle, with relevant equation numbers given. Colours symbolize Hamiltonians/ODEs (yellow), SDEs (red), and PDEs (blue). Single lined rectangles indicate dependence on position coordinates only, while double lined rectangles indicate dependence on position and momentum.

In equilibrium DFT we have the relation

$$\mu = \frac{\delta \mathcal{F}[\rho]}{\delta \rho}. \tag{2.15}$$

Making the adiabatic approximation, and therefore substituting (2.15) into (2.14), the standard DDFT equation (2.12) is derived.

These are phenomenological observations which initially justified the use of DDFT. However, in the following sections, we derive DDFT from the underlying microscopic dynamics. First, we show how to derive SDEs for particle dynamics from the full system of colloidal and bath particles. Then we present two ways of deriving the DDFT from such an SDE model.

From Hamilton's Equations to the Langevin Equation

We consider a thermodynamic system consisting of N_c colloidal particles, with positions and momenta given by $(\{\vec{x}_c\}, \{\vec{p}_c\})$, submerged in a bath of N_s solvent particles, with positions and momenta $(\{\vec{x}_s\}, \{\vec{p}_s\})$. Here, we again employ the notation $\{\vec{x}_c\}$ for $\{\vec{x}_{c,i}\}_{i=1}^{N_c}$. In this section it is discussed how this system can be described by Newton's equations of motion, and how Kramers and Langevin equations can be derived from the Newton system. The steps described in the following are illustrated in the flow chart presented in Figure 2.1.

When the particle number remains constant (i.e., in the canonical ensemble), it is possible to describe the total energy of such a system by the Hamiltonian

$$\begin{aligned}\mathcal{H}(\{\vec{x}_c\}, \{\vec{x}_s\}, \{\vec{p}_c\}, \{\vec{p}_s\}) = &\frac{1}{2\mathfrak{m}_c}\sum_{i=1}^{N_c} |\vec{p}_{c,i}|^2 + \frac{1}{2\mathfrak{m}_s}\sum_{i=1}^{N_s} |\vec{p}_{s,i}|^2 \\ &+ \mathbf{V}^c(\{\vec{x}_c\}) + \mathbf{V}^s(\{\vec{x}_s\}) + \mathbf{V}^{c,s}(\{\vec{x}_c\}, \{\vec{x}_s\}),\end{aligned}$$

where $\mathfrak{m}_c$, $\mathfrak{m}_s$ are the masses of colloid and solvent particles, respectively. This can be compared to the simpler Hamiltonian (2.1). The potential energy terms are defined as

$$\begin{aligned}\mathbf{V}^c(\{\vec{x}_c\}) &= \sum_{i=1}^{N_c} V_1^c(\vec{x}_{c,i}) + \sum_{i<j} V_{i,j}^{c,c}(\vec{x}_{c,i}, \vec{x}_{c,j}) + ... \\ \mathbf{V}^s(\{\vec{x}_s\}) &= \sum_{i=1}^{N_s} V_1^s(\vec{x}_{s,i}) + \sum_{i<j} V_{i,j}^{s,s}(\vec{x}_{s,i}, \vec{x}_{s,j}) + ... \\ \mathbf{V}_{c,s}(\{\vec{x}_c\}, (\{\vec{x}_s\}) &= \sum_{i<j} V_{i,j}^{c,s}(\vec{x}_{c,i}, \vec{x}_{s,j}) + ...\end{aligned}$$

see [55]. *Hamilton's equations of motion* are defined as

$$\frac{d\vec{p}_i}{dt} = -\frac{\partial \mathcal{H}}{\partial \vec{x}_i}, \qquad \frac{d\vec{x}_i}{dt} = \frac{\partial \mathcal{H}}{\partial \vec{p}_i}. \tag{2.16}$$

Applying these to colloid and solvent particles, we obtain *Newton's equations*

$$\begin{aligned}\frac{d\vec{x}_{c,i}}{dt} &= \frac{\vec{p}_{c,i}}{\mathfrak{m}_c}, \\ \frac{d\vec{p}_{c,i}}{dt} &= -\nabla_{\vec{x}_{c,i}} V_1^c(\vec{x}_{c,i}) - \nabla_{\vec{x}_{c,i}} \sum_{\substack{j=1 \\ j\neq i}}^{N_c} V_{i,j}^{c,c}(\vec{x}_{c,i}, \vec{x}_{c,j}) - \nabla_{\vec{x}_{c,i}} \sum_{\substack{j=1 \\ j\neq i}}^{N_s} V_{i,j}^{c,s}(\vec{x}_{c,i}, \vec{x}_{s,j}) + ... ,\end{aligned} \tag{2.17}$$

$$\frac{d\vec{x}_{s,i}}{dt} = \frac{\vec{p}_{s,i}}{\mathfrak{m}_s},$$
$$\frac{d\vec{p}_{s,i}}{dt} = -\nabla_{\vec{x}_{s,i}} V_1^s(\vec{x}_{s,i}) - \nabla_{\vec{x}_{s,i}} \sum_{\substack{j=1 \\ j\neq i}}^{N_s} V_{i,j}^{s,s}(\vec{x}_{s,i}, \vec{x}_{s,j}) - \nabla_{\vec{x}_{s,i}} \sum_{\substack{j=1 \\ j\neq i}}^{N_c} V_{i,j}^{s,c}(\vec{x}_{s,i}, \vec{x}_{c,j}) + \dots ,$$

for $i = 1, ..., N_c$ and $i = 1, ..., N_s$, respectively. As discussed in [55], [15, Chapter 2], for this system we can define a phase-space density function $f^{(c+s)}(t, \{\vec{x}_c\}, \{\vec{x}_s\}, \{\vec{p}_c\}, \{\vec{p}_s\})$. In the following, we neglect interaction terms that are many-body, i.e., higher than two-body. Using the chain rule we obtain

$$\begin{aligned}\frac{Df^{(c+s)}}{Dt} = \frac{\partial f^{(c+s)}}{\partial t} &+ \frac{1}{\mathfrak{m}_c}\sum_{i=1}^{N_c} \vec{p}_{c,i} \cdot \nabla_{\vec{x}_{c,i}} f^{(c+s)} + \frac{1}{\mathfrak{m}_s}\sum_{i=1}^{N_s} \vec{p}_{s,i} \cdot \nabla_{\vec{x}_{s,i}} f^{(c+s)} \\ &- \sum_{i=1}^{N_c} \nabla_{\vec{x}_{c,i}} V_1^c(\vec{x}_{c,i}) \cdot \nabla_{\vec{p}_{c,i}} f^{(c+s)} - \sum_{i=1}^{N_s} \nabla_{\vec{x}_{s,i}} V_1^s(\vec{x}_{s,i}) \cdot \nabla_{\vec{p}_{s,i}} f^{(c+s)} \\ &- \sum_{i=1}^{N_c}\sum_{\substack{j=1 \\ j\neq i}}^{N_s} \nabla_{\vec{x}_{c,i}} V_{i,j}^{c,s}(\vec{x}_{c,i}, \vec{x}_{s,i}) \cdot \nabla_{\vec{p}_{c,i}} f^{(c+s)} - \sum_{i=1}^{N_s}\sum_{\substack{j=1 \\ j\neq i}}^{N_c} \nabla_{\vec{x}_{s,i}} V_{i,j}^{s,c}(\vec{x}_{s,i}, \vec{x}_{c,i}) \cdot \nabla_{\vec{p}_{s,i}} f^{(c+s)}.\end{aligned} \tag{2.18}$$

Applying the Liouville Theorem

$$\frac{Df^{(c+s)}}{Dt} = 0,$$

we obtain the *Liouville equation* for (2.17).

In (2.17), assuming that the time scale separation between the bath and the colloids is large, we can assume the action of the solvent particles to be instantaneous. Following the argument in [55] we can derive an equation for the colloid reduced probability density function. Starting from the Liouville equation (2.18) for the phase-space density $f^{(c+s)}$, we can integrate over the solvent positions and momenta to derive an equation for the phase-space density $f^{(c)}$ for the colloids of the form

$$\frac{\partial f^{(c)}}{\partial t} = -\frac{1}{\mathfrak{m}}\sum_{i=1}^{N_c} \vec{p}_i \cdot \nabla_{\vec{x}_i} f^{(c)} - \sum_{i=1}^{N_c} \mathbf{X}_i \cdot \nabla_{\vec{p}_i} f^{(c)} + \left(\frac{\partial f^{(c)}}{\partial t}\right)_{\text{solvent}},$$

where $\mathbf{X}_i = -\nabla_{\vec{x}_i} V_1 - \sum_{\substack{j=1 \\ j\neq i}}^{N} \nabla_{\vec{x}_i} V_{i,j}(\vec{x}_i, \vec{x}_j)$ denotes the forces due to interactions between the colloids, and $\left(\frac{\partial f^{(c)}}{\partial t}\right)_{\text{solvent}}$ describes the interactions of the colloids with the bath. Note that most subscripts c are dropped for readability. We omit the details of the derivation in [55], which involves an application of the fluctuation-dissipation theorem. We find

$$\left(\frac{\partial f^{(c)}}{\partial t}\right)_{\text{solvent}} = \gamma \sum_{i=1}^{N_c} \nabla_{\vec{p}_i} \cdot \vec{p}_i f^{(c)} + \gamma \mathfrak{m}_c k_B T \sum_{i=1}^{N_c} \nabla_{\vec{p}_i}^2 f^{(c)} - \sum_{i=1}^{N_c} \mathbf{x}_i \cdot \nabla_{\vec{p}_i} f^{(c)},$$

where γ is the friction coefficient and $\mathbf{x}_i$ describes the forces on the particles due to the solvent. This results in the N_c dimensional Kramers (Fokker–Planck) equation

$$\begin{aligned}\frac{\partial f^{(c)}}{\partial t} = &-\frac{1}{\mathfrak{m}_c}\sum_{i=1}^{N_c} \vec{p}_i \cdot \nabla_{\vec{x}_i} f^{(c)} + \gamma \sum_{i=1}^{N_c} \nabla_{\vec{p}_i} \cdot \left(\vec{p}_i f^{(c)}\right) - \sum_{i=1}^{N_c} (\mathbf{X}_i + \mathbf{x}_i) \cdot \nabla_{\vec{p}_i} f^{(c)} \\ &+ \gamma \mathfrak{m}_c k_B T \sum_{i=1}^{N_c} \nabla_{\vec{p}_i}^2 f^{(c)}.\end{aligned} \tag{2.19}$$

The corresponding Langevin system to this Fokker–Planck equation is

$$\begin{aligned}\frac{d\vec{x}_{c,i}}{dt} &= \frac{\vec{p}_{c,i}}{\mathfrak{m}_c},\\ \frac{d\vec{p}_{c,i}}{dt} &= -\nabla_{\vec{x}_{c,i}} V_1^c(\vec{x}_{c,i}) - \nabla_{\vec{x}_{c,i}} \sum_{\substack{j=1\\ j\neq i}}^{N_c} V_{i,j}^{c,c}(\vec{x}_{c,i},\vec{x}_{c,j}) - \gamma\vec{p}_{c,i} + \vec{\eta}_i,\end{aligned} \tag{2.20}$$

where γ is the friction coefficient and $\vec{\eta}_i = [\eta_i^{x_1}, \eta_i^{x_2}, \eta_i^{x_3}]$, for $\vec{x}_i = [x, y, z]$, such that $\langle \eta_i^\alpha(t)\rangle = 0$, $\langle \eta_i^\alpha(t)\eta_j^\beta(t')\rangle = 2\gamma\mathfrak{m}_c k_B T\delta_{i,j}\delta(t-t')$. Note that for a simple toy example, this Langevin system can be derived directly from the Newton's equations (2.17), as described in [56, Chapter 8].

When γ is large, in the high friction limit, it can be assumed that $\frac{d\vec{p}_{c,i}}{dt} \approx 0$ and so the microscopic system reduces to the Langevin equation

$$\frac{d\vec{x}_{c,i}}{dt} = \frac{1}{\mathfrak{m}_c\gamma}\left(-\nabla_{\vec{x}_{c,i}} V_{1c}(\vec{x}_{c,i}) - \nabla_{\vec{x}_{c,i}} \sum_{\substack{j=1\\ j\neq i}}^{N_c} V_{i,j}^{c,c}(\vec{x}_{c,i},\vec{x}_{c,j}) + \vec{\eta}_i\right). \tag{2.21}$$

The Langevin Approach by Marconi and Tarazona

DDFT was first derived from the Langevin equation by Marconi and Tarazona in 1999 [51]. This derivation step is illustrated on the top right on the flow chart in Figure 2.1. We start by considering the Langevin equation

$$\frac{d\vec{x}_i(t)}{dt} = -\Gamma\nabla_{\vec{x}_i}\left(V_1(\vec{x}_i) + \sum_{\substack{j=1\\ j\neq i}}^{N} V_{i,j}(|\vec{x}_i - \vec{x}_j|)\right) + \Gamma\vec{\eta}_i(t),$$

which is equivalent to (2.21), with $\Gamma = \frac{1}{\mathfrak{m}_c\gamma}$, dropping the subscript c in this formulation. Note that we choose $V_{i,j}(|\vec{x}_i - \vec{x}_j|)$, which is more specific than $V_{i,j}(\vec{x}_i, \vec{x}_j)$. However, since $V_{i,j}(\vec{x}_i, \vec{x}_j)$ has to be symmetric since the individual particles are indistinguishable, the derivations below follow analogously. We use Itô's lemma to derive an equation for an arbitrary function $f(\vec{x}_i)$

$$\begin{aligned}df(\vec{x}_i) = &-\Gamma\nabla_{\vec{x}_i}\left(V_1(\vec{x}_i) + \sum_{\substack{j=1\\ j\neq i}}^{N} V_{i,j}(|\vec{x}_i - \vec{x}_j|)\right)\cdot\nabla_{\vec{x}_i} f(\vec{x}_i)dt\\ &+ \Gamma k_B T\nabla_{\vec{x}_i}^2 f(\vec{x}_i)dt + \nabla_{\vec{x}_i} f(\vec{x}_i)\cdot\sqrt{2\Gamma k_B T}d\vec{W}_i,\end{aligned}$$

where $\vec{W}_i$ describes Brownian motion with mean zero and variance one.

Then we use the identity $f(\vec{x}_i) = \int \delta(\vec{x}_i(t) - \vec{x})f(\vec{x})d\vec{x}$ to obtain an equation for the partial density operator $\hat{\rho}_i = \delta(\vec{x}_i(t) - \vec{x})$ and use integration by parts to arrive at

$$\begin{aligned}\frac{\partial\hat{\rho}_i(t,\vec{x})}{\partial t} = &\Gamma\nabla\cdot\left[\hat{\rho}_i(t,\vec{x})\left(\nabla V_1(\vec{x}) + \int\hat{\rho}_i(t,\vec{x}')\nabla V_{1,2}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)\right]\\ &+ \Gamma k_B T\nabla^2\hat{\rho}_i(t,\vec{x}) + \Gamma\nabla\cdot\left(\hat{\rho}_i(t,\vec{x})\vec{\eta}_i(t)\right).\end{aligned}$$

Note that boundary terms vanish, since we integrate a density, which has compact support on the domain $\mathbb{R}^d$. Then, defining the global density operator $\hat{\rho}(t,\vec{x}) = \sum_{i=1}^N \delta(\vec{x}_i(t) - \vec{x})$, we

can sum the individual contributions in $\hat{\rho}_i$ to obtain

$$\frac{\partial \hat{\rho}(t,\vec{x})}{\partial t} = \Gamma\nabla\cdot\left[\hat{\rho}(t,\vec{x})\left(\nabla V_1(\vec{x}) + \int \hat{\rho}(t,\vec{x}')\nabla V_{1,2}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)\right] + \Gamma k_B T\nabla^2\hat{\rho}(t,\vec{x}) + \nabla\cdot\left(\sqrt{\hat{\rho}(t,\vec{x})}\vec{\xi}(t,\vec{x})\right), \tag{2.22}$$

where $\sqrt{\hat{\rho}(t,\vec{x})}\vec{\xi}(t,\vec{x}) = \sum_{i=1}^{N}\delta(\vec{x}_i(t)-\vec{x})\vec{\eta}_i$ and $\vec{\xi}$ is such that $\langle\xi^\alpha(t,\vec{x})\xi^\beta(t',\vec{x}')\rangle = 2\Gamma k_B T\delta(t-t')\delta(\vec{x}-\vec{x}')$, see [57]. An ensemble average can be taken, averaging over all paths and initial conditions, to obtain an equation in terms of the noise-averaged density $\rho(t,\vec{x}) = \left\langle\hat{\rho}(t,\vec{x})\right\rangle$, where $\left\langle\nabla\cdot\left(\sqrt{\hat{\rho}(t,\vec{x})}\vec{\xi}(t,\vec{x})\right)\right\rangle = 0$. We then obtain

$$\frac{\partial \rho(t,\vec{x})}{\partial t} = \Gamma\nabla\cdot\left[\rho(t,\vec{x})\nabla V_1(\vec{x}) + \int\langle\hat{\rho}(t,\vec{x})\hat{\rho}(t,\vec{x}')\rangle\nabla V_{1,2}(|\vec{x}-\vec{x}'|)d\vec{x}'\right] + \Gamma k_B T\nabla^2\rho(t,\vec{x}). \tag{2.23}$$

In order to close this equation, we need to express the two-body correlation $\langle\hat{\rho}(t,\vec{x})\hat{\rho}(t,\vec{x}')\rangle$ in terms of ρ. This is done using the adiabatic approximation, which relates the equilibrium excess free energy $\mathcal{F}_{\text{exc}}[\rho]$, which depends on ρ only, to the non-equilibrium particle correlations. The integral term is approximated by

$$\int\langle\hat{\rho}(t,\vec{x})\hat{\rho}(t,\vec{x}')\rangle\nabla V_{1,2}(|\vec{x}-\vec{x}'|)d\vec{x}' = \rho(t,\vec{x})\nabla\frac{\delta\mathcal{F}_{\text{exc}}[\rho]}{\delta\rho}, \tag{2.24}$$

where interaction terms of higher order are neglected. Substituting this into (2.23) results in

$$\frac{\partial\rho}{\partial t} = \Gamma\nabla\cdot\left(\rho\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\right),$$

which is (2.12).

The Smoluchowski Approach by Archer and Evans

We now consider a second way of deriving DDFT. This Smoluchowski derivation follows Archer and Evans' work from 2004 [49]. This derivation can be seen in the flow chart in Figure 2.1. We can again start from the Langevin equation (2.21). As in the previous section, we can apply Itô's Lemma. However, in this case, the multivariate version is used, so that $f^{(N)}$ is a function of all $\{\vec{x}_i\}$. We obtain

$$\begin{aligned} df^{(N)}(\{\vec{x}_i\}) = &-\Gamma\sum_{i=1}^{N}\nabla_{\vec{x}_i}\left(V_1(\vec{x}_i) + \frac{1}{2}\sum_{\substack{j=1\\ j\neq i}}^{N}V_{i,j}(|\vec{x}_i-\vec{x}_j|)\right)\cdot\nabla_{\vec{x}_i}f^{(N)}(\{\vec{x}_i\})dt \\ &+\Gamma k_B T\sum_{i=1}^{N}\nabla^2_{\vec{x}_i}f^{(N)}(\{\vec{x}_i\})dt + \sum_{i=1}^{N}\nabla_{\vec{x}_i}f^{(N)}(\{\vec{x}_i\})\cdot\sqrt{2\Gamma k_B T}d\vec{W}_i. \end{aligned}$$

We then define the ensemble average over a function $f^{(N)}$ and the initial conditions as

$$\langle f^{(N)}\rangle = \int f^{(N)}(\{\vec{x}_i\})P(t,\{\vec{x}_i\}|t_0,\vec{x}_0)d\{\vec{x}_i\},$$

where $P(t,\{\vec{x}_i\}|t_0,\vec{x}_0)$ is the conditional probability density with an initial condition density $P_0(t_0,\vec{x}_0)$. This probability density describes the probability of the system being in state $\{\vec{x}_i\}$ at time t, given an initial state $\vec{x}_0$ at time t_0. Note, however, that this initial state is also given in terms of a distribution, not a specific chosen particle state.

This can be best understood when considering the following equivalent approach. This is averaging over all particle paths and all initial conditions, by explicitly computing

$$\langle f \rangle = \int \int f(t, \vec{x}(\vec{x}_0, d\vec{W}))d\vec{W}d\vec{x}_0.$$

Here, the current position $\vec{x}$ depends on both the initial position $\vec{x}_0$ and the specific realization of the noise that describes the path to get to $\vec{x}$ at time t. This formulation also demonstrates why the noise term in the SDE vanishes under the ensemble average. Consider the average of $g(t, \vec{x})d\vec{W}$. We obtain

$$\langle g(t, \vec{x})d\vec{W} \rangle = \int \int g(t, \vec{x})d\vec{W}d\vec{W}'d\vec{x}_0 = 0,$$

since the increments $d\vec{W}d\vec{W}'$ are independent from each other.

With this approach, and using integration by parts, we obtain the Smoluchowski equation

$$\frac{\partial P(t, \{\vec{x}_i\})}{\partial t} = \Gamma \sum_{i=1}^{N} \nabla_{\vec{x}_i} \cdot \big(P(t, \{\vec{x}_i\})\nabla_{\vec{x}_i} V_1(\vec{x}_i)\big) + \frac{\Gamma}{2} \sum_{i=1}^{N} \nabla_{\vec{x}_i} \cdot \left(P(t, \{\vec{x}_i\})\nabla_{\vec{x}_i} \sum_{\substack{j=1 \\ j\neq i}}^{N} V_{i,j}(|\vec{x}_i - \vec{x}_j|) \right) \tag{2.25}$$
$$+ \Gamma \sum_{i=1}^{N} k_B T \nabla^2_{\vec{x}_i} P(t, \{\vec{x}_i\}).$$

Note that in [55] the Smoluchowski equation is derived from Kramers equation (2.19) instead, illustrated in Figure 2.1.

In order to derive a DDFT from (2.25), we can multiply by N and integrate over all but one particle positions to obtain a resulting equation in terms of the one-body density

$$\rho(t, \vec{x}) := \rho^{(1)}(t, \vec{x}_1) = N \int P(t, \{\vec{x}_i\})d\vec{x}_2 ... d\vec{x}_N.$$

This gives

$$\begin{aligned} N \int_\Omega \frac{\partial P(t, \{\vec{x}_i\})}{\partial t} d\vec{x}_2 ... d\vec{x}_N = \Gamma N \int_\Omega \Bigg[& \sum_{i=1}^{N} \nabla_{\vec{x}_i} \cdot \big(P(t, \{\vec{x}_i\})\nabla_{\vec{x}_i} V_1(\vec{x}_i)\big) \\ & + \frac{1}{2} \sum_{i=1}^{N} \nabla_{\vec{x}_i} \cdot \left(P(t, \{\vec{x}_i\})\nabla_{\vec{x}_i} \sum_{\substack{j=1 \\ j\neq i}}^{N} V_{i,j}(|\vec{x}_i - \vec{x}_j|) \right) \\ & + \sum_{i=1}^{N} k_B T \nabla^2_{\vec{x}_i} P(t, \{\vec{x}_i\}) \Bigg] d\vec{x}_2 ... d\vec{x}_N, \\ := I_1 + I_2 + I_3. & \end{aligned}$$

The left-hand side satisfies $N \int_\Omega \frac{\partial P(t, \{\vec{x}_i\})}{\partial t} d\vec{x}_2 ... d\vec{x}_N = \frac{\partial \rho(t, \vec{x})}{\partial t}$, by the definition of ρ and the fact that the integral is time-independent.

The integral I_1 satisfies

$$I_1 = N \int_\Omega \sum_{i=1}^{N} \nabla_{\vec{x}_i} \cdot \nabla_{\vec{x}_i} P(t, \{\vec{x}_i\})d\vec{x}_2 ... d\vec{x}_N = \nabla^2 \rho(t, \vec{x}), \tag{2.26}$$

by noting that the sum on the left-hand side of the equation, as well as the integration, are finite. Therefore, Fubini's Theorem can be used to take the sum out of the integral. The result follows by applying the Divergence Theorem for all $i \neq 1$, where the resulting boundary terms vanish by applying the boundary conditions. Note that we could apply no-flux boundary conditions on some boundary $\partial\Omega$ instead of the Dirichlet conditions at infinity, which would result in the DDFT being equipped with no-flux conditions.

Next, the integral I_2 is considered

$$I_2 = N \int_\Omega \sum_{i=1}^{N} \nabla_{\vec{x}_i} \cdot \Big(P(t, \{\vec{x}_i\}) \nabla_{\vec{x}_i} V_1(t, \vec{x}_i) \Big) d\vec{x}_2 ... d\vec{x}_N.$$

By the same argument as for I_1, the integration and summation can be swapped and so the Divergence Theorem can be used for all variables $d\vec{x}_2 ... d\vec{x}_N$, while the equation for $\vec{x}_1$ remains unchanged. This gives

$$I_2 = N \sum_{i=2}^{N} \int_{\partial\Omega} P(t, \{\vec{x}_i\}) \frac{\partial V_1(t, \vec{x}_i)}{\partial n} d\vec{x}_2 ... d\vec{x}_N + N \int_\Omega \nabla_1 \cdot \Big(P(t, \{\vec{x}_i\}) \nabla_1 V_1(t, \vec{x}_1) \Big) d\vec{x}_2 ... d\vec{x}_N,$$

where $\frac{\partial}{\partial n}$ denotes the derivative in the direction of the outward normal $\vec{n}$.

Then, applying the boundary conditions for $P(t, \{\vec{x}_i\})$ the following equation is derived

$$I_2 = N \int_\Omega \nabla_1 \cdot \Big(P(t, \{\vec{x}_i\}) \nabla_1 V_1(t, \vec{x}_1) \Big) d\vec{x}_2 ... d\vec{x}_N.$$

Since $\vec{x}_1$ is constant with respect to the integration variables, all terms only depending on $\vec{x}_1$ can be taken out of the integral, dropping the subscripts, I_2 is

$$I_2 = \nabla \cdot \Big(\rho(t, \vec{x}) \nabla V_1(t, \vec{x}) \Big). \tag{2.27}$$

The final term in the PDE that has to be considered is I_3

$$I_3 = \frac{N}{2} \int_\Omega \sum_{i=1}^{N} \nabla_{\vec{x}_i} \cdot \Big(P(t, \{\vec{x}_i\}) \nabla_{\vec{x}_i} \sum_{i \neq j} V_{i,j}(|\vec{x}_i - \vec{x}_j|) \Big) d\vec{x}_2 ... d\vec{x}_N.$$

As for I_1 and I_2, the integration and summation operations can be swapped, and the Divergence Theorem can be applied to $d\vec{x}_2 ... d\vec{x}_N$ to give

$$\begin{aligned} I_3 &= \frac{N}{2} \sum_{i=1}^{N} \int_\Omega \nabla_{\vec{x}_i} \cdot \Big(P(t, \{\vec{x}_i\}) \nabla_{\vec{x}_i} \sum_{i \neq j} V_{i,j}(|\vec{x}_i - \vec{x}_j|) \Big) d\vec{x}_2 ... d\vec{x}_N \\ &= \frac{N}{2} \sum_{i=2}^{N} \int_{\partial\Omega} \Big(P(t, \{\vec{x}_i\}) \sum_{i \neq j} \frac{\partial V_{i,j}(|\vec{x}_i - \vec{x}_j|)}{\partial n} \Big) d\vec{x}_2 ... d\vec{x}_N \\ &\quad + \frac{N}{2} \int_\Omega \nabla_1 \cdot \Big(P(t, \{\vec{x}_i\}) \nabla_1 \sum_{i \neq j} V_{i,j}(|\vec{x}_i - \vec{x}_j|) \Big) d\vec{x}_2 ... d\vec{x}_N. \end{aligned}$$

The boundary conditions for P are applied to set the first term to zero.

The term $\nabla_1 \sum_{i \neq j} V_{i,j}(|\vec{x}_i - \vec{x}_j|)$ has to be examined in more detail. Since the gradient is applied with respect to $\vec{x}_1$, one of the $\vec{x}_i, \vec{x}_j$ in the double sum has to be $\vec{x}_1$, since all other terms will

be zero when the gradient is applied. Therefore

$$\nabla_1 \sum_{i\neq j} V_{i,j}(|\vec{x}_i - \vec{x}_j|) = \nabla_1 \sum_{j=2}^{N} V_{1,j}(|\vec{x}_1 - \vec{x}_j|) + \nabla_1 \sum_{i=2}^{N} V_{i,1}(|\vec{x}_i - \vec{x}_1|).$$

Since i and j are dummy variables, and since $V_{i,j}$ is symmetric by assumption to satisfy the physical model, the previous equation reduces to

$$\nabla_1 \sum_{i\neq j} V_{i,j}(|\vec{x}_i - \vec{x}_j|) = 2\nabla_1 \sum_{j=2}^{N} V_{1,j}(|\vec{x}_1 - \vec{x}_j|).$$

Then I_3 becomes

$$I_3 = N \int_\Omega \nabla_1 \cdot \left(P(t, \{\vec{x}_i\}) \nabla_1 \sum_{j=2}^{N} V_{1,j}(|\vec{x}_1 - \vec{x}_j|) \right) d\vec{x}_2 ... d\vec{x}_N.$$

Since the particles are indistinguishable, a permutation argument can be employed and the indices $\vec{x}_j$ in the terms $V_{1,j}(|\vec{x}_1 - \vec{x}_j|)$ can be relabelled, such that $\vec{x}_i = \vec{x}_2$, $i = 3, ..., N$, for each term in the sum. Since the integration is symmetric, the integration order can be permuted arbitrarily and hence does not have to be adapted. This results in $N-1$ identical equations, and so I_3 becomes

$$I_3 = N(N-1) \int_\Omega \nabla_1 \cdot \left(P(t, \{\vec{x}_i\}) \nabla_1 V_{1,2}(|\vec{x}_1 - \vec{x}_2|) \right) d\vec{x}_2 ... d\vec{x}_N.$$

Considering now the definition of the n-body particle distributions, I_3 can be written in terms of the two-body distribution $\rho^{(2)}(t, \vec{x}_1, \vec{x}_2) = N(N-1) \int_\Omega P(t, \{\vec{x}_i\}) d\vec{x}_3 ... d\vec{x}_N$. Terms that only depend on $\vec{x}_1$ can be taken out of the integral. Then I_3 is

$$I_3 = N(N-1) \nabla_1 \cdot \left(\int_\Omega \nabla_1 V_{1,2}(|\vec{x}_1 - \vec{x}_2|) P(t, \{\vec{x}_i\}) d\vec{x}_2 ... d\vec{x}_N \right).$$

Since $V_{1,2}$ only depends on $\vec{x}_2$ and the order of integration is arbitrary, the integral can be rewritten and dropping the indices on ∇_1, the equation is

$$I_3 = \nabla \cdot \left(\int_\Omega \nabla V_{1,2}(|\vec{x} - \vec{x}'|) \rho^{(2)}(t, \vec{x}, \vec{x}') d\vec{x}' \right). \tag{2.28}$$

The full PDE for ρ is then given by

$$\frac{\partial \rho(t, \vec{x})}{\partial t} = -\Gamma \nabla \cdot \left(k_B T \nabla \rho(t, \vec{x}) + \rho(t, \vec{x}) \nabla V_1(t, \vec{x}) + \int_\Omega \nabla V_{1,2}(|\vec{x} - \vec{x}'|) \rho^{(2)}(t, \vec{x}, \vec{x}') d\vec{x}' \right).$$

This equation is not closed, since it depends on $\rho^{(2)}(t, \vec{x}, \vec{x}')$ and we make the adiabatic approximation (2.24), using that $\rho^{(2)}(t, \vec{x}, \vec{x}') \approx \langle \hat{\rho}(t, \vec{x}) \hat{\rho}(t, \vec{x}') \rangle$, to close the equation, so that the standard DDFT (2.12) is recovered.

2.3.5 Inertial DDFT

In the standard DDFT, the assumption is made that the momentum equilibrates quickly and can be neglected since its distribution is known. However, in many systems, this assumption may be too strong and one may be interested in deriving equations which contain such a momentum, or velocity, contribution. The review [5] discusses several models that include momentum. In [54], such an extended DDFT is derived, which will be described in this section. It is chosen

as illustration of how to extend the standard DDFT model by loosening the assumptions and approximations made in its derivation. Moreover, as can be seen in Figure 2.1, this derivation is also an alternative way of deriving the standard DDFT.

The starting point for this derivation is the full system of Newton's equations for both colloid and bath particles (2.17), as done in Section 2.3.4. From there, one can either average over the bath particles to get a system of Langevin equations of the form (2.20), or derive the Liouville equation (2.18) for the full system, including bath particles, for which an associated Kramers (Fokker–Planck) equation (2.19) can be defined. Starting from (2.20), the phase-space density $f^{(N)}$ for the colloidal particles can be defined and the N-dimensional Kramers (Fokker–Planck) equation can be derived. This is given by

$$\frac{\partial f^{(N)}}{\partial t} = -\frac{1}{\mathfrak{m}}\sum_{i=1}^{N}\vec{p}_i\cdot\nabla_{\vec{x}_i}f^{(N)} + \gamma\sum_{i=1}^{N}\nabla_{\vec{p}_i}\cdot\left(\vec{p}_i f^{(N)}\right) - \sum_{i=1}^{N}\mathbf{X}_i\cdot\nabla_{\vec{p}_i}f^{N} + \gamma\mathfrak{m}k_BT\sum_{i=1}^{N}\nabla^2_{\vec{p}_i}f^{(N)}, \tag{2.29}$$

where $\mathbf{X}_i = -\nabla V_1 - \sum_{\substack{j=1\\ j\neq i}}^{N}\nabla V_{i,j}$, the external potential and interaction contributions, omitting higher order interactions. Note that this is in fact equivalent to (2.19) in Section 2.3.4, with $f^{(N)} = f^{(c)}$, by setting $\mathbf{X}_i$ in the present version equal to $\mathbf{X}_i + \mathbf{x}_i$ in (2.19).

Time-dependent reduced phase-space distribution functions are defined as

$$f^{(n)}(t,\vec{x}_1,...\vec{x}_n,\vec{p}_1,...,\vec{x}_n) = \frac{N!}{(N-n)!}\int\int f^{(N)}(\{\vec{x}_i\},\{\vec{p}_i\})d\vec{x}_n...d\vec{x}_N d\vec{p}_n...d\vec{p}_N,$$

compare to (2.40). In particular, we define

$$f^{(1)}(t,\vec{x}_1,\vec{p}_1) = N\int\int f^{(N)}(\{\vec{x}_i\},\{\vec{p}_i\})d\vec{x}_2...d\vec{x}_N d\vec{p}_2...d\vec{p}_N.$$

Equation (2.29) is multiplied by N and integrated over $d\vec{x}_2...d\vec{x}_N, d\vec{p}_2...d\vec{p}_N$. The definition of the forces is extended, so that V_1 depends on time, and the interaction potential is chosen as $V_{i,j}(\vec{x}_i,\vec{x}_j) = V_{i,j}(|\vec{x}_i - \vec{x}_j|)$. This results in

$$\mathbf{X}_i(t,\vec{x}_i) = -\nabla_{\vec{x}_i}\left(\sum_{i=1}^{N}V_1(t,\vec{x}_i) + \frac{1}{2}\sum_{i,j}V_{i,j}(|\vec{x}_i - \vec{x}_j|)\right).$$

Applying the definition of the reduced phase-space distributions, using symmetry and the Divergence Theorem, we obtain

$$\begin{aligned}\frac{\partial f^{(1)}}{\partial t} = &-\frac{\vec{p}_1}{\mathfrak{m}}\cdot\nabla_{\vec{x}_1}f^{(1)} + \gamma\nabla_{\vec{p}_1}\cdot\left(\vec{p}_1 f^{(1)}\right) + \nabla_{\vec{x}_1}V_1(t,\vec{x}_1)\cdot\nabla_{\vec{p}_1}f^{(1)} \\ &+ \gamma\mathfrak{m}k_BT\nabla^2_{\vec{p}_1}f^{(1)} + \int\int\nabla_{\vec{x}_1}V_{1,2}(|\vec{x}_1 - \vec{x}_2|)\cdot\nabla_{\vec{p}_1}f^{(2)}d\vec{x}_2 d\vec{p}_2.\end{aligned} \tag{2.30}$$

This result can be compared to (2.41) in the derivation for classical fluids.

Next, by taking two momentum moments one can obtain two equations for the reduced density. The first momentum moment is obtained by integrating (2.30) with respect to $\vec{p}_1$, while the second momentum moment is derived by multiplying (2.30) by $\frac{\vec{p}_1}{\mathfrak{m}}$ and, once again, integrating with respect to $\vec{p}_1$. Taking the first momentum moment

$$\begin{aligned}\int\frac{\partial f^{(1)}}{\partial t}d\vec{p}_1 = \int\Bigg[&-\frac{\vec{p}_1}{\mathfrak{m}}\cdot\nabla_{\vec{x}_1}f^{(1)} + \gamma\nabla_{\vec{p}_1}\cdot\left(\vec{p}_1 f^{(1)}\right) + \nabla_{\vec{x}_1}V_1(t,\vec{x}_1)\cdot\nabla_{\vec{p}_1}f^{(1)} \\ &+ \gamma\mathfrak{m}k_BT\nabla^2_{\vec{p}_1}f^{(1)} + \int\int\nabla_{\vec{x}_1}V_{1,2}(|\vec{x}_1 - \vec{x}_2|)\cdot\nabla_{\vec{p}_1}f^{(2)}d\vec{x}_2 d\vec{p}_2\Bigg]d\vec{p}_1\end{aligned} \tag{2.31}$$

results in

$$\frac{\partial \rho(t,\vec{x}_1)}{\partial t} + \nabla_{\vec{x}_1} \cdot \mathrm{j} = 0, \tag{2.32}$$

where

$$\frac{\partial \rho(t,\vec{x}_1)}{\partial t} := \frac{\partial}{\partial t}\int f^{(1)}(t,\vec{x}_1,\vec{p}_1)d\vec{p}_1, \qquad \nabla_{\vec{x}_1}\cdot \mathrm{j}(t,\vec{x}_1) := \nabla_{\vec{x}_1}\cdot\int \frac{\vec{p}_1}{\mathfrak{m}} f^{(1)}(t,\vec{x}_1,\vec{p}_1)d\vec{p}_1.$$

Note that all other terms in (2.31) are zero, due to the Divergence Theorem. Then, multiplying by $\frac{\vec{p}_1}{\mathfrak{m}}$ and integrating with respect to $\vec{p}_1$ results in

$$\begin{aligned}\int \frac{\vec{p}_1}{\mathfrak{m}}\frac{\partial f^{(1)}}{\partial t}d\vec{p}_1 = \int \frac{\vec{p}_1}{\mathfrak{m}}\Big[&- \frac{\vec{p}_1}{\mathfrak{m}}\cdot\nabla_{\vec{x}_1} f^{(1)} + \gamma\nabla_{\vec{p}_1}\cdot\Big(\vec{p}_1 f^{(1)}\Big) + \nabla_{\vec{x}_1}V_1(t,\vec{x}_1)\cdot\nabla_{\vec{p}_1}f^{(1)} \\ &+ \gamma\mathfrak{m}k_BT\nabla^2_{\vec{p}_1}f^{(1)} + \int\int \nabla_{\vec{x}_1}V_{1,2}(|\vec{x}_1-\vec{x}_2|)\cdot\nabla_{\vec{p}_1}f^{(2)}d\vec{x}_2 d\vec{p}_2\Big]d\vec{p}_1.\end{aligned}$$

Using the definition of ρ and j, this is

$$\begin{aligned}\frac{\partial \mathrm{j}(t,\vec{x}_1)}{\partial t} = &-\nabla_{\vec{x}_1}\cdot\int\frac{\vec{p}_1\otimes\vec{p}_1}{\mathfrak{m}^2}f^{(1)}d\vec{p}_1 - \gamma\mathrm{j}(t,\vec{x}_1) - \frac{1}{\mathfrak{m}}\rho(t,\vec{x}_1)\nabla V_1(t,\vec{x}_1) \\ &- \frac{1}{\mathfrak{m}}\int\rho^{(2)}(t,\vec{x}_1,\vec{x}_2)\nabla_{\vec{x}_1}V_{1,2}(|\vec{x}_1-\vec{x}_2|)d\vec{x}_2,\end{aligned} \tag{2.33}$$

where

$$\rho^{(2)}(t,\vec{x}_1,\vec{x}_2) = \int\int f^{(2)}(t,\vec{x}_1,\vec{x}_2,\vec{p}_1,\vec{p}_2)d\vec{p}_1 d\vec{p}_2.$$

We add and subtract the same term

$$\frac{k_BT}{\mathfrak{m}}\nabla\cdot\int \mathrm{Diag}\Big(f^{(1)}\Big)d\vec{p}_1 - \frac{k_bT}{\mathfrak{m}}\nabla\rho(t,\vec{x}_1)$$

to (2.33) and define

$$\mathbf{A}(t,\vec{x}_1) := -\nabla\cdot\int\frac{\vec{p}_1\otimes\vec{p}_1}{\mathfrak{m}^2}f^{(1)}d\vec{p}_1 + \frac{k_BT}{\mathfrak{m}}\nabla\cdot\int\mathrm{Diag}\Big(f^{(1)}\Big)d\vec{p}_1,$$

where $\mathrm{Diag}\Big(f^{(1)}\Big)$ is a diagonal matrix with $f^{(1)}$ on the diagonal, defined to match the dimension of $\vec{p}_1\otimes\vec{p}_1$. This allows to rewrite (2.33) as

$$\begin{aligned}\frac{\partial \mathrm{j}(t,\vec{x}_1)}{\partial t} = &-\mathbf{A}(t,\vec{x}_1) - \frac{k_BT}{\mathfrak{m}}\nabla\rho(t,\vec{x}_1) - \gamma\mathrm{j}(t,\vec{x}_1) - \frac{1}{\mathfrak{m}}\rho(t,\vec{x}_1)\nabla V_1(t,\vec{x}_1) \\ &- \frac{1}{\mathfrak{m}}\int\rho^{(2)}(t,\vec{x}_1,\vec{x}_2)\nabla V_{1,2}(|\vec{x}_1-\vec{x}_2|)d\vec{x}_2.\end{aligned} \tag{2.34}$$

In order to close the system of equations, two approximations are made: the adiabatic approximation and the local equilibrium approximation. As discussed in Section 2.3.3, the adiabatic approximation states that the particle interactions in the nonequilibrium fluid can be approximated by the interactions in the equilibrium fluid, so that

$$\int\rho^{(2)}(t,\vec{x}_1,\vec{x}_2)\nabla V_{1,2}(|\vec{x}_1-\vec{x}_2|)d\vec{x}_2 \approx \rho(t,\vec{x}_1)\nabla\frac{\delta\mathcal{F}_{\mathrm{exc}}[\rho]}{\delta\rho}.$$

Then Equation (2.34) becomes

$$\begin{aligned}\frac{\partial \mathrm{j}(t,\vec{x}_1)}{\partial t} =& -\mathbf{A}(t,\vec{x}_1) - \frac{k_BT}{\mathfrak{m}}\nabla\rho(t,\vec{x}_1) - \gamma \mathrm{j}(t,\vec{x}_1) - \frac{1}{\mathfrak{m}}\rho(t,\vec{x}_1)\nabla V_1(t,\vec{x}_1)\\ &- \frac{1}{\mathfrak{m}}\rho(t,\vec{x}_1)\nabla\frac{\delta\mathcal{F}_{\mathrm{exc}}[\rho]}{\delta\rho},\end{aligned}$$

and using the identity $\nabla\rho(t,\vec{x}_1) = \rho(t,\vec{x}_1)\ln(\rho(t,\vec{x}_1))$, the equation reduces to

$$\begin{aligned}\frac{\partial \mathrm{j}(t,\vec{x}_1)}{\partial t} =& -\mathbf{A}(t,\vec{x}_1) - \frac{k_BT}{\mathfrak{m}}\rho(t,\vec{x}_1)\ln(\rho(t,\vec{x}_1)) - \gamma \mathrm{j}(t,\vec{x}_1) - \frac{1}{\mathfrak{m}}\rho(t,\vec{x}_1)\nabla V_1(t,\vec{x}_1)\\ &- \frac{1}{\mathfrak{m}}\rho(t,\vec{x}_1)\nabla\frac{\delta\mathcal{F}_{\mathrm{exc}}[\rho]}{\delta\rho}\\ =& -\mathbf{A}(t,\vec{x}_1) - \gamma \mathrm{j}(t,\vec{x}_1) - \frac{1}{\mathfrak{m}}\rho(t,\vec{x}_1)\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}.\end{aligned} \tag{2.35}$$

A local-equilibrium approximation for $f^{(1)}$ can be made

$$f^{(1)}_{\mathrm{l.e.}}(\vec{x}_1,\vec{p}_1,t) = \frac{\rho(t,\vec{x}_1)}{(2\pi\mathfrak{m}k_BT)^{3/2}}\exp\left\{-\frac{(\vec{p}_1-\bar{p})^2}{2\mathfrak{m}k_BT}\right\},$$

which assumes that the momentum $\vec{p}$ is close to the average momentum $\bar{p}$ defined using the average local velocity via $\bar{p} = \mathfrak{m}\vec{v}$. Then

$$\mathrm{j} = \int \frac{\vec{p}_1}{\mathfrak{m}} f^{(1)} d\vec{p}_1 \approx \rho(t,\vec{x}_1)\vec{v}(t,\vec{x}_1),$$

and so (2.32) becomes

$$\frac{\partial\rho(t,\vec{x}_1)}{\partial t} + \nabla_{\vec{x}_1}\cdot\big(\rho(t,\vec{x}_1)\vec{v}(t,\vec{x}_1)\big) = 0.$$

The expression for $\mathbf{A}$ can be simplified

$$\mathbf{A}(t,\vec{x}_1) = -\nabla\cdot\int\frac{\vec{p}_1\otimes\vec{p}_1}{\mathfrak{m}^2}f^{(1)}d\vec{p}_1 + \frac{k_BT}{\mathfrak{m}}\nabla\cdot\int\mathrm{Diag}\left(f^{(1)}\right)d\vec{p}_1 = \nabla\cdot\left(\rho\vec{v}\otimes\vec{v}\right),$$

so that (2.35) is

$$\frac{\partial\rho\vec{v}}{\partial t} = -\nabla\cdot(\rho\vec{v}\otimes\vec{v}) - \gamma\rho\vec{v} - \frac{1}{\mathfrak{m}}\rho\nabla\frac{\delta F[\rho]}{\delta\rho},$$

which becomes

$$\rho\frac{\partial\vec{v}}{\partial t} = -\rho\vec{v}\cdot(\nabla\vec{v}) - \gamma\rho\vec{v} - \frac{1}{\mathfrak{m}}\rho\nabla\frac{\delta F[\rho]}{\delta\rho}, \tag{2.36}$$

by rewriting $\nabla\cdot(\rho\vec{v}\otimes\vec{v})$ and $\frac{\partial\rho\vec{v}}{\partial t}$. Therefore, the one-body inertial equations are

$$\begin{aligned}&\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\vec{v}) = 0,\\ &\frac{\partial\vec{v}}{\partial t} = -\vec{v}\cdot(\nabla\vec{v}) - \gamma\vec{v} - \frac{1}{\mathfrak{m}}\nabla\frac{\delta F[\rho]}{\delta\rho}.\end{aligned} \tag{2.37}$$

The classical DDFT (2.12) can be recovered by taking the overdamped limit of (2.37) when γ

is large. Then $\frac{D\vec{v}}{Dt} := \frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot (\nabla \vec{v}) = 0$ and so, substituting this into (2.37) gives

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0,$$
$$\vec{v} = -\frac{1}{\mathfrak{m}\gamma} \nabla \frac{\delta F[\rho]}{\delta \rho}.$$

From this, an overdamped equation in ρ is obtained, independent of $\vec{v}$, which is precisely (2.12)

$$\frac{\partial \rho}{\partial t} - \frac{1}{\mathfrak{m}\gamma} \nabla \cdot \left(\rho \nabla \frac{\delta F[\rho]}{\delta \rho} \right) = 0.$$

This limit can be taken rigorously, as done in [58]. In this article a similar flow chart to the one in Figure 2.1 is presented, which demonstrates different ways of deriving the overdamped DDFT. Without hydrodynamic interactions present, the derivation in this section agrees with the one discussed in Section 2.3.4. However, as demonstrated in [58], when hydrodynamic interactions are included in the model, different ways of deriving the overdamped DDFT do not lead to the same result.

As shown in [54], one can instead take the limit $\gamma \to 0$ in (2.37). This results in

$$\frac{\partial \vec{v}}{\partial t} = -\vec{v} \cdot (\nabla \vec{v}) - \frac{1}{\mathfrak{m}} \nabla \frac{\delta F[\rho]}{\delta \rho}.$$

Splitting $\mathcal{F}$ into ideal, external, and excess contributions, this can be rewritten to give

$$\frac{\partial \vec{v}}{\partial t} = -\vec{v} \cdot (\nabla \vec{v}) - \frac{1}{\mathfrak{m}} \nabla V_1 - \frac{1}{\rho \mathfrak{m}} \nabla p, \tag{2.38}$$

where $p = -f + \mu\rho$ is a pressure term, with $f = k_B T \rho \left(\ln \Lambda^3 \rho - 1 \right) + f_{\text{exc}}$ and $\mu = \frac{\delta f}{\delta \rho}$, so that the compressible Euler's equation for a classical fluid is recovered.

2.3.6 A Connection to Classical Fluids

In Section 2.3.5 it is shown that a result for classical fluids can be obtained from equations involved in the derivation of DDFT. In this section, we investigate a derivation from the microscopic system for a classical fluid, and see that similar principles to the above can be applied. The key difference is that there is no underlying bath that dampens the particle motion, and so no diffusive forces will enter the system. The steps of the derivation are illustrated in Figure 2.1. This section draws from [7, Chapter 9], [9, Chapter 3], and [59].

As in Section 2.3.4, we assume the total energy of the system to be described by the Hamiltonian of the form

$$\mathcal{H}(\{\vec{x}_i\}, \{\vec{p}_i\}) = \frac{1}{2\mathfrak{m}} \sum_{i=1}^{N} |\vec{p}_i|^2 + \mathbf{V}(\{\vec{x}_i\}),$$

where $\mathfrak{m}$ is the particle mass and

$$\mathbf{V}(\{\vec{x}_i\}) = \sum_{i=1}^{N} V_1(\vec{x}_i) + \sum_{i<j} V_{i,j}(\vec{x}_i, \vec{x}_j).$$

This can be compared to (2.1). As before, Hamilton's equations of motion (2.16) can be applied to derive the system of Newton's equations

$$\begin{aligned}\frac{d\vec{x}_i}{dt} &= \frac{\vec{p}_i}{\mathfrak{m}},\\ \frac{d\vec{p}_i}{dt} &= -\nabla_{\vec{x}_i} V_1(\vec{x}_i) - \nabla_{\vec{x}_i} \sum_{i<j} V_{i,j}(\vec{x}_i, \vec{x}_j).\end{aligned}$$

Since this is not a system of stochastic but of deterministic equations, we do not need to apply Itô's Lemma to derive an equation for a probability distribution $f^{(N)}(t, \{\vec{x}_i\}, \{\vec{p}_i\})$, but merely take the total derivative of $f^{(N)}$, which is an application of Liouville's Theorem, to get the Liouville equations

$$\begin{aligned}\frac{Df^{(N)}(t, \{\vec{x}_i\}, \{\vec{p}_i\})}{Dt} =& \frac{\partial f^{(N)}(t, \{\vec{x}_i\}, \{\vec{p}_i\})}{\partial t} + \sum_{i=1}^{N} \nabla_{\vec{p}_i} f^{(N)}(t, \{\vec{x}_i\}, \{\vec{p}_i\}) \cdot \frac{d\vec{p}_i}{dt} \\ &+ \sum_{i=1}^{N} \nabla_{\vec{x}_i} f^{(N)}(t, \{\vec{x}_i\}, \{\vec{p}_i\}) \cdot \frac{d\vec{x}_i}{dt} = 0.\end{aligned}$$

Substituting the derivatives of $\vec{x}_i$, $\vec{p}_i$, we obtain

$$\begin{aligned}\frac{\partial f^{(N)}(t, \{\vec{x}_i\}, \{\vec{p}_i\})}{\partial t} =& \sum_{i=1}^{N} \nabla_{\vec{p}_i} f^{(N)}(t, \{\vec{x}_i\}, \{\vec{p}_i\}) \cdot \left(\nabla_{\vec{x}_i} V_1(\vec{x}_i) + \nabla_{\vec{x}_i} \sum_{i<j} V_{i,j}(\vec{x}_i, \vec{x}_j) \right) \\ &- \sum_{i=1}^{N} \nabla_{\vec{x}_i} f^{(N)}(t, \{\vec{x}_i\}, \{\vec{p}_i\}) \cdot \frac{\vec{p}_i}{\mathfrak{m}}.\end{aligned} \tag{2.39}$$

As is the case for colloidal fluids, we can define reduced phase-space distributions of the form

$$f^{(n)}(t, \vec{x}_1, ...\vec{x}_n, \vec{p}_1, ..., \vec{p}_n) = \frac{N!}{(N-n)!} \int\int f^{(N)}(t, \{\vec{x}_i\}, \{\vec{p}_i\}) d\vec{x}_n ... d\vec{x}_N d\vec{p}_n ... d\vec{p}_N. \tag{2.40}$$

Multiplying (2.39) by N and integrating over all but one spatial and momentum degrees, we obtain

$$\begin{aligned}\frac{\partial f^{(1)}(t, \vec{x}_1, \vec{p}_1)}{\partial t} =& -\frac{\vec{p}_1}{\mathfrak{m}} \nabla_{\vec{x}_1} f^{(1)}(t, \vec{x}_1, \vec{p}_1) + \nabla_{\vec{x}_1} V_1(\vec{x}_1) \cdot \nabla_{\vec{p}_1} f^{(1)}(t, \vec{x}_1, \vec{p}_1) \\ &+ \int\int \nabla_{\vec{x}_1} V_{1,2}(\vec{x}_1, \vec{x}_2) \cdot \nabla_{\vec{p}_1} f^{(2)}(t, \vec{x}_1, \vec{x}_2, \vec{p}_1, \vec{p}_2) d\vec{x}_2 d\vec{p}_2.\end{aligned} \tag{2.41}$$

This can be compared to (2.31), with $\gamma = 0$. One could, in principle, follow the steps in Section 2.3.5 with $\gamma = 0$ to arrive at (2.38). However, in this section, we take a different approach, to illustrate that another classical result can be obtained.

As in the colloidal case, this equation is not closed, since the interaction term depends on $f^{(2)}$. From this, the Boltzmann Equation can be derived as in [9]. First, we define the so-called collision integral as

$$\left(\frac{\partial f^{(1)}}{\partial t} \right)_{\text{coll}} = \int\int \nabla_{\vec{x}_1} V_{1,2}(\vec{x}_1, \vec{x}_2) \cdot \nabla_{\vec{p}_1} f^{(2)}(t, \vec{x}_1, \vec{x}_2, \vec{p}_1, \vec{p}_2) d\vec{x}_2 d\vec{p}_2. \tag{2.42}$$

We can rewrite this as

$$\left(\frac{\partial f^{(1)}}{\partial t} \right)_{\text{coll}} = \int\int \nabla_{\vec{x}_1} V_{1,2}(\vec{x}_1, \vec{x}_2) \cdot (\nabla_{\vec{p}_1} - \nabla_{\vec{p}_2}) f^{(2)}(t, \vec{x}_1, \vec{x}_2, \vec{p}_1, \vec{p}_2) d\vec{x}_2 d\vec{p}_2.$$

In order to proceed, integrating (2.39) over two degrees of freedom, an equation for $f^{(2)}$ results, which can be compared to the result for $f^{(1)}$ in (2.41). This is

$$\begin{aligned}&\left[\frac{\partial}{\partial t}+\frac{\vec{p}_1}{\mathfrak{m}}\cdot\nabla_{\vec{x}_1}+\frac{\vec{p}_2}{\mathfrak{m}}\cdot\nabla_{\vec{x}_2}+\nabla_{\vec{x}_1}V_1(\vec{x}_1)\cdot\nabla_{\vec{p}_1}+\nabla_{\vec{x}_2}V_1(\vec{x}_2)\cdot\nabla_{\vec{p}_2}\right.\\&\qquad\left.+\frac{1}{2}\nabla_{\vec{x}_1}V_{1,2}(\vec{x}_1,\vec{x}_2)\cdot(\nabla_{\vec{p}_1}-\nabla_{\vec{p}_2})\right]f^{(2)}\\&=-\int(\nabla_{\vec{x}_1}V_{1,3}(\vec{x}_1,\vec{x}_3)\cdot\nabla_{\vec{p}_1}+\nabla_{\vec{x}_2}V_{2,3}(\vec{x}_2,\vec{x}_3)\cdot\nabla_{\vec{p}_2})f^{(3)}d\vec{x}_3d\vec{p}_3,\end{aligned}$$

which depends on $f^{(3)}$, and is therefore unclosed. Assuming that $f^{(2)}$ equilibrates much faster than $f^{(1)}$, we set $\frac{\partial f^{(2)}}{\partial t}=0$. Furthermore, assuming that the right-hand side is much smaller than the left-hand side, we neglect the contribution of $f^{(3)}$ and can rearrange the second level equation to give

$$\begin{aligned}&\left[\frac{\vec{p}_1}{\mathfrak{m}}\cdot\nabla_{\vec{x}_1}+\frac{\vec{p}_2}{\mathfrak{m}}\cdot\nabla_{\vec{x}_2}+\nabla_{\vec{x}_1}V_1(\vec{x}_1)\cdot\nabla_{\vec{p}_1}+\nabla_{\vec{x}_2}V_1(\vec{x}_2)\cdot\nabla_{\vec{p}_2}\right]f^{(2)}\\&=-\frac{1}{2}\nabla_{\vec{x}_1}V_{1,2}(\vec{x}_1,\vec{x}_2)\cdot(\nabla_{\vec{p}_1}-\nabla_{\vec{p}_2})f^{(2)}.\end{aligned}$$

Substituting this into (2.42), we obtain

$$\begin{aligned}\left(\frac{\partial f^{(1)}}{\partial t}\right)_{\text{coll}}=-2\int\int&\left(\frac{\vec{p}_1}{\mathfrak{m}}\cdot\nabla_{\vec{x}_1}+\frac{\vec{p}_2}{\mathfrak{m}}\cdot\nabla_{\vec{x}_2}+\nabla_{\vec{x}_1}V_1(\vec{x}_1)\cdot\nabla_{\vec{p}_1}+\nabla_{\vec{x}_2}V_1(\vec{x}_2)\cdot\nabla_{\vec{p}_2}\right)\\&\cdot f^{(2)}(t,\vec{x}_1,\vec{x}_2,\vec{p}_1,\vec{p}_2)d\vec{x}_2d\vec{p}_2.\end{aligned}$$

Following the argument in [9], which is omitted here for brevity, one can define a new reference system, which allows for simplifications of the integral considered. Furthermore, the external potential contributions are neglected and the integration region is simplified. The assumption of molecular chaos (Stosszahlenanzatz) can be made

$$f^{(2)}(t,\vec{x},\vec{p}_1,\vec{p}_2)=f^{(1)}(t,\vec{x},\vec{p}_1)f^{(1)}(t,\vec{x},\vec{p}_2),$$

where, at the collision location, $\vec{x}_1=\vec{x}_2=\vec{x}$. Then the classical Boltzmann term

$$\left(\frac{\partial f^{(1)}}{\partial t}\right)_{\text{coll}}=\int\int|\vec{p}_1-\vec{p}_2|\left(f^{(1)}(t,\vec{x},\vec{p}_1)f^{(1)}(t,\vec{x},\vec{p}_2)-f^{(1)}(t,\vec{x},\vec{p}_1{}')f^{(1)}(t,\vec{x},\vec{p}_2{}')\right)d\Omega d\vec{p}_2$$

arises, where $d\Omega$ is the cross-section of collision.

2.3.7 Generalizations of DDFT and Related Theories

There are several important extensions to DDFT, which introduce crucial physical effects into the models. Some of these extensions are discussed below, such as the inclusion of hydrodynamic interactions, solvent flows, and orientational degrees of freedom. Some additional extensions are the inclusion of sources and sinks, as well as memory effects, which are not discussed further in this section, but can be found in the review article [5]. Moreover, in Chapter 7 an additional extension to particle mixtures is discussed, while in Chapter 8 volume exclusion is added to the DDFT model. After discussing generalizations of DDFT, two separate, but related theories are briefly presented. First, stochastic DDFT is discussed, before introducing Power Functional Theory.

Hydrodynamic Interactions

Hydrodynamic interactions describe the effect that moving colloids have on other colloids through moving the bath around them. These are solvent-mediated interactions and enter the

equation through the diffusion operator. The Smoluchowski equation (2.25) can be modified to that effect, by replacing the constant mobility Γ by the diffusion tensor [21]

$$D_{ij}(\{\vec{x}_k\}) \approx D\mathbf{1}\delta_{ij} + D\left(\delta_{ij}\sum_{l=1,l\neq i}^{N}\omega_{11}(\vec{x}_i-\vec{x}_l) + (1-\delta_{ij})\omega_{12}(\vec{x}_i-\vec{x}_j)\right),$$

where δ_{ij} is the Kronecker delta. The variables ω_{11} and ω_{12} are often defined by the Rotne–Prager approximation [60], which is a two-body approximation for the long range limit. For small particle distances, such an approximation blows up, due to it involving terms of the form $\frac{1}{r}$, where r is the distance between two particles. In certain configurations, for a large number of particles, N, the above sum of such terms diverges. Such approximations can be made to arbitrary order, by taking Taylor expansions of the solution to Stokes flow, and similar approximations can be made for the short-range limit, see [61]. Reconsidering the DDFT calculations in Section 2.3.2, the constant mobility can be replaced by a diffusion tensor, for which $\sqrt{D}$ has to have real entries. Following the argument in [21], the resulting DDFT is

$$\begin{aligned}\frac{\partial\rho}{\partial t} =& D\nabla\cdot\left(\rho\nabla_{\vec{x}}\frac{\delta F[\rho]}{\delta\rho}\right) + D\nabla\cdot\left[\int\rho^{(2)}(\vec{x},\vec{x}')\omega_{11}(\vec{x}-\vec{x}')d\vec{x}'\left(\nabla_{\vec{x}}\frac{\delta F[\rho]}{\delta\rho(\vec{x})}\right)\right]\\ &+ D\nabla\cdot\left[\int\rho^{(2)}(\vec{x},\vec{x}')\omega_{12}(\vec{x}-\vec{x}')\left(\nabla_{\vec{x}'}\frac{\delta F[\rho]}{\delta\rho(\vec{x}')}\right)d\vec{x}'\right].\end{aligned}$$

The two-body density $\rho^{(2)}$ is present in the hydrodynamic interaction, as well as in the particle-particle interaction terms. One way to approximate $\rho^{(2)}$ is to solve the Ornstein–Zernike equation

$$\rho^{(2)}(\vec{x},\vec{x}') = (1+c^{(2)}(\vec{x},\vec{x}'))\rho(\vec{x})\rho(\vec{x}') + \rho(\vec{x}')\int\left((\rho^{(2)}(\vec{x},\vec{x}'')-\rho(\vec{x})\rho(\vec{x}''))c^{(2)}(\vec{x}'',\vec{x}')\right),$$

where

$$c^{(2)}(\vec{x},\vec{x}') = \frac{\delta^2\mathcal{F}_{\text{exc}[\rho]}}{\delta\rho(\vec{x})\delta\rho(\vec{x}')}.$$

This is not computationally efficient, since it doubles the dimension of the problem of interest. Moreover, the equation has to be solved at each time step and each iteration of the PDE solver. This approach is taken in, for example, [21, 23]. Another way of approximating $\rho^{(2)}$ in the hydrodynamic interaction term is by introducing the function $g(\vec{x})$, which is one for all distances $r > 2R$, where R is the particle radius, and zero otherwise. Then $\rho^{(2)}(\vec{x},\vec{x}') = g(|\vec{x}-\vec{x}'|)\rho(\vec{x})\rho(\vec{x}')$ models the two-body density with volume exclusion. However, in the particle–particle interaction term, such volume exclusion is best modelled using FMT, so that $\rho^{(2)}$ is approximated in two different ways, see [62].

The numerical implementation of this approach also harbours additional challenges, since the chosen numerical scheme needs to be able to accurately integrate over a chosen domain, excluding the regions where $|\vec{x}-\vec{x}'| < 2R$ for each pair of positions $\vec{x},\vec{x}'$ in the domain. An additional challenge in including hydrodynamic interactions into DDFT models is the added nonlocality that such terms introduce, which also propagates into no-flux boundary conditions of such models. There are a range of applications in which the inclusion of these effects is important, such as in sedimentation [63], oscillations in optical traps [64], and microswimmer modelling [65]. Further applications are discussed in [5]. Extended models may be necessary for some of such applications, for example by combining hydrodynamic interactions with inertial effects, as done in [62].

Inclusion of Solvent Flows

There are two types of flows that have been considered in DDFT literature. The most straightforward way to include solvent flows into DDFT problems is by requiring such flow fields to be conservative, i.e., they need to satisfy

$$\vec{v} = -\nabla V_{\text{fl}},$$

where V_{fl} is a potential. Such potential flows can easily be incorporated into the existing DDFT formalism by redefining V_1 to include V_{fl}. This has been shown by Rauscher [66], who first introduced advected DDFT. Note that this definition necessarily excludes flow profiles with a rotational component.
A more complicated case arises when one considers shear flow. This is demonstrated in [67] by considering shear flow with two parallel plane walls, with flow profile

$$\vec{v} = x_2 \dot{\gamma} \vec{e}_{x_1},$$

where $\dot{\gamma}$ is a shear rate, and $\vec{e}_{x_1}$ a unit vector in the direction of x_1. While the term $\nabla \cdot (\rho \vec{v})$ is zero, the effect of shear flow is clearly non-zero, an issue which is rooted in the breakdown of the adiabatic approximation. A phenomenological correction for this issue was proposed by Brader and Krüger [67] using the mean-field term

$$\vec{v} = \int \rho(\vec{x}') \mathbf{K}(\vec{x} - \vec{x}') d\vec{x},$$

which describes the effect of the shear solvent flow on the particle-particle interactions via a flow kernel $\mathbf{K}$. Applications of the inclusion of solvent flow fields include modelling time dependent shear in sedimentation, the interplay between shear and crystallization, laning instability, and effects of shear on nonequilibrium phase transitions, as discussed in [5].

Orientational Degrees of Freedom

The inclusion of orientational degrees of freedom is interesting for a range of applications, including modelling particles of non-spherical shapes, magnetic or active particles. We refer to [5] for a discussion of the different models and references for various modelling approaches. Applications of such models include protein solvation [68], plate-like particles [69], and liquid crystals [70, 71].

As discussed in [5], a standard DDFT model for nonspherical particles can be defined as

$$\frac{\partial \varrho}{\partial t} = D \nabla_{\vec{x}} \cdot \left(\varrho \nabla_{\vec{x}} \frac{\delta \mathcal{F}}{\delta \varrho} \right) + D_R \nabla_{\vec{u}} \cdot \left(\varrho \nabla_{\vec{u}} \frac{\delta \mathcal{F}}{\delta \varrho} \right),$$

where the particle density ϱ depends on an additional, orientational, dimension, defined by $\vec{u}$, which describes an angle with respect to an axis of symmetry, corresponding to the particle orientation. In case of active particles, this is the direction of self-propulsion. Note that the spatial diffusion coefficient D can depend on $\vec{u}$ as well, and that orientational diffusion with coefficient D_R is included in the model. An additional challenge with this model is the introduction of (at least one) additional dimension, which is computationally expensive.

In order to model active particles with arbitrary shapes, as described in [72], one can extend the above model to include cross-terms of translational and rotational effects

$$\frac{\partial \varrho}{\partial t} = \begin{bmatrix} \nabla_{\vec{x}} \\ \nabla_{\vec{\omega}} \end{bmatrix} \cdot \left(D(\vec{x}, \vec{\omega}) \varrho \left(\begin{bmatrix} \nabla_{\vec{x}} \\ \nabla_{\vec{\omega}} \end{bmatrix} \frac{\delta \mathcal{F}}{\delta \varrho} - \begin{bmatrix} \vec{F}(\vec{x}, \vec{\omega}, t) \\ \vec{T}(\vec{x}, \vec{\omega}, t) \end{bmatrix} \right) \right),$$

where $\vec{\omega}$ is an extension of $\vec{u}$ to include an additional angular direction, in order to capture arbitrary particle shapes. The contribution of the active force is denoted by $\vec{F}$, while the torque

contribution is denoted by $\vec{T}$. Such models are only valid for weak activity, since, as is the case for driven colloids, the particle correlations are too distorted for the adiabatic approximation to be valid. One can extend this model to include hydrodynamic interactions and inertial effects, as done in [73].

As is the case for driven colloids, instead of adding an additional degree of freedom, an effective contribution to the free energy can be included. This contribution approximates the steady state active force present in the system.

In order to model magnetic particles, instead of adding another degree of freedom, one can consider particles with spin $\vec{\sigma}_s$. This can enter a DDFT model through a nonlocal interaction term, where the spin contribution is described by the Heisenberg model

$$U_2(|\vec{x}-\vec{x}'|,\vec{u},\vec{u}') = \mathcal{J}(|\vec{x}-\vec{x}'|)\vec{\sigma}_s\cdot\vec{\sigma}_s',$$

where $\mathcal{J}$ describes the ferromagnetic ordering.

Nonisothermal Systems

In order to include temperature gradients into DDFT systems, an energy density or entropy density field has to be included, see [74, 75]. Including the energy density field, as done in [74], introduces a stress tensor into the DDFT model, which originates from considering an additional momentum moment in deriving the inertial DDFT, compare with the derivation in Section 2.3.5. Such terms are of the form $\mathbf{E} = \frac{1}{\mathrm{m}^2}\int(\vec{p}\otimes\vec{p})\,f(\vec{x},\vec{p})d\vec{p}$, and describe the kinetic energy contribution to the system. Note that considering such terms introduces an additional equation for $\mathbf{E}$ into the system. This is a tensor, which, through symmetry, adds six (as opposed to nine) additional degrees of freedom to the problem. Taking the trace of such terms, this reduces to the average, non-directional kinetic energy, or temperature. Note that this $\mathbf{E}$ refers to the kinetic energy of the particles, while the constant T denotes the temperature of the solvent. While these are not equal, $\mathbf{E}$ should relax to T over time.

When considering the entropy density, one can consider an alternative equilibrium energy functional, which depends on the entropy, as well as the particle density ρ, see [75]. From this point a dynamic equation for the entropy density can be derived. Details of such calculations, as well as additional references for work on nonisothermal systems, can be found in [5].

Stochastic DDFT

In this section, we briefly discuss two stochastic DDFTs which can be derived from Langevin equations. These look equivalent, but are different in meaning, depending on the definition of ρ. The difference between stochastic and deterministic DDFT is that while in the deterministic derivations averages over both the initial conditions and the particle paths are taken, in the stochastic derivation only the initial condition is averaged over.

We first consider the coarse-grained stochastic DDFT. The starting point of the derivation in [76] is a Smoluchowski equation of the form (2.25). Instead of taking ensemble averages, as in Section 2.3.4, we divide space into cells of volume v_ρ, that contain many particles. After addressing a range of technical details, the authors arrive at a Fokker–Planck equation of the form

$$\frac{\partial P(t,[\rho])}{\partial t} = -\frac{D}{k_B T}\int\frac{\delta}{\delta\rho(t,\vec{x})}\nabla\cdot\left(\rho(t,\vec{x})\nabla\left(\frac{\delta}{\delta\rho}+\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\right)\right)P(t,[\rho])d\vec{x},$$

where ρ is the coarse-grained density, defined on the volume cells v_ρ and $\mathcal{F}[\rho]$ the coarse-grained free energy. From the Fokker–Planck equation the underlying stochastic process in ρ can be

derived, given by

$$\frac{\partial \rho(t,\vec{x})}{\partial t} = \Gamma \nabla \cdot \left(\rho(t,\vec{x}) \nabla \frac{\delta \mathcal{F}[\rho]}{\delta \rho} \right) + \nabla \cdot \left(\sqrt{\rho(t,\vec{x})} \vec{\xi}(t,\vec{x}) \right), \tag{2.43}$$

where $\vec{\xi}$ is defined as in Section 2.3.4.

A second version of the stochastic DDFT was derived by Dean [77]. The starting point for this derivation is the Langevin equation for N particles (2.22) and its corresponding equation for $\hat{\rho}$ as in the deterministic Langevin derivation in Section 2.3.4. This is in the form of a stochastic DDFT, when making the adiabatic approximation, so that we have

$$\frac{\partial \hat{\rho}(\vec{x},t)}{\partial t} = \Gamma \nabla \cdot \left(\hat{\rho}(t,\vec{x}) \nabla \frac{\delta \mathcal{F}[\hat{\rho}]}{\delta \rho} \right) + \nabla \cdot \left(\sqrt{\hat{\rho}(t,\vec{x})} \vec{\xi}(t,\vec{x}) \right),$$

where $\hat{\rho}(t,\vec{x}) = \sum_{i=1}^{N} \delta(\vec{x}_i(t) - \vec{x})$ is now a distribution of delta functions. Note that this result is exact, while the coarse-grained stochastic DDFT (2.43) is not.

Power Functional Theory

Power Functional Theory (PFT) is theory related to classical DDFT that describes particle dynamics in terms of a time-dependent *free power functional* instead of the equilibrium free energy. This is an exact theory, which enables the description of *superadiabatic* effects and in which the flux j is a central variable, alongside the density ρ. This is contrasted by standard DDFT, which is based on the adiabatic approximation, and which is described in terms of the variable ρ only. While DDFT models are sufficient to describe many slowly evolving systems, it is unable to accurately capture many effects arising from time-dependent particle correlations and nonconservative forces, such as dissipation of power and shear effects. The account below follows the presentation in [78] of results published in [79]. Discussions of different extensions and applications of PFT can be found in review article [5].

A variational principle for PFT in terms of a time-dependent free power functional, denoted by $\mathcal{R}$, can be derived. This is comparable to the variational principle in DFT in terms of the free energy, as introduced in Section 2.2. Details of the derivation from a microscopic description of the system and a discussion of links to DFT are omitted here in the interest of brevity and can be found in [79]. The PFT variational principle is given by

$$\frac{\delta \mathcal{R}[\rho, \mathrm{j}]}{\delta \mathrm{j}(t,\vec{x})} = 0. \tag{2.44}$$

The free power can be split into the following contributions

$$\mathcal{R} = \mathcal{P} + \frac{d\mathcal{F}[\rho]}{dt} - \mathcal{X},$$

which are a *dissipation functional* $\mathcal{P}$, an adiabatic contribution from the equilibrium free energy $\mathcal{F}$, and external powers $\mathcal{X}$. We can split $\mathcal{P}$ into an ideal gas contribution $\mathcal{P}_{\mathrm{id}}$ and an excess, or superadiabatic, part $\mathcal{P}_{\mathrm{exc}}$, analogous to the process for $\mathcal{F}$. The ideal contribution is given explicitly by

$$\mathcal{P}_{\mathrm{id}} = \frac{\gamma}{2} \int \frac{\mathrm{j}(t,\vec{x})^2}{\rho(t,\vec{x})} d\vec{x},$$

while $\mathcal{P}_{\mathrm{exc}}$ needs to be approximated. One can show that the time derivative of $\mathcal{F}$ is

$$\frac{d\mathcal{F}[\rho]}{dt} = \int \left(\nabla \frac{\delta \mathcal{F}[\rho]}{\delta \rho(t,\vec{x})} \right) \cdot \mathrm{j}(t,\vec{x}) d\vec{x},$$

noting that $\mathcal{F}$ depends on time only through the time-dependent choice of ρ. The external power $\mathcal{X}$ is defined by

$$\mathcal{X} = \int \left(\mathrm{j}(t,\vec{x}) \cdot \vec{f}_1 - \rho(t,\vec{x}) \frac{dV_1(t,\vec{x})}{dt} \right) d\vec{x},$$

where $\vec{f}_1$ denotes external forces, including conservative forces ∇V_1 and nonconservative forces. Evaluating (2.44) explicitly results in

$$\begin{aligned} \frac{\delta \mathcal{R}[\rho,\mathrm{j}]}{\delta \mathrm{j}(t,\vec{x})} &= \frac{\delta \mathcal{P}_{\mathrm{id}}}{\delta \mathrm{j}} + \frac{\delta \mathcal{P}_{\mathrm{exc}}}{\delta \mathrm{j}} + \frac{\delta \dot{\mathcal{F}}_{\mathrm{id}}}{\delta \mathrm{j}} + \frac{\delta \dot{\mathcal{F}}_{\mathrm{exc}}}{\delta \mathrm{j}} - \frac{\delta \mathcal{X}}{\delta \mathrm{j}} \\ &= \frac{\gamma \mathrm{j}}{\rho} + \frac{\delta \mathcal{P}_{\mathrm{exc}}}{\delta \mathrm{j}} + k_B T \nabla \ln \rho + \frac{\delta \dot{\mathcal{F}}_{\mathrm{exc}}}{\delta \mathrm{j}} - \vec{f}_1 = 0, \end{aligned}$$

where $\dot{\mathcal{F}} = \frac{d\mathcal{F}}{dt}$. Rearranging for j and recalling the continuity equation

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathrm{j},$$

we have an equation of motion based the formally exact power functional $\mathcal{R}$. Using that $\rho \nabla \ln \rho = \nabla \rho$, this is

$$\frac{\partial \rho}{\partial t} = \frac{1}{\gamma} \nabla \cdot \left(k_B T \nabla \rho - \rho \vec{f}_1 + \rho \nabla \frac{\delta \mathcal{F}_{\mathrm{exc}}}{\delta \rho} + \rho \frac{\delta \mathcal{P}_{\mathrm{exc}}}{\delta \mathrm{j}} \right).$$

From this formulation it is evident that the additional terms in comparison to standard DDFT are the nonconservative forces contained in $\vec{f}_1$ and the superadiabatic contributions from $\mathcal{P}_{\mathrm{exc}}$. Neglecting those terms recovers standard DDFT. The power dissipation functional $\mathcal{P}_{\mathrm{exc}}$ is generally a function of both ρ and j, and nonlocal in space and time. The exact form of $\mathcal{P}_{\mathrm{exc}}$ is unknown, but different approximations are available. In [80], a change of variables from j to $\vec{v}$ is made and a power series expansion of $\mathcal{P}_{\mathrm{exc}}$ is taken to derive such an approximation. The second order term in this expansion has the form

$$\mathcal{P}_{\mathrm{exc}} = \int \int \int_0^t \rho(t,\vec{x}) \nabla \vec{v}(t,\vec{x}) : M(t-t', \vec{x}-\vec{x}') : \nabla \vec{v}(t',\vec{x}') \rho(t',\vec{x}') dt' d\vec{x}' d\vec{x}, \tag{2.45}$$

where M is a fourth-rank tensor. This is able to capture shear and viscous superadiabatic forces. Depending on the model problem at hand, different simplifications and extensions to (2.45) can be made. In [80] and [81], simplified versions of (2.45) were obtained by assuming the model to be Markovian and local in space, while in [82] the model was extended to higher orders in time. Using this extended form of $\mathcal{P}_{\mathrm{exc}}$, the authors were able to identify viscous and structural contributions to the superadiabatic force in shear flow. Finally, in [83], a systematic way to split superadiabatic forces into structural forces and flow contributions was introduced, and fourth order terms in $\vec{v}$ of the expansion of $\mathcal{P}_{\mathrm{exc}}$ were included to capture these effects accurately. Approximations to $\mathcal{P}_{\mathrm{exc}}$, such as those developed in [80, 81, 83], enable PFT to describe a larger variety of physical phenomena than standard DDFT.

Chapter 3

Theoretical Results and Numerical Methods for PDEs

In this chapter we discuss theoretical results for PDEs and numerical methods for solving them. These are important in their own right, as well as the foundation for the work on PDE-constrained optimization in the next chapters. We first discuss PDE theory in Section 3.1. Then, in Section 3.2 we introduce a class of numerical methods to solve PDEs, that will be used throughout this thesis – spectral methods. Moreover, other commonly used methods for solving PDEs are discussed. In Section 3.3 the implementation of a specific spectral method in MATLAB is presented. This enables the solution of integro-PDEs arising from DDFT, as is demonstrated by the results achieved with this implementation, which are shown in Section 3.3. While this implementation is based on [84], extensions are made, which allow us to tackle additional boundary conditions, as well as problems in three dimensions.

3.1 Theoretical Results on PDEs

Theoretical results for PDEs concern the existence, uniqueness, and regularity of their solutions. This is important to investigate, so that we know whether a solution to the problem at hand exists and whether there is more than one solution. Moreover, we would like to know whether small changes in initial conditions can cause large differences in the PDE solution. It can be important for practical applications to know if there are two different solutions to a given problem and how sensitive a solution is to the given starting point. It is also important to know how 'smooth' a PDE solution may be, since this can again be relevant in applications. For example, it could be important to know if a solution has sudden jumps. If a PDE model describes material durability under changing temperature, a sudden jump could describe a sudden fracturing of the material. Such considerations can also inform the choice and construction of reliable numerical methods.

We first introduce some general results on function spaces, then discuss linear elliptic and parabolic equations before briefly touching on results for nonlinear PDEs. In this section, a brief collection of such results is given to inform the development of numerical methods for PDEs later in this chapter, as well as results on PDE-constrained optimization in the next. Definitions and theorems in the following sections are direct quotes from the stated reference. Proofs are generally omitted in this chapter, but can be found in the corresponding reference text from which a theorem was cited. Most of this section follows [85], with some use made of [86] and [87].

3.1.1 Some Required Results on Function Spaces

We begin with a few definitions and prerequisites for the following discussions. This section first introduces Banach and Hilbert spaces, before discussing dual spaces and the Riesz Representation Theorem for such spaces, and ends with introducing the necessary framework to

analyse PDEs of interest, namely Sobolev spaces.

Normed Spaces

We begin by introducing the notion of a normed space.

Definition 3.1.1 (Normed Space [86, Chapter 2])**.** Let Y be a linear space over $\mathbb{R}$. A mapping $\|\cdot\| : Y \to \mathbb{R}$ is called a norm on Y if the following hold for all $y, v \in Y$ and $\lambda \in \mathbb{R}$:

1. $\|y\| \geq 0$, and $\|y\| = 0 \Leftrightarrow y = 0$,
2. $\|y + v\| \leq \|y\| + \|v\|$ (triangle inequality),
3. $\|\lambda y\| = |\lambda| \|y\|$ (homogeneity).

If $\|\cdot\|$ is a norm on Y, then $\{Y, \|\cdot\|\}$ is called a (real) normed space.

Definition 3.1.2 (Banach Space [86, Chapter 2])**.** A normed space $\{Y, \|\cdot\|\}$ is said to be complete if every Cauchy sequence in Y converges, i.e., has a limit in Y. A complete normed space is called a Banach space.

Another important concept is that of a scalar product.

Definition 3.1.3 (Scalar Product, Pre-Hilbert Space [86, Chapter 2])**.** Let H be a real linear space. A mapping $(\cdot, \cdot) : H \to \mathbb{R}$ is called a scalar product on H if the following conditions are satisfied for all $y, v, y_1, y_2 \in H$ and $\lambda \in \mathbb{R}$:

1. $(y, y) \geq 0$, and $(y, y) = 0 \Leftrightarrow y = 0$,
2. $(y, v) = (v, y)$,
3. $(y_1 + y_2, v) = (y_1, v) + (y_2, v)$,
4. $(\lambda y, v) = \lambda (y, v)$.

If $(\cdot, \cdot)$ is a scalar product on H, then $\{H, (\cdot, \cdot)\}$ is called a pre-Hilbert space.

Definition 3.1.4 (Hilbert Space [86, Chapter 2])**.** A pre-Hilbert space $\{H, (\cdot, \cdot)\}$ is called a Hilbert space if it is complete with respect to the norm

$$\|y\| := \sqrt{(y, y)}.$$

Note that Hilbert spaces show up frequently in the following discussion. Let us now look at a special normed space: the space of linear operators.

Definition 3.1.5 (Linear Operator [87, Chapter 1])**.** Let Y, V be normed real vector spaces with norms $\|\cdot\|_Y, \|\cdot\|_V$.

1. A mapping $A : Y \to V$ is called a linear operator if it satisfies

   $$A(\lambda y + \mu v) = \lambda A y + \mu A v \quad \forall y, v \in Y, \lambda, \mu \in \mathbb{R}.$$

   The range of A is defined by $R(A) := \{v \in V : \exists y \in Y : v = Ay\}$ and the null space of A by $N(A) := \{y \in Y : Ay = 0\}$.

2. By $\mathcal{L}(Y, V)$ we denote the space of all linear operators $A : Y \to V$ that are bounded in the sense that

   $$\|A\|_{Y,V} := \sup_{\|y\|_Y = 1} \|Ay\|_V < \infty.$$

   $\mathcal{L}(Y, V)$ is a normed space with the operator norm $\|\cdot\|_{Y,V}$.

Theorem 3.1.1. [87, Chapter 1] If V is a Banach space then $\mathcal{L}(Y, V)$ is a Banach space.

We can now introduce the notion of a dual space, and the related concept of dual operators, leading us to a very important theorem: the Riesz Representation Theorem.

Definition 3.1.6 (Linear Functionals, Dual Spaces [87, Chapter 1])**.**

1. Let Y be a Banach space. A bounded linear operator $y^* : Y \to \mathbb{R}$, i.e., $y^* \in \mathcal{L}(Y, \mathbb{R})$, is called a bounded linear functional on Y.

2. The space $Y^* := \mathcal{L}(Y, \mathbb{R})$ of linear functionals on Y is called dual space of Y and is (by Theorem 3.1.1) a Banach space with the operator norm

$$\|y^*\| := \sup_{\|y\|_Y = 1} |y^*(y)|.$$

3. We use the notation

$$\langle y^*, y \rangle_{Y^*,Y} := y^*(y).$$

The term $\langle y^*, y \rangle_{Y^*,Y}$ is called the dual pairing of Y^* and Y.

Definition 3.1.7 (Dual Operator [87, Chapter 1])**.** Let Y, V be Banach spaces. Then for an operator $A \in \mathcal{L}(Y, V)$ the dual operator $A^* \in \mathcal{L}(V^*, Y^*)$ is defined by

$$\langle A^* v, y \rangle_{Y^*,Y} = \langle v, Ay \rangle_{V^*,V} \quad \forall v \in V^*, y \in Y.$$

It is easy to check that $\|A^*\|_{V^*,Y^*} = \|A\|_{V,Y}$.

A closely related concept is the Hilbert space adjoint, which is defined as follows:

Definition 3.1.8 (Adjoint Operator [86, Chapter 2])**.** Let real Hilbert spaces $\{Y, (\cdot,\cdot)_Y\}$ and $\{V, (\cdot,\cdot)_V\}$ as well as an operator $A \in \mathcal{L}(Y, V)$ be given. An operator A^* is called the Hilbert space adjoint or adjoint of A if

$$(v, Ay)_V = (A^* v, y)_Y \quad \forall y \in Y, \quad \forall v \in V.$$

Definition 3.1.9 (Reflexive Banach Space [87, Chapter 1])**.** A Banach space Y is called reflexive if the mapping $y \in Y \mapsto \langle \cdot, y \rangle_{Y^*,Y} \in (Y^*)^*$ is surjective, i.e., if for any $y^{**} \in (Y^*)^*$ there exists $y \in Y$ with

$$\langle y^{**}, y^* \rangle_{(Y^*)^*,Y^*} = \langle y^*, y \rangle_{Y^*,Y} \quad \forall y^* \in Y^*.$$

Finally, we are able to state the Riesz Representation Theorem.

Theorem 3.1.2 (Riesz Representation Theorem [87, Chapter 1])**.** The dual space H^* of a Hilbert space H is isometric to H itself. More precisely, for every $v \in H$ the linear functional y^* defined by

$$\langle y^*, y \rangle_{H^*,H} := (v, y)_H \quad \forall y \in H$$

is in H^* with norm $\|y^*\|_{H^*} = \|v\|_H$. Vice versa, for any $y^* \in H^*$ there exists a unique $v \in H$ such that

$$\langle y^*, y \rangle_{H^*,H} := (v, y)_H \quad \forall y \in H$$

and $\|y^*\|_{H^*} = \|v\|_H$. In particular, a Hilbert space is reflexive.

Sobolev Spaces

Let $\Omega \subset \mathbb{R}^N$ be a non-empty, bounded, Lebesgue measurable set. We can define the following spaces:

Definition 3.1.10 (Lebesgue Spaces [86, Chapter 2])**.** We denote by $L^p(\Omega)$, $1 \leq p < \infty$, the linear space of all (equivalence classes of) Lebesgue measurable functions y that satisfy

$$\int_\Omega |y(x)|^p dx < \infty.$$

With the norm

$$\|y\|_{L^p(\Omega)} = \left(\int_\Omega |y(x)|^p dx \right)^{1/p},$$

the spaces $L^p(\Omega)$ are reflexive Banach spaces. This means we have $(L^p)^* = L^q$, $\frac{1}{p} + \frac{1}{q} = 1$ and $((L^p)^*)^* = L^p$. Moreover, the space L^2 with the inner product $(y, v)_{L^2} := \int_\Omega yv dx$ is a Hilbert space [87, Chapter 1].

Definition 3.1.11. [86, Chapter 2] We denote by $L^\infty(\Omega)$ the Banach space of all (equivalence classes of) Lebesgue measurable and essentially bounded functions, equipped with the norm

$$\|y\|_{L^\infty(\Omega)} = \operatorname{ess\,sup}_{x \in \Omega} |y(x)| := \inf_{|F|=0} \left(\sup_{x \in \Omega \setminus F} |y(x)| \right).$$

As defined in [87, Chapter 1], the space of k-times continuously differentiable functions is given by

$$C(\Omega) = \{y : \Omega \to \mathbb{R} : y \text{ continuous}\},$$
$$C^k(\Omega) = \{y \in C(\Omega) : D^\alpha y \in C(\Omega) \text{ for } |\alpha| \leq k\}.$$

One important example of such spaces is C_0^∞, the space of test functions, which is the space of infinitely differentiable functions of compact support, i.e., functions such that their derivatives vanish on $\partial\Omega$, the boundary of the domain. For bounded Ω, $C(\bar{\Omega})$ is a Banach space and the associated norm is $\|y\|_{C(\bar{\Omega})} = \sup_{x \in \bar{\Omega}} |y(x)|$. Then the spaces $C^k(\bar{\Omega})$ are equipped with the norm $\|y\|_{C^k(\bar{\Omega})} := \sum_{|\alpha| \leq k} \|D^\alpha y\|_{C(\bar{\Omega})}$.

Next, we require the definition of a *Lipschitz domain*, which can be found in [86, Chapter 2] and [87, Chapter 1] and is omitted here. The essential requirement is that locally the boundary of the considered domain can be represented by a Lipschitz continuous function. If this is given, then the Gauss–Green Theorem (i.e., integration by parts) is valid, which is of crucial importance in defining weak derivatives, and, later on, the weak formulation of PDE problems. We furthermore denote locally (on compact subsets of Ω) integrable functions by L_{loc} and define a so-called *multi-index*, which is a vector $\alpha = [\alpha_1, \alpha_2, ..., \alpha_N]$, where α_i denotes the number of derivatives with respect to x_i. We can then introduce the notion of a weak derivative:

Definition 3.1.12. [86, Chapter 2] Let $y \in L^1_{\text{loc}}(\Omega)$ and some multi-index α be given. If a function $w \in L^1_{\text{loc}}(\Omega)$ satisfies

$$\int_\Omega y(x) D^\alpha v(x) dx = (-1)^{|\alpha|} \int_\Omega w(x) v(x) dx \quad \forall v \in C_0^\infty(\Omega),$$

then w is called the weak derivative of y (associated with α).

Now, finally we can define Sobolev spaces as follows:

Definition 3.1.13 (Sobolev Spaces $W^{k,p}(\Omega)$ [86, Chapter 2])**.** Let $1 \leq p < \infty$ and $k \in \mathbb{N}$. We denote by $W^{k,p}(\Omega)$ the linear space of all functions $y \in L^p(\Omega)$ having weak derivatives $D^\alpha y$ in $L^p(\Omega)$ for all multi-indices α of length $|\alpha| \leq k$, endowed with the norm

$$\|y\|_{W^{k,p}(\Omega)} = \left(\sum_{|\alpha| \leq k} \int_\Omega |D^\alpha y(x)|^p dx \right)^{1/p}.$$

Analogously, for $p = \infty$, the space $W^{k,\infty}(\Omega)$ can be defined, equipped with the norm

$$\|y\|_{W^{k,\infty}(\Omega)} = \max_{|\alpha| \leq k} \|D^\alpha y\|_{L^\infty(\Omega)}.$$

For $p = 2$ we define $H^k(\Omega) := W^{k,2}(\Omega)$, which are Hilbert spaces with inner product $(y, v)_{H^k}(\Omega) = \sum_{|\alpha| \leq k} (D^\alpha y, D^\alpha v)_{L^2(\Omega)}$, see [87, Chapter 1]. In particular, $H^1(\Omega)$ is the space of functions $y \in L^2(\Omega)$ with all first partial derivatives being in $L^2(\Omega)$ as well, and which is equipped with the norm $\|y\|_{H^1(\Omega)} = \left(\int_\Omega \left(y^2 + |\nabla y|^2\right) dx\right)^{1/2}$, see [86, Chapter 2].

Definition 3.1.14. [86, Chapter 2] The closure of $C_0^\infty(\Omega)$ in $W^{k,p}(\Omega)$ is denoted by $W_0^{k,p}(\Omega)$. Moreover, we define $H_0^k(\Omega) := W_0^{k,2}(\Omega)$.

The $W_0^{k,p}(\Omega)$ are subspaces of $W^{k,p}(\Omega)$ and, by definition, $C_0^\infty(\Omega)$ is dense in $H_0^1(\Omega)$, see [86, Chapter 2]. Now we are in a position to address the following issue: since functions in Sobolev spaces are equivalent to each other if they differ only on a set of measure zero, a complication arises when we want to consider the boundary of a set, $\partial\Omega$, since it is a set of measure zero. In order to define the boundary values of functions in Sobolev spaces, we need the following Trace Theorem:

Theorem 3.1.3 (Trace Theorem [86, Chapter 2])**.** Let $\Omega \subset \mathbb{R}^N$ be a bounded Lipschitz domain and let $1 \leq p < \infty$. Then there exists a linear and continuous mapping $\tau : W^{1,p}(\Omega) \to L^p(\partial\Omega)$ such that for all $y \in W^{1,p}(\Omega) \cap C(\bar{\Omega})$ we have $(\tau y)(x) = y(x)$ for all $x \in \partial\Omega$.

Furthermore, we have the following embedding results for Sobolev spaces:

Theorem 3.1.4. [86, Chapter 2] Let $\Omega \subset \mathbb{R}^N$ be a bounded Lipschitz domain. Moreover, let $1 \leq p < \infty$ and let m be a nonnegative integer. Then the following embeddings exist and are continuous:

- for $mp < N$: $W^{m,p}(\Omega) \hookrightarrow L^q(\Omega)$ if $1 \leq q \leq \frac{Np}{N-mp}$,
- for $mp = N$: $W^{m,p}(\Omega) \hookrightarrow L^q(\Omega)$ if $1 \leq q \leq \infty$,
- for $mp > N$: $W^{m,p}(\Omega) \hookrightarrow C(\bar{\Omega})$.

Finally, it is useful to define the dual space of $H_0^1(\Omega)$. We denote this space by $H^{-1}(\Omega)$ and note that $H_0^1(\Omega) \subset L^2(\Omega) \subset H^{-1}(\Omega)$. We can define a norm on H^{-1} and characterize the space as follows:

Definition 3.1.15. [85, Chapter 5] If $f \in H^{-1}(\Omega)$, we define the norm

$$\|f\|_{H^{-1}(\Omega)} := \sup \left\{ \langle f, y \rangle | y \in H_0^1(\Omega), \|y\|_{H_0^1(\Omega)} \leq 1 \right\}.$$

Theorem 3.1.5 (Characterization of H^{-1} [85, Chapter 5])**.**

1. Assume $f \in H^{-1}(\Omega)$. Then there exist functions $f^0, f^1, ..., f^N$ in $L^2(\Omega)$ such that

$$\langle f, v \rangle = \int_\Omega f^0 v + \sum_{i=1}^N f^i \frac{\partial v}{\partial x_i} dx, \quad v \in H_0^1(\Omega), \tag{3.1}$$

   where x_i are the elements of the vector x.

2. Furthermore,

$$\|f\|_{H^{-1}(\Omega)} = \inf \left\{ \left(\int_\Omega \sum_{i=0}^N |f^i|^2 dx \right)^{1/2} \middle| f \text{ satisfies (3.1) for } f^0, f^1, ..., f^N \in L^2(\Omega) \right\}.$$

3. In particular, we have

$$(v, y)_{L^2(\Omega)} = \langle v, y \rangle$$

for all $y \in H_0^1(\Omega), v \in L^2(\Omega) \subset H^{-1}(\Omega)$.

When (3.1) holds, we denote $f = f^0 - \sum_{i=1}^N f^i_{x_i}$.

3.1.2 Weak Solutions to Linear Elliptic PDEs

Weak Solution to the Poisson Equation

Following [86, Chapter 2], we consider a simple example to illustrate the methods used to analyse elliptic PDEs, before moving to parabolic equations. The first problem considered is the Poisson equation

$$\begin{aligned} -\nabla^2 y &= f \quad \text{in } \Omega, \\ y &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{3.2}$$

where $f \in L^2(\Omega)$, $\Omega \subset \mathbb{R}^N$ is a bounded Lipschitz domain, and $\partial\Omega$ is the boundary of Ω. A solution $y \in C^2(\Omega) \cap C^1(\bar{\Omega})$ to (3.2) is called a strong, or classical, solution. However, this does not allow for solutions to (3.2) that admit all possible $f \in L^2(\Omega)$, which could, for example, be a piecewise continuous function. Therefore, we are interested in deriving the weak form of this problem, to which we can find a weak solution $y \in H_0^1(\Omega)$. The weak form is found by multiplying (3.2) by a test function $v \in C_0^\infty(\Omega)$ and integrating over the domain. Note that all boundary terms vanish, since v has compact support. Since $C_0^\infty(\Omega)$ is dense in $H_0^1(\Omega)$, by (3.1.14), and the equation depends continuously on v, we arrive at the weak formulation of the Poisson problem

$$\int_\Omega \nabla y \cdot \nabla v dx = \int_\Omega f v dx \quad \forall v \in H_0^1(\Omega),$$

with weak solution $y \in H_0^1(\Omega)$. We can generalize this by denoting $V = H_0^1(\Omega)$ and define a bilinear map $a : V \times V \to \mathbb{R}$:

$$a[y, v] := \int_\Omega \nabla y \cdot \nabla v dx,$$

and the functional $F : V \to \mathbb{R}$:

$$F(v) := \int_\Omega f v dx,$$

so that the general form of the variational formulation becomes

$$a[y, v] = F(v) \quad \forall v \in V, \tag{3.3}$$

and $F \in V^*$, where V^* is the dual space of V.

The following theorem is very important for proving existence of solutions for elliptic PDEs.

Theorem 3.1.6 (Lax and Milgram [86, Chapter 2])**.** Let V be a real Hilbert space, and let $a : V \times V \to \mathbb{R}$ denote a bilinear form. Moreover, suppose that there exist positive constants α_0 and β_0 such that the following conditions are satisfied for all $v, y \in V$:

$$\begin{aligned} |a[y, v]| &\leq \alpha_0 \|y\|_V \|v\|_V \quad \text{(boundedness)}, \\ a[y, y] &\geq \beta_0 \|y\|_V^2 \quad \text{(V-ellipticity)}. \end{aligned}$$

Then for every $F \in V^*$ the variational equation (3.3) admits a unique solution $y \in V$. Moreover, there is some constant $c_a > 0$, which does not depend on F, such that $\|y\|_V \leq c_a \|F\|_{V^*}$.

Using this theorem, the Cauchy–Schwarz and Friedrichs inequalities, we can show that the Poisson problem admits a unique weak solution, see [86, Chapter 2] for details.

In the same way, the existence of a unique weak solution to a similar problem with Robin boundary conditions can be shown:

$$\begin{aligned} -\nabla^2 y + c_0 y &= f \quad \text{in } \Omega, \\ \frac{\partial y}{\partial n} + \alpha y &= g \quad \text{on } \partial\Omega, \end{aligned}$$

where $f \in L^2(\Omega)$, $g \in L^2(\Omega)$, $c_0 \in L^\infty(\Omega)$, and $\alpha \in L^\infty(\Omega)$. The challenge is to show that the boundedness and V-ellipticity conditions in Theorem 3.1.6 hold. For this, the generalized Poincaré inequality and the Trace Theorem (Theorem 3.1.3) may be applied, as shown in [86, Chapter 2].

Weak Solutions to General Linear Elliptic PDEs

We have now seen one example of an elliptic equation and results on the existence and uniqueness of a weak solution. In this section, following [85, Chapter 6], more general elliptic equations are considered. For an open, bounded $\Omega \subset \mathbb{R}^N$, $y : \bar{\Omega} \to \mathbb{R}$, these are of the form

$$\begin{aligned} Ly &= f \quad \text{in } \Omega, \\ y &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{3.4}$$

where Ly is defined in divergence form

$$Ly = -\sum_{i,j=1}^{N} \left(a_{ij}(x) y_{x_i}\right)_{x_j} + \sum_{i=1}^{N} b_i(x) y_{x_i} + c(x) y, \tag{3.5}$$

where y_{x_i} denotes the derivative of y with respect to component x_i of the vector x, and $a_{ij} = a_{ji}$ by assumption.

Definition 3.1.16 (Ellipticity Condition [85, Chapter 6])**.** We say the partial differential operator L is (uniformly) elliptic if there exists a constant $\theta > 0$ such that

$$\sum_{i,j=1}^{N} a_{ij}(x)\xi_i\xi_j \geq \theta|\xi|^2$$

for a.e. $x \in \Omega$ and all $\xi \in \mathbb{R}^N$.

Under this condition, we have that the matrix with entries a_{ij} is positive definite with the smallest eigenvalue $\lambda_0 \geq \theta$. For the following argument, we assume

$$a_{ij}, b_i, c \in L^\infty(\Omega) \quad \forall i, j = 1, ..., N, \quad f \in L^2(\Omega).$$

As done for the Poisson equation, we multiply by a test function $v \in C_0^\infty(\Omega)$ and integrate by parts. We can define the bilinear form

$$a[y, v] := \int_\Omega \left(\sum_{i,j=1}^{N} a_{ij} y_{x_i} v_{x_j} + \sum_{i=1}^{N} b_i y_{x_i} v + cyv \right) dx,$$

for $y, v \in H_0^1(\Omega)$ and the weak formulation of (3.4) reads:
A weak solution $y \in H_0^1(\Omega)$ to (3.4) satisfies

$$a[y, v] = (f, v) \quad \forall v \in H_0^1(\Omega). \tag{3.6}$$

If this weak formulation satisfies the estimates in the Lax–Milgram Lemma introduced above, then there exists a unique solution to the weak formulation. We introduce so-called energy

estimates, to inform whether we can apply Lax–Milgram.

Theorem 3.1.7 (Energy Estimates [85, Chapter 6])**.** There exist constants $\alpha, \beta > 0$ and $\gamma \geq 0$ such that

$$|a[y, v]| \leq \alpha \|y\|_{H_0^1(\Omega)} \|v\|_{H_0^1(\Omega)}$$

and

$$\beta \|y\|^2_{H_0^1(\Omega)} \leq a[y, y] + \gamma \|y\|^2_{L^2(\Omega)}$$

for all $y, v \in H_0^1(\Omega)$.

These estimates are proved in [85, Chapter 6] using the ellipticity condition (Definition 3.1.16) and Poincaré's inequality. Now, for $\gamma = 0$, these are precisely the estimates required for the Lax–Milgram Lemma. However, for $\gamma > 0$ the following extended result is available.

Theorem 3.1.8 (First Existence Theorem for Weak Solutions [85, Chapter 6])**.** There is a number $\gamma \geq 0$ such that for each $\mu \geq \gamma$ and each function $f \in L^2(\Omega)$, there exists a unique weak solution $y \in H_0^1(\Omega)$ of the boundary-value problem

$$\begin{aligned} Ly + \mu y &= f \quad \text{in } \Omega, \\ y &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

Regularity

We are also interested in how smooth a solution to (3.4) might be, i.e., how many continuous derivatives it may have. First, a result is presented that is relevant for interior regularity, i.e., regularity in the domain Ω.

Theorem 3.1.9 (Interior regularity [85, Chapter 6])**.** Let m be a non-negative integer, and assume

$$a_{ij}, b_i, c \in C^{m+1}(\Omega) \quad \text{for } i, j = 1, ..., N$$

and $f \in H^m(\Omega)$. Suppose $y \in H^1(\Omega)$ is a weak solution of the elliptic PDE

$$Ly = f \quad \text{in } \Omega.$$

Then $y \in H^{m+2}_{\text{loc}}(\Omega)$, and for each V compactly contained in Ω we have the estimate

$$\|y\|_{H^{m+2}(V)} \leq C \left(\|f\|_{H^m(\Omega)} + \|y\|_{L^2(\Omega)} \right), \tag{3.7}$$

the constant C depending only on m, Ω, V and the coefficients of L.

For the problem (3.4) with Dirichlet boundary conditions, a similar theorem can be proved, when all $a_{ij}, b_i, c \in C^{m+1}(\bar{\Omega})$ and $\partial\Omega$ is C^{m+2}. Then $y \in H^{m+2}(\Omega)$, as opposed to $H^{m+2}_{\text{loc}}(\Omega)$ and the left hand side of the estimate (3.7) is with respect to $H^{m+2}(\Omega)$ instead of $H^{m+2}(V)$. Both results can be extended to considering infinite differentiability, as discussed in [85, Chapter 6].

3.1.3 Weak Solutions to Linear Parabolic PDEs

Spaces that Include Time

In order to talk about solutions to parabolic PDEs, we need to extend the collection of spaces introduced so far, to include spaces that involve time. For this, let X be a real Banach space with associated norm $\|\cdot\|$.

Definition 3.1.17. [85, Chapter 5] The space

$$L^p(0,T;X)$$

consists of all strongly measurable functions $\tilde{y} : [0,T] \to X$ with

$$\|\tilde{y}\|_{L^p(0,T;X)} := \left(\int_0^T \|\tilde{y}(t)\|^p dt \right)^{1/p} < \infty$$

for $1 \leq p < \infty$ and

$$\|\tilde{y}\|_{L^\infty(0,T;X)} := \operatorname{ess\,sup}_{0 \leq t \leq T} \|\tilde{y}(t)\| < \infty.$$

Definition 3.1.18. [85, Chapter 5] The space

$$C(0,T;X)$$

comprises all continuous functions $\tilde{y} : [0,T] \to X$ with

$$\|\tilde{y}\|_{C([0,T];X)} := \max_{0 \leq t \leq T} \|\tilde{y}(t)\| < \infty.$$

Definition 3.1.19. [85, Chapter 5]

1. The Sobolev space

   $$W^{1,p}(0,T;X)$$

   consists of all functions $\tilde{y} \in L^p(0,T;X)$ such that their derivatives $\tilde{y}'$ exist in the weak sense and belong to $L^p(0,T;X)$. Furthermore,

   $$\|\tilde{y}\|_{W^{1,p}(0,T;X)} := \begin{cases} \left(\int_0^T \|\tilde{y}(t)\|^p + \|\tilde{y}'(t)\|^p dt \right)^{1/p} & (1 \leq p < \infty) \\ \operatorname{ess\,sup}_{0 \leq t \leq T} (\|\tilde{y}(t)\| + \|\tilde{y}'(t)\|) & (p = \infty) \end{cases}$$

2. We write $H^1(0,T;X) = W^{1,2}(0,T;X)$.

Weak Solutions to General Linear Parabolic PDEs

Following the account in [85, Chapter 7], we consider the initial-boundary value problem for the variable y of the form

$$\begin{aligned} \tfrac{\partial y}{\partial t} + Ly &= f \quad \text{in } (0,T] \times \Omega, \\ y &= 0 \quad \text{on } [0,T] \times \partial\Omega, \\ y &= g \quad \text{on } \{t = 0\} \times \Omega, \end{aligned} \tag{3.8}$$

where $T > 0$, and the functions $f : (0,T] \times \Omega \to \mathbb{R}$ and $g : \Omega \to \mathbb{R}$ are given. The operator L is defined as in (3.5), but note that now the coefficients a_{ij}, b_i and c can also depend on time.

Definition 3.1.20 (Parabolicity Condition [85, Chapter 7])**.** We say that the partial differential operator $\frac{\partial}{\partial t} + L$ is (uniformly) parabolic if there exists a constant $\theta > 0$ such that

$$\sum_{i,j=1}^{N} a_{ij}(t,x)\xi_i\xi_j \geq \theta|\xi|^2$$

for all $(t,x) \in (0,T] \times \Omega$, $\xi \in \mathbb{R}^N$.

In order to characterize weak solutions, we first assume

$$a_{ij}, b_i, c \in L^\infty((0,T]\times\Omega) \quad \forall i,j = 1,...,N,$$
$$f \in L^2((0,T]\times\Omega), \quad g \in L^2(\Omega),$$

and note that the a_{ij} define a symmetric matrix. Then, as in the elliptic case, the bilinear form corresponding to (3.8), for $y, v \in H_0^1(\Omega)$, is defined by

$$a[t;y,v] := \int_\Omega \left(\sum_{i,j=1}^N a_{ij}(t,\cdot) y_{x_i} v_{x_j} + \sum_{i=1}^N b_i(t,\cdot) y_{x_i} v + c(t,\cdot) yv \right) dx,$$

and we can characterize a weak solution to (3.8) as follows, where $\langle\cdot,\cdot\rangle$ denotes the pairing of $H^{-1}(\Omega)$ and $H_0^1(\Omega)$:

Definition 3.1.21. [85, Chapter 7] We say that a function

$$\tilde{y} \in L^2(0,T;H_0^1(\Omega)), \text{ with } \tilde{y}' \in L^2(0,T;H^{-1}(\Omega)),$$

is a weak solution of the parabolic initial-boundary value problem (3.8) provided

$$\langle \tilde{y}', v\rangle + a[\tilde{y}, v; t] = (f, v)$$

for each $v \in H_0^1(\Omega)$, a.e. time $0 \le t \le T$ and $\tilde{y}(0) = g$.

Existence of Weak Solutions

Existence of a weak solution to (3.8) can be shown using so-called Galerkin approximations. A brief outline is provided here, while more details can be found in [85, Chapter 7]. The idea is to construct solutions in a finite-dimensional subspace, spanned by the following basis. Let $\{w_k\}_{k=1}^\infty$ be smooth functions, which form an orthogonal basis of $H_0^1(\Omega)$ and an orthonormal basis of $L^2(\Omega)$. We then take the m first terms of this basis and construct a function $\tilde{y}_m : [0,T] \to H_0^1(\Omega)$ as follows:

$$\tilde{y}_m(t) := \sum_{k=1}^m d_m^k(t) w_k, \tag{3.9}$$

where the w_k contain the spatial dependence of the problem and the coefficients d_m^k are such that

$$d_m^k(0) = (g, w_k) \quad \forall k = 1,...,m, \tag{3.10}$$

and the function $\tilde{y}_m$ satisfies the weak formulation

$$(\tilde{y}_m', w_k) + a[\tilde{y}_m, w_k; t] = (f, w_k), \tag{3.11}$$

for $k = 1,...,m$ and $0 \le t \le T$. With this setup we have the following result for approximate solutions.

Theorem 3.1.10 (Construction of approximate solutions [85, Chapter 7])**.** For each integer $m = 1, 2, ...$, there exists a unique function $\tilde{y}_m$ of the form (3.9) satisfying (3.10), (3.11).

The next step is to show that as $m \to \infty$, the approximation $\tilde{y}_m$ converges to $\tilde{y}$, the weak solution of the infinite-dimensional problem (3.8). The existence of such a solution is shown using energy estimates bounding $\tilde{y}_m$ and $\tilde{y}_m'$ by f and g in appropriate L^2 norms. Then it can also be shown that the weak solution to (3.8) is unique. The details of these calculations can be found in [85, Chapter 7].

Regularity

As in the case of elliptic equations, the idea is to repeatedly apply energy estimates to derive results on higher regularity of solutions. Additionally, the results on Galerkin approximations can be used to aid the construction of solutions with higher regularity. We assume that the basis $\{w_k\}_{k=1}^{\infty}$ is the basis of eigenfunctions for $-\nabla^2$ on $H_0^1(\Omega)$. Furthermore, we require Ω to be bounded, open and $\partial\Omega$ smooth. Moreover, a_{ij}, b_i, c do not depend on t and are smooth on $\bar{\Omega}$.

Theorem 3.1.11 (Regularity [85, Chapter 7])**.** Assume that

$$g \in H^{2m+1}(\Omega), \quad \frac{d^k f}{dt^k} \in L^2(0,T;H^{2m-2k}(\Omega)) \quad \text{for } k = 0, ..., m.$$

Suppose also that the following mth-order compatibility conditions hold:

$$\begin{cases} g_0 := g \in H_0^1(\Omega), g_1 := f(0) - Lg_0 \in H_0^1(\Omega), g_2 := \frac{df}{dt}(0) - Lg_1 \in H_0^1(\Omega), \\ ..., g_m := \frac{d^{m-1}f}{d^{m-1}t}(0) - Lg_{m-1} \in H_0^1(\Omega). \end{cases}$$

Then

$$\frac{d^k \tilde{y}}{dt^k} \in L^2(0,T;H^{2m+2-2k}(\Omega)) \quad \text{for } k = 0, ..., m+1,$$

and we have the estimate

$$\sum_{k=0}^{m+1} \left\| \frac{d^k \tilde{y}}{dt^k} \right\|_{L^2(0,T;H^{2m+2-2k}(\Omega))} \leq C \left(\sum_{k=0}^{m} \left\| \frac{d^k f}{dt^k} \right\|_{L^2(0,T;H^{2m-2k}(\Omega))} + \|g\|_{H^{2m+1}(\Omega)} \right),$$

the constant C depending only on m, Ω, T and the coefficients of L.

If we let $m \to \infty$, we can conclude that if $g \in C^\infty(\bar{\Omega})$ and $f \in C^\infty((0,T] \times \bar{\Omega})$, and the other assumptions of the above theorem hold, then $\tilde{y} \in C^\infty((0,T] \times \bar{\Omega})$.

3.1.4 Results for Nonlinear PDEs – Calculus of Variations

So far, we have only considered linear PDEs. There are some ways in which the results above can be extended to certain types of nonlinearities. For example in [86, Chapter 4] it is demonstrated how the results of the Lax–Milgram Lemma can be extended to certain semilinear elliptic problems, and in [86, Chapter 7] how the Galerkin method for linear parabolic problems can be extended to the semilinear case. Both of these techniques require results on monotone operators and the case of Robin boundary conditions is included, which involves the application of the Trace operator introduced above. A more detailed discussion is omitted here for brevity.

The class of results for nonlinear results that we wish to discuss here, also in view to its links with PDE-constrained optimization, is the calculus of variations. This discussion again follows [85, Chapter 8]. We would like to solve the (nonlinear) PDE

$$A[y] = 0, \tag{3.12}$$

where A is a (nonlinear) operator. The main idea of the calculus of variations is to assume that A is the derivative of another functional E, such that

$$A[y] = E'[y].$$

Then, we can replace finding a solution to (3.12) by minimizing $E[y]$, in the hope that it is an easier task to find critical points of $E[y]$ than to solve (3.12).

First Variation: Euler–Lagrange Equation

We can define a Lagrangian $\mathcal{L}$ as

$$\mathcal{L}(p, z, x),$$

where $p = Dw(x)$, $z = w(x)$ are chosen for clarity of notation. Then we assume that E has the form

$$E[w] := \int_\Omega \mathcal{L}(Dw(x), w(x), x)dx,$$

for smooth functions w satisfying the boundary condition

$$w = g \quad \text{on } \partial\Omega.$$

Now, assume that a function y, satisfying the above boundary condition, is a minimizer of E. Then it is the solution to a PDE called the Euler–Lagrange equation, associated with the energy functional E. We derive the equation as follows. First, we take a perturbation around the minimizer y as $y + \tau v$, where $\tau \in \mathbb{R}$ and $v \in C_0^\infty(\Omega)$ is a test function, and define

$$e(\tau) := E[y + \tau v] = \int_\Omega \mathcal{L}(Dy + \tau Dv, y + \tau v, x)dx.$$

Then, taking the derivative, or first variation, we have

$$e'(\tau) = \int_\Omega \left[\sum_{i=1}^N \mathcal{L}_{p_i}(Dy + \tau Dv, y + \tau v, x)v_{x_i} + \mathcal{L}_z(Dy + \tau Dv, y + \tau v, x)v \right] dx. \tag{3.13}$$

Noticing that, since y is a minimizer of E and $y + \tau v = y = g$ on $\partial\Omega$, we have a critical point at $\tau = 0$, we find $e'(0) = 0$. Substituting $\tau = 0$ into (3.13) and integrating by parts results in

$$\int_\Omega \left(-\sum_{i=1}^N \left(\mathcal{L}_{p_i}(Dy, y, x)\right)_{x_i} + \mathcal{L}_z(Dy, y, x) \right) v dx.$$

Since this holds for all test functions v, we arrive at the Euler–Lagrange equation

$$-\sum_{i=1}^N \left(\mathcal{L}_{p_i}(Dy, y, x)\right)_{x_i} + \mathcal{L}_z(Dy, y, x) = 0 \quad \text{in } \Omega,$$

which is a nonlinear PDE.

Existence of Minimizers

In order to state results on the existence of minimizers, some conditions have to be satisfied. Initially, we need to define an admissible set of functions, which are taken from the Sobolev space $W^{1,q}(\Omega)$, as follows:

$$\mathcal{A} := \{w \in W^{1,q}(\Omega) | w = g \text{ on } \partial\Omega \text{ in the trace sense}\}.$$

We furthermore require a coercivity condition on $\mathcal{L}$ as follows. We assume that for a fixed $1 < q < \infty$ there exist constants $\alpha > 0, \beta \geq 0$ such that

$$\mathcal{L}(p, z, x) \geq \alpha|p|^q - \beta, \quad \forall p \in \mathbb{R}^N, z \in \mathbb{R}, x \in \Omega. \tag{3.14}$$

This implies a coercivity condition for E as

$$E[w] \geq \delta \|Dw\|^q_{L^q(\Omega)} - \gamma,$$

for $\gamma := \beta|\Omega|$, and $\delta > 0$ a constant. We moreover require E to be weakly lower semicontinuous, which is necessary for the existence of solutions. This is defined as follows:

Definition 3.1.22. [85, Chapter 8] We say that a function $E[\cdot]$ is (sequentially) weakly lower semicontinuous on $W^{1,q}(\Omega)$, provided

$$E[y] \leq \liminf_{k\to\infty} E[y_k]$$

whenever

$$y_k \rightharpoonup y \text{ weakly in } W^{1,q}(\Omega).$$

We will need to make some further assumptions regarding $\mathcal{L}$ to conclude such weak lower semicontinuity of E.

Theorem 3.1.12 (Weak lower semicontinuity [85, Chapter 8])**.** Assume that $\mathcal{L}$ is smooth, bounded below and in addition the mapping $p \mapsto \mathcal{L}(p, z, x)$ is convex for each $z \in \mathbb{R}$, $x \in \Omega$. Then $E[\cdot]$ is weakly lower semicontinuous on $W^{1,q}(\Omega)$.

Finally, we can state the existence theorem for minimizers of the energy functional.

Theorem 3.1.13 (Existence of minimizer [85, Chapter 8])**.** Assume that $\mathcal{L}$ satisfies the coercivity inequality (3.14) and is convex in the variable p. Suppose also the admissible set $\mathcal{A}$ is nonempty. Then there exists at least one function $y \in \mathcal{A}$ solving

$$E[y] = \min_{w\in\mathcal{A}} E[w].$$

If we make two further assumptions we can establish a unique minimizer, namely that

$$\mathcal{L} = \mathcal{L}(p, x) \text{ does not depend on } z \tag{3.15}$$

and there exists $\theta > 0$ such that

$$\sum_{i,j=1}^{N} \mathcal{L}_{p_i p_j}(p, x)\xi_i\xi_j \geq \theta|\xi|^2 \quad (p, \xi \in \mathbb{R}^N, x \in \Omega). \tag{3.16}$$

We hence conclude that the mapping $p \mapsto \mathcal{L}(p, x)$ is uniformly convex for each x.

Theorem 3.1.14 (Uniqueness of minimizer [85, Chapter 8])**.** Suppose (3.15), (3.16) hold. Then a minimizer $y \in \mathcal{A}$ of $E[\cdot]$ is unique.

Weak Solutions of the Euler–Lagrange Equation

Finally, we would like to be able to link the above derivations to finding solutions to nonlinear PDEs, and in particular to Euler–Lagrange equations. Therefore, we consider the boundary-value problem

$$\begin{aligned} -\sum_{i=1}^{N} \big(\mathcal{L}_{p_i}(Dy, y, x)\big)_{x_i} + \mathcal{L}_z(Dy, y, x) = 0 \quad &\text{in } \Omega, \\ y = g \quad &\text{on } \partial\Omega. \end{aligned} \tag{3.17}$$

We impose some growth conditions on $\mathcal{L}$ and its derivatives as follows:

$$|\mathcal{L}(p, z, x)| \leq C(|p|^q + |z|^q + 1), \tag{3.18}$$

and

$$\begin{cases} |\mathcal{L}_p(p, z, x)| \leq C(|p|^{q-1} + |z|^{q-1} + 1), \\ |\mathcal{L}_z(p, z, x)| \leq C(|p|^{q-1} + |z|^{q-1} + 1), \end{cases} \tag{3.19}$$

with C some constant, $p \in \mathbb{R}^N$, $z \in \mathbb{R}$ and $x \in \Omega$. Then, as before, for $y \in W^{1,q}(\Omega)$ we can define a weak solution to the boundary value problem (3.17) by multiplying by a test function $v \in C_0^\infty(\Omega)$ and integrating by parts.

Definition 3.1.23. [85, Chapter 8] We say $y \in \mathcal{A}$ is a weak solution of the boundary-value problem (3.17) for the Euler–Lagrange equation provided

$$\int_\Omega \sum_{i=1}^{N} \left(\mathcal{L}_{p_i}(Dy, y, x) v_{x_i} + \mathcal{L}_z(Dy, y, x) v \right) dx = 0 \tag{3.20}$$

for all $v \in W_0^{1,q}(\Omega)$.

Theorem 3.1.15 (Solution of Euler–Lagrange equation [85, Chapter 8])**.** Assume $\mathcal{L}$ satisfies the growth conditions (3.18), (3.19) and $y \in \mathcal{A}$ satisfies

$$e[y] = \min_{w \in \mathcal{A}} E[w].$$

Then y is a weak solution of (3.17).

Note that in general (3.17) can have solutions that are not minimizers of E. Only when $(p, z) \mapsto \mathcal{L}(p, z, x)$ is convex for each x, then each weak solution to (3.17) is also a minimizer of E.

The discussion on the calculus of variations can be extended to various topics such as local minimizers, regularity, or constrained minimization. The reader is referred to [85, Chapter 8] for a detailed introduction. Further theory on nonlinear PDEs can also be found in [85]. Similar ideas will be encountered when discussing results on PDE-constrained optimization in Section 4.1.

3.2 Spectral Methods

After discussing theoretical results concerning PDEs, we now turn to their numerical solution. In particular, we will focus on one class of methods that can be very effective in solving different kinds of PDEs: spectral methods. The first person to be credited for work on spectral collocation methods was Lanczos in 1938, see [88]. Only in the seventies were such methods applied to PDEs, see [89, 90]. After the first book on spectral methods was published in 1977, see [91], many theoretical and practical advances have been made in the field and several books have been written on the subject since, such as [92, 93], as well as [94] on implementations in MATLAB. The first applications of spectral methods to PDEs were in the context of fluid dynamics problems [95]. Since then, various fields have applied these methods, such as in molecular dynamics [96], modelling of protein folding [97], vibration analysis in mechanical engineering [98], or option pricing [99]. More importantly for the work in this thesis, a spectral method has been applied to solving DDFT models, see [84], which will be discussed in more detail in Section 3.3. The account in this thesis of spectral methods follows [92]. First, results on polynomial expansions and their theoretical properties are discussed. Then, the difference between collocation and Galerkin spectral methods is established, before providing some convergence results for such methods. These theoretical discussions of spectral methods are largely included for context, since most of the original work in this thesis is focussed on numerical considerations.

3.2.1 Polynomial Expansions

As discussed in [92, Chapter 2], the basis of spectral methods is the idea of approximating an unknown function y. If we know the value of y on a set of points, we can approximate y by a polynomial. A polynomial expansion is of the form

$$y(x) = \sum_{k=0}^{\infty} \hat{y}_k p_k(x), \tag{3.21}$$

where p_k denotes a polynomial of degree k with coefficients $\hat{y}_k \in \mathbb{R}$. We would like the set of polynomials $\{p_k\}_{k=0,1,2,\ldots}$ to satisfy specific properties. The main one is that any two polynomials p_k and p_m in the set, for $k \neq m$, are orthogonal to each other with respect to a weight function ω in the canonical interval $(-1, 1)$. This can be defined by

$$\int_{-1}^{1} p_k(x) p_m(x) \omega(x) dx = 0 \quad \text{for } m \neq k.$$

Such a set of polynomials is complete in the weighted L^2 space denoted by $L^2_\omega(-1, 1)$, equipped with the weighted norm

$$\|v\|_\omega = \left(\int_{-1}^{1} |v(x)|^2 \omega(x) dx \right)^{1/2} < \infty, \tag{3.22}$$

and inner product

$$(y, v)_\omega = \int_{-1}^{1} y(x) v(x) \omega(x) dx. \tag{3.23}$$

For a function $y \in L^2_\omega(-1, 1)$, its polynomial expansion (3.21) has coefficients given by

$$\hat{y}_k = \frac{1}{\|p_k\|^2_\omega} \int_{-1}^{1} y(x) p_k(x) \omega(x) dx, \tag{3.24}$$

called the *polynomial transform* of y. Note that the notation $\hat{y}$ differs from the one for a desired state $\widehat{y}$ in optimal control theory. Expansion (3.21) is not yet instructive for finding a computable approximation to y, since it is defined in terms of an infinite sum. The next step in the approach is therefore to truncate this polynomial expansion at the N-th term

$$P_N y(x) = \sum_{k=0}^{N} \hat{y}_k p_k(x), \tag{3.25}$$

so that $P_N y \in \mathbb{P}_N$, the space of polynomials up to degree N. Since $P_N y$ is also an element of $L^2_\omega(-1, 1)$, for all test functions $v \in \mathbb{P}_N$, it satisfies

$$(P_N y, v)_\omega = (y, v)_\omega. \tag{3.26}$$

This means, $P_N y$ is the *orthogonal projection* of y onto $\mathbb{P}_N$ and therefore the best approximation to y in $\mathbb{P}_N$ as measured by the norm (3.22). Moreover, due to the completeness of $\{p_k\}_{0,1,2,..}$, the truncated series $P_N y$ converges to y in $L^2_\omega(-1, 1)$:

$$\|y - P_N y\|_\omega \to 0 \quad \text{as} \quad N \to \infty.$$

This approach still requires the computation of the coefficients $\hat{y}_k$, defined in (3.24), in terms of a continuous integral, which we cannot resolve computationally. The reason for this is that computing this integral requires knowledge of y for all x, when we only know y on a set of discrete points.

In order to address this difficulty, Gaussian quadrature methods can be considered. There are different formulae, three of which are stated in [92, Chapter 2]. Here we will present the so-called Gauss–Lobatto integration, since it is most in line with the numerical approach in this thesis, which was first introduced in [84]. If we consider the polynomial

$$q(x) = p_{N+1}(x) + a p_N(x) + b p_{N-1}(x), \tag{3.27}$$

with a,b such that $q(-1) = q(1) = 0$, then q will have $N+1$ roots in the interval $[-1, 1]$.

Theorem 3.2.1 (Gauss–Lobatto Integration [92, Chapter 2])**.** Let $-1 = x_0 < x_1 < ... < x_N = 1$ be the $N+1$ roots of the polynomial (3.27), and let $\omega_0, ..., \omega_N$ be the solution of the linear system

$$\sum_{j=0}^{N} (x_j)^k \omega_j = \int_{-1}^{1} x^k \omega(x) dx, \quad 0 \leq k \leq N.$$

Then

$$\sum_{j=0}^{N} p(x_j)\omega_j = \int_{-1}^{1} p(x)\omega(x)dx \quad \text{for all } p \in \mathbb{P}_{2N-1}.$$

The constants ω are called the *integration weights*. Note that this result can be generalized from the choice (3.27) to an arbitrary polynomial of degree $N+1$ that satisfies the orthogonality conditions outlined above. With this relationship between summation and integration, we can translate the concepts introduced above into a discrete setting. First, we can define a discrete inner product on $\mathbb{P}_N$ as

$$(y, v)_N = \sum_{j=0}^{N} y(x_j)v(x_j)w_j, \tag{3.28}$$

which is related to the continuous inner product (3.23) via Theorem 3.2.1

$$(y, v)_N = (y, v)_\omega \quad \text{for } y, v \in \mathbb{P}_{2N-1}.$$

Moreover, we can define the discrete norm

$$\|y\|_N = \sqrt{(y, y)_N}.$$

Using Theorem 3.2.1, we can also develop a discrete version of the polynomial transform (3.24), which we denote by $\tilde{y}_k$. The *discrete polynomial transform* is given by

$$\tilde{y}_k = \frac{1}{\gamma_k} \sum_{j=0}^{N} y(x_j)p_k(x_j)\omega_j, \tag{3.29}$$

for $k = 0, .., N$ and with

$$\gamma_k = \sum_{j=0}^{N} p_k^2(x_j)\omega_j,$$

the discrete version of the prefactor $\|p_k\|_\omega^2$ in (3.24). These coefficients can be computed, provided that the $y(x_j)$ are known, while the continuous analogue cannot, since knowledge of y on the whole domain would be required. Then, analogous to the truncated expansion (3.25), we can define the *interpolation polynomial*

$$I_N y(x) = \sum_{k=0}^{N} \tilde{y}_k p_k(x). \tag{3.30}$$

This interpolation polynomial equals y at the integration nodes $\{x_j\}_{j=0}^N$ given in Theorem 3.2.1 so that

$$I_N y(x_j) = y(x_j).$$

When using these nodes for interpolation, they are called *collocation points*, and numerical methods using interpolation polynomials to approximate y are *collocation methods*. Finally, equivalently to $P_N y$ being a projection onto $\mathbb{P}_N$ with respect to $(\cdot,\cdot)_\omega$, $I_N y$ defines the orthogonal projection onto $\mathbb{P}_N$ with respect to the discrete inner product $(\cdot,\cdot)_N$. Therefore, we have, for continuous v,

$$(I_N y, v)_N = (y, v)_N,$$

compare with (3.26).

One can relate the interpolation and truncation approaches by considering the difference in the coefficients $\hat{y}_k$ and $\tilde{y}_k$, see [92, Chapter 2]. This is given by

$$\tilde{y}_k - \hat{y}_k = \frac{1}{\gamma_k} \sum_{l>N} (p_l, p_k)_N \hat{y}_l := r_k, \tag{3.31}$$

for $k = 0, ..., N$. Then the difference between the two approximations of y can be quantified as

$$R_N y(x) = I_N y(x) - P_N y(x) = \sum_{k=0}^{N} r_k p_k(x).$$

The issue is that for the discrete inner product $(p_k, p_l)_N$, for $l > N$, orthogonality cannot be guaranteed, and so the r_k, and therefore the $R_N y$, are not necessarily zero in this case. The error $R_N y$ is called the *aliasing error* caused by interpolation.

There are different kinds of polynomials that satisfy the requirements set out so far. These include Legendre, Chebyshev, Jacobi and Laguerre polynomials, see [92, Chapter 2]. Moreover, using trigonometric polynomials on the interval $(0, 2\pi)$ (or any other 2π length interval) instead of $(-1, 1)$, the Fourier system also falls under the umbrella of polynomial approximation methods. In general, these polynomials arise as eigenfunctions of Sturm–Liouville problems, as discussed in [92, Chapter 2]. In the following, we will discuss two of these choices, which will underpin the numerical implementations in this thesis: Chebyshev and Fourier methods.

A Note on the Runge Phenomenon

A well known result is that if a function is interpolated using equispaced points, the polynomial interpolant will exhibit large oscillations close to the boundary. This is called the *Runge phenomenon*, see [93, Chapter 4], [94, Chapter 5]. The error between an (at least) $(N + 2)$ times differentiable function y on $[-1, 1]$ and the interpolation polynomial p_{N+1} is given by

$$y(x) - p_{N+1}(x) = \frac{1}{(N+2)!} y^{(N+2)}(\xi) \prod_{i=0}^{N+1} (x - x_i),$$

where $\xi \in [-1, 1]$, see [93, Chapter 4]. While we cannot influence the factor $y^{(N+1)}(\xi)$, we can try to find the interpolant p_{N+1} that will provide the smallest maximum amplitude over $[-1, 1]$. This is described in the following theorem:

Theorem 3.2.2 (Chebyshev Minimal Amplitude Theorem [93, Chapter 4])**.** Of all polynomials of degree $N+1$ with leading coefficient equal to 1, the unique polynomial which has the smallest maximum on $[-1, 1]$ is $\frac{T_{N+1}(x)}{2^N}$, the Chebyshev polynomial of degree $N + 1$ divided by 2^N. In other words, all polynomials of the same degree and leading coefficient unity satisfy the inequality

$$\max_{x\in[-1,1]} |p_{N+1}(x)| \geq \max_{x\in[-1,1]} \left| \frac{T_{N+1}(x)}{2^N} \right| = \frac{1}{2^N}.$$

We can associate this polynomial with the product of linear factors as

$$\frac{T_{N+1}(x)}{2^N} = \prod_{i=1}^{N+1}(x - x_i),$$

where the x_i are the roots of the polynomial. Then it is evident that the optimal distribution of interpolation points is given by the roots of the Chebyshev polynomial. This motivates the discussion in the next section.

Chebyshev Polynomials

Most of the work in this thesis will be done using Chebyshev pseudospectral methods. The main ideas of this method will be introduced in this section and can be found in [92, Chapter 2]. The expansion of y, (3.21), in terms of Chebyshev polynomials, is given by

$$y(x) = \sum_{k=0}^{\infty} \hat{y}_k T_k(x),$$

where the $T_k(x)$ are Chebyshev polynomials. The coefficients are given by formula (3.24) and are

$$\hat{y}_k = \frac{2}{\pi c_k} \int_{-1}^{1} y(x) T_k(x) \omega(x) dx,$$

where $c_k = 2$ for $k = 0$ and $c_k = 1$ otherwise. The polynomials T_k can be defined by

$$T_k(x) = \cos(k\theta), \quad \theta = \arccos x, \tag{3.32}$$

and are normalized by satisfying $T_k(1) = 1$.
The quadrature nodes and weights, as given in Theorem 3.2.1, can be defined as

$$x_j = \cos\left(\frac{\pi j}{N}\right), \tag{3.33}$$
$$\omega_j = \begin{cases} \frac{\pi}{2N}, & j = 0, N, \\ \frac{\pi}{N}, & j = 1, ..., N-1. \end{cases}$$

The nodes x_j are called (Chebyshev–)Gauss–Lobatto points, and are the ones used in this thesis. Using other versions of Theorem 3.2.1 will result in slightly different nodes and weights, see [92, Chapter 2]. Given these nodes and weights, the interpolation polynomial (3.30) is defined as

$$I_N y(x) = \sum_{k=0}^{N} \tilde{y}_k T_k(x), \tag{3.34}$$

where the coefficients are given by (3.29). Substituting nodes and weights, the coefficients reduce to

$$\tilde{y}_k = \frac{1}{\gamma_k}\left(\frac{\pi}{2N} y(x_0) + \frac{\pi}{2N} y(x_N) + \sum_{j=1}^{N-1} \frac{\pi}{N} y(x_j) \cos\left(\frac{\pi k j}{N}\right)\right),$$

where

$$\gamma_k = \begin{cases} \pi, & k = 0, N, \\ \frac{\pi}{2}, & 1 \le k \le N-1. \end{cases}$$

Given this, we can now quantify the aliasing error $R_N y$. Considering the right hand side of

(3.31) given by

$$r_k = \frac{1}{\gamma_k} \sum_{l>N} (T_l, T_k)_N \hat{y}_l,$$

we note that the Chebyshev polynomials satisfy

$$(T_l, T_k)_N = \begin{cases} (T_k, T_k)_N, & \text{for } l = 2mN \pm k,\ m \in \mathbb{Z}_+, \\ 0, & \text{otherwise,} \end{cases}$$

since $T_l(x_j) = \cos\left(\frac{\pi l j}{N}\right)$, which satisfies $T_k = T_l$ precisely when $l = 2mN \pm k$, due to the 2π periodicity of the cosine function. This demonstrates why the error $R_N y$ is called the aliasing error, since it describes how the Chebyshev mode k is aliased with higher modes l. This notion is more standard when discussed in terms of Fourier modes in the next section. The discrete inner products cancel out with $\frac{1}{\gamma_k}$ and so the r_k reduce to

$$r_k = \sum_{\substack{l=2mN\pm k, \\ l>N}} \hat{y}_l.$$

Given the above definitions, the approximation of the derivative y' can be defined. The expansion of y' is given by

$$y' = \sum_{k=0}^{\infty} \hat{y}_k^{(1)} T_k(x),$$

where

$$\hat{y}_k^{(1)} = \frac{2}{c_k} + \sum_{\substack{s=s+1 \\ s+k \text{ odd}}} s\hat{y}_s,$$

with $c_k = 2$ for $k = 0$ and $c_k = 1$, otherwise. This expression is derived by applying a recurrence relation on the T_k, see [92, Chapter 2]. One can now either find the approximation to y' by forming $P_{N-1}y'$ and $I_{N-1}y'$ or by differentiating the approximations $P_N y$ and $I_N y$, denoted by $(P_N y)'$ and $(I_N y)'$, respectively. Letting $D_N y := (I_N y)'$, the *interpolation derivative* is given by

$$(D_N y)(x_j) = \sum_{k=0}^{N} (D_N)_{jk} y(x_k),$$

for $j = 0, ..., N$. Now, since interpolation polynomials of degree $\leq N+1$ for the set $\{x_j\}_{j=0}^N$ are unique, see [94, Chapter 6], a more computationally convenient version of the interpolation polynomial (3.34) through the (Chebyshev–)Gauss–Lobatto points $\{x_j\}_{j=0}^N$ can be applied. This is *Lagrange interpolation*, given by

$$I_N y(x) = \sum_{j=0}^{N} y(x_j) C_j(x),$$

$$C_j = \prod_{\substack{k=0 \\ k\neq j}}^{N} \frac{x - x_k}{x_j - x_k},$$

which satisfies $C_j(x_k) = \delta_{j,k}$, see [93, Chapter 4]. However, as shown in [100], barycentric Lagrange interpolation is more stable than the simple formula introduced above. The barycentric

formula is

$$I_N y(x) = \frac{\sum_{j=0}^{N} \frac{\tilde{\omega}_j}{x - x_j} y(x_j)}{\sum_{j=0}^{N} \frac{\tilde{\omega}_j}{x - x_j}}, \tag{3.35}$$

where the weights are defined as

$$\tilde{\omega}_j = (-1)^j d_j, \quad d_j = \begin{cases} \frac{1}{2} & \text{for } j = 0, j = N, \\ 1 & \text{otherwise,} \end{cases} \tag{3.36}$$

see [84, 100]. Differentiating the Lagrange interpolation polynomial (3.35) and evaluating it at the (Chebyshev–)Gauss–Lobatto points, results in the $(N+1) \times (N+1)$ *differentiation matrix* D_N, which has the following entries

$$(D_N)_{jk} = \begin{cases} \frac{2N^2+1}{6}, & j = k = 0, \\ -\frac{2N^2+1}{6}, & j = k = N, \\ -\frac{x_j}{2(1-x_j^2)}, & j = k = 1, ..., N-1, \\ \frac{c_i}{c_j} \frac{(-1)^{i+j}}{(x_i - x_j)}, & j \neq k, \quad i, j = 0, ..., N, \end{cases} \tag{3.37}$$

where

$$c_i = \begin{cases} 2, & i = 0 \text{ or } N, \\ 1, & \text{otherwise,} \end{cases}$$

compare with [94, Chapter 6], [92, Chapter 2]. One can then construct higher derivative matrices in two ways; either by taking powers of D_N or by repeatedly differentiating the interpolation polynomial and evaluating it at the nodes.

Fourier Spectral Methods

While introduced last in this chapter, the Fourier spectral method presented in [92, Chapter 2] is the most commonly known of the spectral methods. Here, approximation (3.21) is defined as

$$y(x) = \sum_{k=-\infty}^{\infty} \hat{y}_k \phi_k(x), \tag{3.38}$$

using trigonometric polynomials of the form

$$\phi_k(x) = e^{ikx},$$

which are orthogonal over any 2π interval. Here we fix this to be $(0, 2\pi)$ and the orthogonality condition is given by

$$\int_0^{2\pi} \phi_k(x) \overline{\phi_m(x)} dx = 2\pi \delta_{km},$$

where $\overline{\phi_m(x)}$ is the complex conjugate of $\phi_m(x)$. In the same way as in the general polynomial case (see (3.22) and (3.23)), $y \in L^2(0, 2\pi)$, which is equipped with

$$\|y\| = \left(\int_0^{2\pi} |y(x)|^2 \right)^{1/2},$$

and

$$(y, v) = \int_0^{2\pi} y(x)\overline{v(x)}dx.$$

Note that the only differences in the present definitions to the ones given by (3.22) and (3.23) are the interval considered and the fact that the complex conjugation has to be accounted for. Moreover $\omega = 1$ in the Fourier case.

In line with the definition of the polynomial transform (3.24), we can define the much better-known concept of a *Fourier transform* given by

$$\hat{y}_k = \frac{1}{2\pi} \int_0^{2\pi} y(x)e^{-ikx}dx, \tag{3.39}$$

for $k = ..., -2, -1, 0, 1, 2, ...$, which form the coefficients of the *Fourier expansion* (3.38). The truncated expansion, compare to (3.25), is given by

$$P_N y(x) = \sum_{k=-N/2}^{N/2-1} \hat{y}_k \phi_k(x).$$

Now $P_N y$ is the orthogonal projection of y onto the space of trigonometric polynomials of degree $\leq N/2$, given by

$$\mathbb{P}_{\mathbb{T}} := \text{span}\{e^{ikx} | -N/2 \leq k \leq N/2 - 1\}.$$

Hence, $P_N y$ satisfies

$$(P_N y, v) = (y, v) \quad \forall v \in \mathbb{P}_{\mathbb{T}},$$

and is therefore the best approximation to y in $\mathbb{P}_{\mathbb{T}}$. As done for the general polynomial case, the *Fourier nodes* can be derived and are given by

$$x_j = \frac{2\pi j}{N}, \quad j = 0, ..., N - 1. \tag{3.40}$$

In the same way as Theorem 3.2.1 states for Gauss–Lobatto quadrature, these nodes are found by approximating integrals via sums, using appropriate weight functions. Here, the composite trapezoidal rule is used, with associated weights $\omega_j = 1$. The discrete approximation to coefficients (3.39) at the nodes (3.40) is called the *discrete Fourier Transform*

$$\tilde{y}_k = \frac{1}{N} \sum_{j=0}^{N-1} y(x_j)e^{-ikx_j}, \quad k = -N/2, ..., N/2 - 1. \tag{3.41}$$

Moreover, the interpolation polynomial is defined, as before, by replacing the continuous $\hat{y}_k$ by the discrete ones, to obtain

$$I_N y(x) = \sum_{k=-N/2}^{N/2-1} \tilde{y}_k e^{ikx}, \tag{3.42}$$

satisfying $I_N y(x_j) = y(x_j)$.

Using these definitions, we can now quantify the aliasing error $R_N y$ in terms of the contin-

uous and discrete coefficients $\hat{y}$ and $\tilde{y}$, as defined in (3.31). Then

$$r_k = \sum_{\substack{l=-\infty \\ l\neq 0}}^{\infty} \hat{y}_{k+mN}, \quad k = -N/2, ..., N/2-1, \quad m \in \mathbb{Z}_+.$$

In order to see this, consider the set of trigonometric polynomials $\phi \in \mathbb{P}_{\mathbb{T}}$ that are mutually orthogonal in the interval $(0, 2\pi)$. Outside of this interval these polynomials are not mutually orthogonal, since each *wavenumber* $k \in [-N/2, N/2-1]$ aliases with all other wavenumbers $k + mN$ that lie in $(-\infty, -N/2) \cup (N/2-1, \infty)$. This can be seen when writing ϕ_k in its trigonometric components

$$\phi_k(x) = e^{ikx} = \cos(kx) + i\sin(kx).$$

Since both cos and sin are 2π-periodic, $\cos(kx) = \cos((k+mN)x)$ and $\sin(kx) = \sin((k+mN)x)$, for $m = 0, 1, ...$, and so $\phi_k(x) = e^{ikx} = e^{i(k+mN)x} = \phi_{k+mN}(x)$. In other words, the functions ϕ_k are 2π-periodic and at the interpolation nodes $\phi_k(x_j) = \phi_{k+mN}(x_j)$. Note the similarity between this result and that for the Chebyshev polynomials. While by no means a coincidence, as we will see shortly, this result is rather intuitive in the case of Fourier systems. Having found the r_k, the aliasing error is defined as

$$R_N y(x) = \sum_{k=-N/2}^{N/2-1} r_k \phi_k(x).$$

Differentiation of y is in fact simpler than for the Chebyshev polynomials. Considering the full Fourier series (3.38), we notice that differentiating it with respect to x only depends on the ϕ_k. Therefore,

$$y'(x) = \sum_{k=-\infty}^{\infty} ik\hat{y}_k \phi_k(x),$$

i.e., differentiating the Fourier series corresponds to multiplying by a factor of ik. Since this is also true for $P_N y$, in contrast to the Chebyshev case we have $P_N y' = (P_N y)'$.
In the same way as done for Chebyshev approximations, Lagrange polynomials can be used to rewrite the interpolation polynomial for computational convenience. Here the *Lagrange trigonometric polynomials* are defined by

$$\psi_k(x) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} e^{ik(x-x_j)},$$

satisfying $\psi_j(x_k) = \delta_{j,k}$. Then

$$I_N y(x) = \sum_{j=0}^{N-1} y(x_j)\psi_j(x).$$

Using these, the differentiation matrix is given by

$$(D_N)_{jl} = \psi_l'(x_j) = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} ike^{2ik(j-l)\pi/N}.$$

This is a good choice for odd N. However, if N is even, when $k = -N/2$, there is no counterpart in the sum at $k = N/2$. This means that if $\hat{y}_{-N/2}$ is not real, then neither is the approximation to y. Therefore, in this case $\hat{y}_{-N/2} = 0$ is enforced to avoid complications, see [92, Chapter 3]. Alternatively, a mode $k = N/2$ could be added, with $\hat{y}_{N/2} = \hat{y}_{-N/2}$, see [94, Chapter 3]. Now

these terms can be evaluated explicitly to give

$$(D_N)_{jl} = \begin{cases} \frac{1}{2}(-1)^{j+l} \cot\left[\frac{(j-l)\pi}{N}\right], & j \neq l \\ 0, & j = l. \end{cases} \tag{3.43}$$

The relationship between Chebyshev and Fourier Methods

We have seen that, while the different methods presented here all fall within the general framework of polynomial approximation, there is an even closer relationship between Chebyshev and trigonometric polynomials, which explains some of the similar features the two methods exhibit. Recalling the definition of Chebyshev polynomials (3.32) and choosing $x = \cos(\theta)$ results in

$$T_k(\cos\theta) = \cos(k\theta), \tag{3.44}$$

which is exactly the real part of a trigonometric polynomial $\phi_k = e^{ik\theta} = \cos(k\theta) + i\sin(k\theta)$. Therefore, many of the results that are established for Fourier series also hold for Chebyshev expansions. The key advantage here is that while Fourier series best approximate periodic functions, Chebyshev series do not require periodicity of y, while exhibiting some of the same behaviour as their Fourier counterparts.

3.2.2 Approximation Theory for Polynomials

In this subsection, some results on both the truncation and interpolation errors made in polynomial approximation are stated, before discussing stability and convergence of different spectral schemes. Here we focus on Fourier and Chebyshev results, but other cases can be found in [92, Chapter 5]. Contrary to the presentation above, here we start by discussing Fourier approximations, since these results are a little more intuitive to understand and some of the Chebyshev results follow directly, due to the relation (3.44).

Fourier Approximation

For Fourier approximations, one can show that the decay of coefficients $\hat{y}$ depends on the regularity of y, i.e., the number of its continuous derivatives, and on the fact that y is periodic. This can be shown following [92, Chapter 2]. Note that for the problems considered in this thesis we expect solutions y to the PDE models to have high regularity so that the coefficients decay fast. This will be discussed in more detail later on.

Integrating the Fourier coefficients given in (3.39) by parts results in

$$\hat{y}_k = \frac{1}{2\pi}\frac{1}{ik}\left(-[y(2\pi) - y(0)] + \int_0^{2\pi} y'(x)e^{-ikx}dx\right).$$

We know that $y(2\pi) = y(0)$ due to periodicity. The integral can be bounded independent of k as follows

$$\begin{aligned} \left|\int_0^{2\pi} y'(x)e^{-ikx}dx\right| &\leq 2\pi \max_{x\in[0,2\pi]} \left|y'(x)\right| \max_{x\in[0,2\pi]} \left|\cos(kx) + i\sin(kx)\right| \\ &\leq 2\pi \max_{x\in[0,2\pi]} \left|y'(x)\right| \max_{x\in[0,2\pi]} \left(\left|\cos(kx)\right| + \left|\sin(kx)\right|\right) \\ &\leq 2\pi \max_{x\in[0,2\pi]} \left|y'(x)\right|, \end{aligned}$$

where the second inequality follows by the triangle inequality and the third by noting that $|\cos(kx)|$ and $|\sin(kx)|$ are bounded by 1 for any choice of k. Since this integral is bounded independently of k, we conclude that $\hat{y}_k = O(k^{-1})$. This can be extended to the following result:

Theorem 3.2.3. [92, Chapter 5] If y is m-times continuously differentiable in $[0, 2\pi]$ ($m \geq 1$), and if $y^{(j)}$ is periodic for all $j \leq m-2$, then

$$\hat{y}_k = O(k^{-m}), \quad k = \pm 1, \pm 2, \dots .$$

Corollary 3.2.1. [92, Chapter 5] The k-th Fourier coefficient of a function which is infinitely differentiable and periodic with all its derivatives on $[0, 2\pi]$ decays faster than any negative power of k.

The following discussion on Fourier approximation theory follows [92, Chapter 5]. For the following result, we introduce the Sobolev norm of integer order $m \geq 0$ denoted by

$$\|y\|^2_{H^m(0,2\pi)} = \sum_{k=0}^{m} \int_0^{2\pi} |y^{(k)}(x)|^2 dx.$$

The subspace $H^m_p(0, 2\pi)$ of $H^m(0, 2\pi)$ is defined by the property that each of the $m-1$ derivatives is periodic. Moreover, the space $G_{\eta,m}(0, 2\pi)$ is defined as

$$\|y\|^2_{G_{\eta,m}(0,2\pi)} = \sum_{k\in\mathbb{Z}} e^{2\eta(1+|k|)}(1 + |k|^{2m})|\hat{y}_k|^2 < \infty.$$

Given the rate of decay of Fourier coefficients, error bounds for the Fourier approximations can be formulated. For the truncated Fourier expansion the truncation error is

$$\|y - P_N y\|_{L^2(0,2\pi)} \leq CN^{-m}\|y^{(m)}\|_{L^2(0,2\pi)}, \tag{3.45}$$

with $y \in H^m_p(0, 2\pi)$ and $m \geq 0$. Equivalent bounds for different Sobolev norms are available and can be found in [92, Chapter 5].

Given these definitions, the following result holds: If y is 2π-periodic and analytic in a strip $|\mathrm{Im} z| < \eta$, then the truncation error becomes

$$\|y - P_N y\|_{H^l(0,2\pi)} \leq CN^{l-m}e^{-\eta N}\|y\|_{G_{\eta,m}(0,2\pi)},$$

for $0 \leq l \leq m$, i.e., the error decays exponentially in N.

The interpolation error is of the form (3.45), with $m \geq 1$. This enables the formulation of an error estimate for the interpolation derivative $D_N y$ as

$$\|y' - D_N y\|_{L^2(0,2\pi)} \leq CN^{1-m}\|y^{(m)}\|_{L^2(0,2\pi)}.$$

As for the truncation error, this result can be improved for a variable y, which is 2π-periodic and analytic in $|\mathrm{Im} z| < \eta_0$, to give

$$\|y' - D_N y\|_{L^2(0,2\pi)} \leq \frac{4}{\sinh(\eta)} N e^{-\eta N} \max_{|\mathrm{Im} z|\leq\eta} |y(z)|,$$

for all η in $0 < \eta < \eta_0$. Again, we can see that the error in the interpolation derivative decays exponentially for such y. Given the errors for $P_N y$ and $I_N y$ we can quantify the aliasing error by

$$\|R_N y\|_{L^2(0,2\pi)} \leq CN^{-m}\|y^{(m)}\|_{L^2(0,2\pi)},$$

so that asymptotically it is of the same order as the truncation and interpolation errors.

Chebyshev Approximation

Since Fourier and Chebyshev approximations are related by (3.44), the results on the decay of Fourier coefficients given by (3.2.1) and (3.2.3) directly apply to the coefficients of Chebyshev expansions.

We consider the weighted Sobolev spaces

$$H^m_\omega(-1,1) = \left\{ v \in L^2_\omega(-1,1) : \quad v^{(k)} \in L^2_\omega(-1,1), \text{ for } 0 \le k \le m \right\},$$

with the norm

$$\|y\|^2_{H^m_\omega(-1,1)} = \sum_{k=0}^{m} \left\| y^{(k)} \right\|^2_{L^2_\omega(-1,1)},$$

and the semi-norm

$$|y|^2_{H^{m;N}_\omega(-1,1)} = \sum_{k=\min(m,N+1)}^{m} \left\| y^{(k)} \right\|^2_{L^2_\omega(-1,1)},$$

for the following result: The truncation error for the Chebyshev approximation of y is given by

$$\|y - P_N y\|_{L^2_\omega(-1,1)} \le C N^{-m} |y|_{H^{m;N}_\omega(-1,1)}, \quad \forall y \in H^m_\omega(-1,1), \tag{3.46}$$

with $m \ge 0$. As stated in [92, Chapter 5], similar bounds can be found in terms of weighted L^p norms and higher order Sobolev norms. The interpolation error $I_N y$ is of the form (3.46), but with $m \ge 1$ instead of 0 and higher order estimates can be established. We can make the interpolation error sharper for analytic y, as done for the Fourier case. In particular, for analytic y in $[-1,1]$, with analytic extension to ellipse E_η with foci $z = \pm 1$ and sum of semi-axes equal to $e^\eta > 1$ for some $\eta > 0$, we have

$$\|y^{(l)} - (I_N y)^{(l)}\|_{L^2_\omega(-1,1)} \le \frac{C(l)}{\sinh(\eta)} N^{2l} e^{-\eta N} \max_{z \in E_\eta} |y(z)|,$$

for all $l > 0$, see [92, Chapter 5]. This shows that the interpolation error, as well as the error in the interpolation differentiation matrices decays exponentially quickly in N.

3.2.3 Collocation and Galerkin Methods

Now that we know how to approximate a function y using polynomial expansions, we can apply this to solving partial differential equations, following [92, Chapter 6]. In particular, as an example, problem (3.4) from Section 3.1 is considered. Let X, V, W be Hilbert spaces and consider for $y \in X$

$$\begin{aligned} Ly &= f \quad \text{in } \Omega, \\ y &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

where L is a linear operator acting on y. The weak formulation (3.6) given in the same section is: For $y \in W$,

$$a[y,v] = (f,v) \quad \forall v \in V. \tag{3.47}$$

During integration by parts derivatives on y are shifted onto v. Therefore, $y \in W$ does not need to be as regular as in the strong form of the PDE, which is a key motivation for using the weak formulation. As a consequence, more regularity is required for $v \in V$ than before integrating by parts. In the cases considered here, we use $V = W$. Spectral methods can be used to solve the weak and strong form of the PDE. In the following, we present two of these: collocation methods are used in the context of the strong form of the PDE, while Galerkin methods utilize the weak form. There is a third approach, called the tau method, which is omitted here but can be found in [92, 93].

In the following, let X_N and Y_N be finite-dimensional subspaces of X. Moreover, we assume that X_N is also a subspace of V. If we consider solving the weak form of the PDE, we can use

a *Galerkin method*, in which y_N and v are chosen to be in the same subspace X_N of X. In particular, let $\{\Phi_k\}$ be a basis of X_N, so that $y_N = \sum_k \hat{y}_k \Phi_k$ and $v = \sum_k \alpha_k \Phi_k$. The basis $\{\Phi_k\}$ could for example be the set of orthogonal Fourier or Chebyshev polynomials in X_N. The discrete weak formulation is

$$\begin{aligned} &y_N \in X_N, \\ &(Ly_N, v) = (f, v) \quad \forall v \in X_N, \end{aligned} \tag{3.48}$$

or, after integration by parts,

$$\begin{aligned} &y_N \in X_N, \\ &a[y_N, v] = (f, v) \quad \forall v \in X_N. \end{aligned} \tag{3.49}$$

One can numerically integrate this weak formulation, so that a is replaced by its discrete version a_N. Alternatively, one can consider a *collocation method* which uses the strong form of the PDE. This is

$$\begin{aligned} L_N y_N(x_k) &= f(x_k) \quad &&\text{for } x_k \in \Omega, \\ y_N(x_k) &= 0 &&\text{for } x_k \in \partial\Omega. \end{aligned} \tag{3.50}$$

Here, L_N is the discretized operator L, which is obtained using differentiation matrices D_N.

3.2.4 Stability and Convergence

Stability of the chosen method is given when we can find a bound on the approximate solution, in some norm, in terms of given data and independent of N. Convergence is determined by a bound on the error of the approximation to the true solution in an appropriate norm.

Galerkin Method

We recall the Lax–Milgram Theorem 3.1.6 from Section 3.1. If the weak formulation (3.47) satisfies the conditions of this theorem, we can consider (3.48) and assume that each X_N is in V. Then Theorem 3.1.6 gives that

$$\|y_N\|_V \leq C\|f\|_X. \tag{3.51}$$

Therefore, the approximation y_N is bounded by given data f, and the numerical scheme is stable. In cases where the conditions of the Lax–Milgram Theorem are not directly satisfied, one can use other results to arrive at a similar stability estimate, see [92, Chapter 6]. Convergence of Galerkin methods is given by the Lax–Richtmyer equivalence theorem [101, 102]. This states that if the given numerical method is *consistent*, then convergence follows if the method is stable. A consistency condition for a numerical method generally requires that an exact solution y satisfies the discrete Galerkin approximation. Therefore, it is required that X is well approximated by the family of X_N, which is discussed in [92, Chapter 6]. Under this condition the convergence is dictated by the best approximation of y in X_N with respect to the V-norm. This is an application of Céa's Lemma [103, Chapter 2], which states

$$\|y - y_N\|_V \leq C \inf_{v \in X_N} \|y - v\|_V.$$

Choosing $X_N = \mathbb{P}_N$ and $V = L^2_\omega$, the error bounds for Chebyshev expansions can be used to find the convergence bound. Equivalent approaches can be taken for the Fourier case.

Collocation Method

As before, we assume that the conditions of the Lax–Milgram Theorem hold and that the X_N are contained in V. Then result (3.51) holds for collocation methods as well and the collocation scheme (3.50) is stable. Extensions of this are discussed in [92, Chapter 6]. The consistency condition for collocation methods is similar to that of Galerkin methods and can be found in

[92, Chapter 6]. The convergence bound is more complicated than in the Galerkin case and is given by

$$\|y - y_N\|_W \leq C_1 \|y - R_N y\|_W + C_2 \sup_{\substack{v \in Y_N \\ v \neq 0}} \frac{|(LR_N y, v) - (L_N R_N y, v)_N|}{\|v\|_V} + C_2 \sup_{\substack{v \in Y_N \\ v \neq 0}} \frac{|(f, v) - (f, v)_N|}{\|v\|_V}.$$

The positive constants C_1 and C_2 depend on those given in the Lax–Milgram Theorem and are independent of N. Here, $W \subseteq X$, $V \subseteq X$, $X_N \subset W$, $Y_N \subset V$, and $R_N y$ is a projection from a dense subspace of the domain of L, denoted by $D(L) \subset X$, into $X_N \cap D(L)$. One can see how when choosing for example a Chebyshev expansion with $X_N = Y_N = \mathbb{P}_N$, $X = L^2_w$ with $W \subseteq X$, $V \subseteq X$, the error estimates from Section 3.2.2 can be applied to find the rate of convergence. In [92, Chapter 6] further stability and convergence results are discussed, including results for problems in higher dimensions and problems that depend on time.

Poisson Example

The theoretical results discussed in the previous section can be illustrated by considering a simple example. Following [92, Chapter 7], we consider the Poisson problem

$$\begin{aligned} -\nabla^2 y &= f \quad \text{in } \Omega, \\ y &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

where $\Omega = (-1, 1)$. Given that Chebyshev approximations are the most relevant for this thesis, the Galerkin and collocation approaches with respect to this expansion are discussed. The domain of L, $D(L)$ is now given by

$$D(L) = \{v \in H^2_\omega(\Omega) : v|_{\partial\Omega} = 0\},$$

and $y \in D(L)$. Integrating by parts one can derive the weak formulation of the Poisson problem, which can be compared to the approach in deriving (3.3). The difference is that the weak form is now weighted by the Chebyshev weight ω. For $y, v \in H^1_{\omega,0} := \{v \in H^1(-1,1) | v(-1) = v(1) = 0\}$, we have that

$$\begin{aligned} a[y, v] &= \int_\Omega \nabla y \cdot \nabla(v\omega) dx, \\ F(v) &= \int_\Omega f v \omega dx. \end{aligned}$$

In this example, it is evident that the regularity of y is decreased from $H^2_{\omega,0}$ in the strong Poisson problem to $H^1_{\omega,0}$ in the weak formulation. One can show, see [92, Chapter 7], that this weak formulation satisfies the condition of the Lax–Milgram Theorem 3.1.6. Therefore, it follows that the method is stable in the norm induced by $H^1_{\omega,0}$. Convergence is then measured in this norm and so we choose $y, v \in \mathbb{P}^0_N(\Omega)$, i.e., polynomials that vanish on the boundary of the domain. Then, a similar estimate to the one given by (3.46) can be formed, by considering the H^1_ω norm instead of the L^2_ω norm. This is

$$\|y - y_N\|_{H^1_\omega(-1,1)} \leq C N^{1-m} |y|_{H^{m;N}_\omega(-1,1)},$$

with $m \geq 1$.

For collocation methods, in order to show stability and convergence of the method, one has to distinguish between one and more dimensions. In one dimension, stability can be shown by using the fact that that the collocation points are nodes of Gauss–Lobatto quadrature given in

Theorem 3.2.1. This relates the discretized and continuous versions of the problem

$$\sum_{j=0}^{N} y_{xx}(x_j)v(x_j)\omega_j = \int_{-1}^{1} y_{xx}v\omega dx, \quad \forall v \in \mathbb{P}_N.$$

Using this weak form, we can use the Lax–Milgram Theorem to establish stability. For higher dimensions this is a bit more involved. The approximation $y_N \in \mathbb{P}_N^0(\Omega)$ and the discrete inner product are defined. This is comparable to (3.28), but now extended to two dimensions and higher-dimensional generalizations are possible. The inner product is

$$(y, v)_N = \sum_{i=0}^{N}\sum_{j=0}^{N} y(\vec{x}_{i,j})v(\vec{x}_{i,j})\omega_i\omega_j,$$

where the $\vec{x}_{i,j} = \left[\cos\left(\frac{i\pi}{N}\right), \cos\left(\frac{j\pi}{N}\right)\right]$. Then the discrete bilinear form is

$$a_N[y, v] = -(\nabla^2 y, v)_N,$$

and the weak formulation is

$$a_N[y_N, v] = (f, v)_N \quad \forall v \in \mathbb{P}_N^0(\Omega).$$

It is shown in [92, Chapter 7] that the conditions of the Lax–Milgram Theorem hold with respect to the $H_\omega^1(\Omega)$ norm, and so stability is given. Then the convergence result for the method in this norm is given by

$$\|y - y_N\|_{H_\omega^1(\Omega)} \leq CN^{1-m}\left(|y|_{H_\omega^{m;N}(\Omega)} + |f|_{H_\omega^{m-1;N}(\Omega)}\right),$$

for $m > 2$.

3.2.5 Other Numerical Methods

In this section, a brief summary of other common approaches is given and it is discussed how they compare to spectral methods. Namely, we will focus on finite element methods, finite difference methods, and finite volume methods. Moreover, a brief discussion on time stepping methods is provided.

Finite Element Methods

The discussion on finite elements provided here mostly follows [103]. The development of the finite element method starts with the Galerkin formalism introduced above. One considers the weak formulation of a PDE given by (3.47). Then, a basis of test functions $\{\Phi_k\}$ on a subspace X_N of X is chosen and the discrete version of the weak form given by (3.49) is derived. The key difference is now that the Φ_k are low order polynomials, such as piecewise linear functions defined by $\Phi_i(x_j) = \delta_{ij}$, which are not given on the whole domain, as in spectral methods, but on a single *element*, i.e., a part of the domain. In one dimension, elements are intervals $[a+ih, a+2ih]$, $i = 0, ..., N$, formed by subdividing the domain $[a, b]$. In two dimensions the domain can be discretized by triangular elements, for example, when solving a (time-independent) PDE.

This approach results in a large, sparse matrix–vector system of equations, in which each sub-block corresponds to the discretized PDE on an element in the domain. This is contrasted by spectral methods, which discretize the PDE on the whole domain, resulting in dense, but comparatively small matrices. This is due to rapid convergence of spectral approximations, allowing comparatively low-dimensional discretizations of PDEs. Finite element methods are an active area of research and have gained much popularity in engineering and industry applications. For many problems the matrix–vector systems are relatively easy to solve, due to the

sparse matrix structure, and the discretization of the domain into elements allows one to tackle problems on complicated domains, making it an attractive tool in applications. However, due to the low order approximations on each elements by, most often, piecewise linear, and sometimes quadratic or cubic polynomials, one cannot expect high orders of accuracy.

In order to obtain a high resolution approximation, one would have to use a large number of elements, which increases the dimension of the matrix system to solve. Increasing the degree of the interpolating polynomial is another approach for increasing accuracy. However, with equispaced points within each element, one faces the Runge phenomenon discussed above. An approach that can overcome this is by using *spectral elements*, which use high order polynomials and non-equispaced points on each element. This is discussed in Chapter 6. In [92, Chapter 1], an illustrative comparison between the numerical results achieved with a spectral and a finite element method is provided, demonstrating the superior convergence of the spectral method. However, we note that this is highly problem-specific. Spectral methods are advantageous for problems on simple domains that have high regularity due to their rapid convergence properties in such cases. For problems on more complicated domains one may use a spectral or finite element method, see Chapter 6 for a discussion. For problems with solutions of low regularity, a finite element method may be superior.

In this thesis, we consider problems that are 'smooth' in the sense that they are diffusion-dominated and in that we do not expect any discontinuities or shocks in the solutions. Moreover, the PDE models considered are nonlocal, which means that the finite element discretization would be dense, as opposed to its usual sparse structure. The reason this dense matrix structure arises is that at each point $\vec{x}$ the particle–particle interaction term requires information at all other points in the domain, coupling all elements of the discretization with each other. The sparsity of the finite element matrices are a key feature of such algorithms, that makes it feasible to solve the resulting large matrix systems efficiently. Problems that result in large, dense matrix systems, such as nonlocal problems, can therefore make a finite element approach very expensive and spectral methods preferable.

Finite Difference Methods

Finite difference methods are again polynomial expansion methods, but in contrast to the finite element method, one considers the whole domain. This section follows the account in [104, Chapter 2] and is restricted to one-dimensional considerations in the interest of brevity. Higher-dimensional cases are discussed in the reference text. In finite difference methods, one divides the grid using a constant step-size Δx, so that each two adjacent grid points satisfy $x_{i+1} - x_i = \Delta x$, $i = 0, ..., N$. Assuming sufficient regularity of y, one considers the Taylor expansions

$$\begin{aligned} y(x_{i+1}) = y(x_i) + \Delta x \frac{dy(x_i)}{dx} + (\Delta x)^2 \frac{1}{2!} \frac{d^2y(x_i)}{dx^2} + (\Delta x)^3 \frac{1}{3!} \frac{d^3y(x_i)}{dx^3} \\ + (\Delta x)^4 \frac{1}{4!} \frac{d^4y(x_i)}{dx^4} + O\left((\Delta x)^5\right), \end{aligned} \tag{3.52}$$

and

$$\begin{aligned} y(x_{i-1}) = y(x_i) - \Delta x \frac{dy(x_i)}{dx} + (\Delta x)^2 \frac{1}{2!} \frac{d^2y(x_i)}{dx^2} - (\Delta x)^3 \frac{1}{3!} \frac{d^3y(x_i)}{dx^3} \\ + (\Delta x)^4 \frac{1}{4!} \frac{d^4y(x_i)}{dx^4} + O\left((\Delta x)^5\right). \end{aligned} \tag{3.53}$$

One can now add these two equations to each other, to obtain

$$y(x_{i+1}) + y(x_{i-1}) = 2y(x_i) + 2(\Delta x)^2 \frac{1}{2!} \frac{d^2y(x_i)}{dx^2} + O\left((\Delta x)^4\right).$$

Rearranging this gives an approximation to the second derivative of y at x_i

$$\frac{d^2y(x_i)}{dx^2} = \frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{(\Delta x)^2} + O\left((\Delta x)^2\right). \tag{3.54}$$

We can then approximate $\frac{d^2y(x_i)}{dx^2}$ on each grid point using this relationship, which is called a second-order central difference scheme. This name stems from the fact that values of y at both sides of x_i are needed for the approximation. For this scheme, the error is of order $(\Delta x)^2$, and it decreases with decreasing step size. There are many different finite difference schemes with higher accuracy, but they are, in essence, based on the above idea. Using this approximation, we can discretize the Poisson equation in one dimension by

$$-\frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{(\Delta x)^2} = f(x_i).$$

Boundary conditions are applied at the relevant nodes. If Dirichlet conditions are applied then these are simply prescribed at $y(x_0)$ and $y(x_N)$. For Neumann boundary conditions, we can derive a finite difference discretization of the first derivative $\frac{dy(x_i)}{dx}$ at the boundary points, see [104, Chapter 2] for a discussion on the application of different boundary conditions.

As illustrated, finite difference methods approximate the PDE by low-order, overlapping polynomial expansions found by means of Taylor series. These are simple tools for the discretization of numerical methods, which are easily implementable, and give quick results. However, their main drawback is that the accuracy to low order depends on the constant step size Δx, meaning that a high number of points has to be used for a high resolution result. Using higher order polynomial approximations can somewhat improve the convergence, but, as discussed before, high order polynomial approximations on equispaced points are generally not as accurate as one might hope, due to the Runge phenomenon.

Finite Volume Methods

Following [104, Chapter 6], the idea of the finite volume method is to discretize the spatial domain into cells of volumes V_i. One can then average the value of the variable of interest over this cell by computing

$$\overline{y}_i = \frac{1}{|V_i|}\int_{V_i} y dV.$$

The Poisson equation in one dimension, neglecting boundary conditions for the time being, is

$$-\frac{d^2y}{dx^2} = \frac{d}{dx}\mathrm{j} = f \quad \text{in } [a,b],$$

so that the flux is defined by

$$\mathrm{j} = -\frac{dy}{dx}.$$

We split up the interval (or volume) $[a,b]$ into equispaced cells, such that the point x_i is the midpoint of the 1D volume $V_i = [x_{i-1} + \frac{\Delta x_i}{2}, x_{i+1} - \frac{\Delta x_i}{2}] := [l_i, u_i]$, for $i = 0, ..., N$. The size of each cell is Δx_i. Integrating the Poisson equation over each cell results in

$$-\left.\frac{dy}{dx}\right|_{u_i} + \left.\frac{dy}{dx}\right|_{l_i} = f_i \Delta x_i,$$

which is

$$\mathrm{j}_{u_i} - \mathrm{j}_{l_i} = f_i \Delta x_i. \tag{3.55}$$

This is a continuity condition, describing the difference in flux over a volume V_i. Now, enforcing flux continuity between two neighbouring cells, we require that $\mathrm{j}_{u_i} = \mathrm{j}_{l_{i+1}}$. Doing this for all the volume cells results in the equation

$$\mathrm{j}_{u_N} - \mathrm{j}_{l_0} = f_1 \Delta x_1 + f_2 \Delta x_2 + ... + f_N \Delta x_N = \int_a^b f dx,$$

which enforces mass conservation over the whole domain.

The idea is now to approximate $\frac{dy}{dx}$ using a finite difference approach, and use this to solve (3.55) for each cell. One can rewrite this equation to recover the finite difference scheme (3.54) in the case of uniform cell sizes. For non-uniform cells, the finite volume method has the potential to be more versatile than the finite difference method. The application of boundary conditions is illustrated in [104, Chapter 6]. The easiest case for this method is the application of Neumann conditions, by replacing the terms at l_1 and u_N in (3.55) by the boundary conditions. Finite volume methods can have similar limitations to finite difference methods, especially at constant step size. One key advantage of finite volume methods is that, since it matches the flux j, it preserves the mass of the system, which is a desirable feature in real-world models.

A Brief Discussion of Time Stepping

We very briefly discuss some standard choices of time stepping schemes, following [104, Chapter 5]. Now that we have discussed how to discretize spatial operators, another question is how to solve a time dependent PDE, such as the one-dimensional heat equation

$$\begin{aligned} \frac{\partial y}{\partial t} &= \frac{\partial^2 y}{\partial x^2} && \text{in } (0,T) \times \Omega, \\ y(t, \vec{x}) &= 0 && \text{on } (0,T) \times \partial\Omega, \\ y(0, \vec{x}) &= y_0(\vec{x}) && \text{at } \{t = 0\} \times \Omega. \end{aligned} \tag{3.56}$$

First, we can apply the spatial discretization scheme of our choice for continuous time. Employing the finite difference scheme (3.54), we obtain

$$\begin{bmatrix} 0 \\ \frac{d}{dt} y(t, x_1) \\ \frac{d}{dt} y(t, x_2) \\ \vdots \\ \frac{d}{dt} y(t, x_{N-2}) \\ \frac{d}{dt} y(t, x_{N-1}) \\ 0 \end{bmatrix} = \frac{1}{(\Delta x)^2} \begin{bmatrix} 1 & & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & & \ddots & & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \\ & & & & & & 1 \end{bmatrix} \begin{bmatrix} y(t, x_0) \\ y(t, x_1) \\ y(t, x_2) \\ \vdots \\ y(t, x_{N-2}) \\ y(t, x_{N-1}) \\ y(t, x_N) \end{bmatrix}, \tag{3.57}$$

subject to the vector of initial conditions

$$\left[y_0(x_0), y_0(x_1), y_0(x_2), ..., y_0(x_{N-2}), y_0(x_{N-1}), y_0(x_N) \right]^\top.$$

We can see that, for $i = 1, ..., N-1$, each row corresponds to an ODE that is of the form

$$\begin{aligned} \frac{d}{dt} y(t, x_i) &= \frac{1}{(\Delta x)^2} \left(y(t, x_{i+1}) - 2y(t, x_i) + y(t, x_{i-1}) \right) && \text{in } (0,T), \\ y(0, x_i) &= y_0(x_i) && \text{at } \{t = 0\}, \end{aligned} \tag{3.58}$$

while for $i = 0$ and $i = N$ we have

$$y(t, x_0) = y(t, x_N) = 0 \quad \text{in } (0,T).$$

Each row of the matrix vector system, apart from the first and last, is an ODE with the right-hand side corresponding to finite difference discretization (3.54). The first and last row of the left-hand side of the system are set to zero, so that the computation of these rows results in Dirichlet boundary conditions. We have reduced the PDE problem to a system of ODEs and algebraic equations. In this case the algebraic equations are simply the constant Dirichlet conditions. However, for other boundary conditions, these become more involved.

There are several different time stepping schemes available that can now be applied to each of the ODEs given by (3.58). The simplest of these is possibly the *forward Euler method*, which uses the Taylor expansion (3.52) with x_i replaced by t_j. The update t_{j+1} is given by the constant time step

$$t_{j+1} = t_j + \Delta t.$$

We rearrange (3.52) to get the forward Euler method

$$\frac{dy(t_j)}{dt} = \frac{y(t_{j+1}) - y(t_j)}{\Delta t} + O(\Delta t).$$

Combining this with the finite difference approximation for the spatial derivative (3.54), we can approximate the heat equation (3.56) by the scheme

$$\frac{y_{j+1,i} - y_{j,i}}{\Delta t} = \frac{y_{j,i+1} - 2y_{j,i} + y_{j,i-1}}{\Delta x^2},$$

where $y_{i,j} := y(t_j, x_i)$, and which is accurate to $O(\Delta x^2)$ and $O(\Delta t)$. This scheme is *explicit*, since we can rearrange to solve for the next time step $y_{j+1,i}$. In order for this method to be stable, one has to choose a step size Δt adhering to

$$\Delta t \leq \frac{1}{2}\Delta x^2,$$

i.e., the time step has to be chosen dependent on the spatial grid size. The method is therefore *conditionally stable*. If one uses (3.53) instead, the resulting scheme is called the *backward Euler method*

$$\frac{dy_{j+1}}{dt} = \frac{y_{j+1} - y_j}{\Delta t} + O(\Delta t).$$

In this case however, the discretization of (3.56) becomes

$$\frac{y_{j+1,i} - y_{j,i}}{\Delta t} = \frac{y_{j+1,i+1} - 2y_{j+1,i} + y_{j+1,i-1}}{\Delta x^2},$$

which cannot be rearranged explicitly for the next time step t_{j+1}. Therefore, this is called an *implicit method*, which results in a linear system that needs to be solved. The reason why this is preferred over the forward Euler method, is because this method is *unconditionally stable* for our problem, so that the step size Δt can be chosen independently from the spatial discretization. This is not the case for the forward Euler method, as seen above. While the stability is improved by the backward Euler method, it is still only $O(\Delta t)$ accurate. A method with $O(\Delta t^2)$ accuracy is the *Crank-Nicolson scheme*

$$\frac{y_{j+1,i} - y_{j,i}}{\Delta t} = \frac{1}{2}\frac{y_{j,i+1} - 2y_{j,i} + y_{j,i-1}}{\Delta x^2} + \frac{1}{2}\frac{y_{j+1,i+1} - 2y_{j+1,i} + y_{j+1,i-1}}{\Delta x^2}.$$

This is an implicit method and again unconditionally stable for the considered PDE. Details of this method can be found in [104, Chapter 5].

In this thesis, the MATLAB solver `ode15s` is used, which is based on numerical differentiation formulas, and which can tackle the differential–algebraic equations that arise from solving the integro-PDE models from particle dynamics. This solver is discussed in Section 3.3.6.

3.3 Pseudospectral Implementation

After discussing theoretical results on PDEs and some important numerical approaches to solving such equations, we now turn towards solving the PDEs that are of interest in this thesis. They are of the form (1.1), which are nonlinear, nonlocal DDFT models such as

$$\frac{\partial \rho}{\partial t} = \nabla^2 \rho + \nabla \cdot (\rho \nabla V_1) - \nabla \cdot (\rho \vec{w}) + \nabla \cdot \int_\Omega \rho(t, \vec{x}) \rho(t, \vec{x}') \nabla V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \quad \text{in } (0, T) \times \Omega, \tag{3.59}$$

with Dirichlet, periodic, or nonlinear, nonlocal no-flux boundary conditions, where the latter are given by

$$\frac{\partial \rho}{\partial n} + \rho \frac{\partial V_1}{\partial n} - \rho \vec{w} \cdot \vec{n} + \int_\Omega \rho(t, \vec{x}) \rho(t, \vec{x}') \frac{\partial V_2(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' = 0 \qquad \text{on } (0, T) \times \partial\Omega. \tag{3.60}$$

Note that the notation of the PDE variable has now changed from y to ρ, to align with the standard notation in DDFT. Solving such models numerically is challenging due to their nonlinear and nonlocal nature. In designing an efficient numerical method, the following considerations have to be made: How to effectively

- interpolate from one set of collocation points onto another,
- discretize spatial derivatives,
- approximate integrals,
- compute convolution integrals,
- apply nonlocal, nonlinear boundary conditions, and
- compute time stepping.

Note that the interpolation is necessary for the plotting of solutions, as well as for the computation of quantities on subdomains. Moreover, it is used in the optimal control solver that is introduced in Chapter 5. In this section it is discussed how to implement such a method in one, two, and three dimensions for periodic and non-periodic grids. For interpolation, differentiation, and integration, Chebyshev or Fourier collocation methods are employed, depending on the periodicity requirements of the problem at hand.

Applying no-flux boundary conditions, and in particular nonlocal no-flux boundary conditions, is highly challenging using standard approaches. Most commonly, boundary conditions are applied using *boundary bordering*, see [93, Chapter 6], [94, Chapter 13]. In order to illustrate the method, a simple example is presented. We solve the Poisson equation with zero Neumann boundary conditions, as done in [94, Chapter 13], given by

$$\begin{aligned} \nabla^2 \rho &= f \quad \text{in } \Omega, \\ \frac{\partial \rho}{\partial n} &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

On a domain $\Omega = [-1, 1]$, discretized by $N + 1$ points $-1 = x_0 < x_1 < ... < x_{N-1} < x_N = 1$, the discretized PDE is $D_{xx}\boldsymbol{\rho} = \mathbf{f}$, where D_{xx} is a matrix and $\boldsymbol{\rho}$, $\mathbf{f}$ are vectors. The boundary conditions are implemented by replacing the first and last rows and columns of D_{xx}, which correspond to the domain boundaries $x = -1$ and $x = 1$, with the equivalent entries of D_x. Moreover, we set $\mathbf{f}(-1) = \mathbf{f}(1) = 0$. Then, computing this modified version of the discretized Poisson equation applies the correct boundary conditions. While this is a straightforward approach for simple, linear equations, this becomes highly non-trivial for PDEs with nonlinear boundary conditions. The numerical method described in this section takes another approach, incorporating complicated boundary conditions into the PDE discretization as algebraic equations, which can be tackled by using a differential–algebraic equation solver.

Equilibrium DFT problems have been solved using finite element methods, see [105, 106]. The convolution integrals of the form

$$(\rho \star \chi)(\vec{y}) = \int \chi(\vec{y} - \vec{z})\rho(\vec{z})d\vec{z},$$

arising from particle interactions have to be computed numerically. Traditionally, this is done using Fast Fourier Transforms, see [107, 108, 109]. This means one has to compute

$$(\rho \star \chi)(\vec{y}) = F^{-1}\{F\{\rho\} \cdot F\{\chi\}\}(\vec{y}). \tag{3.61}$$

The issue is that these computations have to be done on a periodic grid, which is not convenient for many real-world applications that are better captured by non-periodic conditions, such as no-flux boundary conditions. Moreover, in the case of DDFT, this has to be done once for each time step in a PDE solver, which becomes expensive.

DDFT models have been computed using finite element methods [110], finite difference methods for lattice models [111], and, more recently, the finite volume method has also been applied to a range of DDFT problems, see [112, 113, 114, 115, 116, 117]. In this thesis, we are using a Chebyshev collocation method to compute (3.59), as first developed in [84]. This approach allows for the efficient computation of convolution integrals arising in both DFT and DDFT, not relying on FFT or periodic grids and not having to be recomputed at each time step of a PDE solver.

In this section, the computational implementation of this Chebyshev collocation scheme in MATLAB is discussed. We will refer to this as *pseudospectral method*. Most of the approach presented here is part of the package 2DChebClass [118], presented in [84], and its extensions. Note that the 2DChebClass implementation is also able to compute problems on infinite and semi-infinite domains. These are not used in this thesis and are therefore omitted. We choose the spatial domain $\Omega = [a, b]$ in one, $\Omega = [a, b] \times [c, d]$ in two, and $\Omega = [a, b] \times [c, d] \times [e, f]$ in three dimensions, where $a, b, c, d, e, f \in \mathbb{R}$. We furthermore briefly discuss a Fourier collocation scheme, used for periodic domains.

3.3.1 Interval (1D)

First, we discuss the spectral discretization of a finite interval in one dimension. In order to construct such a line in the 2DChebClass library, three inputs have to be given; the two endpoints of the interval and the number of discretization points N.

Discretization Points, Boundaries, Normals

The Chebyshev spectral method presented in Section 3.2.1 is applied. The collocation points $\{x_j\}$ are defined by (3.33). Using this definition, the points are distributed from 1 to -1, which is counter-intuitive. Therefore, in the code library 2DChebClass [84], the Chebyshev points are automatically flipped to run from -1 to 1. These points are clustered at the endpoints of the interval, and sparse around the centre of the domain, as can be seen in Figure 3.1. Most com-

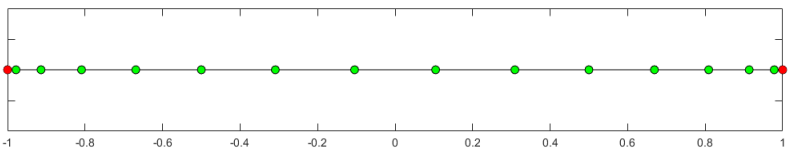

Figure 3.1: Chebyshev points with $N = 16$ in the interval $[-1, 1]$. The red points denote the domain boundary.

putations in 2DChebClass are done on this canonical domain $[-1, 1]$, called the *computational*

domain. A vector $\vec{x}$ on $[-1,1]$ can then be mapped onto a *physical domain* $[a,b]$ of interest via the following linear map

$$\vec{y} = a + \frac{(\vec{x}+1)(b-a)}{2}. \tag{3.62}$$

The inverse map is

$$\vec{x} = -1 + \frac{2(\vec{y}-a)}{b-a}, \tag{3.63}$$

see [93, Chapter 2]. In the implementation, the boundary of the domain, as well as the outward normal vector at each boundary point, are found automatically. The boundary of the domain, illustrated in Figure 3.1, is found by evaluating the computational minimum and maximum $x_{\min}$ and $x_{\max}$, rather than the physical $y_{\min}$ and $y_{\max}$, since they are in one-to-one correspondence via the linear map (3.62). The result is stored in a vector of size N, which contains ones at the boundary points and zeros everywhere else. The two one-dimensional outward unit normals are defined as $n_1 = -1$, located at $y_{\min}$, and $n_2 = 1$ at $y_{\max}$.

Interpolation

Interpolation is an important feature of the given numerical method, needed for plotting of numerical results, as well as for the computation of quantities on subdomains. As discussed in Section 3.2.1, there exists a unique polynomial of degree $\leq N$ that can be used to interpolate a function ρ defined on the grid points x_j. Here, barycentric Lagrange interpolation (3.35) is used. The interpolant $I_N\rho$ satisfies, by definition, the following relationship

$$I_N\rho(x_j) = \rho(x_j), \tag{3.64}$$

so that the residual $I_N\rho(x_j) - \rho_j$ is zero at these points. In the code library 2DChebClass, interpolating from the set of points $\{x_j\}$, onto another set of points $\{x_i\}$, where $i = 0, ..., M-1$ and $j = 0, ..., N-1$, is implemented as a matrix–vector product. The interpolation matrix is of the form

$$\text{Interp}_{ji} = \frac{1}{\omega_j}\left(\frac{\tilde{w}_i}{x_j - x_i}\right), \qquad \omega_j = \sum_{k=0}^{N-1} \frac{\tilde{w}_k}{x_i - x_k},$$

where the $\tilde{w}_i$ are defined in (3.36). The set of collocation points $\{x_j\}$ lies in computational space $[-1,1]$, while the second set of points can be customized by the user, to be any M points in $[-1,1]$. If the user would like to interpolate onto a set of M points $\{x_i\}$ in physical space, this is possible in 2DChebClass as well. This set of points is then first mapped onto the computational domain, using the linear map (3.63), before the interpolation matrix is computed, as described above. This method can then be applied to the interpolation of an arbitrary set of values $\{\rho_j\} := \{\rho(x_j)\}$, the set of values found by evaluating ρ at the collocation points $\{x_j\}$ onto the new set of points $\{\rho_i\}$, which denotes ρ evaluated at the set of interpolation points $\{x_i\}$.

Differentiation

The Chebyshev differentiation matrices are given by (3.37). The second derivative of a function ρ with respect to x, evaluated at the Chebyshev points $\{x_j\}$, is represented by the second differentiation matrix D_{xx}. This can be found by squaring the first differentiation matrix; $D_{xx} = D_x^2$, and more generally the j^{th} differentiation matrix D_{x^j} is defined to be

$$D_{x^j} = D_x^j.$$

However, in [84], D_{xx} is computed by taking the second derivative of the interpolant $I_N\rho$, since it is more accurate than squaring D_x. All differentiation matrices are derived in computational space and then mapped to physical space using the Jacobian transformation of the linear map

(3.62), which is defined to be

$$J = \operatorname{diag}\left(\tilde{J}\right), \quad \tilde{J}_j = \frac{dy_j}{dx_j}, \quad j = 0, ..., N. \tag{3.65}$$

Since the map (3.62) is linear, we have $\frac{dy_j}{dx_j} = \frac{b-a}{2}$. Then the physical differentiation matrix D_y is defined as $D_y = JD_x$. Higher-order differentiation matrices are mapped to physical space in a similar manner.

Integration

In order to evaluate integrals, the so-called *Clenshaw–Curtis quadrature* is used, which is derived in [119]. For the integral over a smooth function ρ, this is

$$\int_{-1}^{1} \rho(x)dx = \sum_{j=0}^{N-1} w_j \rho(x_j), \tag{3.66}$$

where the weights are defined as:

$$w_j = \frac{2d_j}{N} \begin{cases} 1 - \displaystyle\sum_{k=1}^{(N-2)/2} \frac{2\cos(2kt_j)}{4k^2-1} - \frac{\cos(\pi j)}{N^2-1} & \text{for } N \text{ even}, \\ 1 - \displaystyle\sum_{k=1}^{(N-2)/2} \frac{2\cos(2kt_j)}{4k^2-1} & \text{for } N \text{ odd}, \end{cases}$$

see [84]. In 2DChebClass integration is implemented as a row vector, such that $\left(\text{Int}_{\text{comp}}\right)_j = w_j$. Again, this is done in computational space, so that we multiply Int_{comp} pointwise with the transposed Jacobian transformation (3.65) to map onto a desired physical domain $[a, b]$. This is

$$\left(\text{Int}\right)_j = \left(\text{Int}_{\text{comp}}\right)_j \tilde{J}_j.$$

The integration vector can then be applied to a vector $\boldsymbol{\rho}$, denoting the variable ρ evaluated at the set of collocation points. This is

$$\int_a^b \rho(x)dx \approx \text{Int}\boldsymbol{\rho}.$$

Convolution

The final aspect to be considered is the convolution matrix, which is needed to compute convolution integrals arising from particle–particle interactions. As stated above, the convolution integral is defined as

$$\left(\rho \star \chi\right)(\vec{y}) = \int \chi(\vec{y} - \vec{z})\rho(\vec{z})d\vec{z},$$

for a density ρ and a weight function χ. We apply the integration formula (3.66), discretize $\vec{z}$ at the collocation points $\{x_j\}$, denoted by $\{z_j\}$, and evaluate ρ at these points. This results in

$$\left(\rho \star \chi\right)(\vec{y}) \approx \sum_{j=0}^{N-1} w_j \tilde{J}_j \chi(\vec{y} - z_j)\rho(z_j).$$

Then, we replace the sum with the integration vector Int, and discretize $\vec{y}$ to give the vector $\boldsymbol{y}$. At an index j, the convolution integral becomes

$$\left[\left(\rho \star \chi\right)(\boldsymbol{y})\right]_j = \text{Int} \;\; \chi(\boldsymbol{y} - z_j)\rho(z_j).$$

It is clear that z_j is subtracted from each entry in $\boldsymbol{y}$. So if we consider an entry i of the vector $\boldsymbol{y}$, the convolution integral is defined as

$$\left[(\rho \star \chi)(\boldsymbol{y})\right]_{ij} = \text{Int} \quad \chi(y_i - z_j)\rho(z_j). \tag{3.67}$$

The convolution matrix Conv can be created as

$$\text{Conv}_{i,j} := \text{Int}_j \quad \chi(y_i - y_j). \tag{3.68}$$

The convolution integral is now defined as matrix vector multiplication, by applying the matrix Conv to any density vector $\boldsymbol{\rho}$

$$(\rho \star \chi)(\vec{y}) \approx \text{Conv}\boldsymbol{\rho}.$$

The convolution matrix can be applied to different densities ρ, which saves computational time. This is contrasted by the fast Fourier transform (FFT) approach for solving convolution integrals [20]. Here, at each step in time a Fourier transform of ρ is taken, and the convolution theorem is applied, as discussed above. It is clear that the present approach does not exhibit these complications, and therefore can lead to noticeable computational advantages when used within a sub-routine. Moreover, as discussed above, while the FFT method requires periodic domains, the pseudospectral approach does not, since it only requires the discrete integration vector for a given domain.

3.3.2 Periodic Line (1D)

The implementation of a periodic line is not part of the 2DChebClass implementation, but was added as an extension. Again, the end points of the interval, as well as the number of discretization points, need to be provided to initialize the construction of the line.

Domain Construction

The periodic line is defined in Fourier space, using an equispaced discretization with stepsize $h = 2\pi/N$, so that the coordinate $\vec{x}$ is periodic. In order to map from the canonical interval $[0, 1]$ to physical space $[a, b]$ we have the map

$$\vec{y} = a + (b - a)\vec{x},$$

and the corresponding inverse map

$$\vec{x} = \frac{\vec{y} - a}{b - a}.$$

Note that there are no boundaries defined in this case, and hence computation of outward normal vectors is not required.

Interpolation

The interpolation matrix on the periodic line is constructed using Discrete Fourier Transforms. The account below follows the work in [94]. The Discrete Fourier Transform formula is given by (3.41). The inverse transform is the Fourier interpolant (3.42), evaluated at the Fourier nodes $\vec{x}$. One small adjustment has to be made. We need to define $\hat{y}_{N/2} = \hat{y}_{-N/2}$ to get the interpolant

$$I_N y(x) = \sum_{k=-N/2}^{N/2} \hat{y}_k e^{ikx}, \tag{3.69}$$

where $x \in [-\pi/h, \pi/h]$, $h = 2\pi/N$, compare to (3.42). An equivalent definition of the interpolant is

$$I_N y(x) = \sum_{m=1}^{N} y_m S_N(x - x_m), \quad \text{with } S_N(x) = \frac{\sin(\pi x/h)}{(2\pi/h)\tan(x/2)}, \tag{3.70}$$

see [94] for a derivation. In the code, the interpolation matrix for the periodic line is defined as

$$\text{Interp}_F = \left[e^{2\pi i \vec{x}^I 0}, e^{2\pi i \vec{x}^I j_1}, \cos(2\pi \vec{x}^I M), e^{2\pi i \vec{x}^I j_2} \right],$$

where $j_1 = 1, ..., M-1$, $j_2 = M+1, ..., N$. Further, $M = (N+1)/2$ if N is odd and $M = N/2$ if N is even. Note that the cosine term results from combining the first two terms in (3.69). The set $\vec{x}^I$ is the set of points on which the x_j are interpolated. In order to map back to computational space, the Fast Fourier Transform matrix is considered

$$\text{FFTM}_{jk} = e^{-2\pi i jk/N},$$

where $j, k = 1, ..., N$. Then the interpolation matrix is given by

$$\text{Interp} = \Re\left(\text{Interp}_F \ \text{FFTM}\right).$$

Differentiation

In order to derive the differentiation matrices for the periodic grid, we return to the periodic interpolant (3.70). This can be differentiated to give

$$S'_N(x_j) = \begin{cases} 0, & j \equiv 0 \mod N, \\ \frac{1}{2}(-1)^j \cot(jh/2), & j \not\equiv 0 \mod N. \end{cases}$$

compare with (3.43). This gives one column of the differentiation matrix, which has Toeplitz structure and is created in the code as

$$D = 2\pi \begin{bmatrix} 0 & 0 & \cdots & 0 & -\frac{1}{2}\cot(h/2) \\ -\frac{1}{2}\cot(h/2) & & & & \frac{1}{2}\cot(h) \\ \frac{1}{2}\cot(h) & & & & -\frac{1}{2}\cot(3h/2) \\ -\frac{1}{2}\cot(3h/2) & & & & \\ & & \ddots & & \vdots \\ \vdots & & & & \\ & & & & \frac{1}{2}\cot(h/2) \\ \frac{1}{2}\cot(h/2) & 0 & \cdots & 0 & 0 \end{bmatrix},$$

see [94]. The factor of 2π maps the matrix back onto the grid $[0, 1]$.

Integration and convolution

In the Fourier domain, the integration weights defining the integration vector Int are given by $w_j = 1/N$. These are multiplied by the one-dimensional Fourier Jacobian of the form (3.65) to map the integration vector into physical space. In order to compute the convolution integral

$$(\rho \star \chi)(\vec{y}) = \int \chi(\vec{y} - \vec{z})\rho(\vec{z})d\vec{z},$$

the distance between the two vectors $\vec{y}$ and $\vec{z}$ on the periodic line needs to be computed. In particular, considering (3.67), we need to compute the distance between two points y_i and z_j,

$i, j = 1, .., N$. The periodic distance $d_{i,j} := y_i - z_j$ is given by

$$d_{i,j} = c_{i,j} - \frac{L}{2}, \quad \text{where} \quad c_{i,j} = z_i - y_j + \frac{L}{2} \mod L, \tag{3.71}$$

and $L = b - a$, the length of the domain. The convolution matrix is defined by

$$\text{Conv}_{i,j} = \text{Int}_j \ \ \chi(d_{i,j}),$$

where Int is the integration vector for the periodic box, compare to (3.68). If the function χ only takes one input, we compute

$$\text{Conv}_{i,j} = \text{Int}_j \ \ \chi(||d_{i,j}||_{l_2}).$$

3.3.3 Box (2D)

In order to construct a rectangular shape, or box, in 2DChebClass, the set of vertices $\{Y_i\} = \{(Y_1^{\min}, Y_2^{\min}, Y_1^{\max}, Y_2^{\max})\}$, has to be provided, as well as the number of discretization points in each direction, N_1 and N_2. The canonical domain is $[-1, 1]^2$.

Discretization points and domains

In order to extend the one-dimensional considerations to two-dimensional grids, a so-called tensor product grid has to be defined. First, Chebyshev points x_1^j, for $j = 0, ..., N_1 - 1$, on the x_1-axis and another set of Chebyshev points x_2^i, for $i = 0, ..., N_2 - 1$, on the x_2-axis are taken, both between $[-1, 1]$. Then the following two *Kronecker vectors* are defined:

$$\begin{aligned} \mathbf{x}_1^K &= \left[x_1^0, x_1^0, ..., x_1^0, x_1^1, x_1^1, ..., x_1^1, ..., x_1^n, x_1^n, ..., x_1^n\right]^\top, \\ \mathbf{x}_2^K &= \left[x_2^0, x_2^1, ..., x_2^m, x_2^0, x_2^1 ..., x_2^m, ..., x_2^1, x_2^2, ..., x_2^m\right]^\top, \end{aligned} \tag{3.72}$$

where $n = N_1 - 1$ and $m = N_2 - 1$. These are found by computing

$$\mathbf{x}_1^K = \mathbf{x}_1 \otimes \mathbf{e}_{N_2}, \qquad \mathbf{x}_2^K = \mathbf{e}_{N_1} \otimes \mathbf{x}_2, \tag{3.73}$$

where $\mathbf{x}_1$ is size $N_1 \times 1$ and denotes the set of Chebyshev points $\{x_1^j\}$ and $\mathbf{e}_N$ is the vector of ones of size $N \times 1$. The definitions for x_2 follow. In $\mathbf{x}_1^K$, each x_1^j is repeated N_2 times, while in $\mathbf{x}_2^K$ each sequence $x_2^0, x_2^1, ..., x_2^m$ is repeated N_1 times. The total length of each vector is $N_1 \times N_2$. These vectors are defined, so that the set $(\mathbf{x}_1^K, \mathbf{x}_2^K)$ is a full set of all Chebyshev points on the two-dimensional tensor grid in computational space. An equivalent set can be defined for the points on the physical domain. Note that the points are clustered around the boundary of the two-dimensional grid and sparse in the middle of the grid, see Figure (3.2). As in the one-dimensional case we can map the points $(\mathbf{x}_1^K, \mathbf{x}_2^K)$ to an arbitrary rectangular domain, using the linear map (3.62) in each coordinate direction.

Boundaries and Normals

We can define the boundary of the box by defining vectors for each of the four sides of the box as follows. We do this on the computational domain $[-1, 1]^2$, since each of the sides of the computational domain is conformally mapped onto a corresponding side of the rectangle in physical space. It is then straightforward to define a Boolean vector of size $N_1 \times N_2$, which contains ones where $x_1 = x_{1,\min}$ and zeros everywhere else, to define the left boundary of the square, and analogously for the other sides. We combine these four vectors to one boundary vector of length $N_1 \times N_2$, which contains ones at each of the four faces and zeros everywhere else. The normal vectors of the rectangle are found by considering the four corners of the shape. We sort the four corners $\{Y_i\} = \{(y_1^i, y_2^i)\}$, starting from the left bottom corner and in clockwise order. We consider the following set of normals $\vec{n}_l$, $\vec{n}_r$, $\vec{n}_t$, and $\vec{n}_b$, for the left, right, top, and

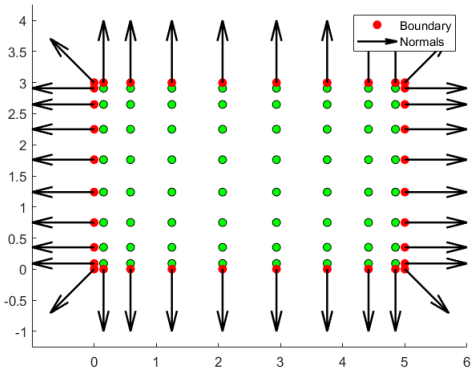

Figure 3.2: Chebyshev points with $N = 10$ on the domain $[0, 5] \times [0, 3]$. The red points symbolize the domain boundary, while the arrows indicate outward normals on the boundary.

bottom normal respectively. For the left normal we first define the vector

$$\vec{m}_l = \left[y_2^2 - y_2^1, y_1^2 - y_1^1\right]^\top := \left[m_{l,1}, m_{l,2}\right]^\top. \tag{3.74}$$

We then define the normal to be

$$\vec{n}_l = \text{sgn}(m_{l,1}) \frac{\left[-m_{l,1}, m_{l,2}\right]^\top}{\sqrt{m_{l,1}^2 + m_{l,2}^2}},$$

where the negative first component and $\text{sgn}(m_{l,1})$ are taken so that we consider the outward normal. We furthermore want to work with the unit normal, which is the reason for the normalization term. The other three normals are therefore defined in an equivalent way

$$\vec{n}_r = \text{sgn}(m_{r,1}) \frac{\left[m_{r,1}, -m_{r,2}\right]^\top}{\sqrt{m_{r,1}^2 + m_{r,2}^2}}, \quad \vec{n}_t = \text{sgn}(m_{t,1}) \frac{\left[-m_{t,1}, m_{t,2}\right]^\top}{\sqrt{m_{t,1}^2 + m_{t,2}^2}},$$
$$\vec{n}_b = \text{sgn}(m_{b,1}) \frac{\left[m_{b,1}, -m_{b,2}\right]^\top}{\sqrt{m_{b,1}^2 + m_{b,2}^2}}.$$

Note that this piece of code is general, and can be used for computing normals of more complicated shapes, such as quadrilaterals, which will be introduced in a later chapter. For the rectangle, as is evident in Figure 3.2, the four outward normals are simply defined by

$$\vec{n}_l = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \vec{n}_r = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \vec{n}_t = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \vec{n}_b = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

for the left, right, top, and bottom normals, respectively. Each of these normal vectors has values on its respective face of the rectangle and zeros everywhere else. We then combine these four vectors using the boundary Boolean vector which we have defined in the above step. This means that the normals on the corners are summed up, since each of the two faces meeting at a corner has a normal defined at the corner. The normal at a corner is not uniquely defined. However, it is convenient to set it to be the average of the two normals from each shape. This is done by normalising the sum of the two normals, as demonstrated in Figure 3.2. We illustrate this for the corner normal between the left and top face of the quadrilateral:

$$\vec{n}_c = \frac{\vec{n}_l + \vec{n}_t}{||\vec{n}_l + \vec{n}_t||_{l_2}}. \tag{3.75}$$

Interpolation

Interpolation in two dimensions is based on the idea of Kronecker tensor products. The one-dimensional interpolation matrices Interp_1 and Interp_2 are computed for each set of points $\{x_1^j\}$ on the x_1-axis and $\{x_2^i\}$ in the x_2 direction, as explained in Section 3.3.1. The Kronecker product of these two matrices is taken, resulting in the two-dimensional interpolation matrix. For an example with $i = j = 0, 1, 2$, this is the block matrix of size 9×9

$$\begin{aligned} \text{Interp} &= \text{Interp}_1 \otimes \text{Interp}_2 \\ &= \begin{bmatrix} \text{Interp}_1(0,0) \times \text{Interp}_2 & \text{Interp}_1(0,1) \times \text{Interp}_2 & \text{Interp}_1(0,2) \times \text{Interp}_2 \\ \text{Interp}_1(1,0) \times \text{Interp}_2 & \text{Interp}_1(1,1) \times \text{Interp}_2 & \text{Interp}_1(1,2) \times \text{Interp}_2 \\ \text{Interp}_1(2,0) \times \text{Interp}_2 & \text{Interp}_1(2,1) \times \text{Interp}_2 & \text{Interp}_1(2,2) \times \text{Interp}_2 \end{bmatrix}. \end{aligned} \tag{3.76}$$

Differentiation

A similar approach, using Kronecker vectors, can be used to find the Chebyshev differentiation matrices for two-dimensional problems as follows, compare to [94]. For a first derivative D in the x_1 direction, a Kronecker product is taken of the one-dimensional Chebyshev differentiation matrix, which is computed as explained in Section 3.3.1, with the identity, as demonstrated here with three points

$$\begin{aligned} D_{x_1} = I \otimes D &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} \\ &= \begin{bmatrix} d_{11} & d_{12} & d_{13} & & & & & & \\ d_{21} & d_{22} & d_{23} & & & & & & \\ d_{31} & d_{32} & d_{33} & & & & & & \\ & & & d_{11} & d_{12} & d_{13} & & & \\ & & & d_{21} & d_{22} & d_{23} & & & \\ & & & d_{31} & d_{32} & d_{33} & & & \\ & & & & & & d_{11} & d_{12} & d_{13} \\ & & & & & & d_{21} & d_{22} & d_{23} \\ & & & & & & d_{31} & d_{32} & d_{33} \end{bmatrix}, \end{aligned}$$

where the block structure matches the repetition of each x_1^j in $\mathbf{x}_1^K$, see (3.72). The second derivative with respect to x_1, $D_{x_1x_1}$ can be found equivalently by computing $D_{x_1x_1} = I \otimes D_2$, where D_2 is the second order differentiation matrix in one dimension, as explained in Section 3.3.1. The derivative with respect to y is found by taking the Kronecker product the other way around

$$\begin{aligned} D_{x_2} = D \otimes I &= \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} d_{11} & & & d_{12} & & & d_{13} & & \\ & d_{11} & & & d_{12} & & & d_{13} & \\ & & d_{11} & & & d_{12} & & & d_{13} \\ d_{21} & & & d_{22} & & & d_{23} & & \\ & d_{21} & & & d_{22} & & & d_{23} & \\ & & d_{21} & & & d_{22} & & & d_{23} \\ d_{31} & & & d_{32} & & & d_{33} & & \\ & d_{31} & & & d_{32} & & & d_{33} & \\ & & d_{31} & & & d_{32} & & & d_{33} \end{bmatrix}, \end{aligned}$$

which now matches the repetition of each $x_2^0, ...x_2^m$ in $\mathbf{x}_2^K$, see (3.72). We have the following operators for the rectangle in computational space

$$\begin{aligned} \text{Grad} &= [D_{x_1}, D_{x_2}]^\top, \\ \text{Div} &= [D_{x_1}, D_{x_2}], \\ \text{Lap} &= D_{x_1x_1} + D_{x_2x_2}. \end{aligned}$$

Since these operators are defined in computational space, the Jacobian (3.65) is applied to $D_{x_1},\ D_{x_2},\ D_{x_1x_1},\ D_{x_2x_2}$ to map into physical space before defining Grad, Div, and Lap.

Integration and Convolution

The two-dimensional integration vector is found by considering the two one-dimensional integration vectors for the points in the directions of the x_1- and x_2-axes. The Kronecker product of these is taken and then multiplied by the determinant of the Jacobian of the mapping to the physical domain (3.65), since the integration vectors are computed on the computational grid. This results in the two-dimensional integration vector $\text{Int} = \text{Int}_1 \otimes \text{Int}_2$, where Int_1, Int_2 denote the one-dimensional integration vectors in physical space.

The construction of the convolution matrix is identical to the one in one dimension, see Section 3.3.1, except that χ takes both $\mathbf{y}_1^K$ and $\mathbf{y}_2^K$ as inputs and the integration vector involved is the two-dimensional integration vector. We have

$$\text{Conv}_{i,j} = \text{Int}_j \ \ \chi(y_1^i - y_1^j, y_2^i - y_2^j). \tag{3.77}$$

If the function χ only takes one input, we consider the two given differences of points

$$d_1^{i,j} = y_1^i - y_1^j, \quad d_2^{i,j} = y_2^i - y_2^j,$$

and compute the Euclidean distance between d_1 and d_2 to obtain the convolution matrix

$$\text{Conv}_{i,j} = \text{Int}_j \ \ \chi(||d_1^{i,j}, d_2^{i,j}||_{l_2}).$$

3.3.4 Periodic Box (2D)

The definition of the periodic box follows more or less exactly from the discussion of the periodic line and the Chebyshev box. We highlight some of the differences in the following.

Domains and Boundaries

We can construct a periodic box which is either periodic in both directions, or periodic in one and nonperiodic in the other coordinate direction. Kronecker vectors are created as discussed for the box, regardless of whether the coordinate is discretized using Fourier or Chebyshev points. Note that there are only boundaries in any directions that do not have periodic boundary conditions.

Interpolation

In order to construct the interpolation matrix for this shape, we need to consider the interpolation matrices for each coordinate and take the Kronecker product of them. For a two-dimensional grid in which the second coordinate is defined by Fourier approximations, in order to define interpolation, the Fast Fourier Transform matrix is considered

$$\text{FFTM} = I \otimes F, \quad \text{where} \quad \text{F}_{jk} = e^{-2\pi ijk/N},$$

where $j, k = 1, ..., N$. Then the interpolation matrix is given by

$$\text{Interp} = \Re\left((\text{Interp}_1 \otimes \text{Interp}_2)\,\text{FFTM}\right),$$

where Interp_1, Interp_2 are the interpolation matrices in the respective coordinate.

Differentiation, Integration and Convolution

The appropriate Kronecker products of the differentiation matrices in physical space in each coordinate are taken to create Grad, Div, and Lap for the periodic box, following the approach in Section 3.3.3. The integration vector for the periodic box is the Kronecker product of the two one-dimensional integration vectors. The convolution matrix is constructed in the same way as in Section 3.3.3. The aspect to pay careful attention to is to compute the periodic distance (3.71) in any periodic direction instead of the Euclidean distance.

3.3.5 Cube (3D)

We can extend the ideas from constructing the box discussed in Section 3.3.3 to construct a three-dimensional cube. This is not part of the original 2DChebClass library, but an extension of it. We construct three vectors of Chebyshev points $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ of sizes $N_1 \times 1$, $N_2 \times 1$, and $N_3 \times 1$, respectively. Following the construction of a grid in 2D given by (3.73), we compute

$$\mathbf{x}_1^K = (\mathbf{x}_1 \otimes \mathbf{e}_{N_2}) \otimes \mathbf{e}_{N_3}, \quad \mathbf{x}_2^K = (\mathbf{e}_{N_1} \otimes \mathbf{x}_2) \otimes \mathbf{e}_{N_3}, \quad \mathbf{x}_3^K = (\mathbf{e}_{N_1} \otimes \mathbf{e}_{N_2}) \otimes \mathbf{x}_3.$$

Using the same pattern, one can construct the 3D interpolation matrix from the 1D case as

$$\text{Interp} = (\text{Interp}_1 \otimes \text{Interp}_2) \otimes \text{Interp}_3.$$

Furthermore, constructing the differentiation matrices in each direction D_{x_1}, D_{x_2} and D_{x_3}, the same Kronecker products are taken as in constructing the 3D grid. The vectors $\mathbf{x}_d$, $d = 1, 2, 3$ are replaced with the 1D differentiation matrix D of dimension N_1, N_2, and N_3, respectively. Similarly, the vectors $\mathbf{e}_{N_d}$ are replaced by identity matrices of size $N_d \times N_d$, for $d = 1, 2, 3$. Then the differentiation operators are defined as

$$\begin{aligned} \text{Grad} &= [D_{x_1}, D_{x_2}, D_{x_3}]^\top, \\ \text{Div} &= [D_{x_1}, D_{x_2}, D_{x_3}], \\ \text{Lap} &= D_{x_1x_1} + D_{x_2x_2} + D_{x_3x_3}, \end{aligned}$$

which can be compared to the 2D box. Again, the integration vector in 3D is constructed by taking Kronecker products of the 1D version in each coordinate. Finally, convolution cannot be constructed using Kronecker products, since it is a global operation. However, it is essentially equivalent to the approaches for the line in Section 3.3.1 and box in Section 3.3.3. Now χ has inputs $\mathbf{y}_1^K$, $\mathbf{y}_2^K$, and $\mathbf{y}_3^K$ and the integration vector involved is the three-dimensional integration vector. We have, for $y_d^i \in \mathbf{y}_d^K$, $d = 1, 2, 3$

$$\text{Conv}_{i,j} = \text{Int}_j \quad \chi(y_1^i - y_1^j, y_2^i - y_2^j, y_3^i - y_3^j).$$

And, as before, if χ only takes one input, we consider differences of points

$$d_1^{i,j} = y_1^i - y_1^j, \quad d_2^{i,j} = y_2^i - y_2^j, \quad d_3^{i,j} = y_3^i - y_3^j,$$

in order to compute the Euclidean distance between d_1, d_2, and d_3. Then the convolution matrix is

$$\text{Conv}_{i,j} = \text{Int}_j \quad \chi(||d_1^{i,j}, d_2^{i,j}, d_3^{i,j}||_{l_2}).$$

3.3.6 Solving the Integro-PDE

Regardless of the chosen domain, a PDE model can be solved as follows. The initial condition ρ_0 of the PDE is discretized using Chebyshev points, or in the case of the periodic box, equispaced points. In a similar spirit to the approach in Section 3.2.5, the PDE is discretized using the differentiation and convolution matrices constructed in this section, which results in a system

of ODEs

$$M\boldsymbol{\rho}' = \mathbf{f}(\boldsymbol{\rho}). \tag{3.78}$$

Here, $\mathbf{f}(\boldsymbol{\rho})$ symbolizes the discretized right-hand side of the PDE, including initial and boundary conditions, The matrix M is a mass matrix, which is diagonal with ones on the diagonal for an interior point and zero for a boundary point. One possible method for solving differential–algebraic equations (DAEs), such as (3.78), is the MATLAB solver `ode15s` [120, 121]. It is a variable-order, variable-step-size DAE solver, which is based on numerical differentiation formulas (NDFs) for solving ODEs of the form $\boldsymbol{\rho}' = \mathbf{f}(t, \boldsymbol{\rho})$. Time is discretized, with time step $h = t_{j+1} - t_j$, and we denote $\boldsymbol{\rho}(t_j) := \boldsymbol{\rho}_j$. The NDF of order k, for $k \in \{1, ..., 5\}$, is given by

$$\sum_{m=1}^{k} \left[\frac{1}{m} \nabla^m \boldsymbol{\rho}_{j+1} - h\mathbf{f}(t_{j+1}, \boldsymbol{\rho}_{j+1}) - \frac{c}{m} \left(\boldsymbol{\rho}_{j+1} - \boldsymbol{\rho}_{j+1}^{(0)} \right) \right] = \mathbf{0},$$

where $\boldsymbol{\rho}_{j+1}^{(0)} = \sum_{m=0}^{k} \nabla^m \boldsymbol{\rho}_j$, and ∇^m is the backward differentiation operator defined recursively by

$$\nabla^m \boldsymbol{\rho}_{j+1} = \nabla \left(\nabla^{m-1} \boldsymbol{\rho}_{j+1} \right), \text{ with } \nabla \boldsymbol{\rho}_{j+1} := \boldsymbol{\rho}_{j+1} - \boldsymbol{\rho}_j.$$

The parameter c, depending on k, is chosen to maximize the stability region, and, where possible, minimize the truncation error of the numerical method. In `ode15s`, $\boldsymbol{\rho}_{j+1}$ is found using a modified Newton iteration. In order to do so, an approximation to the Jacobian of $\mathbf{f}(t, \boldsymbol{\rho})$ is formed, which is updated whenever convergence cannot be achieved within four iterations. Moreover, in such cases the step size of the solver can be changed, which is done efficiently using interpolation techniques. This method is adapted to DAEs of the form (3.78) by modifying the Newton step so that it involves the mass matrix M, and by adding a procedure for finding consistent initial conditions for the solver. Listings 3.1–3.3 demonstrate how to set up a Chebyshev box, discretize the PDE and solve it using the MATLAB solver.

Listing 3.1: Setting up a Chebyshev box and extracting operators

```
% Settting up a box
geom.x1Min = 0; geom.x1Max = 3; geom.x2Min = 0; geom.x2Max = 3;
geom.N = [20;30];
aBox = Box(geom);

% Extract pre-computed operators from the box respectively: this improves
% performance of the ODE solve, since accessing structures in MATLAB
% is comparatively slow
grad = aBox.Diff.grad; div = aBox.Diff.div; Lap = aBox.Diff.Lap;
Int = aBox.Int; bound = aBox.Ind.bound; normal = aBox.Ind.normal;
x1 = aBox.Pts.y1_kv; x2 = aBox.Pts.y2_kv;

% Compute convolution matrix, for a kernel defined by a function V2Fun
Conv = aBox.ComputeConvolutionMatrix(@V2Fun,true);
```

Listing 3.2: Discretizing the right-hand side of the PDE system

```
% Right-hand side function, including boundary & intersection conditions
function drhodt = drho_dt(t,rho)

% Define discretized PDE using known external potential vector
% (Vext), differentiation matrices (grad, div), and convolution
% matrices between species a and b (Convaa, Convab, Convbb, Convba)
rhoflux = Flux(rho,Conv);
drhodt  = -div*rhoflux;

% Apply no flux boundary conditions at boundary
drhodt(bound) = normal*rhoflux;

```

```
13      end
14      function rhoFlux = Flux(rho,Conv)
15      % Stack rho - this is now a size 2M x 1 vector
16      rho2 = [rho;rhoa];
17
18      rhoFlux = -(grad*rho + rho2.*(grad*Vext) + rho2.*(grad*(Conv*rho)));
19      end
```

Listing 3.3: Solving a PDE system on a box using MATLAB's DAE solver

```
1       % Solving a system of PDEs using MATLAB's inbuilt DAE solver
2       % Set the mass matrix
3       mM = ones(size(x1));    % vector of ones, of size M x 1
4       % M = length of discretized spatial domain
5       mM(bound) = 0;          % set to zero at boundaries
6
7       % Set options for ODE solver: relative & absolute tolerance, mass matrix
8       opts = odeset('RelTol',10^-9,'AbsTol',10^-9,'Mass',diag(mM));
9       % Solve ODE, given the following:
10      % - vector of time points comp_times
11      % - initial condition rho_ic
12      % - right-hand side function 'drho_dt', computed in Listing 3
13      [outTimes, rho_t] = ode15s(@drho_dt,comp_times,rho_ic,opts);
```

3.4 Numerical Experiments

In this section, we demonstrates how the pseudospectral method performs in solving one-, two-, and three-dimensional problems with different boundary conditions. First, we compare the results to exact solutions, and test them for convergence when refining the grid size. Then some models of interacting particles are solved for different boundary conditions. Unless stated otherwise, the spatial domain is $[-1,1]^d$, $d = 1,2,3$, the time horizon of the simulation is $(0,T)$, and the problems are solved using MATLAB's solver `ode15s` [120, 121]. Note that other choices of domain do not affect the results adversely.

Measures of Accuracy

All errors presented in this section are calculated as a measure of the difference between a variable of interest, y, and a reference value y_R, which is either the analytic solution to a test problem or a solution computed with a large number of discretization points. The error measure $\mathcal{E}$ is composed of an L^2 error in space and an L^∞ error in time. We define absolute and relative L^2 spatial errors

$$\mathcal{E}_{Abs}(t) = \left\| y(\vec{x},t) - y_R(\vec{x},t) \right\|_{L^2(\Omega)}, \qquad \mathcal{E}_{Rel}(t) = \frac{\left\| y(\vec{x},t) - y_R(\vec{x},t) \right\|_{L^2(\Omega)}}{\left\| y_R(\vec{x},t) \right\|_{L^2(\Omega)} + 10^{-10}},$$

where the small additional term on the denominator prevents division by zero. These are used in the full error measure:

$$\mathcal{E} = \max_{t\in[0,T]} \left[\min\left(\mathcal{E}_{Rel}(t), \mathcal{E}_{Abs}(t) \right) \right]. \tag{3.79}$$

The minimum between absolute and relative spatial error is taken to avoid choosing an erroneously large relative error, caused by division of one numerically very small term by another.

3.4.1 Validation Tests

Since the 2DChebClass implementation was first published in [84], it has been validated extensively, so that this is omitted in this thesis. Further omissions in this section include tests of the discretized integration, differentiation, and convolution operators for the extensions made

since the original publication, in the interest of brevity. In this section, the numerical method is tested by constructing exact solutions to an advection–diffusion PDE of the form

$$\frac{\partial \rho}{\partial t} = \nabla^2 \rho - \nabla \cdot (\rho \vec{w}) + f \quad \text{in } (0,T) \times \Omega, \tag{3.80}$$

with Dirichlet, no-flux, and periodic boundary conditions, in one, two and three dimensions. In order to construct an exact solution for this PDE, we choose ρ_{ex} and $\vec{w}$ that satisfy initial and boundary conditions. Then we define the source term so that it satisfies the PDE:

$$f = \frac{\partial \rho_{\text{ex}}}{\partial t} - \nabla^2 \rho_{\text{ex}} + \nabla \cdot (\rho_{\text{ex}} \vec{w}) \,. \tag{3.81}$$

The error between the numerical and exact solutions is measured using (3.79). The time horizon is $t \in (0,1)$ and the ODE tolerances are set to 10^{-9} unless stated otherwise and $\rho(0,x) = \rho_{\text{ex}}(0,x)$.

One-Dimensional Problems

First exact solutions for problems with Dirichlet, no-flux conditions, and periodic boundary conditions are constructed. The first problem is the advection–diffusion equation (3.80) with a source term and with non-zero Dirichlet conditions

$$\rho(t,x) = 1 \qquad \text{on } (0,T) \times \partial\Omega. \tag{3.82}$$

We choose

$$\begin{aligned} \rho_{\text{ex}}(t,x) &= \frac{1}{2} \cos\left(\frac{\pi k x}{2}\right)(t^2 - 1) + 1, \\ \vec{w}(t,x) &= (1-t)x, \end{aligned}$$

with $k = 1,3,5,7$. Given these inputs, f can be constructed using (3.81). The error between the exact and numerical solution, measured by (3.79), is shown for a different number of discretization points in Figure 3.3. It is evident that with increasing frequency k, the convergence becomes slower, since the solution oscillates more, and therefore steeper gradients have to be resolved numerically. Next, problem (3.80) is considered with no-flux conditions

$$\frac{\partial \rho}{\partial n} - \rho \vec{w} \cdot \vec{n} = 0 \quad \text{on } (0,T) \times \partial\Omega.$$

The exact solution is constructed in the same manner as above, choosing

$$\begin{aligned} \rho_{\text{ex}}(t,x) &= \frac{1}{2} \cos(\pi k x)(t^2 - 1), \\ \vec{w}(t,x) &= e^{-t}(x^2 - 1), \end{aligned}$$

with $k = 1,2,3,4$. The convergence in N can be seen in Figure 3.3 and is again slower with the increase in k. Finally, the exact solution for the problem with periodic boundary conditions

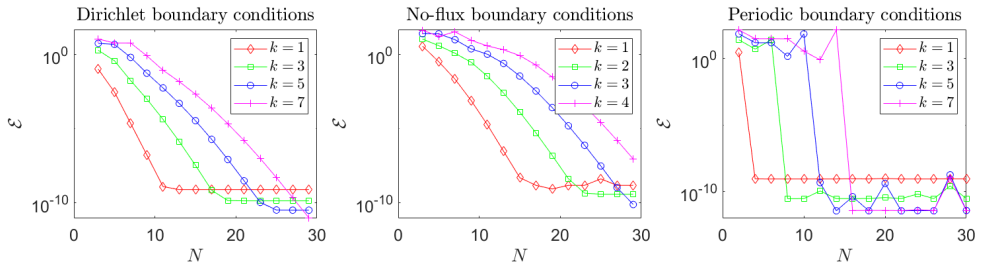

Figure 3.3: The error $\mathcal{E}$, given by (3.79), in the exact solution to (3.80), with Dirichlet (left), no-flux (middle), and periodic (right) boundary conditions, for an increasing number of discretization points N, and for different frequencies k.

$$\rho(t,-1) = \rho(t,1) \quad \text{on } (0,T)\times\partial\Omega$$

is constructed. The choices for ρ and $\vec{w}$ are

$$\rho_{\text{ex}}(t,x) = \frac{1}{2}\cos(\pi k x)\,(t^2-1),$$
$$\vec{w}(t,x) = \cos(\pi x),$$

with $k = 1,3,5,7$. Again, convergence in N is shown in Figure 3.3 and more points are needed to reach exponential accuracy with increasing k.

Two-Dimensional Problems

As in the one-dimensional case, the first problem that is considered is an advection–diffusion equation (3.80) with a source term and the result is compared to an exact solution. The exact solutions are constructed in the same way as in the one-dimensional case, and we choose again $t \in (0,1)$. The following inputs for a Dirichlet problem are given:

$$\rho_{\text{ex}}(t,x) = \frac{1}{4}e^t \cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{\pi x_2}{2}\right),$$
$$\vec{w}(t,x) = \left[x_1^2 - x_2, x_2^2 - x_1\right]^\top.$$

The convergence in N can be seen in Figure 3.4. For the no-flux problem the inputs are

$$\rho_{\text{ex}}(t,x) = \frac{1}{8}e^t \cos(\pi x_1)\cos(\pi x_2),$$
$$\vec{w}(t,x) = 0.1\left[(x_1^2-1)(x_2^2-1), (x_1^2-1)(x_2^2-1)\right]^\top.$$

The convergence results for different N are again displayed in Figure 3.4. Finally, as in the one-dimensional case, we test a problem with periodic boundary conditions against an exact solution. Here, we choose no-flux boundary conditions for the x_1 coordinate, while the x_2 coordinate has periodic boundary conditions. Therefore, all input choices have to be periodic in x_2. These are

$$\rho_{\text{ex}}(t,x) = \frac{1}{8}e^t \cos(\pi x_1)\cos(\pi x_2),$$
$$\vec{w}(t,x) = 0.1\left[0, \cos(\pi x_2)\right]^\top.$$

The convergence in N is shown in Figure 3.4. It is evident that for choices of N greater than 15, the error plateaus. This signals that, given the problem setup, configuration of solver, and discretization method, a maximum in precision is reached.

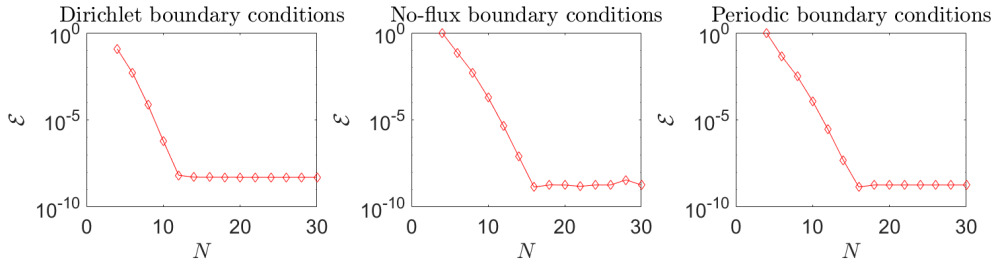

Figure 3.4: The error $\mathcal{E}$, given by (3.79), in the exact solution to (3.80) in two dimensions with zero Dirichlet (left), no-flux (middle), and periodic (right) boundary conditions for an increasing number of discretization points $N = N_1 = N_2$.

Three-Dimensional Problems

Equivalently to the one- and two-dimensional case, we can create a test problem in three dimensions of the form (3.80). For the exact problem with zero Dirichlet boundary conditions we choose

$$\rho_{\text{ex}}(t,x) = \frac{1}{8}e^t \cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{\pi x_2}{2}\right)\cos\left(\frac{\pi x_3}{2}\right),$$
$$\vec{w}(t,x) = 0.1\left[(x_1^2-1)(x_2^2-1)(x_3^2-1),(x_1^2-1)(x_2^2-1)(x_3^2-1),(x_1^2-1)(x_2^2-1)(x_3^2-1)\right]^\top.$$

For the no-flux problem the choices are

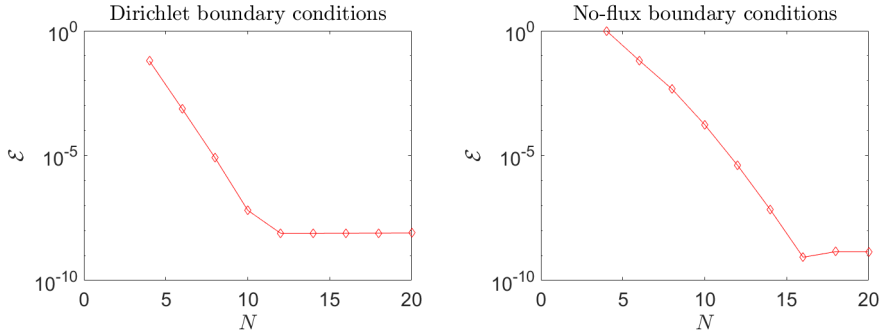

Figure 3.5: The error $\mathcal{E}$, given by (3.79), in the exact solution to (3.80) in two dimensions with zero Dirichlet conditions (left) and with no-flux conditions (right) for an increasing number of discretization points $N = N_1 = N_2 = N_3$.

$$\rho_{\text{ex}}(t,x) = \frac{1}{8}e^t \cos(\pi x_1)\cos(\pi x_2)\cos(\pi x_3),$$
$$\vec{w}(t,x) = 0.1\left[(x_1^2-1)(x_2^2-1)(x_3^2-1),(x_1^2-1)(x_2^2-1)(x_3^2-1),(x_1^2-1)(x_2^2-1)(x_3^2-1)\right]^\top.$$

The convergence in N for both problems can be seen in Figure 3.5. Given the increased computational complexity in three dimensions, we only consider convergence up to $N = N_1 = N_2 = N_3 = 20$, instead of $N = 30$, as done in the one- and two-dimensional cases. Moreover, periodic conditions in 3D are omitted. The time horizon is again $(0,1)$.

3.4.2 Solving DDFT Models in One, Two, and Three Dimensions

In this section, problems of the form (3.59)

$$\frac{\partial\rho}{\partial t} = \nabla^2\rho + \nabla\cdot(\rho\nabla V_1) - \nabla\cdot(\rho\vec{w}) + \nabla\cdot\int_\Omega \rho(t,\vec{x})\rho(t,\vec{x}')\nabla V_2(|\vec{x}-\vec{x}'|)d\vec{x}' \quad \text{in } (0,T)\times\Omega,$$

with the three different boundary conditions are solved. This is (1.3) introduced in Chapter 1, for which no exact solutions exist. Note that $k_BT = 1$, $\Gamma = 1$ from now on. For the following interacting problems, an interaction potential is specified as

$$V_2(\vec{x}) = \kappa e^{-\|\vec{x}\|^2}. \tag{3.83}$$

This is given for the Dirichlet and no-flux boundary conditions. For the periodic problem another interaction potential is considered, since it is required to be periodic. In the following examples, it is of interest how the interaction strength impacts the particle dynamics. Here, three values are considered: $\kappa = 0$ (no interaction), $\kappa = -a$ (attraction), and $\kappa = a$ (repulsion), where $a > 0$. The parameter a is chosen to be $a = 1$ or $a = 0.3$, depending on the example. If $\kappa = 0$, the PDE constraint reduces to an advection–diffusion equation. This serves as a benchmark, since the advection–diffusion problem has been validated in the previous section.

The interaction strengths $|\kappa| = a$ are chosen as a balance between demonstrable differences in the solution and efficient computation. For smaller $|\kappa|$, diffusion and V_1 dominate and interaction effects are negligible, while for larger $|\kappa|$, steep gradients in the resulting particle density make the numerical solution difficult. The given domain is $[-1, 1]^d$, $d = 1, 2, 3$, and the time horizon is $t \in (0, 1)$. The chosen ODE tolerances in this section are 10^{-9} for one- and two-dimensional problems, and 10^{-7} for three-dimensional problems. The solution for one- and two-dimensional problems takes of the order of a second, while the three-dimensional examples take around 10 minutes to solve.

One-Dimensional Examples

Now that the numerical method is verified against an exact solution, we test the interacting problem (3.59). The initial condition is $\rho_0(x) = 1$ and an external potential $V_1 = (2t-1)((x+0.4)^2 - 1)((x-0.4)^2 - 1)$ is defined. The problem is solved with both the Dirichlet condition (3.82) and a no-flux condition given by

$$\frac{\partial \rho}{\partial n} + \rho \frac{\partial V_1}{\partial n} - \rho \vec{w} \cdot \vec{n} + \int_\Omega \rho(t, \vec{x}) \rho(t, \vec{x}') \frac{\partial V_2(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' = 0 \qquad \text{on } (0, T) \times \partial\Omega.$$

The accuracy of the numerical solution with varying N is tested against a reference solution with $N_{\text{Ref}} = 200$. The result for three different interaction configurations is shown in Figure

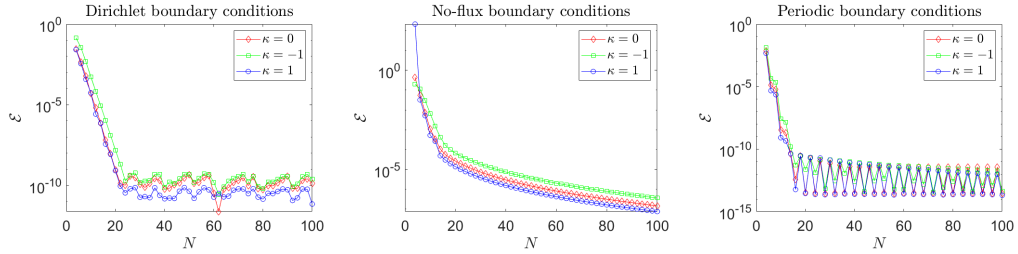

Figure 3.6: Convergence of interacting problems with Dirichlet (left), no-flux (middle), and periodic (right) boundary conditions for an increasing number of discretization points N. The error $\mathcal{E}$, given by (3.79), in the solution with N points is computed with respect to a reference solution with $N_{\text{Ref}} = 200$.

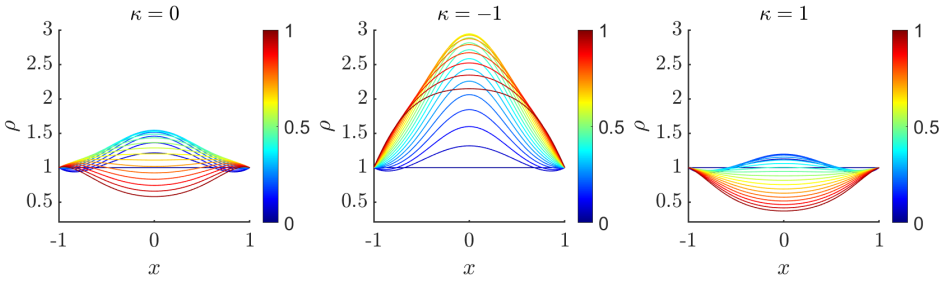

Figure 3.7: Solution to an interacting model with Dirichlet conditions. The colorbars denote the time evolution from $t = 0$ to $t = 1$.

3.6. The solved PDE model with zero Dirichlet boundary conditions is shown in Figure 3.7, with $N = 30$. It is evident that different interaction strengths affect the evolution of the particle density differently. If the particles attract each other ($\kappa = -1$), they build one cluster in the middle of the domain, while they will tend to spread apart in the repulsive case ($\kappa = 1$). If $\kappa = 0$, the dynamics are dominated by the external potential, which changes over time from enforcing clustering in the middle of the domain, to supporting particles to spread out at the end of the time horizon. The same PDE model with no-flux boundary conditions is shown in Figure 3.8, with $N = 30$. In comparison to the problem with Dirichlet boundary conditions in

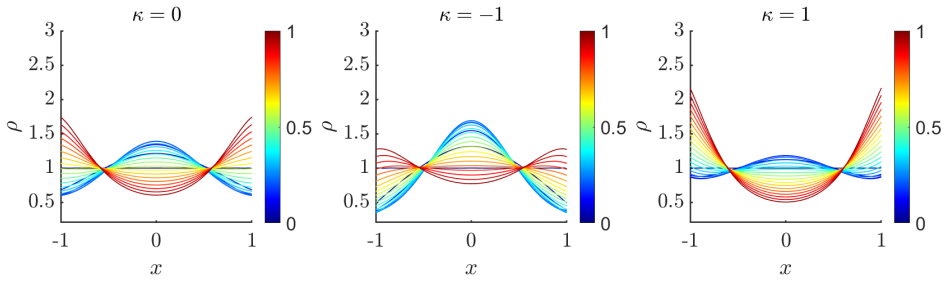

Figure 3.8: Solution to an interacting model with no-flux boundary conditions. The colorbars denote the time evolution from $t = 0$ to $t = 1$.

Figure 3.7, the no-flux case is qualitatively very different. However, the general observations remain the same. In case of $\kappa = 0$, the particles are mainly influenced by V_1, are first forced together and over time forced apart. While attractive particles support the earlier action of V_1, and hence a steeper cluster of particles forms, it counteracts the later action of the external potential. The exact opposite is true for $\kappa = 1$: while the cluster of particles in the centre of the domain is smaller than in the other two cases, the particles accumulate in a much steeper fashion at the boundaries of the domain at later times. Lastly, an interacting problem with

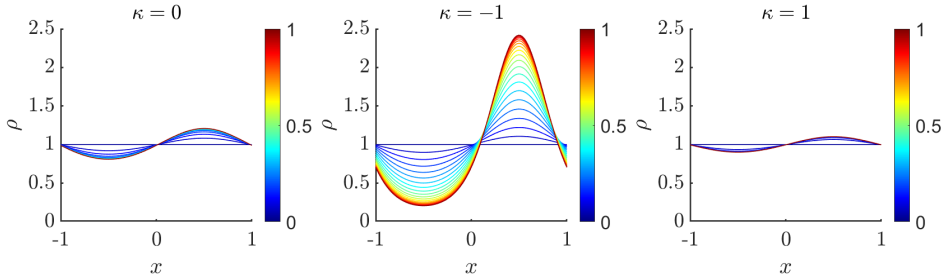

Figure 3.9: Solution to an interacting model with periodic boundary conditions. The colorbars denote the time evolution from $t = 0$ to $t = 1$.

periodic boundary conditions is solved. Here the inputs are chosen to be periodic, so that

$$\rho_0(x) = 1, \quad V_2 = \kappa \cos(\pi x), \quad V_1 = -0.2 \sin(\pi x).$$

Again, the convergence in N is tested using a reference solution with $N = 200$ and shown in Figure 3.6. The solved model is displayed in Figure 3.9.

Two-Dimensional Examples

Analogously to the 1D case, the considered problem is an interacting particle model of the form (3.59). The initial condition is $\rho(0, x) = 1$ and the external potential is chosen as $V_1 = \exp\left((c(x_1 + d)^2 + c(x_2 + d)^2)t + (c(x_1 - d)^2 + c(x_2 - d)^2)(1 - t)\right)$, with $c = -0.5$, $d = 1$. Computing this problem with Dirichlet boundary conditions, the resulting model with different interaction strengths can be seen in Figure 3.10. It is evident that the three different interaction strengths have a strong effect on the resulting particle dynamics. For $\kappa = -0.3$, the particles clump together, while for $\kappa = 0, 3$ they aim to spread out. The same problem is computed for no-flux conditions and the results can be seen in Figure 3.11. Here, the interplay between the external potential and the particle interactions can be observed. In the case when no interactions are present ($\kappa = 0$), the particles first accumulate on the top right corner, where V_1 is small. When V_1 becomes steep in this area at later times, and weaker in the bottom left corner, the particles start accumulating there instead. In the case $\kappa = -0.3$, the same overall trend is visible, but instead of accumulating at the boundaries in the two respective corners, they cluster in the middle of that part of the domain (top-right or bottom-left, respectively).

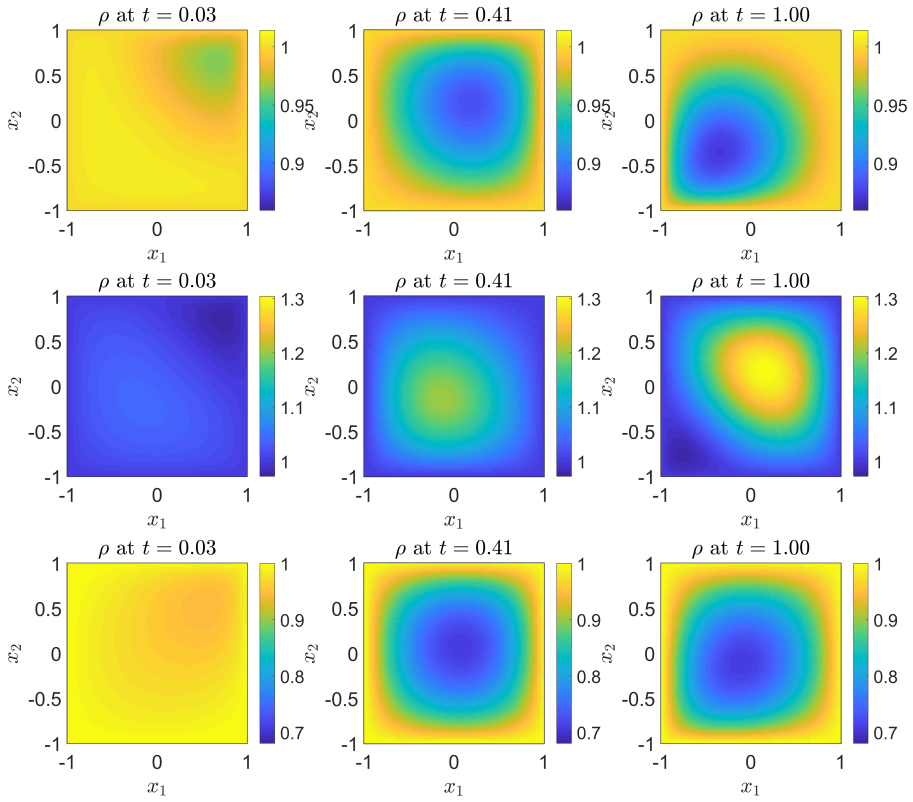

Figure 3.10: Solution to an interacting model with Dirichlet boundary conditions, for $\kappa = 0$ (top), $\kappa = -0.3$ (middle), $\kappa = 0.3$ (bottom).

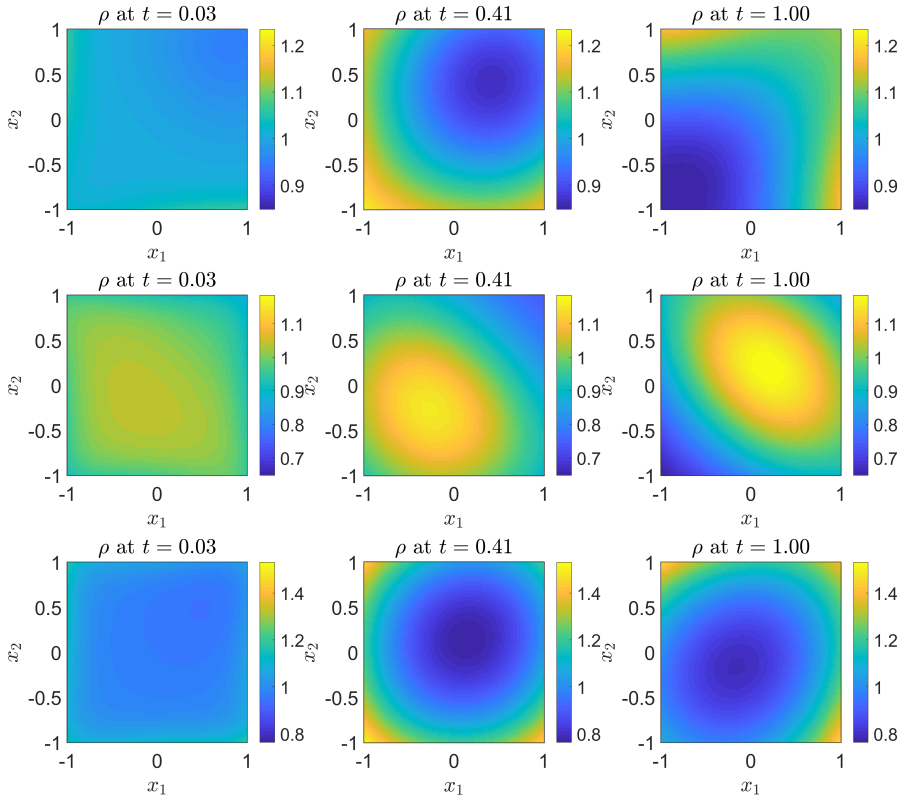

Figure 3.11: Solution to an interacting model with no-flux conditions, for $\kappa = 0$ (top), $\kappa = -0.3$ (middle), $\kappa = 0.3$ (bottom).

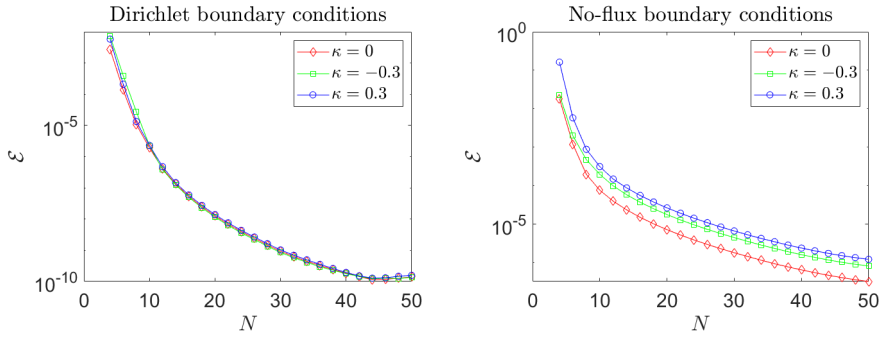

Figure 3.12: Convergence of interacting problems with Dirichlet (left) and with no-flux (right) boundary conditions for an increasing number of discretization points $N = N_1 = N_2$. The error $\mathcal{E}$, given by (3.79), in the solution with N points is computed with respect to a reference solution with $N_{\text{Ref}} = 70$.

For the repulsive particles ($\kappa = 0.3$), the trend to cluster at the boundaries is exacerbated in comparison to the $\kappa = 0$ case. Both the Dirichlet and no-flux problems are computed with $N = N_1 = N_2 = 30$ spatial points. The convergence of the numerical solution with each boundary condition to a reference solution with $N_{\text{Ref}} = N_1 = N_2 = 70$ is shown in Figure 3.12.

As in the one-dimensional case, we would also like to investigate a problem with periodic boundary conditions. Since the periodicity is only applied in the direction of x_2, other boundary conditions for the x_1 boundaries can be imposed. Here, Dirichlet conditions

$$\rho(t, \vec{x}) = 1 \quad \text{on } (0, T) \times \partial\Omega$$

are chosen. In order to satisfy the periodicity condition in x_2, we choose variables

$$\rho_0(\vec{x}) = 1, \quad V_2(\|x_1\|, \|x_2\|) = \kappa \cos(\pi \|x_2\|), \quad V_1 = 0.2 \cos(\pi x_2).$$

A problem with these variable choices can also be computed with no-flux and Dirichlet conditions.

In Figure 3.13 the results for the three choices of boundary conditions are compared for $\kappa = -0.3$. The external potential is steep around $x_2 = 0$, so that particles accumulate around the lines $x_2 = -1$ and $x_2 = 1$. The particle interaction $\kappa = -0.3$ causes steeper accumulation at the two boundaries than for other choices of κ. For the Dirichlet problem, the particles arrange in an additional layer around the lines $x_1 = -1$ and $x_1 = 1$, so that the boundary conditions are satisfied. This is not present in the no-flux problem, in which the particles accumulate according to the given external potential, and hence the particle gradient is steeper than in the Dirichlet problem. The periodic problem displays a cluster of particles at the two boundaries $x_2 = -1$ and $x_2 = 1$, as in both the Dirichlet and no-flux cases. Since Dirichlet conditions are applied in the x_1 direction, there is again a layer of particles at the $x_1 = -1$ and $x_1 = 1$ lines satisfying these conditions. However, in contrast to the Dirichlet case, the periodic boundary conditions in x_2 have to be satisfied. Therefore, a rounder cluster of particles forms on the periodic boundary, the steepness of which lies between Dirichlet and no-flux cases.

A key observation in this example is that the choice of boundary condition severely impacts the solution of the model problem. Especially in real-world applications this is highly relevant when the model informs the design of a process and predictions of the outcome. Convergence of this periodic example can be seen in Figure 3.14. It is clear that convergence for the case $\kappa = -0.3$ is slower than for the other two choices of κ. This is most likely due to the problem being a harder one, with steeper gradients forming, which require more points to be resolved numerically.

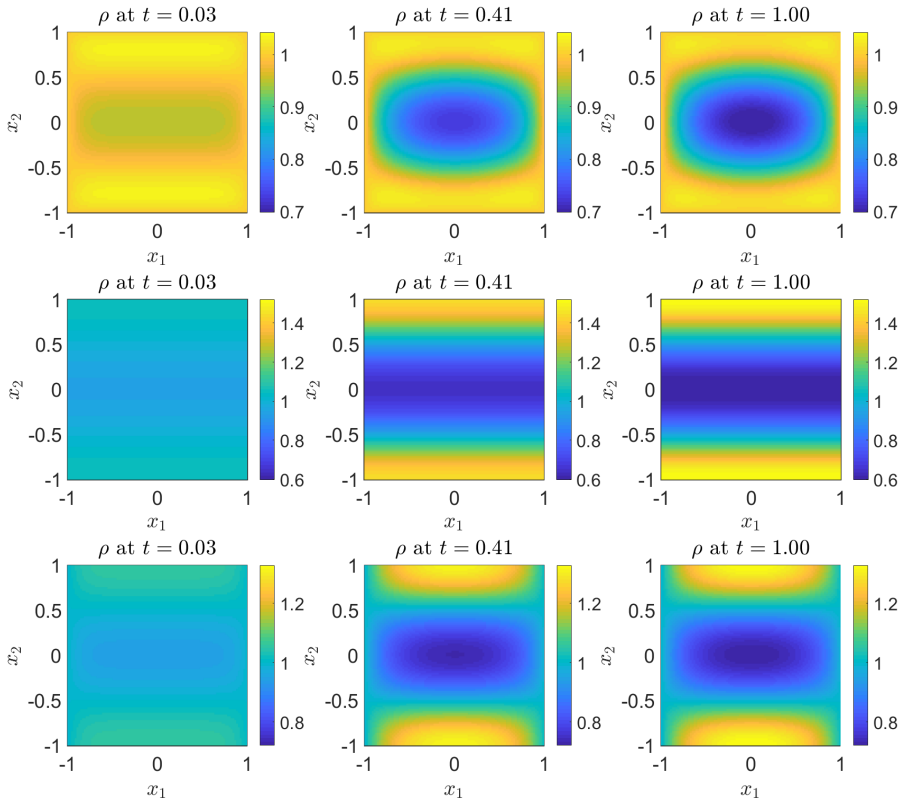

Figure 3.13: Solution to an interacting model with Dirichlet (top), no-flux (middle), and periodic (bottom) boundary conditions, for $\kappa = -0.3$.

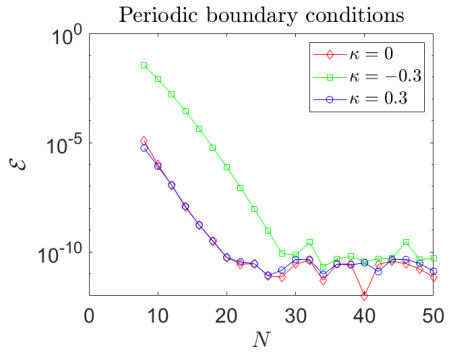

Figure 3.14: Convergence of an interacting problem with periodic boundary conditions in two dimensions and for different interaction strengths. The error $\mathcal{E}$, given by (3.79), in the solution with N points is computed with respect to a reference solution with $N_{\text{Ref}} = 70$.

Three-Dimensional Example

As in the lower-dimensional cases, an interacting problem is considered in three dimensions. Given the increased computational complexity in three dimensions, only one problem is computed and the convergence analysis is omitted. This is an example of the 'curse of dimensionality', demonstrated by the following calculation: If a one-dimensional problem is size M, then the two-dimensional equivalent is M^2, and so the equivalent three-dimensional problem is size M^3. Choosing $M = 20$ points, the two-dimensional problem has $M^2 = 400$, while the three-dimensional problem has $M^3 = 8000$ points, which demonstrates the significantly increased complexity. In this example, the number of discretization points in each direction is $N = N_1 = N_2 = N_3 = 20$. Here, we choose an example of the form (3.59) with no-flux

boundary conditions, an initial condition $\rho(0, \vec{x}) = \frac{1}{8}$, external potential

$$V_1 = 0.3 \sin(\pi x_1) \cos\left(\frac{\pi x_2}{2}\right) \cos\left(\frac{\pi x_3}{2}\right),$$

and an advective field

$$\vec{w} = 0.3[-1, 0, 0]^\top.$$

The action of V_1 causes the particles to prefer the $x_1 < 0$ half of the domain, and in particular the middle of this part, while making it harder to access the half $x_1 > 0$, again specifically in the middle. However, the advective field pushes the particles towards the $x_1 > 0$ half that contains a steep potential. This creates interesting particle dynamics, which can be observed in Figure 3.15. In each of the snapshots, these two effects play out: while particles do prefer the plane $x_1 < 0$, the advection pushes the particles into the other half. Since the middle of the half-volume $x_1 > 0$ is especially hard to access due to the large potential V_1, particles cluster at the far end, close to the $x_1 = 1$ plane instead. Contrastingly, the plane $x_1 = -1$ contains a low particle density, since they accumulate in the potential well in the middle of the half volume $x_1 < 0$. Moreover, particles are advected away from $x_1 = -1$ by $\vec{w}$. Figure 3.15 furthermore displays the effects of different interaction strengths. It can be observed that for attractive particles the effect of the potential well is stronger, and the effect of the advective field is weaker, while for the repulsive particles the effect is the opposite. This is due to the fact that the attractive forces between particles support the clustering in the potential well, while counteracting a broad spread of particles that is prescribed by the advective field $\vec{w}$. The effect of repulsion is exactly the opposite, with more particles spreading out along the plane $x_1 = 1$, and in particular in the corners of this plane, to be as far apart as possible.

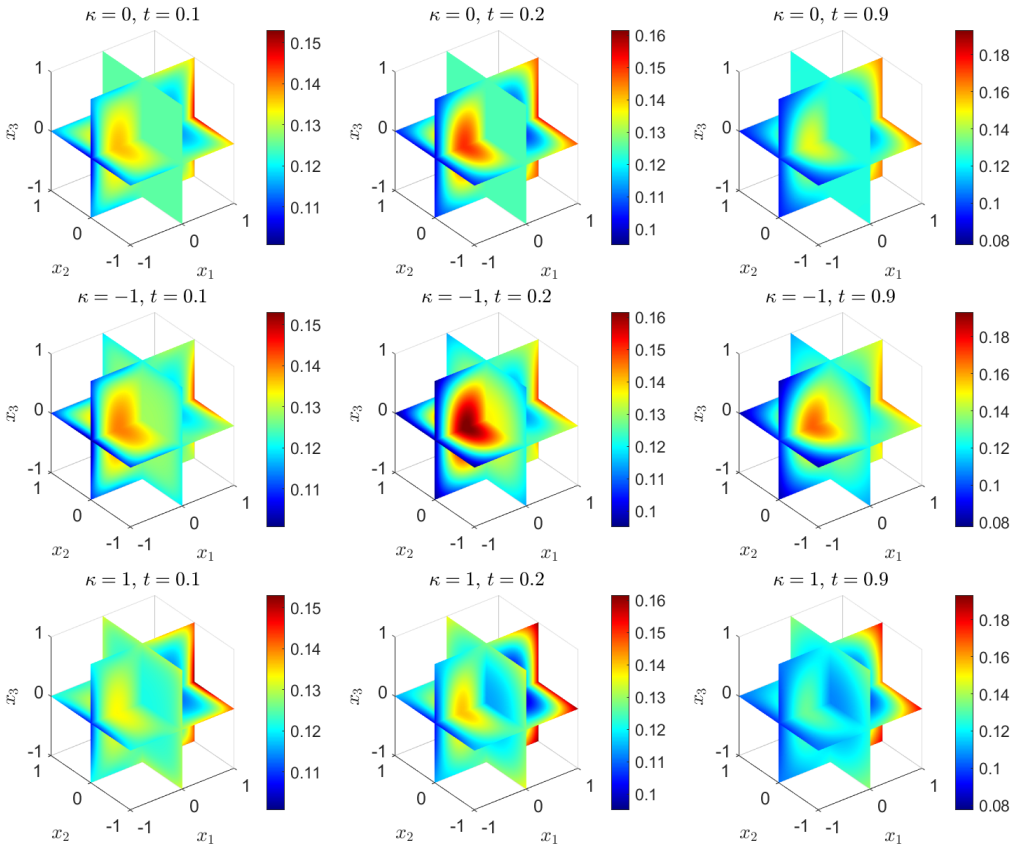

Figure 3.15: Solution to an interacting model with no-flux conditions, for $\kappa = 0$ (top), $\kappa = -1$ (middle), $\kappa = 1$ (bottom).

Chapter 4

PDE-Constrained Optimization

PDE-constrained optimization is a large and active field of research that is concerned with the theoretical properties and numerical solutions of optimization problems that are constrained by PDE models. This includes problems with various types of PDEs as constraints, different optimization targets or outcomes, as well as approaches to meeting these targets by controlling certain parameters of the problem. A survey of PDE-constrained optimization, discussing both theoretical results and numerical methods used to solve such problems is given in this chapter. This provides the necessary broader context for the, mostly numerical, work done in this thesis. The chapter is structured as follows: in Section 4.1 theoretical results for PDE-constrained optimization are presented. In Section 4.2, we present a survey of numerical approaches that are often used to tackle such optimization problems, before discussing Krylov subspace methods in more detail in Section 4.3, in preparation for Chapter 5.

4.1 Theoretical Results for PDE-Constrained Optimization

In this section we will survey some of the classical results in the theory of PDE-constrained optimization and make some comments on further, more theoretically challenging problems. An example of a classical PDE-constrained optimization problem, chosen in view to being related to the work in this thesis, is the following. Here, the PDE constraint is an advection–diffusion problem with an additional source term:

$$\min_{y,u} \mathcal{J}(y,u) := \frac{1}{2}\|y-\widehat{y}\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2((0,T)\times\Omega)} \tag{4.1}$$

subject to

$$\begin{aligned} \frac{\partial y}{\partial t} &= \nabla^2 y - \nabla\cdot(y\vec{w}) + u && \text{in } (0,T)\times\Omega, \\ y(t,\vec{x}) &= 0 && \text{on } (0,T)\times\partial\Omega, \\ y(0,\vec{x}) &= y_0(\vec{x}) && \text{at } \{t=0\}\times\Omega. \end{aligned}$$

Here, we aim to minimize the *cost functional* $\mathcal{J}$ by minimizing the distance between a *state variable* y and a desired state, or *target state*, $\widehat{y}$, while also minimizing the amount of exerted *control*, u, in reaching the desired state.

The weight of the control is influenced by the *regularization parameter* $\beta > 0$. If β is small, the desired state will typically be approached more closely, but much control may have to be exerted. If β is large, the control will be more heavily penalized, with the consequence that the desired state might not be reached as closely. In the PDE-constrained optimization literature it is generally of interest that a numerical solution for the optimization problem is robust for different choices of β. The choice of a particular β then depends on the application involved. We note that in the related field of *inverse problems*, in which $\widehat{y}$ is derived from noisy data, different

strategies of choosing an appropriate β are employed. In such problems, the control functions as a noise smoothing term, so that the choice of β determines how much of this smoothing is necessary, depending on the level of noise in the target state. Different strategies, such as the discrepancy principle and the L-curve criterion, are discussed in, e.g., [122, Chapter 5].

The minimization problem is constrained by the given PDE model, since the state y is dictated by the solution to this model. The state can be changed by varying the control u, which is a source term in the PDE. The PDE constraint is also called the *state equation* or *forward problem* and these terms will be used interchangeably throughout the remainder of this thesis.

The problem (4.1) is a linear, distributed optimal control problem with Dirichlet boundary conditions. It is linear, both in the PDE and in the way the control enters the constraint. It is called a distributed control problem, since the cost functional $\mathcal{J}$ is acting on the whole space–time domain. There are other optimal control setups, in which the target state $\widehat{y}$ is only given at a final time T, or where control is only measured on parts, or on the boundary, of the domain, which changes the form of $\mathcal{J}$. Some of the theoretical results discussed below are extended to such setups in [86]. Dirichlet boundary conditions are the conditions applied in most classical texts. Other possible choices are Neumann, Robin, no-flux, or periodic boundary conditions, all of which will be discussed in this thesis and some of which are addressed in [86]. Most of the discussion in this section follows [123], unless stated otherwise.

4.1.1 An Abstract Optimization Problem in Banach Spaces

As illustrated in [123, Chapter 2], in order to understand the PDE-constrained optimization problem, we first consider a more general setting. The abstract problem is the following: Let U be a real reflexive Banach space with U_{ad} a closed convex subset of U. The problem of interest is

$$\min_{u \in U_{\text{ad}}} f(u),$$

i.e., we would like to minimize f by *controlling* u. A control $\bar{u} \in U_{\text{ad}}$ is called optimal, if

$$f(\bar{u}) \leq f(u) \quad \forall u \in U_{\text{ad}}. \tag{4.2}$$

In order to continue, a definition of lower semicontinuity is necessary.

Definition 4.1.1 (Weak lower semicontinuity [86, Chapter 2])**.** Every continuous and convex functional $f : U \to \mathbb{R}$ on a Banach space U is weakly lower semicontinuous; that is, for any sequence $\{u_n\}_{n=1}^{\infty} \subset U$ such that $u_n \rightharpoonup u$ as $n \to \infty$ we have

$$\lim \inf_{n \to \infty} f(u_n) \geq f(u).$$

Note that this is essentially Definition 3.1.22 in Section 3.1.4 and that the results in the following theorem are very similar to those used in the calculus of variations. Existence and uniqueness of an optimal control can be shown using the Weierstrass Theorem:

Theorem 4.1.1 (Weierstrass Theorem [123, Chapter 2])**.** Suppose $f : U \to \mathbb{R}$ is weakly lower semicontinuous with U a reflexive Banach space and $U_{\text{ad}} \subset U$ is closed and convex. Let the lower γ-level set $\{u \in U_{ad} : f(u) \leq \gamma\}$ of f be nonempty and bounded for some $\gamma \in \mathbb{R}$. Then problem (4.2) has an optimal solution, i.e., there exists $\bar{u} \in U_{\text{ad}}$ such that $f(\bar{u}) \leq f(u)$ for all $u \in U_{\text{ad}}$. If f is strictly convex then the solution is unique.

Differentiability

In order to be able to take derivatives in Banach spaces the notions of directional, Gâteaux, and Fréchet derivatives need to be introduced. Throughout this section, let U, V, and Z be real Banach spaces.

Definition 4.1.2. [124, Chapter 3] If, for given elements $u, h \in U$, the limit

$$\delta F(u)(h) := \lim_{t \to 0^+} \frac{1}{t} \Big(F(u + th) - F(u) \Big)$$

exists, then $\delta F(u)(h)$ is called the *directional derivative* of F at u in direction h. If this limit exists for all $h \in U$, then F is called directionally differentiable at u.

Definition 4.1.3. [124, Chapter 3] If, for some $u \in U$ and all $h \in U$, the limit

$$\delta F(u)(h) := \lim_{t \to 0} \frac{1}{t} \Big(F(u + th) - F(u) \Big)$$

exists and $\delta F(u)(h)$ is a continuous mapping from U to V, then $\delta F(u)$ is denoted by $F'(u)$ and is called the Gâteaux derivative of F at u, and F is called *Gâteaux differentiable* at u.

Definition 4.1.4. [124, Chapter 3] If F is Gâteaux differentiable at $u \in U$, and satisfies in addition that

$$\lim_{\|h\|_U \to 0} \frac{\big\| F(u + h) - F(u) - F'(u)h \big\|_V}{\|h\|_U} = 0,$$

then $F'(u)$ is called the Fréchet derivative of F at u and F is called *Fréchet differentiable*.

Definition 4.1.5 (Chain Rule [124, Chapter 3])**.** Let $F : U \to V$ and $G : V \to Z$ be Fréchet differentiable at u and $F(u)$, respectively. Then

$$E(u) = G(F(u))$$

is also Fréchet differentiable and its derivative is given by

$$E'(u) = G'(F(u))F'(u).$$

Definition 4.1.6 (Product Rule [125])**.** Let U and V be Banach spaces, let $\mathcal{U}$ be an open subset of U, and let $F : \mathcal{U} \to \mathbb{R}$ and $G : \mathcal{U} \to V$ be functions. If both F and G are differentiable at $u \in \mathcal{U}$, then FG is differentiable at u and

$$(FG)'(u) = F(u)G'(u) + G(u)F'(u).$$

First-Order Optimality Conditions

It is often of interest to define conditions that the optimization problem has to satisfy, since they can be used to characterize solutions and to solve the optimization problem numerically. The first-order necessary optimality condition for the reduced problem is characterized as follows:

Lemma 4.1.1. [86, Chapter 2] Let U_{ad} denote a nonempty and convex subset of a real Banach space U, and let the real-valued mapping f be Gâteaux differentiable in an open subset of U containing U_{ad}. If $\bar{u} \in U_{\text{ad}}$ is a solution to the problem

$$\min_{u \in U_{\text{ad}}} f(u),$$

then it solves the *variational inequality*

$$f'(\bar{u})(u - \bar{u}) \geq 0 \quad \forall u \in U_{\text{ad}}. \tag{4.3}$$

Conversely, if $\bar{u} \in U_{\text{ad}}$ solves the variational inequality (4.3) and f is convex, then $\bar{u}$ is a solution to the minimization problem $\min_{u \in U_{\text{ad}}} f(u)$.

This result constitutes the necessary first-order optimality condition, and, if f is convex, also the sufficient condition. Otherwise, second-order conditions have to be considered, but are omitted here in the interest of brevity, see [86, Chapter 4] and [124, Chapter 3].

4.1.2 PDE-Constrained Optimization in Banach Spaces

The above theory can now be applied to PDE-constrained optimization problems, as illustrated in [123, Chapter 3]. We consider a real Banach space Y and a real reflexive Banach space U, denoting the state and control space, respectively. Furthermore, we consider the closed convex set $U_{\text{ad}} \subset U$, called the set of admissible controls. Note that here we consider a problem without state constraints beyond the PDE. In an even more general case we could consider an admissible state space $Y_{\text{ad}} \subset Y$. The considered optimization problem is of the form

$$\min_{(y,u)\in Y\times U} \mathcal{J}(y,u) \quad \text{subject to} \quad e(y,u)=0, \quad y\in Y, u\in U_{\text{ad}}. \tag{4.4}$$

Here, $\mathcal{J}$ is a function to be minimized by varying the state variable y and the control variable u, subject to the constraint $e(y,u)=0$, which, in the case of PDE-constrained optimization, describes a PDE in terms of the variable y, depending also on the values of u. In many cases the cost functional $\mathcal{J}$ can be split into two parts

$$\mathcal{J}(y,u) := \mathcal{J}_1(y) + \mathcal{J}_2(u).$$

Often the first term drives the state y to a given target state $\widehat{y}$, while the second term reduces the control in some norm, as illustrated by the specific example (4.1).

One way of solving problem (4.4) is by building the *reduced cost functional*, which only depends on the variable u, the control. In order to do this, there has to exist a solution operator, or *control-to-state operator*, S, which maps each u to a state y, i.e. $u \mapsto S(u) = y(u)$, such that $y(u)$ satisfies $e(y(u),u)=0$. Then the reduced problem reads

$$\min_{u\in U_{\text{ad}}} \mathcal{J}(S(u),u). \tag{4.5}$$

For this reduced problem, one can utilize Theorem 4.1.1, which is summarized in the following corollary:

Corollary 4.1.1. [123, Chapter 3] Let the following assumptions hold:

1. $U_{\text{ad}} \subseteq U$ is closed and convex.
2. For each $u \in U_{\text{ad}}$, there exists a unique map $S(u)$ that solves $e(S(u),u)=0$.
3. S is weakly continuous, i.e., if $u_n \rightharpoonup u$ in U_{ad} then $S(u_n) \rightharpoonup S(u)$ in Y.
4. The lower γ-level set $\{u \in U_{\text{ad}} : \mathcal{J}(S(u),u) \leq \gamma\}$ of $\mathcal{J}$ is non-empty and bounded for some $\gamma \in \mathbb{R}$.
5. $\mathcal{J}_1$ is continuous and convex and $\mathcal{J}_2$ is weakly lower semicontinuous.

Then there exists a solution to (4.5).

Note that instead of using the γ-level set condition, boundedness of U_{ad} or coercivity of $\mathcal{J}_2$ can be utilized.

First-Order Optimality Conditions – Reduced Form

As in the abstract setting, we can apply Lemma 4.1.1 to the minimization problem (4.4) to derive the first-order optimality conditions.

Corollary 4.1.2. [123, Chapter 3] Let all the assumptions of Corollary 4.1.1 hold except U being reflexive. Let $\mathcal{U}$ be an open set in U such that $U_{\text{ad}} \subset \mathcal{U}$ and $u \mapsto S(u) : \mathcal{U} \mapsto Y$ is Gâteaux differentiable with derivative

$$S'(u) \in \mathcal{L}(U,Y),$$

and $(y,u) \mapsto \mathcal{J}(y,u) : Y \times U \to \mathbb{R}$ is Fréchet differentiable with

$$\mathcal{J}'(y,u) \in \mathcal{L}(Y \times U, \mathbb{R}).$$

If $\bar{u}$ is a minimizer of (4.4) over U_{ad} then the first-order necessary optimality conditions are given by

$$\langle S'(\bar{u})^* \mathcal{J}_y(S(\bar{u}),\bar{u}) + \mathcal{J}_u(S(\bar{u}),\bar{u}), u - \bar{u}\rangle_{U^*,U} \geq 0 \quad \forall u \in U_{\text{ad}}, \tag{4.6}$$

where $\mathcal{J}_y$ and $\mathcal{J}_u$ are the partial derivatives of $\mathcal{J}$. If $\mathcal{J}$ is convex then the condition (4.6) is sufficient.

Note that in (4.6) we use that $\mathcal{J}'(S(\bar{u}),\bar{u}) = S'(\bar{u})^* \mathcal{J}_y(S(\bar{u}),\bar{u}) + \mathcal{J}_u(S(\bar{u}),\bar{u})$. Here, the notation S^* denotes the dual operator for S as stated in Definition 3.1.7. As discussed in [123, Chapter 3], in order to understand how to deal with this variational inequality, the form of $S'(u)$ can be investigated further. To do this, we assume that the state equation $e(S(u),u) = 0$ can be differentiated, so that we have

$$e_y(S(\bar{u}),\bar{u})S'(\bar{u})h + e_u(S(\bar{u}),\bar{u})h = 0.$$

We can rearrange this for $S'(\bar{u})$ and substitute into (4.6) giving

$$\left\langle - e_u(S(\bar{u}),\bar{u})^* \left((e_y(S(\bar{u}),\bar{u})^{-1})^* \mathcal{J}_y(S(\bar{u}),\bar{u}) \right) + \mathcal{J}_u(S(\bar{u}),\bar{u}), u - \bar{u} \right\rangle_{U^*,U} \geq 0,$$

which would be quite hard to solve, since we would have to construct the inverse of e_y and find its dual operator. Instead, we introduce a new variable q, the *adjoint variable*, to solve the following relation

$$e_y(S(\bar{u}),\bar{u})^* q = \mathcal{J}_y(S(\bar{u}),\bar{u}), \tag{4.7}$$

called the *adjoint equation*. This simplifies the variational inequality so that it reads

$$\langle -e_u(S(\bar{u}),\bar{u})^* q + \mathcal{J}_u(S(\bar{u}),\bar{u}), u - \bar{u}\rangle_{U^*,U} \geq 0, \tag{4.8}$$

and the derivative of $\mathcal{J}$ is

$$\mathcal{J}'(S(\bar{u}),\bar{u}) = -e_u(S(\bar{u}),\bar{u})^* q + \mathcal{J}_u(S(\bar{u}),\bar{u}) \quad \in U^*.$$

This concludes the derivation of the optimality conditions for the reduced system, consisting of the state equation $e(S(\bar{u}),\bar{u}) = 0$, the adjoint equation (4.7), and the variational inequality (4.8).

First-Order Optimality Conditions – Lagrangian Approach

Instead of using the reduced form of the problem, an alternative strategy starts by considering the full form (4.4). The following is called the formal Lagrangian method, which can be made rigorous using Karush–Kuhn–Tucker (KKT) theory for optimization in Banach spaces, see for example [86, Chapter 6] for an introduction. Here, we will focus on the formal Lagrangian method, widely applied in PDE-constrained optimization, following the discussion in [123, Chapter 3]. First, let X be a real Banach space and denote the state map $e : Y \times U_{\text{ad}} \to X$. Then we can formulate the Lagrangian $\mathcal{L} : Y \times U_{\text{ad}} \times X^* \to \mathbb{R}$ as follows:

$$\mathcal{L}(y,u,q) = \mathcal{J}(y,u) - \langle e(y,u), q\rangle_{X,X^*}.$$

Here, X^* is the dual space of X, as introduced in Definition 3.1.6. The idea of the Lagrangian method is to find stationary states of $\mathcal{L}$ with respect to y, u, and q. If we have a triplet $(\bar{y},\bar{u},q)$ denoting such a state, then the partial derivatives with respect to y and q evaluated at this

stationary state are zero, while the derivative with respect to u is minimized. This is

$$\mathcal{L}_y(\bar{y}, \bar{u}, q) = 0, \quad \mathcal{L}_q(\bar{y}, \bar{u}, q) = 0, \quad \langle \mathcal{L}_u(\bar{y}, \bar{u}, q), u - \bar{u} \rangle_{U^*, U} \geq 0 \quad \forall u \in U_{\text{ad}}. \tag{4.9}$$

Note that if $U_{\text{ad}} = U$, i.e., if the problem had no control constraints, then the third expression would also lead to an equality. The derivative with respect to q reduces to the state equation

$$e(\bar{y}, \bar{u}) = 0,$$

and the derivative with respect to y is the adjoint equation, equivalent to the abstract version (4.7). Finally, the derivative with respect to u is exactly the variational formulation (4.6).

4.1.3 PDE-Constrained Optimization in Hilbert Spaces

The above results were all posed in Banach spaces. We now restrict our attention to a common class of PDE-constrained optimization problems, those posed in Sobolev spaces, since such spaces are a natural framework for PDE theory, as seen in Section 3.1. In particular, we often consider Hilbert spaces H^1 and L^2. In the following, we will illustrate the methods using a particular, simple example – that of *Poisson control*. As shown in [123, Chapter 6], such a PDE-constrained optimization problem is formulated as

$$\begin{aligned}
&\min_{y,u} \ \mathcal{J}(y, u) = \frac{1}{2}\|y - \widehat{y}\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)} \\
&\text{subject to} \\
&-\nabla^2 y = u \quad \text{in } \Omega, \\
&\qquad y = 0 \quad \text{on } \partial\Omega, \\
&\text{and pointwise control constraints} \\
&u \in U_{\text{ad}} := \{v \in L^2(\Omega) : u_a(\vec{x}) \leq v(\vec{x}) \leq u_b(\vec{x}) \text{ a.e. } \vec{x} \in \Omega\},
\end{aligned} \tag{4.10}$$

where $\widehat{y} \in L^2(\Omega)$ is the target state, and $\beta > 0$ is the regularization parameter. Note that if $u_a = -\infty, u_b = \infty$, so that $U_{\text{ad}} = L^2(\Omega)$, the control constraint may be removed.

Linear Elliptic Problem – Reduced Cost Functional

Following the account in [86, Chapter 2], we can apply the results of Section 4.1.2 to such an example. The solution map $S : L^2(\Omega) \to L^2(\Omega)$ exists for the problem (4.10), is linear and continuous. The reduced problem is

$$\min_{u \in U_{\text{ad}}} \mathcal{J}(S(u), u) := \frac{1}{2}\|Su - \widehat{y}\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)}, \tag{4.11}$$

and we have the following result:

Theorem 4.1.2. [86, Chapter 2] Suppose that real Hilbert spaces U and H, a nonempty and convex set $U_{\text{ad}} \subset U$, some $\widehat{y} \in H$, and a constant $\beta \geq 0$ are given. Moreover, let $S : U \to H$ denote a continuous linear operator. Then $\bar{u} \in U_{\text{ad}}$ is a solution to the minimization problem (4.11) if and only if $\bar{u}$ solves the variational inequality

$$(S^*(S\bar{u} - \widehat{y}) + \beta\bar{u}, u - \bar{u})_U \geq 0 \quad \forall u \in U_{\text{ad}}.$$

Here, S^* denotes the adjoint operator of S, as defined in Definition 3.1.8 and in our example $U = H = L^2(\Omega)$. We can compare this result to the abstract setting (4.6). However, since we are now working with Hilbert spaces, we have associated the derivative of $\mathcal{J}$ with the gradient

of $\mathcal{J}$

$$(\nabla \mathcal{J}(S(\bar{u}), \bar{u}), v)_U = \langle \mathcal{J}'(S(\bar{u}), \bar{u}), v \rangle_{U^*, U} \quad \forall v \in U,$$

by applying the Riesz Representation Theorem (Theorem 3.1.2), see [123, Chapter 3]. We furthermore have that $\mathcal{J}_u(S(\bar{u}), \bar{u}) = \beta \bar{u}$, $\mathcal{J}_y(S(\bar{u}), \bar{u}) = S\bar{u} - \widehat{y}$, and $S'(\bar{u})^* = S^*$. The latter result follows, since S is linear and so the derivative of Su in direction h is simply Sh. We can then check the assumptions in Corollary 4.1.1 to conclude the existence of solutions and for $\beta > 0$, since $\mathcal{J}$ is strictly convex, the solution is unique, as described in [123, Chapter 6].

Linear Elliptic Problem – Lagrangian Approach

We can now compare this to the Lagrangian approach, as discussed in [123, Chapter 6]. Equivalently to the approach in the Banach setting, we define the Lagrangian
$\mathcal{L} : H_0^1(\Omega) \times U_{\text{ad}} \times H^1(\Omega) \to \mathbb{R}$ as

$$\mathcal{L}(y, u, q) = \mathcal{J}(y, u) - (e(y, u), q),$$

where the brackets $(\cdot, \cdot)$ in the final term denote the L^2 inner product. We can the derive the optimality conditions, as in the Banach setting, by taking the derivatives of the Lagrangian and evaluating those at a stationary point $(\bar{y}, \bar{u}, q)$.

In particular, the Lagrangian, excluding boundary conditions, is

$$\mathcal{L}(y, u, q) = \frac{1}{2}\|y - \widehat{y}\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(\Omega)} - \int_\Omega \left(-\nabla^2 y - u\right) q d\vec{x},$$

and we have the following derivatives, evaluated at $(\bar{y}, \bar{u}, q)$:

$$\begin{aligned} \mathcal{L}_y(\bar{y}, \bar{u}, q)v &= \int_\Omega \left((\bar{y} - \widehat{y})v + q\nabla^2 v\right) d\vec{x} = 0, \\ \mathcal{L}_q(\bar{y}, \bar{u}, q)v &= \int_\Omega \left(-\nabla^2 \bar{y} - \bar{u}\right) v d\vec{x} = 0, \\ \mathcal{L}_u(\bar{y}, \bar{u}, q)(u - \bar{u}) &= \int_\Omega \beta\bar{u}(u - \bar{u}) + q(u - \bar{u}) d\vec{x} \geq 0. \end{aligned} \tag{4.12}$$

We integrate (4.12) by parts twice

$$\begin{aligned} \mathcal{L}_y(\bar{y}, \bar{u}, q)v &= \int_\Omega \left((\bar{y} - \widehat{y})v + q\nabla^2 v\right) d\vec{x} \\ &= \int_\Omega \left((\bar{y} - \widehat{y})v - \nabla v \cdot \nabla q + \nabla \cdot (q\nabla v)\right) d\vec{x} \\ &= \int_\Omega \left((\bar{y} - \widehat{y})v + v\nabla^2 q + \nabla \cdot (q\nabla v) - \nabla \cdot (v\nabla q)\right) d\vec{x} \\ &= \int_\Omega \left((\bar{y} - \widehat{y})v + v\nabla^2 q\right) d\vec{x} = 0, \end{aligned}$$

noting that the terms involving the divergence operator vanish by the Divergence Theorem and since v is zero on the $\partial\Omega$. Since this holds for all sufficiently smooth test functions v, we find the adjoint equation

$$\begin{aligned} -\nabla^2 q &= \bar{y} - \widehat{y} \quad \text{in } \Omega, \\ q &= 0 \qquad\quad \text{on } \partial\Omega. \end{aligned}$$

The derivative with respect to q recovers the state equation, while the equation for the derivative

with respect to u holds for all choices of $v \in U_{\text{ad}}$, so that we obtain

$$\bar{u}(\vec{x}) = \mathcal{P}_{U_{\text{ad}}}\left(-\frac{1}{\beta}q(\vec{x})\right) \quad \text{a.e. } \vec{x} \in \Omega. \tag{4.13}$$

Here, $\mathcal{P}_{U_{\text{ad}}}$ is the projection of the solution onto the admissible set U_{ad}. If $U_{\text{ad}} = U$, then the *gradient equation* of the problem is

$$\bar{u}(\vec{x}) = -\frac{1}{\beta}q(\vec{x}).$$

Linear Parabolic Problem

The example above involved an elliptic PDE constraint. Here we briefly discuss the problem with parabolic PDE constraints. We consider a problem of the form

$$\min_{y,u} \mathcal{J}(y,u) = \frac{1}{2}\|y - \widehat{y}\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2((0,T)\times\Omega)} \tag{4.14}$$

subject to

$$\begin{aligned} \frac{\partial y}{\partial t} - \nabla^2 y &= u && \text{in } (0,T)\times\Omega, \\ y(t,\vec{x}) &= 0 && \text{on } (0,T)\times\partial\Omega, \\ y(0,\vec{x}) &= y_0(\vec{x}) && \text{at } \{t=0\}\times\Omega, \end{aligned}$$

and pointwise control constraints

$$u \in U_{\text{ad}} := \{v \in L^2((0,T)\times\Omega) : u_a(t,\vec{x}) \le v(t,\vec{x}) \le u_b(t,\vec{x}) \text{ a.e. } \vec{x} \in \Omega, t \in (0,T)\}.$$

Importantly, as discussed in Section 3.1.3, we know that under certain assumptions and for each given u, the parabolic PDE that constrains problem (4.14) has a unique solution. In this case, we can define a continuous, linear control-to-state operator S, as done in the elliptic case, and derive the reduce cost functional $\mathcal{J}(Su,u)$. The rest of the analysis follows in much the same way as before, only that now all Hilbert spaces involved also include time, as defined in Section 3.1.3. Further details of how to tackle such problems are given in [86, Chapter 3].

Deriving the first-order optimality system using the Lagrangian approach is also very similar to the elliptic case and a detailed example of deriving this system for a similar parabolic problem will be given in Section 5.2. Here we refer to [86] for more details and just note that the Lagrangian for a parabolic problem is modified so that the state equation is integrated over both the domain Ω, as in the elliptic case, and the time horizon $(0,T)$. Omitting boundary conditions, we obtain

$$\mathcal{L}(y,u,q) = \frac{1}{2}\|y - \widehat{y}\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2((0,T)\times\Omega)} - \int_0^T \int_\Omega \left(\frac{\partial y}{\partial t} - \nabla^2 y - u\right) q\, d\vec{x}\, dt.$$

Following the procedure for elliptic problems we take derivatives with respect to the three variables y, u, q and evaluate these at the steady state $(\bar{y},\bar{u},q)$. The adjoint equation, resulting from differentiating the Lagrangian with respect to y, reads

$$\begin{aligned} -\frac{\partial q}{\partial t} - \nabla^2 q &= \bar{y} - \widehat{y} && \text{in } (0,T)\times\Omega, \\ q &= 0 && \text{on } (0,T)\times\partial\Omega, \\ q(T,x) &= 0 && \text{at } \{t=T\}\times\Omega, \end{aligned}$$

which instead of an initial condition has a final time condition. The state equation is retrieved when differentiating $\mathcal{L}$ with respect to q, and (4.13) remains the result of differentiating with respect to u.

Problems Involving Other PDE Constraints

While extending the ideas from linear elliptic to linear parabolic optimal control problems is reasonably straightforward, this is not the case when considering semilinear or even fully nonlinear problems. In general, for such problems one is interested in finding *locally optimal* solutions. Both elliptic and parabolic semilinear problems are discussed in [86]. Here we just mention that in order for these problems to have locally optimal controls, one often uses that the semilinear PDE has a unique solution, which can only be guaranteed under certain monotonicity conditions on the nonlinear term. We can establish that finding an optimal control is related to the uniqueness of solution to the PDE constraint. If the underlying PDE does not have a unique solution, given one (not necessarily optimal) instance of the control variable, then the control-to-state operator is not injective and hence the assumptions for the uniqueness of optimal controls require more careful analysis. Moreover, even if the underlying PDE does have a unique solution, if the associated reduced cost functional is not strictly convex we cannot generally assume uniqueness of optimal controls. This is the case in so-called *bilinear optimal control*, in which the PDE constraint in question is linear, but the state and control are coupled nonlinearly. One example of such an optimal control problem is one in which the constraint is an advection–diffusion model of the form

$$\frac{\partial y}{\partial t} = \frac{\partial^2 y}{\partial x^2} - u\frac{\partial y}{\partial x},$$

where the control u is a multiplicative advective driving force. Existence results for optimal control problems involving such PDE constraints have been derived in [126, 127], with recent extensions to advection–reaction–diffusion systems [128]. More importantly for the kind of optimal control problems that we consider in this thesis, there are some results on existence of optimal controls to problems that involve Fokker–Planck equations as PDE-constraints, see [129, 130]. In none of these cases can uniqueness of such an optimal control be established, since the control and the state are coupled multiplicatively, resulting in a nonlinear control-to-state operator, and hence a nonconvex reduced cost functional. While for nonlinear, nonconvex problems the task of finding a global, unique optimal control may be too difficult in practice, especially in real-world applications it is highly relevant to find *an* optimal control solution that can improve a process of interest.

4.2 Numerical Methods for PDE-Constrained Optimization

Having reviewed theoretical results for PDE-constrained optimization, this section provides an overview of numerical methods that are available for such problems. We start by classifying the different types of numerical methods before discussing them in turn.

4.2.1 Classification of Methods

As in the previous section, the aim is to solve the general PDE-constrained optimization problem (4.4). Here, the discussion of state and control constraints is omitted, and the reader is referred to [86, 124] for details on numerical methods for such problems. In this section, unconstrained problems of the form

$$\begin{aligned} &\min_{y,u} \mathcal{J}(y,u) \\ &\text{subject to} \\ &e(y,u) = 0, \quad y \in Y, \quad u \in U, \end{aligned} \tag{4.15}$$

are considered, where Y, U are Hilbert spaces. As discussed in [131], one can differentiate between black-box methods and all-at-once methods for solving such problems. Both are briefly discussed below, before focusing on solution techniques for the latter in Section 4.3.

Black-Box Methods

As discussed in Section 4.1.2, if there exists a control-to-state operator S, such that $y = S(u)$, the reduced problem can be defined as

$$\min_u \mathcal{J}(S(u), u).$$

Following Section 4.1.2, the first-order optimality conditions for this problem are derived. First, the derivative of $\mathcal{J}$ is found

$$\mathcal{J}'(S(u), u) = -e_u(S(u), u)^* q + \mathcal{J}_u(S(u), u),$$

where q is the adjoint variable given by the solution to the adjoint equation

$$e_y(S(u), u)^* q = \mathcal{J}_y(S(u), u).$$

Since U is a Hilbert space, $\mathcal{J}'$ is associated with the gradient $\nabla\mathcal{J}$ via

$$(\nabla\mathcal{J}(S(u), u), v)_U = \langle \mathcal{J}'(S(u), u), v\rangle_{U^*,U} \quad \forall v \in U,$$

by applying the Riesz Representation Theorem (Theorem 3.1.2). This gradient can then inform the design of the numerical method at hand, where the obvious choice becomes a gradient-based method. Examples of such methods are steepest decent, Conjugate Gradient, and Newton's methods, the first two of which are first-order methods, and the latter a second-order method. Another class of methods for tacking the reduced problem are trust-region methods. Both gradient based and trust-region methods are discussed below.

All-at-Once Methods

All-at-once methods do not consider the reduced cost functional, but keep the state variable y explicitly. One can then derive the first-order optimality system via the formal Lagrangian method as discussed in Section 4.1.2. This results in a coupled system consisting of the state, adjoint and gradient equations. Without state or control constraints, these satisfy the conditions

$$\mathcal{L}_y(\bar{y}, \bar{u}, q)h = 0, \quad \mathcal{L}_q(\bar{y}, \bar{u}, q)h = 0, \quad \mathcal{L}_u(\bar{y}, \bar{u}, q)h = 0,$$

compare to (4.9). Such systems can be solved either using direct methods, such as LU factorization, or iterative methods. Krylov subspace methods, Sequential Quadratic Programming (SQP), also called Lagrange–Newton methods, penalty, and augmented Lagrangian methods, are all often effective choices of iterative solvers. We refer to [132, Chapter 3] for details on these and focus on Krylov subspace methods for solving the linear system resulting from discretizing the optimality system in the following.

Optimize-then-Discretize vs Discretize-then-Optimize

One question that arises in the design of numerical methods for PDE-constrained optimization problems is that of whether to first derive the optimality conditions from the full problem (4.15) and then choose a discretization method to numerically solve it (*optimize-then-discretize*), or whether to first discretize the system and then derive finite-dimensional optimality conditions (*discretize-then-optimize*). Depending on the chosen discretization, these two approaches do not necessarily lead to the same discretized system, see [133] for an example. As discussed in [124, Chapter 4], an advantage of optimize-then-discretize approaches is that the information of the abstract setting can be used to inform the numerical method. For example, in finite element methods, one constructs solutions in a finite-dimensional subspace of the function space that the infinite-dimensional problem is posed in. Moreover, as pointed out in [124, Chapter 4], if the optimality system is derived in the continuous setting, the equations are mesh independent. Therefore, refining the mesh of the discretized optimality system can help determine the quality of the discretization and the convergence of the numerical method. In this thesis, we consider the

optimize-then-discretize approach since the discretization of the optimality system is achievable using numerical methods available for the forward problem, as introduced in Section 3.3.

4.2.2 Black-Box Solvers

In this section, an overview of black-box methods is provided. While these are not the focus of this thesis, they are included for completeness. Moreover, in the development of the numerical framework presented in this thesis, a fixed-point algorithm, which is a gradient-type method, and the MATLAB inbuilt solver `fsolve`, which uses a trust region algorithm, were used, demonstrating the relevance of black-box methods to problems considered in this thesis. Demonstrations of these early solvers are omitted in this thesis, but are discussed in [1].

Gradient-Based Methods

In the discussion of gradient-based methods, we follow [132, Chapter 3] and [134]. Gradient or descent based methods are aimed at solving the unconstrained, or reduced optimization problem $\mathcal{J}(S(u), u)$. For notational convenience we denote $\mathcal{J}(u) := \mathcal{J}(S(u), u)$ and discretize it to obtain

$$\min J(\mathbf{u}),$$

where $\mathbf{u}$ denotes the discretized variable u, and J denotes the discretized $\mathcal{J}$. Given an initial guess $\mathbf{u}_0$, the control $\mathbf{u}$ is updated iteratively based on a *descent direction* $\mathbf{d}$. At each iteration k this is

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \tau_k \mathbf{d}_k,$$

where τ_k is a *step length*. The descent direction has to satisfy

$$\begin{aligned} \mathbf{d}_k^\top \nabla J(\mathbf{u}_k) < 0 \qquad & \text{if } \nabla J(\mathbf{u}_k) \neq \mathbf{0}, \\ \mathbf{d}_k = \mathbf{0} \qquad & \text{if } \nabla J(\mathbf{u}_k) = \mathbf{0}. \end{aligned}$$

There are different ways of choosing $\mathbf{d}_k$ and τ_k. For the *steepest descent method*, the choice is simply

$$\mathbf{d}_k = -\nabla J(\mathbf{u}_k),$$

which converges linearly, see [132, Chapter 3]. An improved method is the *Conjugate Gradient method*, which updates the descent direction via

$$\begin{aligned} \mathbf{d}_0 &= -\nabla J(\mathbf{u}_0), \\ \mathbf{d}_k &= -\nabla J(\mathbf{u}_k) + \beta_k \mathbf{d}_{k-1}, \quad k > 0, \end{aligned} \tag{4.16}$$

where β_k needs to be determined based on the criterion that $\mathbf{d}_k^\top H_k \mathbf{d}_{k-1} = 0$, i.e., that $\mathbf{d}_k$ and $\mathbf{d}_{k-1}$ are *conjugated* with respect to the Hessian matrix H_k. For quadratic $\mathcal{J}$, $H_k = H$ and so

$$\beta_k = \frac{\nabla J(\mathbf{u}_k)^\top H \mathbf{d}_{k-1}}{\mathbf{d}_{k-1}^\top H \mathbf{d}_{k-1}}.$$

For non-quadratic $\mathcal{J}$, other formulations for β_k exist and can be found in [132, Chapter 3]. A third possible choice of descent direction is given by *Newton's method*, where $\mathbf{d}_k$ is given by the solution to the linear system

$$D^2 J(\mathbf{u}_k)\mathbf{d}_k = -\nabla J(\mathbf{u}_k),$$

which, provided $D^2J(\mathbf{u}_k)$ is positive definite, minimizes the following quadratic approximation to $\mathcal{J}$

$$J(\mathbf{u}_k + \mathbf{d}) = J(\mathbf{u}_k) + \mathbf{d}^\top \nabla J(\mathbf{u}_k) + \frac{1}{2}\mathbf{d}^\top D^2 J(\mathbf{u}_k)\mathbf{d}.$$

One issue with this method is that the computation of the Hessian $H_k = D^2J(\mathbf{u}_k)$ can become very expensive. Approximations to this term lead to the class of *quasi-Newton methods*.

Once the descent direction has been determined, one has to choose the step size τ_k. In general the *line search* approach involves finding

$$\tau_k = \arg\min_{\tau \in \mathbb{R}} J(\mathbf{u}_k + \tau \mathbf{d}_k).$$

For quadratic J such as

$$J(\mathbf{u}) = \frac{1}{2}\mathbf{u}^\top A\mathbf{u} - \mathbf{b}^\top \mathbf{u},$$

this can be done exactly by requiring that the derivative with respect to τ of $J(\mathbf{u}_k + \tau\mathbf{d}_k)$ vanishes. This results in the condition

$$\tau_k = \frac{\mathbf{g}_k^\top \mathbf{d}_k}{\mathbf{d}_k^\top A \mathbf{d}_k}, \quad \mathbf{g}_k = -\nabla J(\mathbf{u}_k) = \mathbf{b} - A\mathbf{u}_k.$$

For non-quadratic $\mathcal{J}$ other methods have to be applied to find τ_k, such as quadratic or cubic line-search, which are inexact procedures. Such examples are discussed in [132, Chapter 3].

Trust Region Methods

The following summary of trust region methods can be found in [132, Chapter 3] and [134]. As in gradient based methods, updates of $\mathbf{u}$ are formed, given a direction $\mathbf{d}_k$ and step size τ_k. The difference to gradient based methods is that by introducing trust regions, one can set $\mathbf{p} := \tau\mathbf{d}$, and form updates

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{p}_k. \tag{4.17}$$

Similar to the Newton method, the trust region method considers minimizing the quadratic approximation

$$m_k(\mathbf{p}) := J(\mathbf{u}_k + \mathbf{p}) = J(\mathbf{u}_k) + \mathbf{p}^\top \nabla J(\mathbf{u}_k) + \frac{1}{2}\mathbf{p}^\top H_k \mathbf{p},$$

where $H_k = D^2J(\mathbf{u}_k)$ or its approximation. However, instead of doing this globally, a *trust region*, defined by a ball of radius $\delta_k \in \mathbb{R}$, is considered in which to minimize m_k. The (sub-)problem formulation is

$$\min m_k(\mathbf{p}) \quad \text{subject to} \quad |\mathbf{p}| \leq \delta_k. \tag{4.18}$$

In order for $\mathbf{p}_k$ to lie within the trust region, H_k needs to be positive definite. Then

$$\mathbf{p}_k = -H_k^{-1} \nabla J(\mathbf{u}_k).$$

Otherwise problem (4.18) is considered with equality constraint $|\mathbf{p}| = \delta_k$, which can be solved using the Lagrangian formalism. The remaining question is how to determine whether $\mathbf{p}_k$, the solution to (4.18), is a good update for (4.17). In order to do this, the quadratic approximation is compared with the original model by calculating

$$r_k = \frac{J(\mathbf{u}_k) - J(\mathbf{u}_k + \mathbf{p}_k)}{m_k(\mathbf{0}) - m_k(\mathbf{p}_k)}.$$

If r_k is close to one, the model approximates J well for the step $\mathbf{p}_k$ and is an accepted iterate. If $\mathbf{u}_{k+1}$ is on the trust region boundary, the trust region is expanded. If $r_k > 0$ but much smaller than one, the trust region remains the same. If $r_k \leq 0$, the agreement between the quadratic model and the true model is not good, the step $\mathbf{p}_k$ is rejected, and the trust region is shrunk by a given factor.

There are different strategies for finding approximate solutions to the sub-problem (4.18) to make the method more efficient, such as the Cauchy point method, minimization in two-dimensional subspaces, and the dogleg method. The latter constructs the search direction $\mathbf{p}_k$ within the trust region, using a normalized steepest descent direction, a Newton step, or a linear combination of the two directions. Such methods are omitted here, but we refer to [134, Chapter 4] for a detailed discussion on trust region methods.

4.2.3 All-at-Once Solvers

In order to solve the full first-order optimality system using all-at-once methods, we need to discretize it with a chosen method. In this thesis, pseudospectral methods are used to do this, but one can consider finite difference, finite element, or finite volume methods instead, all of which were briefly discussed in Section 3.3. Irrespective of the discretization chosen, for linear problems, this leads to a matrix–vector equation

$$\mathcal{A}\mathbf{x} = \mathbf{b}. \tag{4.19}$$

In order to solve system (4.19), assuming invertibility, the aim is to find ways to approximate

$$\mathbf{x} = \mathcal{A}^{-1}\mathbf{b}.$$

There are two main avenues for solving such a liner system: (i) direct methods, such as utilizing LU or Cholesky factorizations, and (ii) iterative methods. Direct methods, which focus on simplifying the matrix $\mathcal{A}$ that needs to be inverted, are discussed in [134, Chapter 4] and [135, Chapter 2], but are not the focus of this chapter, since they rely on a small to medium size of the involved matrices. Instead, iterative solvers can be considered, which are based on finding a sequence $\{\mathbf{x}_k\}_{k=0}^{\infty}$ that converges to $\mathbf{x}^*$, the solution to (4.19). In particular, one can define

$$\mathbf{r}_k = \mathbf{b} - \mathcal{A}\mathbf{x}_k. \tag{4.20}$$

The quantity $\mathbf{r}_k$ is called the *residual*, which goes to zero as $\mathbf{x}_k$ approaches $\mathbf{x}^*$.

Saddle-Point and Block Systems

We follow the review article [136] (see also [137, 138] for discussion on preconditioning of such systems and [139] for saddle-point problems in the context of PDE-constrained optimization). One class of matrix vector equations of the form $\mathcal{A}\mathbf{x} = \mathbf{b}$ that has widely been studied is that of *saddle-point systems*, a class of block matrix systems. In the most general form, the involved matrix is defined as

$$\mathcal{A} = \begin{bmatrix} A & B_1^\top \\ B_2 & C \end{bmatrix}. \tag{4.21}$$

However, for the time being we restrict our attention to saddle-point systems where $\mathcal{A}$ has the structure

$$\mathcal{A} = \begin{bmatrix} A & B^\top \\ B & 0 \end{bmatrix}, \tag{4.22}$$

with $A \in \mathbb{R}^{n_1 \times n_1}$, $B \in \mathbb{R}^{n_2 \times n_1}$, $A = A^\top$ and B has full row rank $n_2 \leq n_1$. While the saddle-point matrix (4.22) is indefinite, conditions under which it is invertible can be determined as follows:

Theorem 4.2.1. [136] Assume that A is symmetric positive semidefinite, B has full rank, and $C = 0$. Then a necessary and sufficient condition for the saddle-point matrix $\mathcal{A}$ to be

nonsingular is $\ker A \cap \ker B = \{0\}$.

If A is also positive definite, (4.22) can be factorized as

$$\mathcal{A} = \begin{bmatrix} I & 0 \\ BA^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & A^{-1}B^\top \\ 0 & I \end{bmatrix}.$$

The matrix $S = -BA^{-1}B^\top$ is negative definite and is called the *Schur complement*. Given these conditions, the saddle-point system (4.22) admits a unique solution. Moreover, $\mathcal{A}$ has exactly n_1 positive and n_2 negative eigenvalues.

In [140] the following result was established on the spectrum of $\mathcal{A}$, when is A invertible. One can define the following two intervals satisfying $\mu(\mathcal{A}) \subset \mathcal{I}^- \cup \mathcal{I}^+$:

$$\mathcal{I}^- = \left[\frac{1}{2}\left(\mu_{n_1} - \sqrt{\mu_{n_1}^2 + 4\sigma_1^2}\right), \frac{1}{2}\left(\mu_1 - \sqrt{\mu_1^2 + 4\sigma_{n_2}^2}\right) \right] \subset \mathbb{R}^-,$$
$$\mathcal{I}^+ = \left[\mu_{n_1}, \frac{1}{2}\left(\mu_1 + \sqrt{\mu_1^2 + 4\sigma_1^2}\right) \right] \subset \mathbb{R}^+,$$

where μ_1 and μ_{n_1} are the largest and smallest eigenvalues of A, and σ_1, σ_{n_2} the largest and smallest singular values of B. These bounds help approximate the condition number of $\mathcal{A}$. Since then, these estimates have been improved, some of which are listed in the review article [136].

The reason that saddle-point systems are mentioned in this thesis is that they arise from discretizing the first-order optimality system of PDE-constrained optimization problems. This is demonstrated by the following example. The optimality system for the linear elliptic problem (4.10) in Section 4.1.3, neglecting boundary conditions for ease of argument, is given by

$$\begin{aligned} -\nabla^2 y &= u, \\ -\nabla^2 q &= y - \widehat{y}, \\ -\frac{1}{\beta} q &= u. \end{aligned} \tag{4.23}$$

The discretized versions of the spatial operator ∇^2 is denoted by D_{xx}, and the discretized variables become $\mathbf{y}$, $\mathbf{q}$ and $\mathbf{u}$. Rearranging terms and the order of equations we have

$$\begin{aligned} \mathrm{Id}\,\mathbf{y} \qquad\qquad\quad + \; D_{xx}\mathbf{q} &= \widehat{\mathbf{y}}, \\ \beta\mathrm{Id}\,\mathbf{u} + \quad \mathrm{Id}\,\mathbf{q} &= \mathbf{0}, \\ D_{xx}\mathbf{y} \; + \; \mathrm{Id}\,\mathbf{u} \qquad\quad &= \mathbf{0}, \end{aligned}$$

where Id is the discretized identity operator. Writing this as a matrix–vector system gives

$$\begin{bmatrix} \mathrm{Id} & 0 & D_{xx} \\ 0 & \beta\mathrm{Id} & \mathrm{Id} \\ D_{xx} & \mathrm{Id} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{u} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \widehat{\mathbf{y}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

which is precisely of the form (4.19). This discretized optimality system can be redefined using terms

$$A := \begin{bmatrix} \mathrm{Id} & 0 \\ 0 & \beta\mathrm{Id} \end{bmatrix}, \quad B := \begin{bmatrix} D_{xx} & \mathrm{Id} \end{bmatrix}.$$

Noting that the Laplacian operator is self-adjoint, so that $D_{xx} = D_{xx}^\top$, it is clear that the discretized optimality system has saddle-point structure given by (4.22). Discretized optimality systems of linear parabolic PDEs can also be cast in this framework, as well as those of linearized

versions of nonlinear PDEs. This link stems from KKT theory for the continuous problem, see [86, Chapter 6]. For a convex cost functional, the continuous optimality system finds the unique *saddle-point* of the Lagrangian. This saddle-point condition can be expressed as

$$\mathcal{L}(x^*, q) \leq \mathcal{L}(x^*, q^*) \leq \mathcal{L}(x, q^*),$$

where $x := [y, u]^\top$.

4.3 Krylov Subspace Methods

As discussed in the previous section, iterative methods are a valuable approach for solving linear systems of equations. Since Krylov subspace methods are a large class of iterative methods for solving such systems, this section outlines some of their most important features, introduces three of the most popular such methods, and discusses preconditioning techniques. As presented in [137], Krylov subspace methods aim to solve the linear system

$$\mathcal{A}\mathbf{x} = \mathbf{b}$$

based on the iterative procedure

$$\mathbf{x}_{k+1} = \mathbf{x}_0 + \alpha_0\mathbf{r}_0 + \alpha_1\mathcal{A}\mathbf{r}_0 + ... + \alpha_k\mathcal{A}^k\mathbf{r}_0 = \mathbf{x}_0 + \sum_{j=0}^{k} \alpha_j\mathcal{A}^j\mathbf{r}_0. \tag{4.24}$$

This implies that $\mathbf{x}_{k+1} - \mathbf{x}_0 \in \text{span}\{\mathbf{r}_0, \mathcal{A}\mathbf{r}_0, ..., \mathcal{A}^k\mathbf{r}_0\}$. This set forms a subspace of $\mathbb{R}^n$, where n is the size of $\mathbf{x}_k$. This subspace is called a *Krylov subspace* and is denoted by

$$\mathcal{K}_k(\mathcal{A}, \mathbf{r}_0) := \text{span}\{\mathbf{r}_0, \mathcal{A}\mathbf{r}_0, ..., \mathcal{A}^k\mathbf{r}_0\}.$$

Some additional observations can be made, such as $\sum_{j=0}^{k} \alpha_j\mathcal{A}^j$ being a polynomial in $\mathcal{A}$. Therefore, the iteration can be redefined as

$$\mathbf{x}_{k+1} = \mathbf{x}_0 + q(\mathcal{A})\mathbf{r}_0, \tag{4.25}$$

where

$$q(z) = \sum_{j=0}^{k} \alpha_j z^j.$$

Moreover, by manipulating (4.25), as shown in [137], it can be established that the k-th residual depends on $\mathbf{r}_0$ in terms of $q(\mathcal{A})$. This is

$$\mathbf{r}_{k+1} = \mathbf{b} - \mathcal{A}\mathbf{x}_{k+1} = \mathbf{r}_0 - \mathcal{A}q(\mathcal{A})\mathbf{r}_0 = p(\mathcal{A})\mathbf{r}_0,$$

where

$$p(z) = 1 - \sum_{j=1}^{k+1} \alpha_{j-1} z^j. \tag{4.26}$$

This shows that the $\mathbf{r}_k$ are spanned by $\mathcal{K}_k(\mathcal{A}, \mathbf{r}_0)$.

As discussed in [137], there are many types of Krylov subspace methods, the choice of which depends on the properties of $\mathcal{A}$. The three most popular methods are briefly outlined below. The Conjugate Gradient (CG) method, developed in [141], is a good choice for symmetric positive definite $\mathcal{A}$. For symmetric indefinite $\mathcal{A}$, the most used method is called Minimum Residual (MINRES) method, introduced in [142], whereas for non-symmetric $\mathcal{A}$, the Generalized Minimum Residual (GMRES) method, published in [143], is the most popular.

4.3.1 The Conjugate Gradient Method

As discussed in Section 4.2, Conjugate Gradient methods can be used as a black-box algorithm in the setting of an optimization problem. However, here we present a CG method for solving linear systems of the form (4.19), following [144, Chapter 2]. Given an initial guess $\mathbf{x}_0$, we want to find the solution $\mathbf{x}^*$ to (4.19) by an iterative procedure

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \tau_k \mathbf{d}_k. \tag{4.27}$$

The aim of the CG method is to minimize $\mathbf{x}^* - \mathbf{x}_k$ in the $\mathcal{A}$-norm over $\mathcal{K}_k(\mathcal{A}, \mathbf{r}_0)$. The full algorithm is detailed in Algorithm 1. For the CG method, we have the following results:

Lemma 4.3.1. [144, Chapter 2] For any k such that $\mathbf{x}_k \neq \mathbf{x}^*$, the vectors generated by the Conjugate Gradient method satisfy

1. $\langle \mathbf{r}_k, \mathbf{d}_j \rangle = \langle \mathbf{r}_k, \mathbf{r}_j \rangle = 0, \quad j < k,$
2. $\langle \mathcal{A}\mathbf{d}_k, \mathbf{d}_j \rangle = 0, \quad j < k,$
3. $\text{span}\{\mathbf{r}_0, \mathbf{r}_1, ..., \mathbf{r}_{k-1}\} = \text{span}\{\mathbf{d}_0, \mathbf{d}_1, ..., \mathbf{d}_{k-1}\} = \mathcal{K}_k(\mathcal{A}, \mathbf{r}_0).$

Moreover, we can say the following:

Theorem 4.3.1. [144, Chapter 2] The iterate $\mathbf{x}_k$ generated by the conjugate gradient method is the unique member of $\mathbf{x}_0 + \mathcal{K}_k(\mathcal{A}, \mathbf{r}_0)$ for which the $\mathcal{A}$-norm of the error is minimized, or alternatively, the residual $\mathbf{r}_k := \mathbf{b} - \mathcal{A}\mathbf{x}_k$ is orthogonal to $\mathcal{K}_k(\mathcal{A}, \mathbf{r}_0)$.

Algorithm 1: The Conjugate Gradient Method [144, Chapter 2]

Choose $\mathbf{x}_0$, compute $\mathbf{r}_0 = \mathbf{b} - \mathcal{A}\mathbf{x}_0$, set $\mathbf{d}_0 = \mathbf{r}_0$.
for $k = 0, 1, ...$ *until convergence* **do**

$$\tau_k = \frac{\langle \mathbf{r}_k, \mathbf{r}_k \rangle}{\langle \mathcal{A}\mathbf{d}_k, \mathbf{d}_k \rangle}$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \tau_k \mathbf{d}_k$$
$$\mathbf{r}_{k+1} = \mathbf{r}_k - \tau_k \mathcal{A}\mathbf{d}_k$$
$$< \text{Test for convergence} >$$
$$\beta_k = \frac{\langle \mathbf{r}_{k+1}, \mathbf{r}_{k+1} \rangle}{\langle \mathbf{r}_k, \mathbf{r}_k \rangle}$$
$$\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k$$

4.3.2 MINRES

The MINRES algorithm was first introduced in [142], and is used for problems involving symmetric indefinite matrices $\mathcal{A}$. We follow the presentation in [144, Chapter 2]. MINRES is based on ideas from the Lanczos method for the construction of an orthonormal basis for a Krylov subspace. This works as follows. We choose $\mathbf{v}_0 = \mathbf{0}$ and a vector $\mathbf{v}_1$ with unit norm. Then the recurrence relation

$$\gamma_{j+1}\mathbf{v}_{j+1} = \mathcal{A}\mathbf{v}_j - \delta_j \mathbf{v}_j - \gamma_j \mathbf{v}_{j-1}$$

can construct such a basis. Here $\delta_j = \langle \mathcal{A}\mathbf{v}_j, \mathbf{v}_j \rangle$, and the parameter γ_{j+1} ensures that $\mathbf{v}_{j+1}$ has unit norm. This can be rewritten in a compact form as

$$\mathcal{A}V_k = V_k T_k + \gamma_{k+1}[\mathbf{0}, ..., \mathbf{0}, \mathbf{v}_{k+1}], \tag{4.28}$$

where $V_k = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k]$, T_k is a tridiagonal matrix with entries δ_j on the diagonal, γ_j on the sub-diagonal, and γ_{j+1} on the super-diagonal. Then, conveniently, we have

$$V_k^\top \mathcal{A} V_k = T_k.$$

Using this basis, we can write an update of the form (4.27) as

$$\mathbf{x}_k = \mathbf{x}_0 + V_k \mathbf{y}_k,$$

where $\mathbf{y}_k$ is such that $V_k^\top \mathbf{r}_k = \mathbf{0}$. Using (4.28), the update (4.27) can be expressed as

$$\mathbf{r}_k = \mathbf{r}_0 - \mathcal{A} V_k \mathbf{y}_k = V_{k+1} \left(\|\mathbf{r}_0\| \mathbf{e}_1 - \hat{T}_k \mathbf{y}_k \right), \tag{4.29}$$

where $\mathbf{e}_1 = [1, 0, ..., 0]^\top$ and $\hat{T}_k = [T_k; 0, ..., 0, \gamma_{k+1}]$. In MINRES, the aim is to minimize this residual in the standard 2-norm $\|\mathbf{r}_k\|$. Having rewritten (4.29), this can be computed cheaply, using QR factorization of $\hat{T}_k$ and a Givens rotation at each iteration.

4.3.3 GMRES

Now that we understand how the MINRES algorithm works, we can generalize it for problems involving non-symmetric $\mathcal{A}$. Again, we follow [144, Chapter 7] in this section. The key idea is to replace the Lanczos method in the construction of an orthogonal basis with the Arnoldi method, since this can deal with non-symmetric matrices. The Arnoldi method constructs the basis using a modified Gram–Schmidt process, see [144, Chapter 7] for more details. Then in analogous notation to the MINRES method, we can write this as

$$\mathcal{A} V_k = V_k H_k + h_{k+1,k} [\mathbf{0}, ..., \mathbf{0}, \mathbf{v}_{k+1}],$$

with $H_k = V_k^\top \mathcal{A} V_k$. Here H_k with entries h_{ij} is an upper-Hessenberg matrix. In an analogous fashion to the MINRES algorithm we can define the k-th residual in terms of this basis as

$$\mathbf{r}_k = \mathbf{r}_0 - \mathcal{A} V_k \mathbf{y}_k = V_{k+1} \left(\|\mathbf{r}_0\| \mathbf{e} - \hat{H}_k \mathbf{y}_k \right).$$

Then, since the columns of V_{k+1} are orthogonal, the normed residual is

$$\|\mathbf{r}_k\| = \| \|\mathbf{r}_0\| \mathbf{e}_1 - \hat{H}_k \mathbf{y}_k \|.$$

The full GMRES algorithm can be found in Algorithm 2.

4.3.4 Convergence Results for Krylov Subspace Methods

In this section, convergence results for the three methods introduced above are stated.

Conjugate Gradient Method

Following the account in [137], there exists a convergence result for symmetric positive definite $\mathcal{A}$ in terms of the polynomial (4.26). This has been established in [145]. We denote the set of real polynomials of degree $\leq k$ by Π_k and the spectrum of eigenvalues of $\mathcal{A}$ by $\mu(\mathcal{A})$. Let the $\mathcal{A}$-norm satisfy $\|\mathbf{x}\|_\mathcal{A}^2 = \mathbf{x}^\top \mathcal{A} \mathbf{x}$. Then the following result holds:

$$\frac{\|\mathbf{x} - \mathbf{x}_k\|_\mathcal{A}}{\|\mathbf{x} - \mathbf{x}_0\|_\mathcal{A}} \leq \min_{p_k \in \Pi_k, p_k(0)=1} \max_{\lambda \in \mu(\mathcal{A})} |p_k(\lambda)|. \tag{4.30}$$

Now, assuming that we have s (distinct) eigenvalues of $\mathcal{A}$, then they are roots of $p_s(z)$, where $s \leq k$, and the algorithm will converge in s iterations. In particular, if eigenvalues are clustered, a very small error can be reached with a small number of iterations. If eigenvalues lie in an interval $\lambda_{\min}^\mathcal{A} \leq \lambda \leq \lambda_{\max}^\mathcal{A}$, the condition number $\kappa = \frac{\lambda_{\max}^\mathcal{A}}{\lambda_{\min}^\mathcal{A}}$ can be computed and the right-hand

Algorithm 2: The GMRES Method [144, Chapter 7]

Choose $\mathbf{x}^{(0)}$, compute $\mathbf{r}^{(0)} = \mathbf{b} - \mathcal{A}\mathbf{x}^{(0)}$, $\beta_0 = \|\mathbf{r}^{(0)}\|$, $\mathbf{v}^{(1)} = \frac{\mathbf{r}^{(0)}}{\beta_0}$.

for $k = 1, 2, \ldots$ *until* $\beta_k < \tau\beta_0$ **do**

$$\mathbf{w}_1^{(k+1)} = A\mathbf{v}^{(k)}$$

for $l = 1$ *to* k **do**

$$h_{lk} = \langle \mathbf{w}_l^{(k+1)}, \mathbf{v}^{(l)} \rangle$$
$$\mathbf{w}_{l+1}^{(k+1)} = \mathbf{w}_l^{(k+1)} - h_{lk}\mathbf{v}^{(l)}$$

$$h_{k+1,k} = \|\mathbf{w}_{k+1}^{(k+1)}\|$$
$$\mathbf{v}^{(k+1)} = \frac{\mathbf{w}_{k+1}^{(k+1)}}{h_{k+1,k}}$$

Compute $\mathbf{y}^{(k)}$ such that $\beta_k = \|\beta_0\mathbf{e}_1 - \hat{H}_k\mathbf{y}^{(k)}\|$ is minimized, where $\hat{H}_k = [h_{ij}]_{1\leq i\leq k+1, 1\leq j\leq k}$.

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + V_k\mathbf{y}^{(k)}$$

side of (4.30) can be replaced to obtain

$$\frac{\|\mathbf{x} - \mathbf{x}_k\|_{\mathcal{A}}}{\|\mathbf{x} - \mathbf{x}_0\|_{\mathcal{A}}} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k.$$

From this we can observe that the more clustered the eigenvalues are, the faster the convergence of the algorithm becomes.

MINRES

Following [137], similar convergence results for MINRES can be established as follows:

$$\frac{\|\mathbf{r}_k\|_{l_2}}{\|\mathbf{r}_0\|_{l_2}} \leq \min_{p_k \in \Pi_k, p_k(0)=1} \max_{\lambda \in \mu(\mathcal{A})} |p_k(\lambda)|.$$

The differences from the result for Conjugate Gradient methods are twofold and lie in the properties of $\mathcal{A}$. Firstly, the residual is now minimized as opposed to $\mathbf{x} - \mathbf{x}_k$, which is a difference in which norm convergence is measured. Moreover, while the eigenvalues of $\mathcal{A}$ are real, some are positive and some are negative. Therefore, a simplified error bound as for CG cannot be derived as easily. First of all, instead of looking at the case when the eigenvalues are in one interval, two intervals $[-a, -b] \cup [c, d]$, for a, b, c, d positive, need to be considered to account for positive and negative eigenvalues. Only if $d - c = a - b$, does the error bound simplify. For $\kappa = \frac{ad}{bc}$ this is

$$\frac{\|\mathbf{r}_k\|_{l_2}}{\|\mathbf{r}_0\|_{l_2}} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{\lfloor k/2 \rfloor}.$$

While this is a somewhat similar result to the Conjugate Gradient case, here κ does not necessarily correspond to the condition number of $\mathcal{A}$.

GMRES

As discussed in [137], the convergence of methods for non-symmetric problems is harder to establish. One illustrative result is given in this review, in which $\mathcal{A}$ is diagonalizable, i.e., $\mathcal{A} = X\Lambda X^{-1}$. Then we can at least establish the following result:

$$\|\mathbf{r}_k\| \leq \|X\|\|X^{-1}\| \min_{p_k \in \Pi_k, p_k(0)=1} \max_{z \in \mathcal{R}} |p_k(z)| \|\mathbf{r}_0\|,$$

where $\mathcal{R}$ is a region which contains the eigenvalues. However, this region could be rather large, and $\|X\|\|X^{-1}\|$ has to be approximated. There are other ways of estimating GMRES convergence, some of which are mentioned in [137].

4.3.5 Preconditioning for Iterative Solvers

As we have now seen, the considered Krylov subspace methods have error bounds that depend on the spectrum of $\mathcal{A}$. If eigenvalues are clustered, we can expect fast convergence of the numerical method. However, in many problems, the eigenvalues of $\mathcal{A}$ are spread out, and in this case Krylov subspace methods become very expensive, due to the large number of iterations that are necessary for convergence. Therefore their use depends again on the properties of the matrix $\mathcal{A}$. The idea that comes to the rescue in this situation is called *preconditioning*, which is nicely illustrated in [146]. Other resources on this topic include [135, 137, 138, 147]. The aim is to find an invertible matrix P, the *preconditioner*, such that

$$P^{-1}\mathcal{A}\mathbf{x} = P^{-1}\mathbf{b}$$

is solved more easily than the original problem (4.19). In particular, this may mean that $P^{-1}\mathcal{A}$ has better clustered eigenvalues, i.e., a smaller condition number, resulting in faster convergence of the iterative solver. Moreover, P^{-1} should be cheap to compute, in order to be easily applicable at each step of the iterative process. There are different types of preconditioners, depending on the problem at hand. In the below, we will focus on one such class: preconditioners for saddle-point or block systems, since first-order optimality systems are of this structure. For a general discussion on preconditioning, we refer to [137, 138, 147].

Preconditioners for Saddle-Point and Block Systems

There are a range of preconditioners available for systems of saddle-point structure. Many of these are mentioned in the reviews [136, 137], as well as in [146]. For example, [146] is concerned with developing preconditioners based on the saddle-point structure of PDE-constrained optimization problems. Here, however, we only state some classic results. Following the presentation in [146] of results originally shown in [148, 149], the following preconditioners involving the Schur complement $S = -BA^{-1}B^\top$ exist. For an invertible matrix $\mathcal{A}$, given by (4.22), one can define a preconditioner

$$P_1 = \begin{bmatrix} A & 0 \\ 0 & -S \end{bmatrix},$$

with the spectrum of the preconditioned matrix satisfying

$$\mu(P_1^{-1}\mathcal{A}) \in \left\{1, \frac{1 \pm \sqrt{5}}{2}\right\}.$$

A second possible preconditioner under these conditions is given by

$$P_2 = \begin{bmatrix} A & 0 \\ B & S \end{bmatrix},$$

with eigenvalues given by

$$\mu(P_2^{-1}\mathcal{A}) \in \{1\}.$$

A final preconditioner and spectrum pair is given by

$$P_3 = \begin{bmatrix} A & 0 \\ B & -S \end{bmatrix}, \quad \mu(P_3^{-1}\mathcal{A}) \in \{\pm 1\}.$$

With exact application of the preconditioners P_1, P_2, P_3, a Krylov subspace method will converge in $3, 2$ and 2 iterates, respectively. However, as pointed out in [146], while these preconditioners give great results in terms of the spectrum of the resulting matrix, they are often prohibitively expensive to compute. In particular, forming the Schur complement, involving A^{-1}, is an issue to be addressed. Therefore, rather than using these preconditioners as stated above, one is interested in forming approximations to them, and in particular to S. This is highly problem-specific. For example, a Schur complement approximation for the Stokes equation was introduced in [150] and for the Navier–Stokes equation in [151]. For PDE-constrained optimization, Schur complement approximations were developed in [152], and extended in, for example, [153, 154, 155, 156].

Furthermore, the results for preconditioners P_2 and P_3 can be generalized to more general block systems of the form (4.21), in which $C \neq 0$, see [146, 157]. Considering block system (4.21), a matrix–vector system of the form

$$\mathcal{A}\begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix} = \begin{bmatrix} A & B_1^\top \\ B_2 & C \end{bmatrix}\begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix} = \begin{bmatrix} \vec{a} \\ \vec{b} \end{bmatrix}$$

can equivalently be written as

$$\left(C - B_2 A^{-1} B_1\right)\vec{y} = \vec{b} - B_1 A^{-1}\vec{a}, \quad \vec{x} = A^{-1}\left(\vec{a} - B_1\vec{y}\right),$$

provided that A is invertible. If A is cheap to invert, it may be advantageous to solve this system for $\vec{x}$ and $\vec{y}$ instead of the original matrix–vector system. For such problems, the Schur complement becomes $S = C - B_2 A^{-1} B_1$. We will see an example of an approximation to such a generalized Schur complement, when constructing a preconditioner for a block system of the form (4.21) in the derivation of our own numerical method in Section 5.4.

Chapter 5

PDE-Constrained Optimization for Particle Dynamics

In Chapter 2, we discussed how to model particle dynamics using DDFT. In Chapter 3, theoretical results for PDEs, as well as their numerical solution were discussed, including how to solve DDFT models numerically. Chapter 4 introduced theory and numerical methods for PDE-constrained optimization for standard PDEs, such as linear parabolic equations. In this chapter, we are now ready to merge these areas and extend them to create a novel framework for PDE-constrained optimization for DDFT models, and to tackle the optimal control problem we posed in Chapter 1. The content of this chapter is published in [1].

The chapter is structured as follows: in Section 5.1 the optimal control problems of interest are introduced, and in Section 5.2 first-order optimality conditions for these problems are derived. In Section 5.3 related work on the topic of mean-field optimal control is discussed. Then, in Section 5.4, a Newton–Krylov algorithm is introduced to solve DDFT optimal control problems and numerical results are presented in Section 5.5.

5.1 The Optimal Control Problem

We are now in the position to formulate a general optimal control problem that has a particle dynamics DDFT model as constraint. One such problem is given by

$$\begin{aligned}
&\min_{\rho,\vec{w}} \mathcal{J}(\rho,\vec{w}) := \frac{1}{2}\|\rho-\widehat{\rho}\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2}\|\vec{w}\|^2_{L^2((0,T)\times\Omega)},\\
&\text{subject to}\\
&\frac{\partial\rho}{\partial t} = \nabla\cdot\left(\rho\nabla\frac{\delta\mathcal{F}[\rho]}{\delta\rho}\right) - \nabla\cdot(\rho\vec{w}) \quad \text{in } (0,T)\times\Omega,\\
&\rho(0,x) = \rho_0(x) \qquad\qquad\qquad\qquad \text{at } \{t=0\}\times\Omega,
\end{aligned}$$

subject to suitable boundary conditions.

We aim to drive the particle distribution, or state, ρ towards a target state $\widehat{\rho}$, by varying the control variable $\vec{w}$. The PDE constraint is of the form (1.3) introduced in Chapter 1. While we are most interested in the case where the control is applied bilinearly, via the advective field $\vec{w}$, we also derive conditions for the linear control problem. The linear control problem includes a source term w, modelling a particle source throughout the domain. We will refer to the control via the advective field as *flow control*, while control via a source term is called *source control*. Note that in DDFT, the advective field $\vec{w}$ is generally of the form $\vec{w} = \nabla V$, where V is an external potential. We refer to Section 2.3.7 and [5] for considerations of other forms of $\vec{w}$ and their limitations. Enforcing the condition $\vec{w} = \nabla V$ in an optimal control setting is more complicated, however, since enforcing the gradient of the quantity we aim to control is

a problem without a unique solution. At present, we restrict ourselves to control via a general $\vec{w}$ as a test case for the methodology.

The form of the free energy $\mathcal{F}$ in the PDE constraint depends on the model problem at hand. In this chapter we use the free energy given by

$$\mathcal{F}[\rho] = \int_\Omega \left[\rho(\vec{x}) \left(\ln \rho(\vec{x}) - 1 \right) + \rho(\vec{x}) V_1(\vec{x}) + \frac{1}{2} \rho(\vec{x}) \int_\Omega \rho(\vec{x}') V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \right] d\vec{x},$$

compare to (1.2), resulting in a model of the form

$$\min_{\rho, \vec{w}} \mathcal{J}(\rho, \vec{w}) := \frac{1}{2} \|\rho - \widehat{\rho}\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2} \|\vec{w}\|^2_{L^2((0,T)\times\Omega)}, \tag{5.1}$$

subject to

$$\frac{\partial \rho}{\partial t} = \nabla^2 \rho + \nabla \cdot (\rho \nabla V_1) - \nabla \cdot (\rho \vec{w}) + \nabla \cdot \int_\Omega \rho(t, \vec{x}) \rho(t, \vec{x}') \nabla V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \qquad \text{in } (0,T) \times \Omega,$$

$$\rho(0, \vec{x}) = \rho_0(\vec{x}) \qquad \text{at } \{t = 0\} \times \Omega,$$

with either Dirichlet boundary conditions or no-flux boundary conditions of the form

$$0 = \frac{\partial \rho}{\partial n} + \rho \frac{\partial V_1}{\partial n} - \rho \vec{w} \cdot \vec{n} + \int_\Omega \rho(t, \vec{x}) \rho(t, \vec{x}') \frac{\partial V_2(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' \qquad \text{on } (0,T) \times \partial\Omega,$$

compare to (1.1), (1.4). Note that, again, $k_B T = \Gamma = 1$ throughout.

In terms of boundary conditions, we will mostly focus on no-flux boundary conditions in the upcoming chapters, since they are considerably harder to compute numerically than Dirichlet or periodic conditions. Moreover, they are most relevant in real-world applications, because they model mass conservation, i.e., that no particles can escape the domain of interest. However, in this chapter we will give a more general account and we consider both Dirichlet and no-flux boundary conditions. To that end, we introduce a somewhat more general notation for the PDE constraint of interest. This will also be useful in later chapters, when deriving optimality conditions for extended models.

5.1.1 Source Control

We first introduce the source control problem, which we define as

$$\min_{\rho, w} \mathcal{J}(\rho, w) := \frac{1}{2} \|\rho - \widehat{\rho}\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2} \|w\|^2_{L^2((0,T)\times\Omega)} \tag{5.2}$$

subject to

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \big(\mathcal{D}(\rho) + \mathcal{I}(\rho) \big) + \mathcal{C}_s(w) + f \quad \text{in } (0,T) \times \Omega,$$

$$\rho(0, \vec{x}) = \rho_0(\vec{x}) \qquad \text{at } \{t = 0\} \times \Omega,$$

where a source f and target $\widehat{\rho}$ are given. Note that f is included for completion and that usually $f = 0$. Here, w is a scalar function which is applied to the PDE linearly. The operators are defined as

$$\mathcal{D}(\rho) = -\nabla \rho - \rho \nabla V_1 + \rho \vec{f} \tag{5.3}$$

$$\mathcal{I}(\rho) = -\int_\Omega \rho(t, \vec{x}) \rho(t, \vec{x}') \nabla V_2(|\vec{x} - \vec{x}'|) d\vec{x}', \tag{5.4}$$

where V_1, V_2, and $\vec{f}$ are given functions. Note that usually $\vec{f} = \vec{0}$.

The operator involving the control is simply

$$\mathcal{C}_s(w) = w, \tag{5.5}$$

where s denotes 'source control'. We impose either Dirichlet boundary conditions

$$\rho(t, \vec{x}) = g(t, \vec{x}) \quad \text{on } (0,T) \times \partial\Omega, \tag{5.6}$$

with g a given function, or no-flux boundary conditions

$$\big(\mathcal{D}(\rho) + \mathcal{I}(\rho)\big) \cdot \vec{n} = 0 \qquad \text{on } (0,T) \times \partial\Omega, \tag{5.7}$$

where $\vec{n}$ defines the outward unit normal.

5.1.2 Flow Control

We can modify the above optimization problem so that the control is the advective field instead of a particle source. This is more relevant for applications of interest, as discussed above. The optimal control problem is

$$\begin{aligned}
&\min_{\rho,\vec{w}} \mathcal{J}(\rho, \vec{w}) := \frac{1}{2}\|\rho - \widehat{\rho}\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2}\|\vec{w}\|^2_{L^2((0,T)\times\Omega)} \\
&\text{subject to} \\
&\frac{\partial \rho}{\partial t} = -\nabla \cdot \big(\mathcal{D}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho, \vec{w})\big) + f \quad \text{in } (0,T) \times \Omega, \\
&\rho(0, \vec{x}) = \rho_0(\vec{x}) \qquad\qquad\qquad\qquad\quad \text{at } \{t = 0\} \times \Omega,
\end{aligned} \tag{5.8}$$

with $\mathcal{D}(\rho)$ and $\mathcal{I}(\rho)$ as defined in (5.3) and (5.4), respectively. This can be compared to (5.1). The control operator is now defined as

$$C(\rho, \vec{w}) = \rho\vec{w}, \tag{5.9}$$

and we again either impose Dirichlet boundary conditions (5.6), or modify the no-flux conditions above to incorporate the flow control contribution

$$\big(\mathcal{D}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho, \vec{w})\big) \cdot \vec{n} = 0 \quad \text{on } (0,T) \times \partial\Omega. \tag{5.10}$$

Note that often we define $\mathrm{j} := \mathcal{D}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho, \vec{w})$ to be the particle flux. When $f = 0$, a case we will often choose, we can see that the PDE model is a continuity equation of the form

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathrm{j},$$

with no-flux boundary condition $\mathrm{j} \cdot \vec{n} = 0$. Now that the optimal control problems (5.2) and (5.8) are defined, we can approach their solution. In order to do so, the corresponding first-order optimality systems have to be derived, using the formal Lagrange method.

5.2 First-Order Optimality Conditions

In this section, the optimality conditions for problems (5.2) and (5.8) with Dirichlet and no-flux boundary conditions are derived. For an introduction to the Lagrangian method, applied in this section, we refer to [86].

5.2.1 Source Control

First, the source control problem (5.2) with Dirichlet boundary conditions (5.6) is considered. Initially the particle interaction term $\mathcal{I}$ is omitted to introduce the methodology. This is then a linear advection–diffusion control problem, which falls into the framework introduced in Chapter

4. The Lagrangian for (5.2) is defined as follows

$$\begin{aligned}\mathcal{L}(\rho, w, q, q_{\partial\Omega}) &= \mathcal{J}(\rho, w) - \int_0^T \int_\Omega \left(\frac{\partial \rho}{\partial t} + \nabla \cdot \mathcal{D}(\rho) - \mathcal{C}_s(w) - f \right) q d\vec{x} dt - \int_0^T \int_{\partial\Omega} (\rho - g) q_{\partial\Omega} d\vec{x} dt \\ &= \frac{1}{2} \int_0^T \int_\Omega (\rho - \widehat{\rho})^2 d\vec{x} dt + \frac{\beta}{2} \int_0^T \int_\Omega w^2 d\vec{x} dt, \\ &\quad - \int_0^T \int_\Omega \left(\frac{\partial \rho}{\partial t} - \nabla^2 \rho - \nabla \cdot (\rho \nabla V_1) + \nabla \cdot (\rho \vec{f}) - w - f \right) q d\vec{x} dt \\ &\quad - \int_0^T \int_{\partial\Omega} (\rho - g) q_{\partial\Omega} d\vec{x} dt.\end{aligned}$$

Here, q and $q_{\partial\Omega}$ are the Lagrange multipliers corresponding to the interior and boundary of the domain Ω, respectively.

As discussed in Section 4.1, we can formally derive the first-order optimality conditions by taking directional derivatives of the Lagrangian with respect to each of the variables, and evaluating them at the steady state $(\bar{\rho}, \bar{w})$. Since $\bar{\rho}$ is a steady state of the optimal control problem by assumption, we have that the derivative with respect to ρ satisfies

$$\mathcal{L}_\rho(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})(\rho - \bar{\rho}) \geq 0,$$

for all admissible ρ. Following the argument in [86, Chapter 3], we define $h := \rho - \bar{\rho}$. It becomes apparent that $h(0, \vec{x}) = \rho_0(\vec{x}) - \rho_0(\vec{x}) = 0$ and $\mathcal{L}_\rho(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})h \geq 0$ for all h such that $h(0, \vec{x}) = 0$. However, $-h$ is also such a function, and so, necessarily, we obtain

$$\mathcal{L}_\rho(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})h = 0.$$

Given that we do not have any control constraints, we can use a similar argument to derive that

$$\mathcal{L}_w(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})h = 0.$$

The derivative of $\mathcal{L}$ with respect to ρ for test functions h is

$$\begin{aligned}\mathcal{L}_\rho(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})h = \int_0^T \int_\Omega \left[(\bar{\rho} - \widehat{\rho})h - \frac{\partial h}{\partial t} q + (\nabla^2 h) q + \nabla \cdot (h \nabla V_1) q - \nabla \cdot (h \vec{f}) q \right] d\vec{x} dt \\ - \int_0^T \int_{\partial\Omega} h q_{\partial\Omega} d\vec{x} dt.\end{aligned}$$

The aim is to integrate by parts each term in which a derivative of the test function h is taken. Doing this, the term involving the time derivative becomes

$$\int_0^T \int_\Omega \frac{\partial h}{\partial t} q d\vec{x} dt = \int_\Omega \left[h(T, \vec{x}) q(T, \vec{x}) - h(0, \vec{x}) q(0, \vec{x}) \right] d\vec{x} - \int_0^T \int_\Omega \frac{\partial q}{\partial t} h d\vec{x} dt.$$

Similarly, the diffusion term becomes

$$\begin{aligned}\int_0^T \int_\Omega (\nabla^2 h) q d\vec{x} dt &= \int_0^T \int_{\partial\Omega} q \nabla h \cdot \vec{n} d\vec{x} dt - \int_0^T \int_{\partial\Omega} \nabla h \cdot \nabla q d\vec{x} dt \\ &= \int_0^T \int_{\partial\Omega} (q \nabla h - h \nabla q) \cdot \vec{n} d\vec{x} dt + \int_0^T \int_{\partial\Omega} h \nabla^2 q d\vec{x} dt,\end{aligned}$$

by applying the Divergence Theorem. Finally,

$$\int_0^T \int_\Omega \left[\nabla \cdot (h\nabla V_1)q - \nabla \cdot (h\vec{f})q\right] d\vec{x}dt = \int_0^T \int_{\partial\Omega} h(q\nabla V_1 - q\vec{f}) \cdot \vec{n} d\vec{x}dt - \int_0^T \int_\Omega h(\nabla V_1 \cdot \nabla q - \vec{f} \cdot \nabla q) d\vec{x}dt.$$

Grouping all of these terms together results in

$$\begin{aligned}\mathcal{L}_\rho(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})h = &- \int_\Omega \left[h(T,\vec{x})q(T,\vec{x}) - h(0,\vec{x})q(0,\vec{x})\right] d\vec{x} \\ &+ \int_0^T \int_\Omega \left(\bar{\rho} - \widehat{\rho} + \frac{\partial q}{\partial t} + \nabla^2 q - \nabla V_1 \cdot \nabla q + \vec{f} \cdot \nabla q\right) h d\vec{x}dt \\ &+ \int_0^T \int_{\partial\Omega} \left[q\frac{\partial h}{\partial n} - \left(\nabla q - q\nabla V_1 + q\vec{f}\right) \cdot \vec{n}h - q_{\partial\Omega}h\right] d\vec{x}dt.\end{aligned}$$

We know that $h(0,\vec{x}) = 0$ and set $\mathcal{L}_\rho(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})h = 0$ to obtain

$$\begin{aligned}0 = &- \int_\Omega h(T,\vec{x})q(T,\vec{x})d\vec{x} + \int_0^T \int_\Omega \left(\bar{\rho} - \widehat{\rho} + \frac{\partial q}{\partial t} + \nabla^2 q - \nabla V_1 \cdot \nabla q + \vec{f} \cdot \nabla q\right) h d\vec{x}dt \\ &+ \int_0^T \int_{\partial\Omega} \left[q\frac{\partial h}{\partial n} - \left(\nabla q - q\nabla V_1 + q\vec{f}\right) \cdot \vec{n}h - q_{\partial\Omega}h\right] d\vec{x}dt.\end{aligned} \tag{5.11}$$

We can now restrict the choice of h as much as possible as follows: choose $h \in C_0^\infty((0,T) \times \Omega)$, so that $h(T,\vec{x}) = 0$ in Ω, $h = 0$, $\frac{\partial h}{\partial n} = 0$ on $(0,T) \times \partial\Omega$. Then the remaining part of the Lagrangian derivative is

$$0 = \int_0^T \int_\Omega \left(\bar{\rho} - \widehat{\rho} + \frac{\partial q}{\partial t} + \nabla^2 q - \nabla V_1 \cdot \nabla q + \vec{f} \cdot \nabla q\right) h d\vec{x}dt. \tag{5.12}$$

Since this equation needs to hold for all permissible h, and since $C_0^\infty((0,T) \times \Omega)$ is dense in $L^2((0,T) \times \Omega)$, we have the adjoint equation

$$\bar{\rho} - \widehat{\rho} + \frac{\partial q}{\partial t} + \nabla^2 q - \nabla V_1 \cdot \nabla q + \vec{f} \cdot \nabla q = 0 \quad \text{in } (0,T) \times \Omega.$$

For notational purposes we denote the part of the equation corresponding to $\mathcal{D}(\rho, \vec{w})$ by $\mathcal{D}^*(q, \vec{w})$ as follows:

$$\mathcal{D}^*(q) = -\nabla^2 q + \nabla V_1 \cdot \nabla q - \vec{f} \cdot \nabla q. \tag{5.13}$$

Loosening the restriction $h(T,\vec{x}) \neq 0$, so that $h \in C^1(\overline{(0,T) \times \Omega})$, with $h = 0$, $\frac{\partial h}{\partial n} = 0$ on $(0,T) \times \partial\Omega$, we have that

$$0 = -\int_\Omega h(T,\vec{x})q(T,\vec{x})d\vec{x},$$

and, again by a density argument, we claim that the set of possible $h(T,\vec{x})$ is a dense subset of $L^2(\Omega)$, so that

$$q(T,\vec{x}) = 0 \quad \text{at } \{t = T\} \times \Omega.$$

Next, we consider $h \in C^1(\overline{(0,T)\times\Omega})$ with $h = 0$, $\frac{\partial h}{\partial n} \neq 0$ on $(0,T)\times\partial\Omega$. We find

$$0 = \int_0^T \int_{\partial\Omega} q \frac{\partial h}{\partial n} d\vec{x}dt, \tag{5.14}$$

and since this holds for all permissible $\frac{\partial h}{\partial n}$ and by density of the space, we obtain

$$q(t,\vec{x}) = 0 \quad \text{on } (0,T)\times\partial\Omega.$$

Lifting the final restriction that $h = 0$ on $(0,T)\times\partial\Omega$, we find

$$0 = \int_0^T \int_{\partial\Omega} \left[-\left(\nabla q - q\nabla V_1 + q\vec{f}\right)\cdot\vec{n}h - q_{\partial\Omega}h \right] d\vec{x}dt.$$

However, since $q = 0$ on $\partial\Omega$, this reduces to

$$0 = \int_0^T \int_{\partial\Omega} -\left(\nabla q \cdot \vec{n} + q_{\partial\Omega}\right) h d\vec{x}dt,$$

and, since this holds for all permissible h, we find that

$$q_{\partial\Omega} = -\frac{\partial q}{\partial n} \quad \text{on } (0,T)\times\partial\Omega.$$

We are then able to rewrite the adjoint equation in terms of a single Lagrange multiplier q.

Next, the gradient equation is derived by taking the derivative of the Lagrangian with respect to w and setting it to zero. We obtain

$$\mathcal{L}_w(\bar{\rho},\bar{w},q,q_{\partial\Omega})h = \int_0^T \int_\Omega \left[\beta w h + qh\right] d\vec{x}dt = 0.$$

Since this holds for all $h \in L^2((0,T)\times\Omega)$, the resulting gradient equation reads

$$\beta w + q = 0.$$

Finally, taking the derivative of $\mathcal{L}$ with respect to q recovers the state equation. Therefore, the first-order optimality system for the source control problem with an advection–diffusion PDE constraint is

$$\begin{cases} \text{State equation} \\ \quad \frac{\partial\rho}{\partial t} = -\nabla\cdot\mathcal{D}(\rho) + \mathcal{C}_s(w) + f & \text{in } (0,T)\times\Omega, \\ \quad \rho = g & \text{on } (0,T)\times\partial\Omega, \\ \rho(0,\vec{x}) = \rho_0(\vec{x}) & \text{at } \{t=0\}\times\Omega, \\ \\ \text{Adjoint equation} \\ \quad \frac{\partial q}{\partial t} = \mathcal{D}^*(q) - \rho + \widehat{\rho} & \text{in } (0,T)\times\Omega, \\ \quad q = 0 & \text{on } (0,T)\times\partial\Omega, \\ q(T,\vec{x}) = 0 & \text{at } \{t=T\}\times\Omega, \\ \\ \text{Gradient/control equation} \\ \quad w = -\frac{1}{\beta}q & \text{in } (0,T)\times\Omega, \end{cases} \tag{5.15}$$

with the operators defined by (5.3), (5.5), and (5.13).

Including the Integral (Interaction) Term

We would like to include the particle interaction term into the optimal control problem. Since the terms in the Lagrangian are additive, we can simply treat the integral term separately and add it into the optimality system for the advection–diffusion equation. The full Lagrangian is

$$\mathcal{L}(\rho, w, q, q_{\partial\Omega}) = \mathcal{J}(\rho, w) - \int_0^T \int_\Omega \left(\frac{\partial \rho}{\partial t} + \nabla \cdot \mathcal{D}(\rho) + \nabla \cdot \mathcal{I}(\rho) - \mathcal{D}_s(w) \right) q d\vec{x} dt \\ - \int_0^T \int_{\partial\Omega} (\rho - g) q_{\partial\Omega} d\vec{x} dt.$$

Hence, we only need to consider the derivative of the part of the Lagrangian involving the new term, i.e.,

$$T(\rho, q) := - \int_0^T \int_\Omega \nabla \cdot \mathcal{I}(\rho) q d\vec{x} dt = \int_0^T \int_\Omega \left[\nabla \cdot \int_\Omega \rho(t, \vec{x}) \rho(t, \vec{x}') \nabla V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \right] q d\vec{x} dt.$$

Using the product rule, we have

$$T_\rho(\rho, q)h = \int_0^T \int_\Omega q(t, \vec{x}) \left(\nabla \cdot \int_\Omega h(t, \vec{x}) \rho(t, \vec{x}') \nabla V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \right) d\vec{x} dt \\ + \int_0^T \int_\Omega q(t, \vec{x}) \left(\nabla \cdot \int_\Omega \rho(t, \vec{x}) h(t, \vec{x}') \nabla V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \right) d\vec{x} dt.$$

Integrating by parts results in

$$T_\rho(\rho, q)h = \int_0^T \int_{\partial\Omega} q(t, \vec{x}) \left(\int_\Omega (h(t, \vec{x}) \rho(t, \vec{x}') + \rho(t, \vec{x}) h(t, \vec{x}')) \frac{\partial V_2(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' \right) d\vec{x} dt \\ - \int_0^T \int_\Omega \nabla q(t, \vec{x}) \cdot \left(\int_\Omega h(t, \vec{x}) \rho(t, \vec{x}') \nabla V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \right) d\vec{x} dt \\ - \int_0^T \int_\Omega \nabla q(t, \vec{x}) \cdot \left(\int_\Omega \rho(t, \vec{x}) h(t, \vec{x}') \nabla V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \right) d\vec{x} dt.$$

The inner and outer integrals in the second term can be swapped, and, since the particles are indistinguishable, $\vec{x}$ can be relabelled to $\vec{x}'$ (and vice versa) to obtain

$$T_\rho(\rho, q)h = \int_0^T \int_{\partial\Omega} q(t, \vec{x}) \left(\int_\Omega (h(t, \vec{x}) \rho(t, \vec{x}') + \rho(t, \vec{x}) h(t, \vec{x}')) \frac{\partial V_2(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' \right) d\vec{x} dt \\ - \int_0^T \int_\Omega h(t, \vec{x}) \left(\nabla_{\vec{x}} q(t, \vec{x}) \cdot \int_\Omega \rho(t, \vec{x}') \nabla V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \right) d\vec{x} dt \\ - \int_0^T \int_\Omega h(t, \vec{x}) \left(\int_\Omega \rho(t, \vec{x}') \nabla_{\vec{x}'} q(t, \vec{x}') \cdot \nabla V_2(|\vec{x}' - \vec{x}|) d\vec{x}' \right) d\vec{x} dt.$$

This effects h to be outside of the inner integral. The notation $\nabla_{\vec{x}'}$ denotes the gradient with respect to $\vec{x}'$. Finally, since $V_2(|x|)$ is an even function by definition, ∇V_2 is an odd function, and so $\nabla V_2(|\vec{x}' - \vec{x}|) = -\nabla V_2(|\vec{x} - \vec{x}'|)$. Combining these observations, we obtain

$$T_\rho(\rho, q)h = \int_0^T \int_{\partial\Omega} q \left(\int_\Omega (h(t, \vec{x}) \rho(t, \vec{x}') + \rho(t, \vec{x}) h(t, \vec{x}')) \frac{\partial V_2(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' \right) d\vec{x} dt \\ - \int_0^T \int_\Omega h \left[\int_\Omega \rho(t, \vec{x}') \left(\nabla_{\vec{x}} q(t, \vec{x}) - \nabla_{\vec{x}'} q(t, \vec{x}') \right) \cdot \nabla V_2(|\vec{x}' - \vec{x}|) d\vec{x}' \right] d\vec{x} dt \tag{5.16}$$

$$= \int_0^T \int_{\partial\Omega} q \left(\int_\Omega (h(t,\vec{x})\rho(t,\vec{x}') + \rho(t,\vec{x})h(t,\vec{x}')) \frac{\partial V_2(|\vec{x}-\vec{x}'|)}{\partial n} d\vec{x}' \right) d\vec{x}dt - \int_0^T \int_\Omega h\mathcal{I}^*(q,\rho) d\vec{x}dt,$$

where

$$\mathcal{I}^*(q,\rho) := \int_\Omega \rho(t,\vec{x}') \left(\nabla_{\vec{x}} q(t,\vec{x}) - \nabla_{\vec{x}'} q(t,\vec{x}') \right) \cdot \nabla V_2(|\vec{x}-\vec{x}'|) d\vec{x}'. \tag{5.17}$$

This term can be added to the derivation of the adjoint equation from (5.12) so that

$$0 = \int_0^T \int_\Omega \left(\bar{\rho} - \widehat{\rho} + \frac{\partial q}{\partial t} - \mathcal{D}^*(q,\rho) - \mathcal{I}^*(q,\rho) \right) h d\vec{x}dt.$$

Therefore, since this holds for all $h \in C_0^\infty((0,T)\times\Omega)$, we obtain the modified adjoint equation

$$\frac{\partial q}{\partial t} = \mathcal{D}^*(q) + \mathcal{I}^*(q,\rho) - \rho + \widehat{\rho} \quad \text{ in } (0,T)\times\Omega.$$

The boundary terms in $T_\rho(\rho,q)h$ vanish, since in step (5.14) of the derivation we established that $q = 0$ on $\partial\Omega$, and therefore the boundary condition of the adjoint equation remains unaffected.

The modified optimality system for the interacting PDE-constrained optimization problem with source control and Dirichlet boundary conditions is

$$\begin{cases} \text{State equation} \\ \quad \frac{\partial \rho}{\partial t} = -\nabla \cdot \left(\mathcal{D}(\rho) + \mathcal{I}(\rho) \right) + \mathcal{C}_s(w) + f & \text{in } (0,T)\times\Omega, \\ \quad \rho = g & \text{on } (0,T)\times\partial\Omega, \\ \rho(0,\vec{x}) = \rho_0(\vec{x}) & \text{at } \{t=0\}\times\Omega, \\ \\ \text{Adjoint equation} \\ \quad \frac{\partial q}{\partial t} = \mathcal{D}^*(q) + \mathcal{I}^*(q,\rho) - \rho + \widehat{\rho} & \text{in } (0,T)\times\Omega, \\ \quad q = 0 & \text{on } (0,T)\times\partial\Omega, \\ q(T,\vec{x}) = 0 & \text{at } \{t=T\}\times\Omega, \\ \\ \text{Gradient/control equation} \\ \quad w = -\frac{1}{\beta} q & \text{in } (0,T)\times\Omega, \end{cases}$$

where the operators are defined by (5.3), (5.4), (5.5), (5.13), and (5.17).

5.2.2 Flow Control

This approach can be modified to derive the optimality system for the flow control problem (5.8) with Dirichlet boundary conditions. The Lagrangian for this problem is

$$\mathcal{L}(\rho,\vec{w},q,q_{\partial\Omega}) = \mathcal{J}(\rho,\vec{w}) - \int_0^T \int_\Omega \left(\frac{\partial \rho}{\partial t} + \nabla \cdot \left(\mathcal{D}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho,\vec{w}) \right) - f \right) q d\vec{x}dt$$
$$- \int_0^T \int_{\partial\Omega} (\rho - g) q_{\partial\Omega} d\vec{x}dt,$$

with $\mathcal{C}(\rho,\vec{w}) = \rho\vec{w}$. The derivative with respect to ρ is the same as in the source control case, with $\vec{f} := \vec{f} + \vec{w}$. Therefore we define

$$\mathcal{C}^*(q,\vec{w}) := -\vec{w} \cdot \nabla q, \tag{5.18}$$

as the part of the adjoint equation corresponding to the control term $C(\rho, \vec{w})$. The only part of the optimality system that changes fundamentally is the gradient equation, since now the derivative with respect to $\vec{w}$ instead of w is taken. Evaluating $\mathcal{L}_{\vec{w}}(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})\vec{h} = \vec{0}$ results in

$$\mathcal{L}_{\vec{w}}(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})\vec{h} = \int_0^T \int_\Omega \left[\beta\vec{w} \cdot \vec{h} - \nabla \cdot (\rho\vec{h})q\right] d\vec{x}dt = \vec{0}.$$

Applying the Divergence Theorem, and noting that $q = 0$ on $(0, T) \times \partial\Omega$, we obtain

$$\mathcal{L}_{\vec{w}}(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})\vec{h} = \int_0^T \int_\Omega (\beta\vec{w} + \rho\nabla q) \cdot \vec{h} d\vec{x}dt = \vec{0}.$$

Since this holds for all permissible $\vec{h} \in L^2((0, T) \times \Omega)^2$, the gradient equation for a flow control problem is

$$\vec{w} = -\frac{1}{\beta}\rho\nabla q \quad \text{in } (0, T) \times \Omega.$$

The resulting full optimality system is

$$\begin{cases} \text{State equation} & \\ \quad \frac{\partial\rho}{\partial t} = -\nabla \cdot \big(\mathcal{D}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho, \vec{w})\big) + f & \text{in } (0, T) \times \Omega, \\ \quad \rho = g & \text{on } (0, T) \times \Omega, \\ \rho(0, \vec{x}) = \rho_0(\vec{x}) & \text{at } \{t = 0\} \times \partial\Omega, \\ & \\ \text{Adjoint equation} & \\ \quad \frac{\partial q}{\partial t} = \mathcal{D}^*(q) + \mathcal{I}^*(q, \rho) + \mathcal{C}^*(q, \vec{w}) - \rho + \widehat{\rho} & \text{in } (0, T) \times \Omega, \\ \quad q = 0 & \text{on } (0, T) \times \partial\Omega, \\ q(T, \vec{x}) = 0 & \text{at } \{t = T\} \times \Omega, \\ & \\ \text{Gradient/control equation} & \\ \quad \vec{w} = -\frac{1}{\beta}\rho\nabla q & \text{in } (0, T) \times \Omega, \end{cases} \tag{5.19}$$

where (5.3), (5.4), (5.9), (5.13), (5.17) and (5.18) define the operators in the system.

No-Flux Boundary Conditions

Finally, we would like to add no-flux boundary conditions (5.10) to both the flow and source control problems. Starting with flow control, the modified Lagrangian is

$$\begin{aligned} \mathcal{L}(\rho, \vec{w}, q, q_{\partial\Omega}) = \mathcal{J}(\rho, \vec{w}) &- \int_0^T \int_\Omega \left(\frac{\partial\rho}{\partial t} + \nabla \cdot \big(\mathcal{D}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho, \vec{w})\big) - f\right) q d\vec{x}dt \\ &- \int_0^T \int_{\partial\Omega} \big(\mathcal{D}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho, \vec{w})\big) \cdot \vec{n} q_{\partial\Omega} d\vec{x}dt. \end{aligned}$$

The derivative of $\mathcal{L}$ with respect to ρ is taken and all boundary terms of $\mathcal{L}_\rho$ that arise from the integration by parts of the interior terms in (5.11) (splitting $\vec{f}$ as $\vec{f} =: \vec{f} + \vec{w}$) and (5.16) are collected, and added to the terms stemming from the no-flux boundary condition. Neglecting interior terms, since they do not change when the boundary condition is changed, the portion of $\mathcal{L}_\rho(\bar{\rho}, \bar{\vec{w}}, q, q_{\partial\Omega})h$ of interest, arising in the boundary $\partial\Omega$, is

$$\begin{aligned} \mathcal{B}_\rho(\bar{\rho}, \bar{w}, q, q_{\partial\Omega})h := &\int_0^T \int_{\partial\Omega} \left(\nabla h + h\nabla V_1 - h\vec{f} - h\vec{w}\right) \cdot \vec{n} q_{\partial\Omega} d\vec{x}dt \\ &+ \int_0^T \int_{\partial\Omega} \left(\int_0^T \int_\Omega \big(h(t, \vec{x})\rho(t, \vec{x}') + \rho(t, \vec{x})h(t, \vec{x}')\big) \nabla V_2(|\vec{x} - \vec{x}'|)\right) \cdot \vec{n} q_{\partial\Omega} d\vec{x}dt \end{aligned}$$

$$+\int_0^T\int_{\partial\Omega} q\frac{\partial h}{\partial n} - \left(\nabla q - q\nabla V_1 + q\vec{f} + q\vec{w}\right)\cdot\vec{n}h d\vec{x}dt$$
$$+\int_0^T\int_{\partial\Omega} q\left(\int_\Omega (h(t,\vec{x})\rho(t,\vec{x}') + \rho(t,\vec{x})h(t,\vec{x}'))\frac{\partial V_2(|\vec{x}-\vec{x}'|)}{\partial n}d\vec{x}'\right)d\vec{x}dt.$$

Regrouping the terms, we obtain

$$\mathcal{B}_\rho(\bar\rho,\bar w,q,q_{\partial\Omega}) = \int_0^T\int_{\partial\Omega}(q+q_{\partial\Omega})\frac{\partial h}{\partial n}d\vec{x}dt + \int_0^T\int_{\partial\Omega}\left[-\frac{\partial q}{\partial n}h + (q+q_{\partial\Omega})\left(\nabla V_1 - \vec{f} - \vec{w}\right)\cdot nh\right]d\vec{x}dt$$
$$+\int_0^T\int_{\partial\Omega}(q+q_{\partial\Omega})\left(\int_\Omega (h(t,\vec{x})\rho(t,\vec{x}') + \rho(t,\vec{x})h(t,\vec{x}'))\frac{\partial V_2(|\vec{x}-\vec{x}'|)}{\partial n}d\vec{x}'\right)d\vec{x}dt.$$

Equivalently to step (5.14) in the previous derivation, we restrict our attention to $h \in C^1(\overline{(0,T)\times\Omega})$, such that $\frac{\partial h}{\partial n} \neq 0$ and $h = 0$ on $(0,T)\times\partial\Omega$. Then

$$\mathcal{B}_\rho(\bar\rho,\bar w,q,q_{\partial\Omega}) = \int_0^T\int_{\partial\Omega}(q+q_{\partial\Omega})\frac{\partial h}{\partial n}d\vec{x}dt = 0,$$

and since this has to hold for all feasible test functions, we deduce $q = -q_{\partial\Omega}$. Lifting all restrictions on h, so that $h \neq 0$ on $(0,T)\times\partial\Omega$, we obtain

$$\mathcal{B}_\rho(\bar\rho,\bar w,q,q_{\partial\Omega}) = \int_0^T\int_{\partial\Omega}\left[-\frac{\partial q}{\partial n}h + (q+q_{\partial\Omega})\left(\nabla V_1 - \vec{f} - \vec{w}\right)\cdot nh\right]d\vec{x}dt$$
$$+\int_0^T\int_{\partial\Omega}(q+q_{\partial\Omega})\int_\Omega (h(t,\vec{x})\rho(t,\vec{x}') + \rho(t,\vec{x})h(t,\vec{x}'))\frac{\partial V_2(|\vec{x}-\vec{x}'|)}{\partial n}d\vec{x}'d\vec{x}dt = 0,$$

which, by applying $q = -q_{\partial\Omega}$, reduces to

$$\mathcal{B}_\rho(\bar\rho,\bar w,q,q_{\partial\Omega}) = \int_0^T\int_{\partial\Omega} -\frac{\partial q}{\partial n}h d\vec{x}dt = 0. \tag{5.20}$$

Since this holds for all permissible h, we obtain the boundary condition

$$\frac{\partial q}{\partial n} = 0 \quad \text{on } \partial\Omega.$$

The optimality system for a flow control problem with no-flux boundary conditions is

$$\begin{cases}
\text{State equation} & \\
\quad \frac{\partial\rho}{\partial t} = -\nabla\cdot\left(\mathcal{D}(\rho)+\mathcal{I}(\rho)+\mathcal{C}(\rho,\vec{w})\right) + f & \text{in } (0,T)\times\Omega, \\
\quad 0 = \left(\mathcal{D}(\rho)+\mathcal{I}(\rho)+\mathcal{C}(\rho,\vec{w})\right)\cdot\vec{n} & \text{on } (0,T)\times\partial\Omega, \\
\rho(0,\vec{x}) = \rho_0(\vec{x}) & \text{at } \{t=0\}\times\Omega, \\
\\
\text{Adjoint equation} & \\
\quad \frac{\partial q}{\partial t} = \mathcal{D}^*(q)+\mathcal{I}^*(q,\rho)+\mathcal{C}^*(q,\vec{w}) - \rho + \widehat{\rho} & \text{in } (0,T)\times\Omega, \\
\quad \frac{\partial q}{\partial n} = 0 & \text{on } (0,T)\times\partial\Omega, \\
q(T,\vec{x}) = 0 & \text{at } \{t=T\}\times\Omega, \\
\\
\text{Gradient/control equation} & \\
\quad \vec{w} = -\frac{1}{\beta}\rho\nabla q & \text{in } (0,T)\times\Omega,
\end{cases} \tag{5.21}$$

again where the operators are defined in (5.3), (5.4), (5.9), (5.13), (5.17), and (5.18).

Now, considering the source control problem with no-flux boundary conditions, since the source term does not enter the no-flux conditions, we can derive the optimality system for this problem

by only marginally modifying the flow control result. We set the flow control term $\mathcal{C}(\rho, \vec{w}) = 0$ and corresponding adjoint contribution $\mathcal{C}^*(q, \vec{w}) = 0$ in the system above, add the source control term $\mathcal{C}_s(w)$ in the state equation, and exchange the gradient equation from the flow control one to the source control one. Since all parts of the problem are additive, this can be done without concern. We obtain the following optimality system

$$\begin{cases} \text{State equation} \\ \quad \frac{\partial \rho}{\partial t} = -\nabla \cdot \big(\mathcal{D}(\rho) + \mathcal{I}(\rho)\big) + \mathcal{C}(w) + f & \text{in } (0,T) \times \Omega, \\ \quad 0 = \big(\mathcal{D}(\rho) + \mathcal{I}(\rho)\big) \cdot \vec{n} & \text{on } (0,T) \times \partial\Omega, \\ \rho(0, \vec{x}) = \rho_0(\vec{x}) & \text{at } \{t = 0\} \times \Omega, \\ \\ \text{Adjoint equation} \\ \quad \frac{\partial q}{\partial t} = \mathcal{D}^*(q) + \mathcal{I}^*(q, \rho) - \rho + \widehat{\rho} & \text{in } (0,T) \times \Omega, \\ \quad \frac{\partial q}{\partial n} = 0 & \text{on } (0,T) \times \partial\Omega, \\ q(T, \vec{x}) = 0 & \text{at } \{t = T\} \times \Omega, \\ \\ \text{Gradient equation} \\ \quad w = -\frac{1}{\beta} q & \text{in } (0,T) \times \Omega, \end{cases} \tag{5.22}$$

with the operators given by (5.3), (5.4), (5.5), (5.13), and (5.17).

5.3 Survey of Related Work

In comparison to the problems discussed in Chapter 4, the added difficulty is the nonlinear, nonlocal particle interaction term. Therefore, standard results in optimal control theory established in Section 4.1 cannot readily be applied, and new approaches have to be developed to address theoretical and numerical challenges. While this is a new area of research, the work that has been published focuses on two types of models. The most popular model is a deterministic microscopic model, which is a generalization of the well-known Cucker–Smale model, see [158, 159]. In the mean-field limit of the Cucker–Smale model a Vlasov-type PDE arises. Less work has been done on the optimal control of the Fokker–Planck PDE, which arises as the mean-field limit of a stochastic microscopic model.

5.3.1 Theoretical Results

In this section, we will briefly highlight some of the existing theoretical results as well as challenges relating to optimal control problems of similar structure to the ones considered in this thesis. We note that this thesis is not focused on such theoretical results, but it is important that we are aware of such questions and the limited number of results available in this context.

For control problems involving Vlasov-type PDE models, the work by Fornasier et al. provides a range of theoretical results on the convergence of the microscopic optimal control problem to a corresponding macroscopic problem, using methods of optimal transport and a Γ-limit argument, proving existence of optimal controls in the mean-field setting, see [160, 161, 162]. The work focusses on sparse control strategies, where one or more agents influence a larger crowd. Additional work on sparse control strategies can be found in [163], as well as in the review paper [164]. In [165], an alternative method, set in the space of probability measures, is developed, and convergence results are proved. The control in this work is applied through external agents.

For the optimal control of the Fokker–Planck PDE, some theoretical results are known. In [166], the existence of optimal controls for microscopic and macroscopic versions of a class of problems is proved and a rigorous derivation of first-order optimality conditions is given. Following this, [167] discusses the existence and regularity of an optimal control problem of this type on periodic domains, including the well-posedness of the Fokker–Planck equation. In [168]

and [169], the convergence of the microscopic optimal control problem to its mean-field limit for consensus based optimization is proved.

While all of these results are interesting and important contributions to the field, in terms of the model we consider, one of the most relevant papers is [166]. In this work, an optimal control problem is considered that is very closely related to the one studied in this thesis. Moreover, it considers no-flux boundary conditions, as opposed to periodic conditions or domains with compact support. The optimal control problem considered is

$$\min_{\rho,\vec{w}} \mathcal{J}(\rho,\vec{w}) := \frac{1}{2}\int_0^T \int_\Omega |\vec{x}-\widehat{\vec{x}}|^2 \rho d\vec{x}dt + \beta \int_0^T \int_\Omega \Psi(\vec{w})\rho d\vec{x}dt \tag{5.23}$$

subject to

$$\begin{aligned} &\frac{\partial \rho}{\partial t} = \sigma\nabla^2\rho - \nabla\cdot\big(\mathcal{P}[\rho]\rho + \vec{w}\rho\big) && \text{in } [0,T]\times\Omega, \\ &\langle \sigma\nabla\rho - (\mathcal{P}[\rho]+\vec{w})\rho, \vec{n}(\vec{x})\rangle = 0 && \text{on } [0,T]\times\partial\Omega, \\ &\rho(0,\vec{x}) = \rho_0(\vec{x}) && \text{at } \{t=0\}\times\Omega, \end{aligned} \tag{5.24}$$

where Ω is a smooth, open, bounded subset of $\mathbb{R}^n$, $\vec{n}(\vec{x})$ is the outward normal at $\vec{x}\in\partial\Omega$, and $\mathcal{P}$ is defined by

$$\mathcal{P}[\rho] = \int_\Omega P(\vec{x},\vec{x}')(\vec{x}'-\vec{x})\rho(t,\vec{x}')d\vec{x}'.$$

The set of admissible controls is defined by

$$U_{\text{ad}} = \{\|\vec{w}\|_{L^2(0,T;L^\infty(\Omega))} \le M : \vec{w}\in L^2(0,T;L^\infty(\Omega))\},$$

for a given $M>0$. The differences to our problem are as follows. Instead driving ρ to $\widehat{\rho}$ in the cost functional, a target position $\widehat{\vec{x}}$ is aimed for. Moreover, instead of considering control through the advective field only, the paper considers the control through a function Ψ of the advective field. Setting $\Psi(\vec{w}) := \frac{1}{2}\|\vec{w}\|^2$ and removing ρ from the cost functional term recovers how the control is applied in our problem. Finally, the choice of interaction kernel is different from the one in our work, however, it could, in principle be incorporated into our framework.

With this setup, the authors of [166] show that there exists a unique weak solution to the forward problem, i.e., the PDE constraint, and that there exists an optimal control to the PDE-constrained optimization problem. They show that for $\rho_0\in L^2(\Omega)$ and $P\in L^\infty(\Omega^2)$, there exists a unique weak solution ρ to (5.24). Then, an optimal ρ^∞ and $\vec{w}^\infty\in U_{\text{ad}}$ exist, provided that Ψ satisfies a suitable growth condition. Moreover, the paper provides a rigorous derivation of the first-order optimality system. While [166] does not treat exactly the model problem considered in this thesis, the theoretical results provided by the authors give some theoretical justification for the mostly formal and numerical work we have done.

In [167], instead of considering no-flux boundary conditions, the problem is posed on a d-dimensional torus $\mathbb{T}^d$. Again, the cost functional is slightly different from the one considered in this thesis, and more general than the one considered in [166]. It reads

$$\inf_{\rho,\vec{w}} \int_0^T \int_{\mathbb{T}^d} \left(L(\vec{x},\rho) + \frac{|\vec{w}|^2}{2}\right)\rho d\vec{x}dt + \int_{\mathbb{T}^d} \psi_T(\vec{x})\rho(T,\vec{x})d\vec{x},$$

where ψ_T is a continuous function and L given. However, the PDE model is the same as in [166] and as in our work and, most importantly, the interaction kernel is of the form $\nabla V_2(\vec{x}-\vec{x}')$, which is very close to the one used in this thesis, making the results in [167] highly relevant to our work. The main result in [167] shows the existence of solutions to the first-order optimality system of the mean field optimal control problem. This can be shown under the assumption that $V_2\in W^{2,\infty}(\mathbb{T}^d)$ and some regularity assumptions on the derivative of L appearing in the

adjoint equation.

While the results in these papers are promising, they are derived under certain assumptions and have to be treated with care. One underlying condition for these results to hold is the unique solution to the PDE model. However, as demonstrated in [170], this is not always the case. The authors show that the uniform distribution, which is a steady state for the homogeneous problem on the torus, is not the unique steady state solution if the interaction potential V_2 has negative Fourier modes. Then bifurcations can occur. This is an example of when the forward problem does not have a unique solution, which then also propagates into the analysis of optimal controls.

5.3.2 Numerical Methods

Numerical advances to solve the Vlasov-type problem have been made in [171, 172], where sparse and other control strategies through the external agents are considered. In both papers a Strang-splitting scheme, [173], is applied to solve the optimal control problem. The numerical results verify the convergence of the microscopic control problem to its mean-field limit. Furthermore, in [174], different selective control strategies are considered, and an iterative numerical method is chosen, involving a transport and an interaction step, where the interaction term is approximated stochastically.

Numerical results for the Fokker–Planck model include those presented in [168], where a Strang-splitting scheme, [175], is applied, and in which convergence to the mean field optimal control problem is shown numerically. Furthermore, in [166], an optimal control hierarchy, including instantaneous and Boltzmann-type controls, is proposed. The mean-field first-order optimality system in [166] is solved using a Chang–Cooper scheme for the forward equation, and finite differences for the adjoint equation, while approximating the integrals using a Monte-Carlo scheme. This is coupled by a sweeping algorithm, where updates are made through the gradient equation. Some numerical results on a porous media version of the Fokker–Planck equation are presented in [169]. In [176, 177], steady-state solutions to a Fokker–Planck-type PDE are considered, however, the main focus are Boltzmann-type approaches to solving the optimal control problem.

The most common control types in the literature are flow control, e.g., [166], control through the interaction term, e.g., [161], as well as control through external agents, e.g., [172]. Most papers do not consider boundary conditions, because it is assumed that the particle distribution is of compact support, see [162, 165, 172]. No-flux boundary conditions, which are of high relevance in applications, are not often found in the literature, but are considered in [166, 169]. Our work considers the mean-field equation of Fokker–Planck type, flow control (or control through a source term) and no-flux (or Dirichlet) boundary conditions, in order to address a broad range of test problems and real world applications.

As described above, some numerical methods have been developed for solving optimal control problems involving nonlocal, nonlinear PDEs. Most of these papers, however, focus on other methods and use the mean-field optimal control as verification tool, see [166, 168]. It takes large computational effort to solve these problems, which increases with dimensionality, see [171, 172]. By contrast, the work in this thesis aims to solve such problems using a fast and robust numerical method, which is introduced in the next section.

5.4 A Newton–Krylov Algorithm

In this section, a numerical method for solving the derived first-order optimality system is discussed. Note that solving optimality systems of the form introduced in Section 5.2 is highly non-trivial. Considering the optimality system (5.21) for illustration, it is evident that the state equation has an initial condition, while the adjoint equation is equipped with a final-time condition instead. Moreover, $\mathcal{D}$ and $\mathcal{D}^*$ are defined so that the state and adjoint equations

contain Laplacian terms of opposite signs. These two restrictions imply that one cannot solve this coupled system of equations, once discretized in space, by applying a simple time-stepping algorithm to the whole system. The reason is that the adjoint equation solved 'forward' in time, i.e., from $t = 0$ to $t = T$, would be numerically unstable, due to the change in sign of the Laplacian. The same issue arises when trying to solve the state equation 'backward' in time. Different methods are available to circumvent these issues, such as fixed-point methods, which were applied in, e.g., [1, 166, 178], and the Newton–Krylov scheme presented in this section.

The algorithm is based on the Newton–Krylov method introduced in [179], which is a method for solving PDE-constrained optimization problems with nonlinear PDE constraints and linearly applied control. In this thesis, it is extended to also encompass bilinear control strategies through the advective field (i.e., flow control), as well as optimality systems with nonlocal PDEs arising from particle dynamics problems. In line with the solution of forward problems discussed in Section 3.3, we use a pseudospectral discretization both in space and time. While the Newton–Krylov algorithm developed in [179] has been designed to tackle a range of nonlinear problems, it had not been optimized for the resolution of nonlinearities occurring in combination with derivative and integral terms, such as advective fields and convolution integrals, which are part of the models in this thesis. While it is not evident in advance that this algorithm performs well for the considerably harder nonlinear, nonlocal problems addressed in this thesis, it is demonstrated in the next sections that it is indeed able to solve such problems.

In order to initialize the Newton–Krylov algorithm, irrespective of whether a source or flow control problem is solved and which boundary conditions are used, the first-order optimality system is reduced from three equations to a coupled system of PDEs. Since the control variable $\vec{w}$ (flow control) or w (source control) is given explicitly by the gradient equation, one can substitute the gradient equation into the state and (for flow control) adjoint equations. For example, in the case of the flow control problem with no-flux boundary conditions, the optimality system (5.21) reduces to

$$
\begin{cases}
\text{State equation} & \\
\quad \frac{\partial \rho}{\partial t} = -\nabla \cdot \left(\mathcal{D}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho, -\frac{1}{\beta}\rho\nabla q) \right) + f & \text{in } (0,T) \times \Omega, \\
\quad 0 = \left(\mathcal{D}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho, -\frac{1}{\beta}\rho\nabla q) \right) \cdot \vec{n} & \text{on } (0,T) \times \partial\Omega, \\
\rho(0,\vec{x}) = \rho_0(\vec{x}) & \text{at } \{t=0\} \times \Omega, \\
 & \\
\text{Adjoint equation} & \\
\quad \frac{\partial q}{\partial t} = \mathcal{D}^*(q) + \mathcal{I}^*(q,\rho) + \mathcal{C}^*(q, -\frac{1}{\beta}\rho\nabla q) - \rho + \widehat{\rho} & \text{in } (0,T) \times \Omega, \\
\quad \frac{\partial q}{\partial n} = 0 & \text{on } (0,T) \times \partial\Omega, \\
q(T,\vec{x}) = 0 & \text{at } \{t=T\} \times \Omega.
\end{cases}
$$

All spatial operators are discretized using pseudospectral methods, as demonstrated in Section 3.3. Here, we consider a one-dimensional domain Ω, so that the number of discretization points is N. However, generalizations to higher dimensions follow without requiring additional work. Defining a diagonal mass matrix M, where a zero diagonal entry denotes a boundary term and a one denotes an interior term, the resulting two systems of ODEs and algebraic equations for the discretized state and adjoint variables $\boldsymbol{\rho}$ and $\boldsymbol{q}$ are

$$
\begin{aligned}
M\boldsymbol{\rho}'(t) &= \boldsymbol{f}(t, \boldsymbol{\rho}, \boldsymbol{q}), \quad M\boldsymbol{\rho}(0) = M\boldsymbol{\rho}_0, \\
M\boldsymbol{q}'(t) &= \boldsymbol{g}(t, \boldsymbol{\rho}, \boldsymbol{q}), \quad M\boldsymbol{q}(T) = M\boldsymbol{q}_T = \boldsymbol{0}.
\end{aligned}
$$

Here, $\boldsymbol{f}$ corresponds to the discretized right-hand side of the state equation, including boundary conditions. The first row of $\boldsymbol{f}$ corresponds to the initial condition for ρ. Similarly, $\boldsymbol{g}$ corresponds to the discretized adjoint equation, with the last row corresponding to the final-time condition $q(T,\vec{x}) = 0$. The vector quantities $\boldsymbol{\rho}$, $\boldsymbol{q}$, $\boldsymbol{f}$, and $\boldsymbol{g}$, evaluated at a time point t, are of size N,

where N is the total number of spatial discretization points. The matrix M has dimensions $N \times N$.

Pseudospectral interpolation in time can be defined by

$$\widetilde{\boldsymbol{\rho}}(t) = \sum_{j=0}^{n} \ell_j(t)\boldsymbol{\rho}(t_j) \quad \text{and} \quad \widetilde{\boldsymbol{q}}(t) = \sum_{j=0}^{n} \ell_j(t)\boldsymbol{q}(t_j),$$

where $0 = t_0 < t_1 < \cdots < t_n = T$ are $n+1$ Chebyshev collocation points, and $\ell_j(t)$ are Lagrange functions, with $\ell_j(t_i) = \delta_{ij}$. This enables the definition of the residual functions

$$\begin{aligned}
\boldsymbol{r}_{\boldsymbol{\rho}}(t) &:= \int_0^t \boldsymbol{f}(\tau, \widetilde{\boldsymbol{\rho}}(\tau), \widetilde{\boldsymbol{q}}(\tau))\ \mathrm{d}\tau - M\widetilde{\boldsymbol{\rho}}(t) + M\widetilde{\boldsymbol{\rho}}(0), \\
\boldsymbol{r}_{\boldsymbol{q}}(t) &:= \int_0^t \boldsymbol{g}(\tau, \widetilde{\boldsymbol{\rho}}(\tau), \widetilde{\boldsymbol{q}}(\tau))\ \mathrm{d}\tau - M\widetilde{\boldsymbol{q}}(t) + M\widetilde{\boldsymbol{q}}(0).
\end{aligned}$$

The residuals are discretized in time, resulting in a matrix–vector system of the form

$$\begin{aligned}
[\boldsymbol{r}_{\boldsymbol{\rho},0}, \boldsymbol{r}_{\boldsymbol{\rho},1}, \ldots, \boldsymbol{r}_{\boldsymbol{\rho},n}] &= [\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_n]Q + M[\widetilde{\boldsymbol{\rho}}_0 - \widetilde{\boldsymbol{\rho}}_0, \widetilde{\boldsymbol{\rho}}_0 - \widetilde{\boldsymbol{\rho}}_1, \ldots, \widetilde{\boldsymbol{\rho}}_0 - \widetilde{\boldsymbol{\rho}}_n], \\
[\boldsymbol{r}_{\boldsymbol{q},0}, \boldsymbol{r}_{\boldsymbol{q},1}, \ldots, \boldsymbol{r}_{\boldsymbol{q},n}] &= [\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_n]Q + M[\widetilde{\boldsymbol{q}}_0 - \widetilde{\boldsymbol{q}}_0, \widetilde{\boldsymbol{q}}_0 - \widetilde{\boldsymbol{q}}_1, \ldots, \widetilde{\boldsymbol{q}}_0 - \widetilde{\boldsymbol{q}}_n],
\end{aligned}$$

where $Q \in \mathbb{R}^{(n+1)\times(n+1)}$ is such that $q_{ij} = \int_0^{t_j} \ell_i(\tau)\ \mathrm{d}\tau$. We apply the convention that entries run from 0 to n. Then, a *global residual function* can be formed

$$\boldsymbol{R}: \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{\rho}}_0 \\ \widetilde{\boldsymbol{q}}_0 \\ \widetilde{\boldsymbol{\rho}}_1 \\ \widetilde{\boldsymbol{q}}_1 \\ \widetilde{\boldsymbol{\rho}}_2 \\ \widetilde{\boldsymbol{q}}_2 \\ \vdots \\ \widetilde{\boldsymbol{\rho}}_n \\ \widetilde{\boldsymbol{q}}_n \end{bmatrix}}_{=:\boldsymbol{y}} \mapsto \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \sum_{i=0}^n q_{i1}\boldsymbol{f}_i \\ \sum_{i=0}^n q_{i1}\boldsymbol{g}_i \\ \sum_{i=0}^n q_{i2}\boldsymbol{f}_i \\ \sum_{i=0}^n q_{i2}\boldsymbol{g}_i \\ \vdots \\ \sum_{i=0}^n q_{in}\boldsymbol{f}_i \\ \sum_{i=0}^n q_{in}\boldsymbol{g}_i \end{bmatrix} + \begin{bmatrix} \widetilde{\boldsymbol{\rho}}_0 - \boldsymbol{\rho}_0 \\ \widetilde{\boldsymbol{q}}_n \\ M(\widetilde{\boldsymbol{\rho}}_0 - \widetilde{\boldsymbol{\rho}}_1) \\ M(\widetilde{\boldsymbol{q}}_0 - \widetilde{\boldsymbol{q}}_1) \\ M(\widetilde{\boldsymbol{\rho}}_0 - \widetilde{\boldsymbol{\rho}}_2) \\ M(\widetilde{\boldsymbol{q}}_0 - \widetilde{\boldsymbol{q}}_2) \\ \vdots \\ M(\widetilde{\boldsymbol{\rho}}_0 - \widetilde{\boldsymbol{\rho}}_n) \\ M(\widetilde{\boldsymbol{q}}_0 - \widetilde{\boldsymbol{q}}_n) \end{bmatrix},$$

where $\mathbf{R} : \mathbb{R}^{2N(n+1)} \to \mathbb{R}^{2N(n+1)}$. Here, the first row corresponds to the initial condition of the state equation, while the second row describes the final-time condition for the adjoint equation. The following rows correspond to the two discretized PDEs at the individual time steps, including boundary conditions. In order to minimize this residual, $\boldsymbol{R}\boldsymbol{y} = \boldsymbol{0}$ needs to be solved approximately. This is done using Newton's method

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k - [\mathcal{D}\boldsymbol{R}(\boldsymbol{y}_k)]^{-1}\boldsymbol{R}(\boldsymbol{y}_k). \tag{5.25}$$

To do so, the Jacobian, given by

$J = \mathcal{D}\boldsymbol{R}(\boldsymbol{y}) =$

$$\left[\begin{array}{cc|ccccc|c}
I_N & 0 & \cdots & \cdots & \cdots & 0 & 0 \\
0 & 0 & \cdots & \cdots & \cdots & 0 & I_N \\
\hline
q_{01}\mathcal{D}_\rho \boldsymbol{f}_0 + M & q_{01}\mathcal{D}_q \boldsymbol{f}_0 & q_{11}\mathcal{D}_\rho \boldsymbol{f}_1 - M & q_{11}\mathcal{D}_q \boldsymbol{f}_1 & \cdots & q_{n1}\mathcal{D}_\rho \boldsymbol{f}_n & q_{n1}\mathcal{D}_q \boldsymbol{f}_n \\
q_{01}\mathcal{D}_\rho \boldsymbol{g}_0 & q_{01}\mathcal{D}_q \boldsymbol{g}_0 + M & q_{11}\mathcal{D}_\rho \boldsymbol{g}_1 & q_{11}\mathcal{D}_q \boldsymbol{g}_1 - M & \cdots & q_{n1}\mathcal{D}_\rho \boldsymbol{g}_n & q_{n1}\mathcal{D}_q \boldsymbol{g}_n \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
q_{0n}\mathcal{D}_\rho \boldsymbol{f}_0 + M & q_{0n}\mathcal{D}_q \boldsymbol{f}_0 & q_{1n}\mathcal{D}_\rho \boldsymbol{f}_1 & q_{1n}\mathcal{D}_q \boldsymbol{f}_1 & \cdots & q_{nn}\mathcal{D}_\rho \boldsymbol{f}_n - M & q_{nn}\mathcal{D}_q \boldsymbol{f}_n \\
q_{0n}\mathcal{D}_\rho \boldsymbol{g}_0 & q_{0n}\mathcal{D}_q \boldsymbol{g}_0 + M & q_{1n}\mathcal{D}_\rho \boldsymbol{g}_1 & q_{1n}\mathcal{D}_q \boldsymbol{g}_1 & \cdots & q_{nn}\mathcal{D}_\rho \boldsymbol{g}_n & q_{nn}\mathcal{D}_q \boldsymbol{g}_n - M
\end{array}\right],$$

has to be applied, where $\boldsymbol{f}_i$ denotes $\boldsymbol{f}(t_i, \widetilde{\rho}_i, \widetilde{q}_i)$ and $\mathcal{D}_{\boldsymbol{\rho}}$ the Jacobian with respect to $\boldsymbol{\rho}$. The

notation for $\boldsymbol{g}$ and $\boldsymbol{q}$ follow. Moreover, I_N corresponds to the N-dimensional identity matrix, where N corresponds to the total number of discretization points in space. Equally, each of the entries in J are size $N \times N$, so that $J \in \mathbb{R}^{2N(n+1)\times 2N(n+1)}$. This Jacobian contains four blocks, as indicated by the lines. The first two rows correspond to the initial and final time condition of the state and adjoint equation respectively. Since the inverse operation of J is required in the Newton step (5.25), we would like to rewrite the Jacobian to do this as cheaply as possible, following the discussion on block systems in Section 4.3. Therefore, the matrix

$$L = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ \vdots & & \ddots & \\ 1 & & & 1 \end{bmatrix} \otimes I_{2N}.$$

is applied to J. We compute

$\widetilde{J} = J \cdot L =$

$$\left[\begin{array}{cc|cccccc} I_N & 0 & \cdots & \cdots & \cdots & 0 & 0 \\ 0 & I_N & \cdots & \cdots & \cdots & 0 & I_N \\ \hline \mathcal{D}_\rho \widetilde{\boldsymbol{f}}_1 & \mathcal{D}_q \widetilde{\boldsymbol{f}}_1 & q_{11}\mathcal{D}_\rho \boldsymbol{f}_1 - M & q_{11}\mathcal{D}_q \boldsymbol{f}_1 & \cdots & q_{n1}\mathcal{D}_\rho \boldsymbol{f}_n & q_{n1}\mathcal{D}_q \boldsymbol{f}_n \\ \mathcal{D}_\rho \widetilde{\boldsymbol{g}}_1 & \mathcal{D}_q \widetilde{\boldsymbol{g}}_1 & q_{11}\mathcal{D}_\rho \boldsymbol{g}_1 & q_{11}\mathcal{D}_q \boldsymbol{g}_1 - M & \cdots & q_{n1}\mathcal{D}_\rho \boldsymbol{g}_n & q_{n1}\mathcal{D}_q \boldsymbol{g}_n \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathcal{D}_\rho \widetilde{\boldsymbol{f}}_n & \mathcal{D}_q \widetilde{\boldsymbol{f}}_n & q_{1n}\mathcal{D}_\rho \boldsymbol{f}_1 & q_{1n}\mathcal{D}_q \boldsymbol{f}_1 & \cdots & q_{nn}\mathcal{D}_\rho \boldsymbol{f}_n - M & q_{nn}\mathcal{D}_q \boldsymbol{f}_n \\ \mathcal{D}_\rho \widetilde{\boldsymbol{g}}_n & \mathcal{D}_q \widetilde{\boldsymbol{g}}_n & q_{1n}\mathcal{D}_\rho \boldsymbol{g}_1 & q_{1n}\mathcal{D}_q \boldsymbol{g}_1 & \cdots & q_{nn}\mathcal{D}_\rho \boldsymbol{g}_n & q_{nn}\mathcal{D}_q \boldsymbol{g}_n - M \end{array}\right],$$

with

$$\mathcal{D}_\rho \widetilde{\boldsymbol{f}}_i := \sum_{j=0}^{n} q_{ji}\mathcal{D}_\rho \boldsymbol{f}_j, \quad \mathcal{D}_\rho \widetilde{\boldsymbol{g}}_i := \sum_{j=0}^{n} q_{ji}\mathcal{D}_\rho \boldsymbol{g}_j,$$
$$\mathcal{D}_q \widetilde{\boldsymbol{f}}_i := \sum_{j=0}^{n} q_{ji}\mathcal{D}_q \boldsymbol{f}_j, \quad \mathcal{D}_q \widetilde{\boldsymbol{g}}_i := \sum_{j=0}^{n} q_{ji}\mathcal{D}_q \boldsymbol{g}_j,$$

for which the $(1,1)$ block is now cheaply invertible.

The term $[\mathcal{D}\boldsymbol{R}(\boldsymbol{y}_k)]^{-1}\boldsymbol{R}(\boldsymbol{y}_k)$ in (5.25) can be expressed as

$$[\mathcal{D}\boldsymbol{R}(\boldsymbol{y}_k)]^{-1}\boldsymbol{R}(\boldsymbol{y}_k) := \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = J^{-1} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix},$$

where $\boldsymbol{R}(\boldsymbol{y}_k) := \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$. The vector $\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$ is found by solving the linear system

$$J \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \tag{5.26}$$

which in turn is equivalent to the permuted system

$$\widetilde{J} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{a}} \\ \widetilde{\mathbf{b}} \end{bmatrix},$$

where A, B, C, and D correspond to the four blocks of $\tilde{J}$. They are furthermore of the form (4.21) presented in Section 4.2. The system can be rewritten as

$$\begin{bmatrix} A & B \\ 0 & S \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} I_{2N} & B \\ 0 & D - CB \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{a}} \\ \widetilde{\mathbf{b}} - C\widetilde{\mathbf{a}} \end{bmatrix}.$$

Note that if for $\boldsymbol{y}_k$ we have $\widetilde{\boldsymbol{\rho}}_0 = \boldsymbol{\rho}_0$ and $\widetilde{\boldsymbol{q}}_n = \boldsymbol{q}_T$, then $\widetilde{\mathbf{a}} = \mathbf{0}$ and the system simplifies, so that a computation of $C\widetilde{\mathbf{a}}$ is avoided. The aim is now to solve this modified system, and in particular $S\widetilde{\mathbf{x}}_2 = \widetilde{\mathbf{b}}$. As discussed in Section 4.2, computing S^{-1} explicitly would be expensive. Therefore, the approach is to use a GMRES algorithm combined with an appropriate preconditioner, which can help speed up the numerical method. This preconditioner is discussed in the following.

5.4.1 The Preconditioner

The idea, as presented in [179], is to approximate $S = D - CB \in \mathbb{R}^{2Nn \times 2Nn}$ by $\widehat{S}$ and apply the resulting preconditioner $\widehat{S}^{-1}$ to the system, i.e., solve

$$\widehat{S}^{-1} S \widetilde{\mathbf{x}}_2 = \widehat{S}^{-1} \widetilde{\mathbf{b}}.$$

The approximation $\widehat{S}$ can be separated into two parts as $\widehat{S} = \widehat{D} - \widehat{C}B$. We define

$$\widehat{D} = \begin{bmatrix} q_{11} & q_{21} & \cdots & q_{n1} \\ q_{12} & q_{22} & \cdots & q_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ q_{1n} & q_{2n} & \cdots & q_{nn} \end{bmatrix} \otimes G, \quad \widehat{C} = \begin{bmatrix} \hat{q}_1 \\ \hat{q}_2 \\ \vdots \\ \hat{q}_n \end{bmatrix} \otimes G,$$

with $\hat{q}_i = \sum_{j=0}^{n} q_{ji}$.

The matrix $G \in \mathbb{R}^{2N \times 2N}$ is defined as

$$G := \begin{bmatrix} \mathcal{D}_\rho \boldsymbol{f} - M & \mathcal{D}_q \boldsymbol{f} \\ \mathcal{D}_\rho \boldsymbol{g} & \mathcal{D}_q \boldsymbol{g} - M \end{bmatrix},$$

and is evaluated at $t = T/2$. This time point is provided by the authors in [179], but in principle other time points could be chosen. The key idea is to evaluate G at a representative time point instead of at each time, to save computational cost.

Then, by the Woodbury matrix identity [180], we have that

$$\widehat{S}^{-1} = \underbrace{[I_{2Nn} + \widehat{D}^{-1}\widehat{C}(I_{2N} - B\widehat{D}^{-1}\widehat{C})^{-1}B]}_{=:F} \widehat{D}^{-1}.$$

Since $\widehat{D}$ and $\widehat{C}$ are defined using Kronecker products involving Lagrange functions and G, the term $\widehat{D}^{-1}\widehat{C}$ can be formed cheaply and the factor of G is cancelled out. The result is that this product only contains combinations of Lagrange functions, which have to be computed only once for the entire algorithm.

Calculating F explicitly gives

$$F = I_{2Nn} + \widehat{D}^{-1}\widehat{C}(I_{2N} - B\widehat{D}^{-1}\widehat{C})^{-1}B = \left[\begin{array}{ccc|cc} I_N & & & 0 & 0 \\ & I_N & & 0 & \widetilde{q}_1 I_N/(1-\widetilde{q}_n) \\ \hline & I_N & & 0 & 0 \\ & & I_N & 0 & \widetilde{q}_2 I_N/(1-\widetilde{q}_n) \\ \hline & & \ddots & \vdots & \vdots \\ & & \ddots & \vdots & \vdots \\ \hline & & & I_N & 0 \\ & & & 0 & (1+\widetilde{q}_n/(1-\widetilde{q}_n))I_N \end{array}\right].$$

This, again, only has to be computed once in the entire algorithm.

Then $\widehat{S}^{-1} = F\widehat{D}^{-1}$ can be applied to a vector $\boldsymbol{v}$ via

$$\widehat{S}^{-1}\boldsymbol{v} = F\widehat{D}^{-1}\boldsymbol{v} = F\,\mathtt{vec}\left(G^{-1}\mathtt{mat}(\boldsymbol{v})\begin{bmatrix} q_{11} & q_{21} & \cdots & q_{n1} \\ q_{12} & q_{22} & \cdots & q_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ q_{1n} & q_{2n} & \cdots & q_{nn} \end{bmatrix}^{-\top}\right),$$

which is a consequence of the 'vec-trick' introduced in [181]. Note that the operation `vec` stacks the columns of an $2N \times n$ matrix into a vector of size $2Nn$, while the operation `mat` does the opposite. The application of the preconditioner is comparatively cheap, with the main cost lying in the factorization of G, which has to be done only once per Newton iteration. This is done applying a LU factorization, using the backslash command in MATLAB.

5.4.2 The Algorithm Implementation

Now that we have found an appropriate preconditioner for the problem, we briefly summarize the algorithm and provide an idea of how it is implemented in MATLAB. This Newton–Krylov algorithm works as follows. We drive the residual of the given problem towards zero using Newton's method (5.25). However, to do so, the inverse Jacobian of the system has to be determined, which is prohibitively expensive. Therefore, the Jacobian is rewritten, so that instead we are required to accurately solve an equivalent matrix–vector equation given by (5.26). After transforming this system, the main cost involved in finding its solution is solving the equation

$$S\widetilde{\mathbf{x}}_2 = \widetilde{\mathbf{b}}.$$

Therefore, a preconditioner is devised to do this cheaply. The preconditioned system

$$\widehat{S}^{-1}S\widetilde{\mathbf{x}}_2 = \widehat{S}^{-1}\widetilde{\mathbf{b}}$$

is solved using the GMRES algorithm introduced in Section 4.3. While the Newton–Krylov method itself is standard and can be found in [182, Chapter 3], the key challenge in deriving this numerical method is to devise the Krylov strategy with an efficient preconditioner. Once the solution $\widetilde{\mathbf{x}}_2$ is found, $\widetilde{\mathbf{x}}_1$ is computed easily, since the matrices A and B are sparse.

The code below displays the necessary inputs for the Newton–Krylov solver for a two-dimensional flow control problem (5.8) with Dirichlet boundary conditions (5.6). Here, the quantities `JFu`, `JFv`, `JGu`, `JGv` denote the elements of the Jacobian J. Our software [183] allows us to compute these quantities to spectral accuracy, due to the use of the pseudospectral method introduced in Chapter 3.

```
% Definition of state and adjoint PDE operators
K1 = @(t,u,v) L + 2/bet * scalarOperator(dotVectors(grad*u,grad*v)) ...
    + gradVextDotGrad(t) + scalarOperator(LapVext(t)) ...
    + kappa * ( dotVectorOperator(grad*(Conv*u),grad) ...
    + scalarOperator(L*(Conv*u)) );
K2 = @(t,u,v) -1/bet * scalarOperator(u.^2) * L;
K3 = @(t,u,v) -I + 1/bet * scalarOperator(dotVectors(grad*v,grad*v)) ...
    - kappa * ( Dx1 * Conv * scalarOperator(Dx1*v) ...
    + Dx2 * Conv * scalarOperator(Dx2*v) );
K4 = @(t,u,v) L - gradVextDotGrad(t) - kappa * ...
    dotVectorOperator(grad*(Conv*u),grad);
f = @(t,u,v) z(t);  g = @(t,u,v) uhat(t);

F = @(t,u,v) K1(t,u,v)*u - K2(t,u,v)*v + f(t,u,v);
G = @(t,u,v) K3(t,u,v)*u - K4(t,u,v)*v + g(t,u,v);

% Specification of Jacobians
JFu = @(t,u,v) L + 2/bet * scalarOperator(u) * scalarOperator(L*v)  ...
    + 2/bet * scalarOperator(dotVectors(grad*u,grad*v)) ...
```

```
        + 2/bet * scalarOperator(u) * dotVectorOperator(grad*v,grad) ...
        + gradVextDotGrad(t) + scalarOperator(LapVext(t)) ...
        + kappa * ( dotVectorOperator(grad*(Conv*u),grad) ...
        + scalarOperator(L*(Conv*u)) ...
        + dotVectorOperator((grad*u),grad) * Conv ...
        + scalarOperator(u) * L * Conv );
    JFv = @(t,u,v) 1/bet * scalarOperator(u.^2) * L ...
        + 2/bet * scalarOperator(u) * dotVectorOperator(grad*u,grad);
    JGu = @(t,u,v) -I + 1/bet * scalarOperator(dotVectors(grad*v,grad*v)) ...
        + kappa * ( dotVectorOperator(grad*v,grad*Conv) ...
        - Dx1 * Conv * scalarOperator(Dx1*v) ...
        - Dx2 * Conv * scalarOperator(Dx2*v) );
    JGv = @(t,u,v) -L + 2/bet * scalarOperator(u) * ...
        dotVectorOperator(grad*v,grad) ...
        + gradVextDotGrad(t) ...
        + kappa * ( dotVectorOperator(grad*Conv*u,grad) ...
        - Dx1 * Conv * scalarOperator(u) * Dx1 ...
        - Dx2 * Conv * scalarOperator(u) * Dx2 );
```

In more detail, `Dx1` and `Dx2` are matrices applying spatial derivatives in each direction, with `grad`, corresponding to the gradient function, and `L` the spectral discretization of the Laplacian operator, as introduced in Chapter 3. The function `Conv` applies a convolution integral with the function ∇V_2, as described in Chapter 3, `gradVextDotGrad` applies an operator of the form $\nabla V_1 \cdot \nabla$, with `LapVext` evaluating $\nabla^2 V_1$ to spectral accuracy. The function `scalarOperator` forms a scalar function, with `dotVectors` taking an inner product of two vectors, and `dotVectorOperator` similarly taking an inner product of the first argument with the second argument applied to a subsequent term. Finally, `f` and `g` describe the source term of the state equation f and the desired state $\widehat{\rho}$ within the PDE operators.

For no-flux type boundary conditions the interior and boundary nodes need to be separated within the code, with the Jacobians defined separately for the boundary conditions. We refer to the open-source software [183] described in Chapter 3 and [184] for the full implementation with different boundary conditions, as well as for source control problems.

5.5 Numerical Examples

In this section, the numerical method we developed in this chapter is tested and example problems are computed. First, tests in one, two, and three dimensions are run, in which the numerical solution is compared against an exact solution. Then, some numerical examples in two and three dimensions are presented, which do not have analytic solutions. All problems require a given desired state $\widehat{\rho}$, source term f, advective field $\vec{f}$, and external potential V_1. Moreover, the Newton–Krylov solver is initiated with a guess for ρ and q at all time points. In this thesis, we will always use the initial and final time conditions for ρ and q, respectively, as initial guesses for all time points. All tests are carried out on Dell PowerEdge R430 running Scientific Linux 7, four Intel Xeon E5-2680 v3 2.5GHz, 30M Cache, 9.6 GT/s QPI 192 GB RAM.

We recall the error measure introduced in Section 3.3, and restate it for convenience:

$$\mathcal{E} = \max_{t\in[0,T]} \left[\min\left(\mathcal{E}_{Rel}(t), \mathcal{E}_{Abs}(t) \right) \right], \tag{5.27}$$

where

$$\mathcal{E}_{Abs}(t) = \left\| y(t,\vec{x}) - y_R(t,\vec{x}) \right\|_{L^2(\Omega)}, \qquad \mathcal{E}_{Rel}(t) = \frac{\left\| y(t,\vec{x}) - y_R(t,\vec{x}) \right\|_{L^2(\Omega)}}{\left\| y_R(t,\vec{x}) \right\|_{L^2(\Omega)} + 10^{-10}}.$$

The error considers the difference between a variable of interest, y, and a reference value y_R, which in the following validation section is the analytic solution to a test problem.

5.5.1 Validation Tests

In the same way as for the validation of the code for forward problems, the implementation is validated using exact solutions. The 1D problems are tested with $N = 30$, $n = 20$ points, the 2D problems with $N_1 = N_2 = 20$ and $n = 10$ points, and the 3D problem with $N_1 = N_2 = N_3 = 20$ and $n = 10$ points. The spatial domain is $[-1, 1]^d$, $d = 1, 2, 3$, and the time horizon is $(0, 1)$. The number of outer iterations of the Newton–Krylov solver is fixed at 10 iterations, while the inner iterations are allowed to go up to a maximum of 200 iterations, but mostly terminate before this, using a stopping criterion given in [185]. The tolerance for the Newton–Krylov is 10^{-16}. The initial guesses for ρ and q for all problems are chosen as $\rho(t, \vec{x}) = \rho_{\text{ex}}(0, \vec{x})$ and $q(t, \vec{x}) = 0$ for all $t \in (0, T)$. The errors between the numerical and exact solutions are measured with the standard measure defined in (5.27). The computational time for each one-dimensional problem is between one and five seconds, each two-dimensional problem takes between 30 and 90 seconds, while the three-dimensional problem takes around 11 hours, demonstrating the large increase of computational cost with increasing dimension.

Flow Control Problems

We first consider the flow control problem (5.8) with $\mathcal{I}(\rho) = 0$. This corresponds to the first-order optimality system (5.19) for Dirichlet conditions and (5.21) for no-flux conditions. We construct exact solutions to the optimality system by choosing ρ and q that satisfy the initial, final-time, and boundary conditions. Given these choices, we can compute the control $\vec{w}$. Finally we construct f and $\widehat{\rho}$ by computing

$$\begin{aligned} f_{\text{ex}} &= \frac{\partial \rho_{\text{ex}}}{\partial t} + \nabla \cdot \left(\mathcal{D}(\rho_{\text{ex}}) + \mathcal{C}(\rho_{\text{ex}}, \vec{w}_{\text{ex}}) \right), \\ \widehat{\rho}_{\text{ex}} &= \frac{\partial q_{\text{ex}}}{\partial t} - \mathcal{D}^*(q_{\text{ex}}) - \mathcal{C}^*(q_{\text{ex}}, \vec{w}_{\text{ex}}) + \rho_{\text{ex}}, \end{aligned} \tag{5.28}$$

so that the PDEs are satisfied exactly. For a 1D flow control problem with no-flux boundary conditions we choose

$$\begin{aligned} \rho_{\text{ex}}(t, x) &= \frac{1}{4}\beta^{1/2} e^t (\cos(\pi x) + 1), \\ q_{\text{ex}}(t, x) &= \frac{1}{4}\beta^{1/2} \left(e^T - e^t \right) (\cos(\pi x) + 1), \end{aligned}$$

where the pre-factor $\beta^{1/2}$ is chosen so that the control is independent of β, as demonstrated by

$$\vec{w}_{\text{ex}} = -\frac{1}{\beta}\rho \nabla q = \frac{\pi}{16} e^t (e^T - e^t)(\cos(\pi x) + 1) \sin(\pi x).$$

The reason for this is that if β is small, the advection term $\vec{w}$ is large, making the problem advection-dominated and therefore harder to solve numerically. We furthermore choose $\vec{f} = \vec{0}$ and $V_1 = \frac{1}{10}x^2$. The result of computing the Newton–Krylov algorithm for different β can be seen in Table 5.1. It is clear that for all values of β, both the numerically computed state and adjoint variable are within machine precision of the constructed exact solution.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_\rho$	2.00×10^{-17}	1.93×10^{-16}	1.81×10^{-15}	5.31×10^{-15}	5.41×10^{-15}
$\mathcal{E}_q$	2.55×10^{-17}	2.76×10^{-16}	2.68×10^{-15}	1.17×10^{-14}	1.10×10^{-14}

Table 5.1: Numerical errors for a 1D flow control problem with no-flux boundary conditions found by comparing to an analytical exact solution. Errors in the state ρ and the adjoint q are displayed for different values of β.

The approach for a 2D problem is exactly the same as described for the 1D problem above.

Here the choices of functions are

$$\begin{aligned}
\rho_{\mathrm{ex}}(t,\vec{x}) &= \frac{1}{4}\beta^{1/2}e^{t}(\cos(\pi x_1)+1)(\cos(\pi x_2)+1),\\
q_{\mathrm{ex}}(t,\vec{x}) &= \frac{1}{4}\beta^{1/2}\left(e^{T}-e^{t}\right)(\cos(\pi x_1)+1)(\cos(\pi x_2)+1),\\
V_1(\vec{x}) &= \cos(\pi x_1)\cos(\pi x_2),\\
\vec{f} &= \vec{0}.
\end{aligned}$$

The resulting error in the numerical solutions for ρ and q, which are within orders of machine precision, can be seen in Table 5.2.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_\rho$	1.39×10^{-15}	1.39×10^{-14}	1.39×10^{-13}	2.16×10^{-13}	2.15×10^{-13}
$\mathcal{E}_q$	1.81×10^{-15}	1.81×10^{-14}	1.82×10^{-13}	4.46×10^{-13}	4.45×10^{-13}

Table 5.2: Numerical errors for a 2D flow control problem with no-flux boundary conditions found by comparing to an analytical exact solution. Errors in the state ρ and the adjoint q are displayed for different values of β.

Moreover, in Figure 5.1 convergence for small $N = N_1 = N_2$ for different β and number of time points n is displayed. It is evident that there is a trade-off in the number of time points chosen. If one chooses $n = 6$, the solution converges up to a threshold of 10^{-10}, regardless of the choice of spatial discretization points. For n between 10 and 20 points, the error in the solution improves continuously with increased N, down to an error of order 10^{-16}, which is machine precision. However, for $n = 30$, the error becomes as bad as for $n = 6$, plateauing around 10^{-10}. The residual error is minimized at each individual time point, so that choosing larger n increases the number of terms to minimize. This may lead to worse conditioning and hence to an overall increase in error.

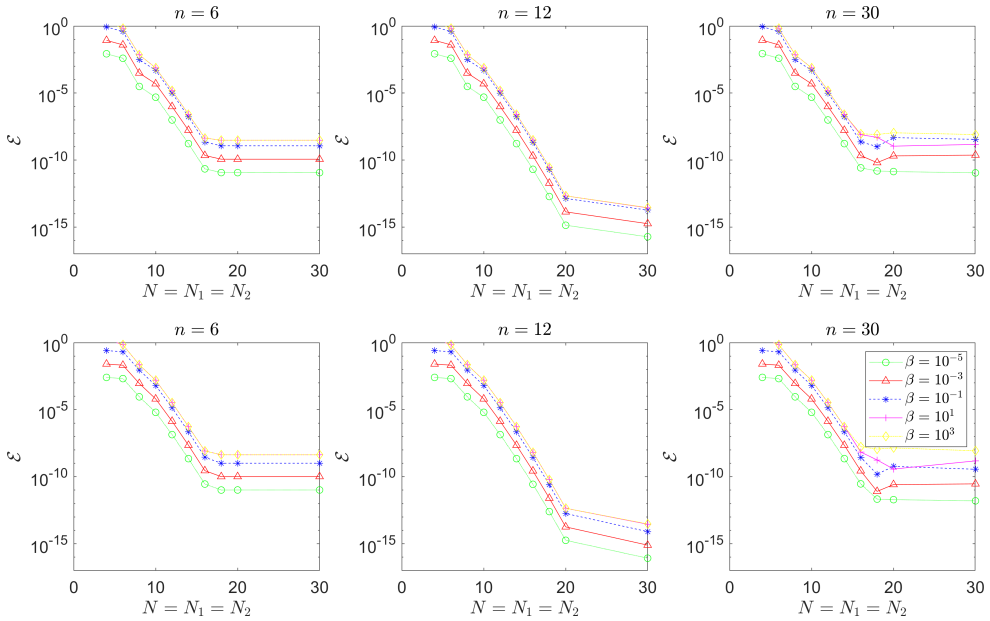

Figure 5.1: Exact flow control, no-flux: Convergence of the Newton–Krylov algorithm. Top row: convergence in the state variable for different numbers of temporal discretization points n. Bottom row: convergence in the adjoint variable. Convergence is measured using the error measure (5.27). Note that the case $n = 20$ (not shown here) is qualitatively similar to $n = 12$.

Finally, for the 3D problem, the exact solutions are

$$\begin{aligned}
\rho_{\text{ex}}(t,\vec{x}) &= \frac{1}{4}\beta^{1/2}e^t(\cos(\pi x_1)+1)(\cos(\pi x_2)+1)(\cos(\pi x_3)+1),\\
q_{\text{ex}}(t,\vec{x}) &= \frac{1}{4}\beta^{1/2}\left(e^T-e^t\right)(\cos(\pi x_1)+1)(\cos(\pi x_2)+1)(\cos(\pi x_3)+1),\\
V_1(\vec{x}) &= \frac{1}{10}x_1^2x_2^2x_3^2,\\
\vec{f} &= \vec{0}.
\end{aligned}$$

The numerical error as compared to the exact solution, which is of similar order to the lower-dimensional examples, can be seen in Table 5.3.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_\rho$	2.49×10^{-15}	2.49×10^{-14}	2.24×10^{-13}	2.23×10^{-13}	2.22×10^{-13}
$\mathcal{E}_q$	2.41×10^{-15}	2.41×10^{-14}	2.41×10^{-13}	3.41×10^{-13}	3.41×10^{-13}

Table 5.3: Numerical solution for a 3D flow control problem with no-flux boundary conditions found by comparing to an analytical exact solution. Errors in the state ρ and the adjoint q are displayed for different values of β.

We proceed in the same way for problems with Dirichlet boundary conditions, now in 1D and 2D only, in the interest of brevity. The choices for the 1D problem are

$$\begin{aligned}
\rho_{\text{ex}}(t,\vec{x}) &= \frac{1}{4}\beta^{1/2}e^t\cos\left(\frac{\pi x}{2}\right)+\frac{1}{4},\\
q_{\text{ex}}(t,\vec{x}) &= \frac{1}{4}\beta^{1/2}\left(e^T-e^t\right)\cos\left(\frac{\pi x}{2}\right),\\
V_1 &= 0, \quad \vec{f} = \vec{0},
\end{aligned}$$

which satisfy non-zero Dirichlet boundary conditions (5.6) with $g = \frac{1}{4}$. The resulting numerical error for such a problem is displayed in Table 5.4.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_\rho$	8.52×10^{-16}	8.52×10^{-16}	8.52×10^{-16}	8.52×10^{-16}	8.52×10^{-16}
$\mathcal{E}_q$	7.48×10^{-17}	7.48×10^{-17}	7.48×10^{-17}	7.48×10^{-17}	7.48×10^{-17}

Table 5.4: Numerical errors for a 1D flow control problem with Dirichlet boundary conditions found by comparing to an analytical exact solution. Errors in the state ρ and the adjoint q are displayed for different values of β.

For the 2D problem with zero Dirichlet boundary conditions we choose

$$\begin{aligned}
\rho_{\text{ex}}(t,\vec{x}) &= 2\beta^{1/2}e^t\cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{\pi x_2}{2}\right),\\
q_{\text{ex}}(t,\vec{x}) &= \beta^{1/2}\left(e^T-e^t\right)\cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{\pi x_2}{2}\right),\\
V_1 &= \frac{1}{2}x_1^2+\frac{1}{2}x_2^2, \quad \vec{f} = \vec{0},
\end{aligned}$$

and the errors between the numerical and exact solutions can be seen in Table 5.5. In both the 1D and 2D example, the tables demonstrate the achieved spectral accuracy.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_\rho$	4.02×10^{-17}	3.85×10^{-16}	4.47×10^{-15}	3.58×10^{-15}	4.49×10^{-15}
$\mathcal{E}_q$	7.67×10^{-18}	6.51×10^{-17}	6.99×10^{-16}	5.47×10^{-15}	5.72×10^{-15}

Table 5.5: Numerical errors for a 2D flow control problem with Dirichlet boundary conditions found by comparing to an analytical exact solution. Errors in the state ρ and the adjoint q are displayed for different values of β.

Source Control Problems

Next, source control problem (5.2) with $\mathcal{I}(\rho) = 0$ is considered. We therefore need to solve optimality system (5.22) for no-flux boundary conditions or (5.15) for Dirichlet conditions. For the source control problem we consider 2D examples only, since neither 1D problems, nor source control problems are the focus of this work. The construction of exact solutions is equivalent to the approach for flow control problems, with the computation of the control variable now replaced by

$$w = -\frac{1}{\beta} q.$$

For the example with Dirichlet conditions, we choose

$$\begin{aligned} \rho_{\text{ex}}(t, \vec{x}) &= 2e^t \cos\left(\frac{\pi x_1}{2}\right) \cos\left(\frac{\pi x_2}{2}\right), \\ q_{\text{ex}}(t, \vec{x}) &= \left(e^T - e^t\right) \cos\left(\frac{\pi x_1}{2}\right) \cos\left(\frac{\pi x_2}{2}\right), \\ V_1 &= \cos\left(\frac{\pi x_1}{2}\right) \cos\left(\frac{\pi x_2}{2}\right), \quad \vec{f} = \vec{0}, \end{aligned}$$

and the numerical errors made by the Newton–Krylov solver are displayed in Table 5.6.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_\rho$	4.19×10^{-14}	1.17×10^{-14}	2.94×10^{-15}	3.37×10^{-15}	3.42×10^{-15}
$\mathcal{E}_q$	3.22×10^{-16}	1.03×10^{-15}	3.14×10^{-15}	3.26×10^{-15}	3.05×10^{-15}

Table 5.6: Numerical errors for a 2D source control problem with Dirichlet boundary conditions found by comparing to an analytical exact solution. Errors in the state ρ and the adjoint q are displayed for different values of β.

Finally, the choices for the no-flux condition example are

$$\begin{aligned} \rho_{\text{ex}}(t, \vec{x}) &= 2e^t \cos(\pi x_1) \cos(\pi x_2), \\ q_{\text{ex}}(t, \vec{x}) &= \left(e^T - e^t\right) \cos(\pi x_1) \cos(\pi x_2), \\ V_1 &= \cos(\pi x_1) \cos(\pi x_2), \quad \vec{f} = \vec{0}, \end{aligned}$$

and errors in the numerical ρ and q with respect to this exact solutions are displayed in Table 5.7. Note that, for source control problems, choosing ρ and q dependent on β is not necessary because having a larger source term does not generally increase the numerical complexity. In contrast, the flow control problem would become advection dominated for ρ, q independent of β. It is evident that for all test problems with analytic solutions, the Newton–Krylov algorithm converges with high accuracy to the exact solution.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_\rho$	3.52×10^{-13}	3.52×10^{-13}	3.52×10^{-13}	3.52×10^{-13}	3.52×10^{-13}
$\mathcal{E}_q$	5.15×10^{-13}	5.15×10^{-13}	5.15×10^{-13}	5.15×10^{-13}	5.15×10^{-13}

Table 5.7: Numerical errors for a 2D source control problem with no-flux boundary conditions found by comparing to an analytical exact solution. Errors in the state ρ and the adjoint q are displayed for different values of β.

5.5.2 Solving Optimal Control Problems in Two Dimensions

Having validated the numerical approach using exact solutions, in this section some numerical examples of interacting particle systems are solved. In particular, two-dimensional examples for the optimal control problems (5.2) and (5.8), with Dirichlet and no-flux boundary conditions are considered. An interaction potential for the particle–particle interactions is specified as

$$V_2(\vec{x}) = \kappa e^{-\|\vec{x}\|^2}. \tag{5.29}$$

In the following examples, it is of interest how the interaction strength impacts the particle dynamics. Here, three values are considered: $\kappa = 0$ (no interaction), $\kappa = -1$ (attraction), and $\kappa = 1$ (repulsion). If $\kappa = 0$, the PDE constraint reduces to an advection–diffusion equation. This serves as a benchmark, since the advection–diffusion problem has been validated in the previous section. The interaction strengths $|\kappa| = 1$ are chosen as a balance between demonstrable differences in the solution and efficient computation. For smaller $|\kappa|$, diffusion and V_1 dominate and interaction effects are negligible, while for larger $|\kappa|$, steep gradients in the resulting particle density make the numerical solution difficult.

In order to compare the optimization result to a benchmark, we consider the cost of the equivalent problem with $\vec{w} = \vec{0}$, which corresponds to solving the forward PDE with $\vec{w} = \vec{0}$. The associated cost functional $\mathcal{J}$ is denoted by $\mathcal{J}_{uc}$, where uc denotes 'uncontrolled'. The optimal cost, denoted by $\mathcal{J}_c$, is then found by evaluating the cost functional for the result found by the Newton–Krylov algorithm. This value is expected to satisfy

$$\mathcal{J}_c \leq \mathcal{J}_{uc},$$

and, moreover, the difference in the two costs depends on β. For smaller β, larger differences are expected, since $\vec{w}$ is allowed to be large. By contrast, as β becomes larger, less control is allowed in the system so that $\vec{w}$ tends to zero and so $\mathcal{J}_c$ to $\mathcal{J}_{uc}$. The spatial-temporal domain is $(0, T) \times \Omega = (0, 1) \times [-1, -1]^2$, however, changing the domain sizes is possible. The number of spatial points is $N_1 = N_2 = 20$ and the number of time points is $n = 11$. The ODE solver tolerances (MATLAB's `ode15s` [120, 121]) for computing the baseline problem with $\vec{w} = \vec{0}$ are set to 10^{-9}, while the tolerance for the Newton–Krylov algorithm is 10^{-16}. The maximum numbers of iterations for the Newton–Krylov solver remain at 200 GMRES and 10 Newton iterations, as for the exact problems.

Flow Control Problem with No-Flux Boundary Conditions

The first problem is of the form (5.8) with no-flux boundary conditions (5.10). The inputs are

$$\rho_0 = \frac{1}{4}, \quad \widehat{\rho} = \frac{1}{4}(1-t) + \frac{t}{Z} \exp\left(-2\left((x_1 + 0.2)^2 + (x_2 + 0.2)^2\right)\right),$$
$$V_1 = \left((x_1 + 0.3)^2 - 1\right)\left((x_1 - 0.4)^2 - 0.5\right)\left((x_2 + 0.3)^2 - 1\right)\left((x_2 - 0.4)^2 - 0.5\right),$$

where $Z \approx 1.3791$ is a normalization constant. In Table 5.8, the value of the cost functional for the benchmark $\mathcal{J}_{uc}$ ($\vec{w} = \vec{0}$) is compared with the optimized case ($\mathcal{J}_c$) for different values of β and for each of the interaction strengths. As expected, in all cases $\mathcal{J}_c \leq \mathcal{J}_{uc}$ and $\mathcal{J}_c$ decreases with decreasing β. For larger β applying control is heavily penalized and $\mathcal{J}_c$ approaches $\mathcal{J}_{uc}$.

		$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\kappa = 0$	$\mathcal{J}_{uc}$	2.67×10^{-2}	2.67×10^{-2}	2.67×10^{-2}	2.67×10^{-2}	2.67×10^{-2}
	$\mathcal{J}_c$	8.23×10^{-5}	3.87×10^{-3}	2.50×10^{-2}	2.67×10^{-2}	2.67×10^{-2}
$\kappa = 1$	$\mathcal{J}_{uc}$	3.29×10^{-2}	3.29×10^{-2}	3.29×10^{-2}	3.29×10^{-2}	3.29×10^{-2}
	$\mathcal{J}_c$	1.16×10^{-4}	5.44×10^{-3}	3.13×10^{-2}	3.29×10^{-2}	3.29×10^{-2}
$\kappa = -1$	$\mathcal{J}_{uc}$	2.09×10^{-2}	2.09×10^{-2}	2.09×10^{-2}	2.09×10^{-2}	2.09×10^{-2}
	$\mathcal{J}_c$	5.71×10^{-5}	2.63×10^{-3}	1.92×10^{-2}	2.09×10^{-2}	2.09×10^{-2}

Table 5.8: Flow control, no-flux: Cost when $\vec{w} = \vec{0}$ and optimal control cost for a range of κ and β. For all β, $\mathcal{J}_c \leq \mathcal{J}_{uc}$, as expected. Note that for $\beta = 10^1$, the cost functionals differ by order of 10^{-5}, while for $\beta = 10^3$ they differ by order of 10^{-7}.

In Figures 5.2 and 5.3, the optimal states and controls are displayed for the three choices of κ and for $\beta = 10^{-3}$. The β value is chosen to showcase differences in optimal control for different κ. If a larger value of β was chosen, since little control would enter the system, these differences would be too small to highlight. In Figure 5.2 the evolution of particle densities over time for different κ is shown. At $t = 0.1$, ρ accumulates in regions with potential wells and the areas where the potential is large are avoided (see contour plot of V_1 in Figure 5.3 for reference).

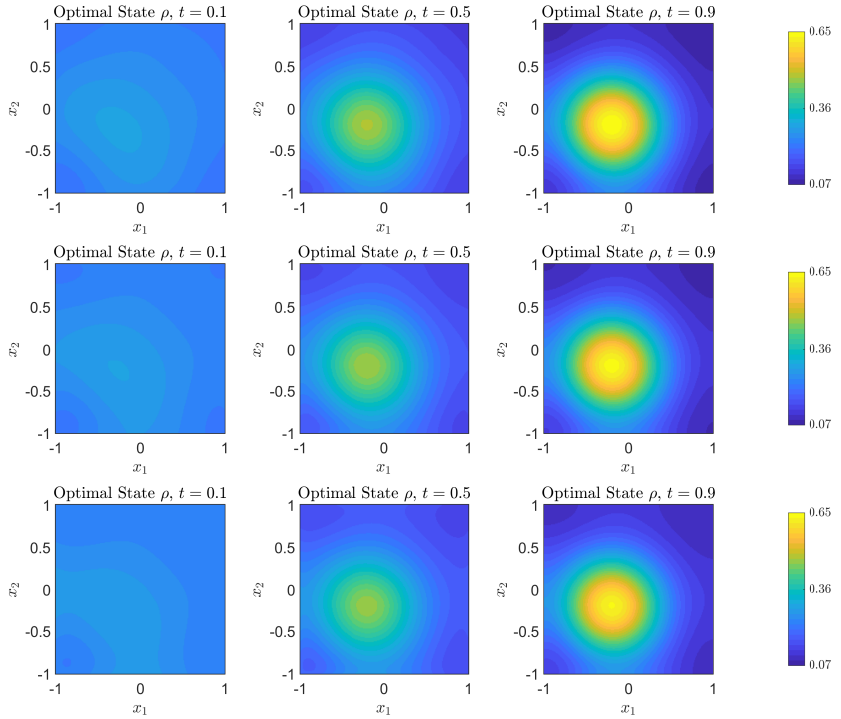

Figure 5.2: Flow control, no-flux: Evolution of the optimal ρ for different interaction strengths, $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom), with $\beta = 10^{-3}$.

In Figure 5.3, it is demonstrated how the control acts to drive the particle distribution towards the desired state. However, it does not act uniformly around the peak of the desired state, but rather acts strongly in the area between the location of the desired peak and the point $(-1, -1)$. This is due to the external potential being large in this area, which requires more control to overcome. In the attractive configuration, the effect of the attraction supports the control action, since the desired state requires the particles to accumulate in one part of the domain, so less control is needed to reach the desired state. By contrast, more control has to

be applied to the repulsive particles, which oppose such clustering. This is evident in Figure 5.3, when comparing the magnitude of the control, symbolized by differently sized arrows, for $\kappa = -1$ (attraction) with $\kappa = 1$ (repulsion).

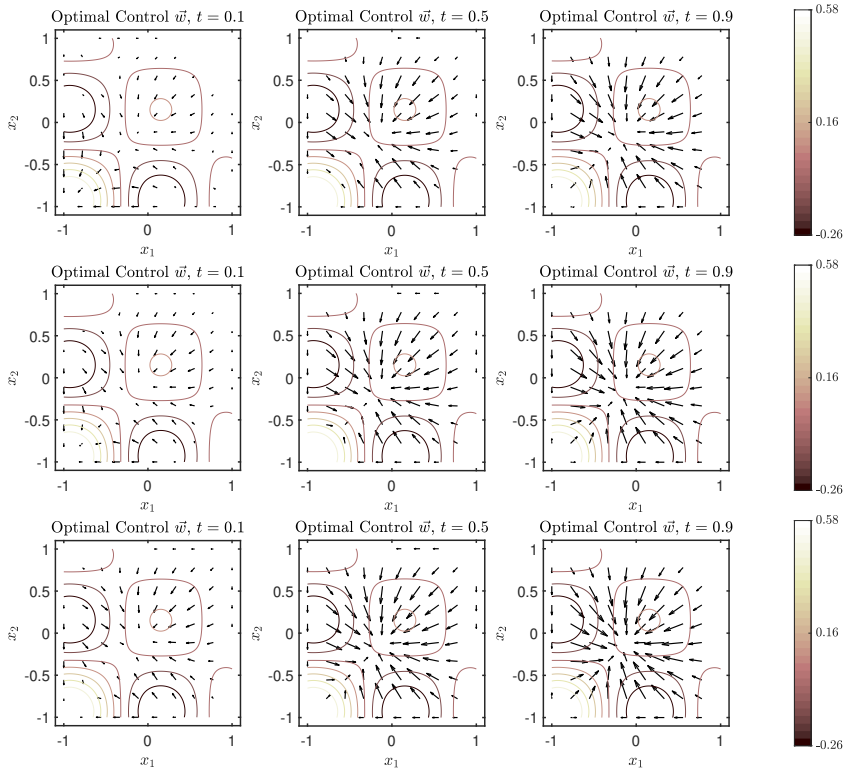

Figure 5.3: Flow control, no-flux: Evolution of the optimal control for different interaction strengths, $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom), with $\beta = 10^{-3}$. The lengths of the arrows are proportional to $\|\vec{w}\|$. A contour plot of the external potential V_1 is superimposed, with a corresponding colorbar on the right-hand side.

This problem is solved with 10 iterations of the Newton–Krylov solver. Table 5.9 shows the performance of the Newton–Krylov algorithm for the problem with $\kappa = -1$ for different numbers of discretization points and β values. In all cases a residual error for ρ and q of the order of machine precision is reached. There is a trend of increasing number of iterations, and therefore computational time, with larger values of β. It can be observed that, for larger n, the inner GMRES iterations reach the maximum of 200 iterations more often. Note that this is not necessarily an issue, as long as the solution is suitably close to the new Newton direction. Alternatively, one can also increase the maximum GMRES iterations, since each iteration is not overly costly due to the preconditioner design, since the dominant cost of the algorithm is a matrix factorization, which is only computed once per Newton iteration and not at each GMRES step. However, in this case, the higher number of iterations is in line with the observations made in Figure 5.1, where the exact error in the Newton–Krylov solution for $n = 30$ was considerably worse than for $n = 12$ at the same number of spatial discretization points. In the same figure it was also evident that $N_1 = N_2 = 11$ discretization points were not sufficient for a high accuracy solution. Therefore, the case $n = 10$, $N_1 = N_2 = 20$ in Table 5.9 appears to be an optimal choice of discretization points, avoiding low accuracy in the spatial discretization and bad conditioning for high number of temporal discretization points. Figure 5.4 shows the convergence of the residual error in the Newton–Krylov algorithm for different values of β. A residual error of 10^{-13} is reached for both state and adjoint variables within 8 iterations for all values of β. For larger β, the convergence is often slightly faster.

n/N	Number of GMRES Iterations	Total	Accuracy ρ	Accuracy q	Time (s)
11/10					
$\beta = 10^{-5}$	8,21,29,40,38,41,56,84,70,19	406	2.78×10^{-13}	3.82×10^{-17}	10.86
$\beta = 10^{-3}$	29,34,40,58,75,99,115,35,32,24	541	1.00×10^{-14}	1.34×10^{-16}	15.51
$\beta = 10^{-1}$	42,61,93,118,150,58,48,53,44,51	718	7.44×10^{-15}	1.20×10^{-15}	20.12
$\beta = 10^{1}$	36,54,81,102,138,65,43,54,48,45	666	9.49×10^{-15}	1.82×10^{-15}	18.62
$\beta = 10^{3}$	33,51,76,98,128,68,49,46,49,38	636	9.38×10^{-15}	1.61×10^{-15}	17.18
11/20					
$\beta = 10^{-5}$	1,22,33,41,49,58,88,53,19,19	383	6.28×10^{-13}	1.34×10^{-15}	98.88
$\beta = 10^{-3}$	26,36,45,60,85,115,64,27,34,24	516	3.25×10^{-13}	5.81×10^{-15}	132.96
$\beta = 10^{-1}$	39,61,91,118,135,64,41,38,54,52	693	2.32×10^{-13}	2.14×10^{-14}	175.06
$\beta = 10^{1}$	33,53,81,103,124,76,39,45,51,48	653	1.96×10^{-13}	2.15×10^{-14}	166.52
$\beta = 10^{3}$	32,49,73,94,118,75,41,43,47,47	619	3.13×10^{-13}	4.62×10^{-14}	156.82
21/20					
$\beta = 10^{-5}$	7,47,62,87,95,109,145,121,37,40	750	3.91×10^{-13}	2.26×10^{-15}	357.15
$\beta = 10^{-3}$	59,81,100,147,187,200,110,54,73,59	1070	1.75×10^{-13}	3.18×10^{-15}	513.35
$\beta = 10^{-1}$	89,130,200,200,200,200,84,94,89,117	1403	1.78×10^{-13}	1.23×10^{-14}	669.59
$\beta = 10^{1}$	80,124,186,200,200,182,96,84,115,92	1359	1.63×10^{-13}	2.99×10^{-14}	657.01
$\beta = 10^{3}$	75,111,167,200,200,167,95,104,105,104	1328	2.01×10^{-13}	1.42×10^{-14}	635.56
31/30					
$\beta = 10^{-5}$	34,72,90,140,136,160,200,170,59,59	1120	1.64×10^{-12}	4.40×10^{-15}	4303.09
$\beta = 10^{-3}$	88,125,155,200,200,200,200,200,97,112	1577	1.28×10^{-12}	1.51×10^{-14}	5986.21
$\beta = 10^{-1}$	143,200,200,200,200,200,200,200,200,200	1943	8.56×10^{-11}	3.78×10^{-12}	7230.79
$\beta = 10^{1}$	134,200,200,200,200,200,200,200,200,200	1934	1.16×10^{-11}	2.48×10^{-12}	7350.47
$\beta = 10^{3}$	126,200,200,200,200,200,200,200,200,200	1926	8.83×10^{-13}	1.09×10^{-13}	7150.56

Table 5.9: Newton–Krylov algorithm performance for different temporal and spatial discretization points n and $N := N_1 = N_2$, and different β. Measured are the number of inner GMRES iterations per Newton iteration, the total number of GMRES iterations for the algorithm, the final accuracy of ρ and q in the residual norm, and the total computation time.

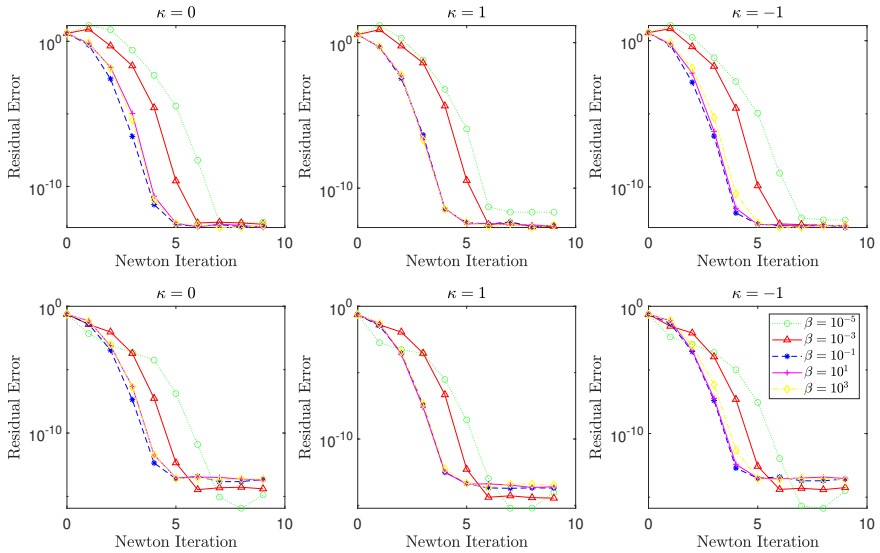

Figure 5.4: Flow control, no-flux: Convergence of the Newton–Krylov algorithm. Top row: convergence in the state variable for different κ. Bottom row: convergence in the adjoint variable. Convergence is measured using the residual error. Note the expected rapid convergence in all cases.

Flow Control Problem with Dirichlet Boundary Conditions

The next control problem is of type (5.8), with zero Dirichlet boundary conditions (5.6) (i.e. $g = 0$), and

$$\rho_0 = \left(\frac{\pi}{4}\right)^2 \cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{\pi x_2}{2}\right) + \left(\frac{\pi}{4}\right)^2, \quad V_1 = 2\sin\left(\frac{\pi x_1}{2}\right)\sin\left(\frac{\pi x_2}{3} - \frac{\pi}{2}\right),$$
$$\widehat{\rho} = (1-t)\left(\left(\frac{\pi}{4}\right)^2 \cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{\pi x_2}{2}\right) + \left(\frac{\pi}{4}\right)^2\right)$$
$$+ t\left(\left(\frac{\pi}{4}\right)^2 \cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{3\pi x_2}{2}\right) + \left(\frac{\pi}{4}\right)^2\right).$$

The costs for different β and κ are displayed in Table 5.10. The solution of each example takes between 50 and 100 seconds.

		$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\kappa = 0$	$\mathcal{J}_{uc}$	1.58×10^{-1}	1.58×10^{-1}	1.58×10^{-1}	1.58×10^{-1}	1.58×10^{-1}
	$\mathcal{J}_c$	4.15×10^{-4}	7.74×10^{-3}	1.30×10^{-1}	1.58×10^{-1}	1.58×10^{-1}
$\kappa = 1$	$\mathcal{J}_{uc}$	2.12×10^{-1}	2.12×10^{-1}	2.12×10^{-1}	2.12×10^{-1}	2.12×10^{-1}
	$\mathcal{J}_c$	4.16×10^{-4}	1.03×10^{-2}	1.85×10^{-1}	2.12×10^{-1}	2.12×10^{-1}
$\kappa = -1$	$\mathcal{J}_{uc}$	4.03×10^{-1}	4.03×10^{-1}	4.03×10^{-1}	4.03×10^{-1}	4.03×10^{-1}
	$\mathcal{J}_c$	4.53×10^{-4}	8.65×10^{-3}	1.74×10^{-1}	3.87×10^{-1}	4.03×10^{-1}

Table 5.10: Flow control, Dirichlet: Cost when $\vec{w} = \vec{0}$ and optimal control cost for a range of κ, β. For all β, $\mathcal{J}_c \leq \mathcal{J}_{uc}$, as expected. For $\beta = 10^1$, the cost functionals differ by order of 10^{-4} for $\kappa = 0$ and $\kappa = 1$ and by 10^{-2} for $\kappa = -1$. For $\beta = 10^3$, the cost functionals differ by order of 10^{-7} for $\kappa = 0$, by 10^{-6} for $\kappa = 1$, and by order of 10^{-4} for $\kappa = -1$.

In Figure 5.5 the optimal state ρ is displayed for different values of β, and for $\kappa = -1$. At small β, the different choices of κ result in similar optimal ρ, since the control is dominant in pushing the particles to the desired state. For larger β (see Figure 5.5, bottom), $\kappa = -1$ causes a more round clustering than V_1 prescribes. The imposed V_1 and the optimal control action can be seen in Figure 5.6. The desired state describes a density moving from one uniform bump in the middle of the domain, to accumulate in a steeper, elongated shape across the x_1-axis. In this example, the effect of the different interaction strengths and the external potential on the control is clearly visible, see Figure 5.6. The external potential is large on the left side of the domain, which naturally drives the density away from this region, so that most effort of the control variable is concentrated on the right side of the domain. However, since the attractive particles oppose the density spreading out along the x_1-axis, additional control is applied on the right part of the domain. For attractive particles, the initially clumped density is a more 'natural' state (as demonstrated in Figure 5.5, bottom). In contrast, for repulsive particles, most of the work of the control is done to push the particles together.

The convergence for the Newton–Krylov algorithm for different choices of κ and β can be seen in Figure 5.7. Note that convergence for $\beta = 10^{-5}$ is noticeably slower than for the other β. This could be explained by the fact that, for $\beta = 10^{-5}$, $\vec{w}$ is allowed to be large. This can cause advection dominance, in which large gradients can form, which in turn propagate into the nonlinear PDE terms. This is harder to resolve numerically by a Newton–Krylov algorithm.

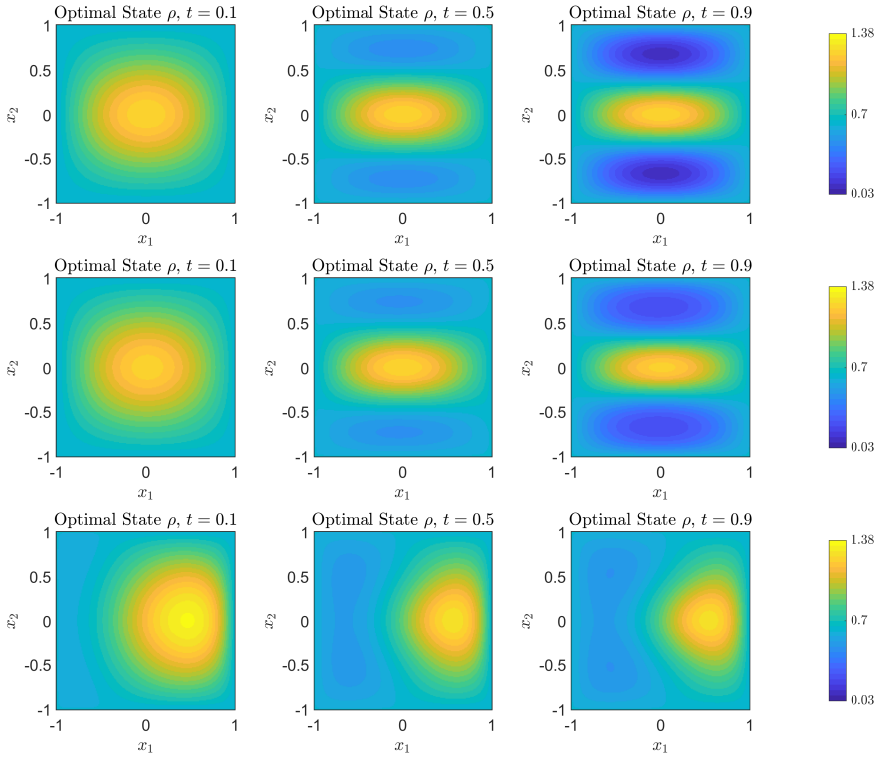

Figure 5.5: Flow control, Dirichlet: Snapshots of the optimal ρ for $\kappa = -1$ and $\beta = 10^{-5}$. $\beta = 10^{-3}$, and $\beta = 10^{-1}$ (top to bottom). Note that the optimal states for $\kappa = 0$ and $\kappa = 1$ (not shown here) look almost identical for smaller values of β, allowing the control to drive the state close to the desired state $\widehat{\rho}$. For $\beta = 10^{-1}$ less control is exerted, and so ρ is influenced more strongly by V_1 (see contour plot in Figure 5.6) and κ.

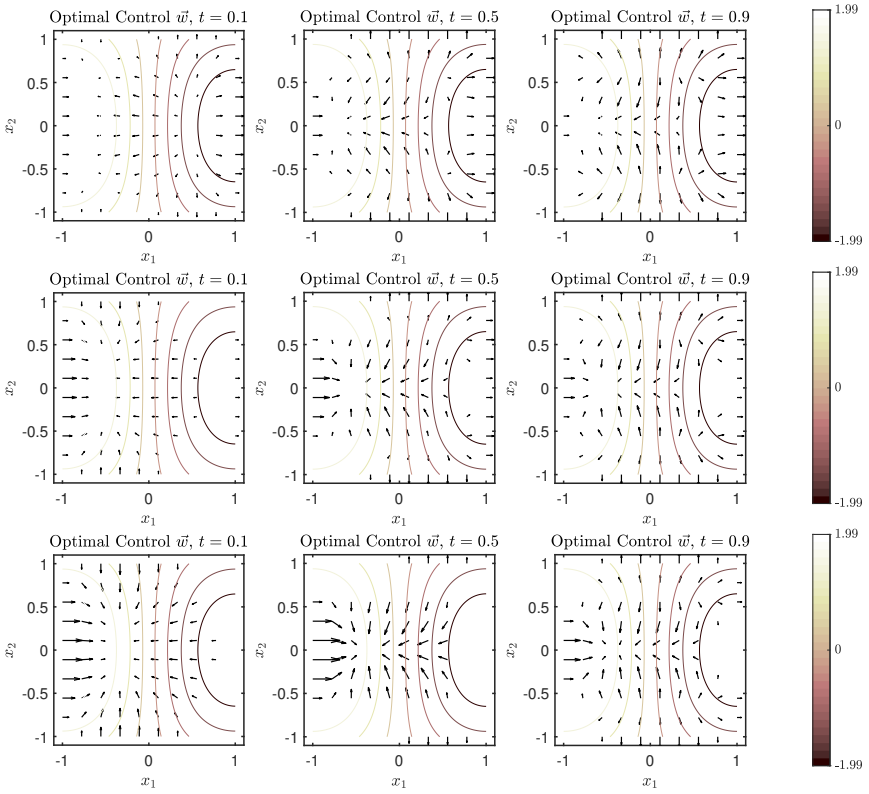

Figure 5.6: Flow control, Dirichlet: Snapshots of the optimal control for $\kappa = -1$, $\kappa = 0$ and $\kappa = 1$ (top to bottom), for $\beta = 10^{-3}$. See Figure 5.3 for further details.

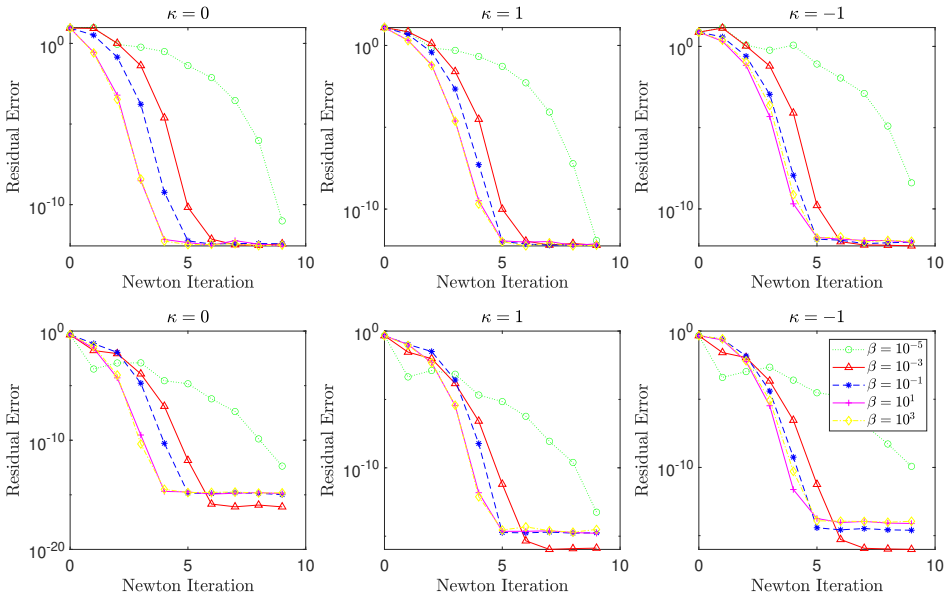

Figure 5.7: Flow control, Dirichlet: Convergence of the Newton–Krylov algorithm. Top row: convergence in the state variable for different κ. Bottom row: convergence in the adjoint variable. Convergence is measured using the residual error. Note the expected rapid convergence in all cases.

Linear (Source) Control Problem with No-Flux Boundary Conditions

We consider problem (5.2) with no-flux boundary conditions (5.7). Inputs for this example are given by

$$\rho_0 = \frac{1}{4}, \quad V_1 = \cos\left(\frac{\pi x_1}{5} - \frac{\pi}{5}\right)\sin\left(\frac{\pi x_2}{5}\right),$$
$$\widehat{\rho} = \frac{1}{4}(1-t) + t\left(\frac{1}{4}\sin\left(\frac{\pi\,(x_1-2)}{2}\right)\sin\left(\frac{\pi(x_2-2)}{2}\right) + \frac{1}{4}\right).$$

The resulting costs for different β and κ are displayed in Table 5.11.

		$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\kappa = 0$	$\mathcal{J}_{uc}$	1.90×10^{-2}	1.90×10^{-2}	1.90×10^{-2}	1.90×10^{-2}	1.90×10^{-2}
	$\mathcal{J}_c$	1.29×10^{-5}	6.65×10^{-4}	1.37×10^{-2}	1.89×10^{-2}	1.90×10^{-2}
$\kappa = 1$	$\mathcal{J}_{uc}$	1.94×10^{-2}	1.94×10^{-2}	1.94×10^{-2}	1.94×10^{-2}	1.94×10^{-2}
	$\mathcal{J}_c$	1.59×10^{-5}	7.43×10^{-4}	1.42×10^{-2}	1.93×10^{-2}	1.94×10^{-2}
$\kappa = -1$	$\mathcal{J}_{uc}$	2.03×10^{-2}	2.03×10^{-2}	2.03×10^{-2}	2.03×10^{-2}	2.03×10^{-2}
	$\mathcal{J}_c$	1.93×10^{-5}	8.17×10^{-4}	1.45×10^{-2}	2.02×10^{-2}	2.03×10^{-2}

Table 5.11: Source control, no-flux: Cost when $w = 0$ and optimal control cost for a range of κ, β. For all β, $\mathcal{J}_c \leq \mathcal{J}_{uc}$, as expected. Note that for $\beta = 10^1$, the cost functionals differ by order of 10^{-4}, while for $\beta = 10^3$ they differ by order of 10^{-7}.

The solution of each example takes between 100 and 200 seconds, apart from the $\beta = 10^{-5}$ case which takes around 25 seconds. The convergence of the Newton–Krylov algorithm for this example is shown in Figure 5.8. In all cases, convergence is already reached by the fifth iteration. Figure 5.9 shows the evolution of the optimal state ρ for different interaction strengths with $\beta = 10^{-3}$. Since β is small, the optimal state is very close to the desired state $\widehat{\rho}$ (not shown). The external potential V_1 has a clear effect on the optimal state and the control.

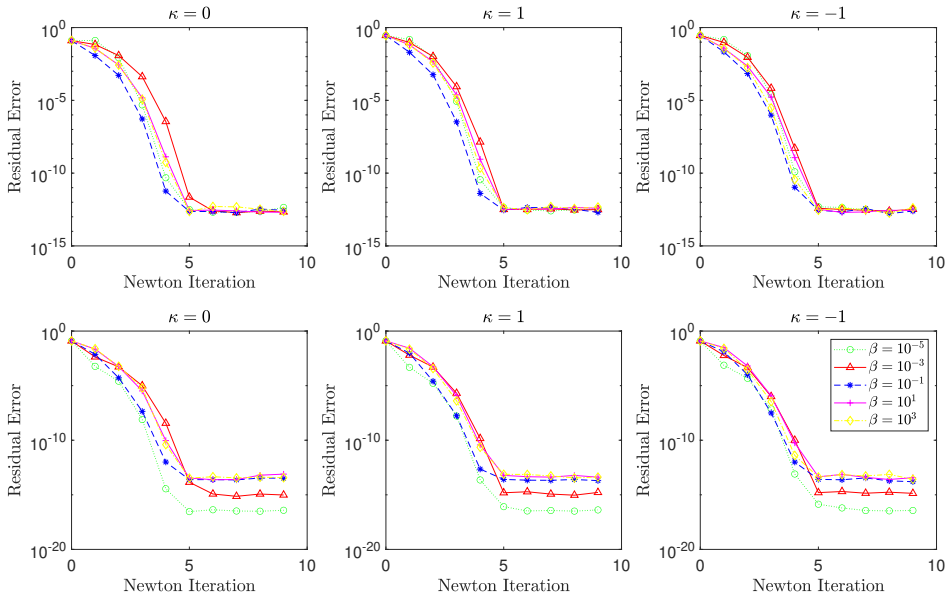

Figure 5.8: Source control, no-flux: Convergence of the Newton–Krylov algorithm. Top row: convergence in the state variable for different κ. Bottom row: convergence in the adjoint variable. Convergence is measured using the residual error. Note the expected rapid convergence in all cases.

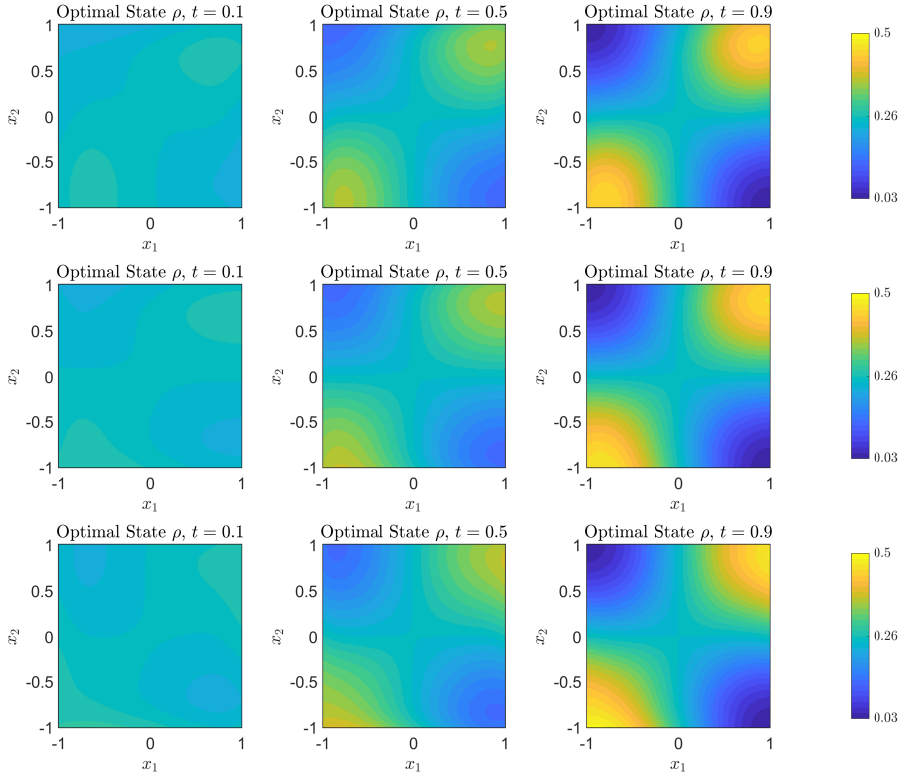

Figure 5.9: Source control, no-flux: Evolution of the optimal ρ for $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom), for $\beta = 10^{-3}$.

Since V_1 is large around $x_2 = 1$, the state is slightly asymmetric because of this effect, despite the given $\widehat{\rho}$ being symmetric. The desired state $\widehat{\rho}$ prescribes higher density near the two corners $(-1, -1)$ and $(1, 1)$. Without control or an external potential, repulsive particles accumulate on

the boundary of the domain, whilst attractive particles gather in the centre. Since the target density is higher near the boundary, less control needs to be applied for repulsive particles. For the attractive case, the particles are arranged in a rounder shape, while the repulsive particles are more spread out, as expected from their interactions.

In Figure 5.10, it is evident that more control has to be applied in the $(1, 1)$ corner, as reflected by the slightly asymmetric state discussed above. The external potential clearly counteracts the desired state, as compared to the corner $(-1, -1)$ in which V_1 is supportive of the desired state. The effect of the different interaction strengths on the control can also be observed. In the case of repulsive particles, more control has to be applied in the centre of the domain, as compared to the attractive case, in which more control is applied around the boundary.

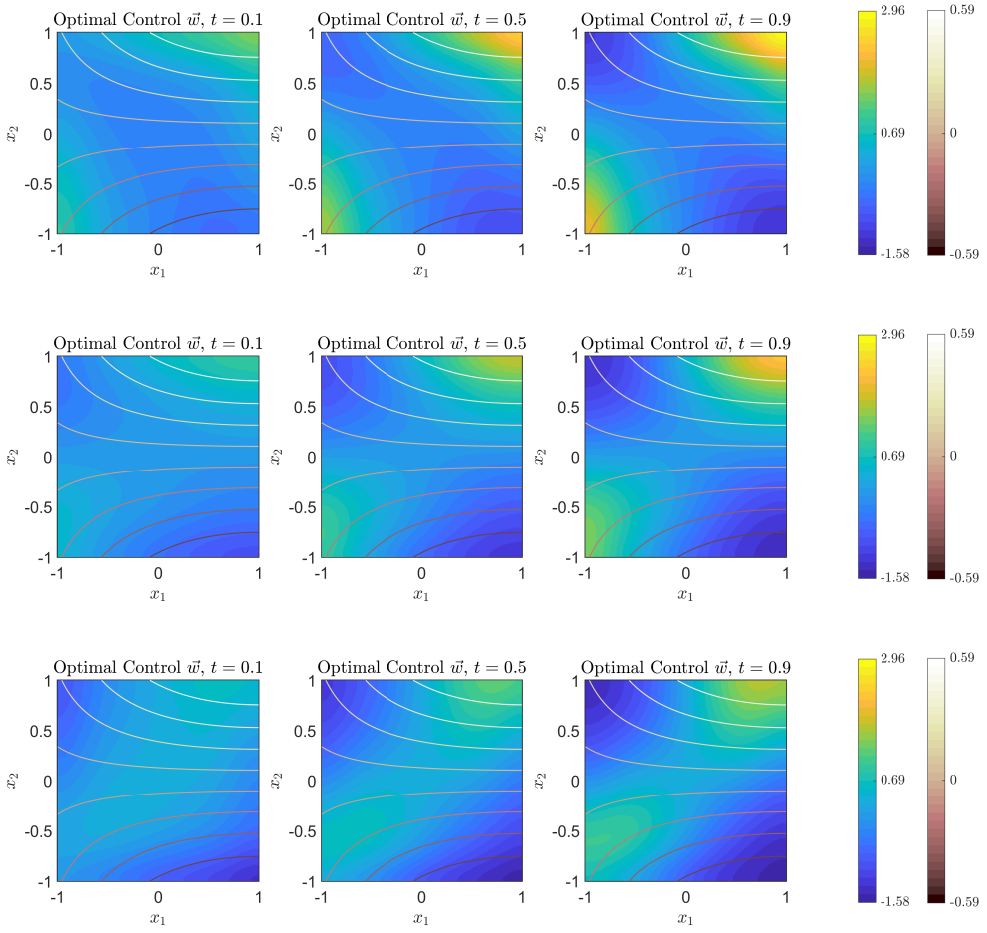

Figure 5.10: Source control, no-flux: Evolution of the optimal control w and superimposed V_1 for $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom), for $\beta = 10^{-3}$.

Linear (Source) Control Problem with Dirichlet Boundary Conditions

The final 2D problem is of the form (5.2) with Dirichlet boundary conditions (5.6). The inputs are given by

$$\rho_0 = \frac{1}{4}\cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{\pi x_2}{2}\right) + \frac{1}{4}, \quad V_1 = \frac{3}{4}(1-t)\left(-\cos\left(\frac{\pi x_1}{2}\right)\sin\left(\frac{\pi x_2}{2}\right) + 1\right),$$

$$\widehat{\rho} = (1-t)\left(\frac{1}{4}\cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{\pi x_2}{2}\right) + \frac{1}{4}\right) - t\left(\frac{1}{4}\sin\left(\pi x_1\right)\sin\left(\frac{\pi x_2}{2} - \frac{\pi}{2}\right) + \frac{1}{4}\right).$$

Here, to demonstrate yet another aspect of the method's versatility, the external potential is time-dependent. Since it decays over time, this results in the strongest effect of V_1 being visible at earlier times. The resulting costs for different β and κ can be seen in Table 5.12. The solution of each example takes between 20 and 90 seconds. The convergence of the algorithm is displayed in Figure 5.11. It can be observed that in all cases convergence only takes four iterations.

		$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\kappa = 0$	$\mathcal{J}_{uc}$	1.50×10^{-2}	1.50×10^{-2}	1.50×10^{-2}	1.50×10^{-2}	1.50×10^{-2}
	$\mathcal{J}_c$	3.40×10^{-5}	1.92×10^{-3}	1.36×10^{-2}	1.50×10^{-2}	1.50×10^{-2}
$\kappa = 1$	$\mathcal{J}_{uc}$	2.10×10^{-2}	2.10×10^{-2}	2.10×10^{-2}	2.10×10^{-2}	2.10×10^{-2}
	$\mathcal{J}_c$	4.27×10^{-5}	2.49×10^{-3}	1.85×10^{-2}	2.06×10^{-2}	2.06×10^{-2}
$\kappa = -1$	$\mathcal{J}_{uc}$	1.33×10^{-2}	1.33×10^{-2}	1.33×10^{-2}	1.33×10^{-2}	1.33×10^{-2}
	$\mathcal{J}_c$	2.88×10^{-5}	1.61×10^{-3}	1.18×10^{-2}	1.27×10^{-2}	1.27×10^{-2}

Table 5.12: Source control, Dirichlet: Cost $\mathcal{J}_{uc}$ of applying no control (i.e., $\vec{w} = \vec{0}$) and optimal control cost $\mathcal{J}_c$ for a range of values of the interaction strength κ and regularization parameter β. For all β, $\mathcal{J}_c \leq \mathcal{J}_{uc}$, as expected. Note that for $\beta = 10$, the cost functionals differ by order of 10^{-5}, while for $\beta = 10^3$ they differ by order of 10^{-7} for $\kappa = 0$ and $\kappa = 1$ and by 10^{-8} for $\kappa = -1$.

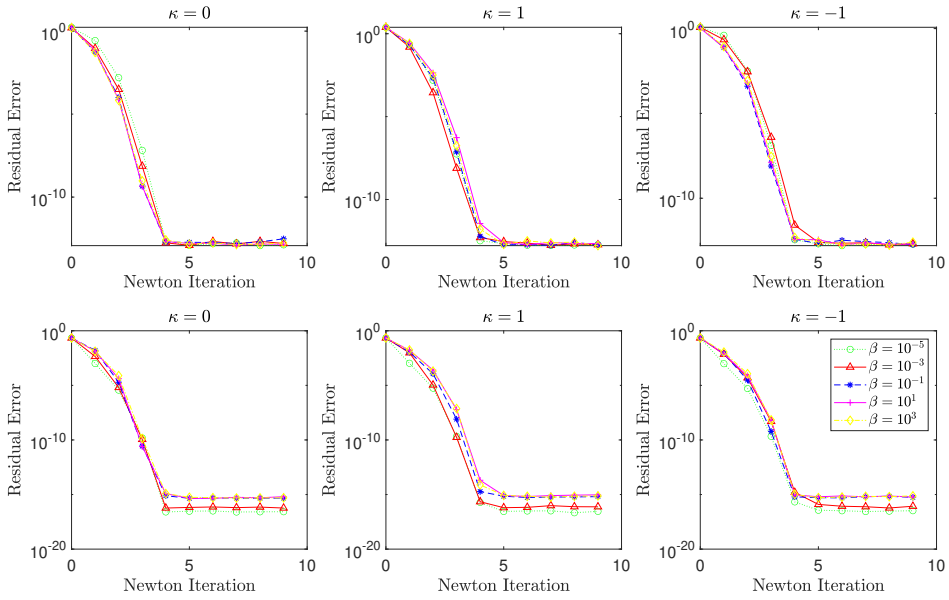

Figure 5.11: Source control, Dirichlet: Convergence of the Newton–Krylov algorithm. Top row: convergence in the state variable for different κ. Bottom row: convergence in the adjoint variable. Convergence is measured using the residual error. Note the expected rapid convergence in all cases.

The evolution of the optimal state for $\beta = 10^{-3}$ and $\kappa = -1$ is shown in Figure 5.12, which is once again close to the target state, $\widehat{\rho}$. While the optimal states for $\kappa = 0$ and $\kappa = 1$ are omitted, since they look identical for this choice of β, the corresponding optimal controls vary significantly and are shown in Figure 5.13. Since the external potential is large around the bottom half of the domain (see contour plots in Figure 5.13 for reference), the density is not centred in the middle of the domain, but shifted slightly upwards. At the same time it can be observed that at $t = 0.1$, the control is applied where the external potential is steep. At later times, the control is mostly applied where the density is prescribed to accumulate approx-

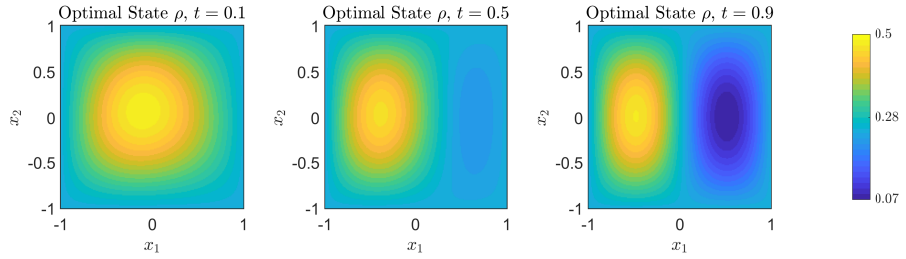

Figure 5.12: Source control, Dirichlet: Evolution of the optimal ρ for $\kappa = -1$, for $\beta = 10^{-3}$. Note that the optimal states for $\kappa = 0$ and $\kappa = 1$ look identical, due to the choice of β allowing the control to drive the state close to $\widehat{\rho}$.

imately in the form of the desired state $\widehat{\rho}$, which is at the left half of the domain. While the qualitative behaviour of the control is similar in each case, it can be seen that less control has to be applied for attractive particles as compared to repulsive ones, since the attraction causes the particles to clump together, which supports the shape of the desired state $\widehat{\rho}$.

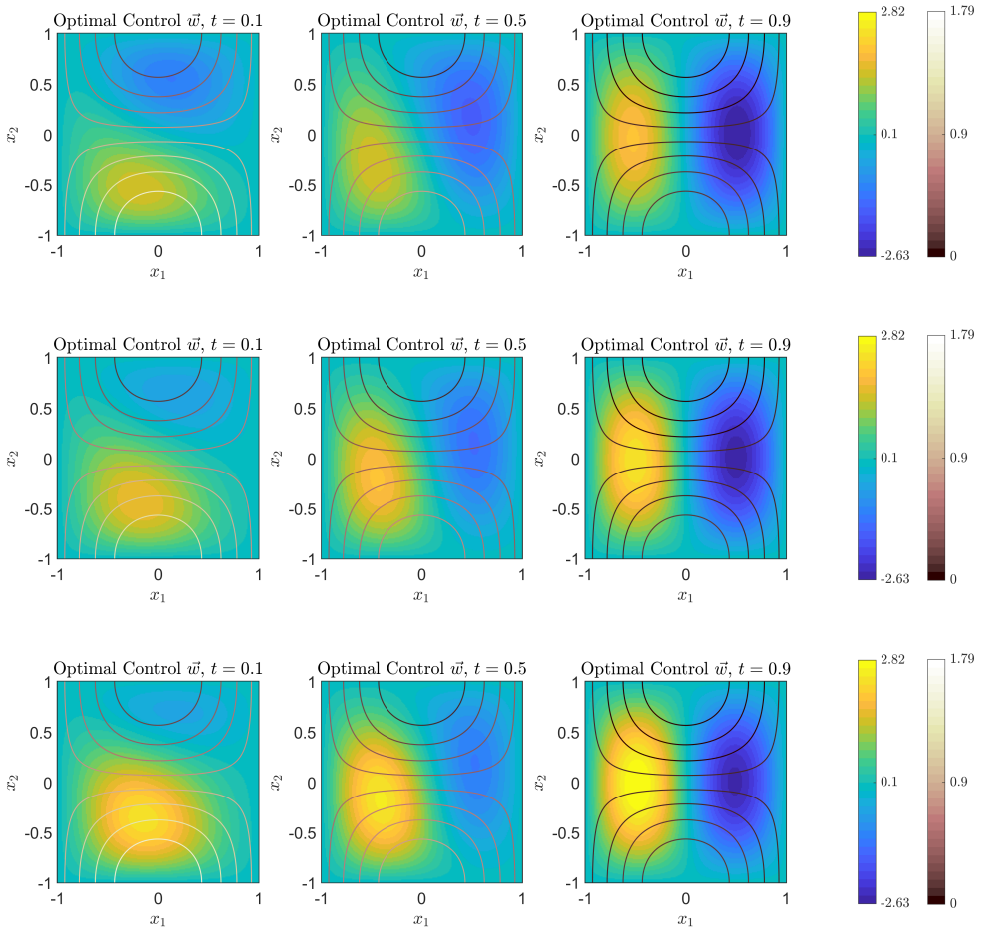

Figure 5.13: Source control, Dirichlet: Evolution of the optimal control for $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom), with $\beta = 10^{-3}$. A contour plot of the external potential V_1 is superimposed on the control plots for reference, with a corresponding colorbar on the right-hand side.

In order to illustrate the effects of the different interaction strengths, in Figure 5.14 the evolution of the optimal ρ for $\beta = 10^1$ is displayed. For this choice of β, control is largely penalized, so that the effects of V_1 and κ become dominant. Since V_1 prescribes accumulation of particles in the top half of the domain at earlier times, this is seen for all three choices of κ. However, it can be observed that the attractive particles form a much bigger cluster than the repulsive particles. Moreover, at later times, when the effect of V_1 is vanishing, for $\kappa = -1$, ρ forms a cluster in the centre of the domain, while for $\kappa = 1$, the particles accumulate at the boundary, which is exactly in line with our expectations.

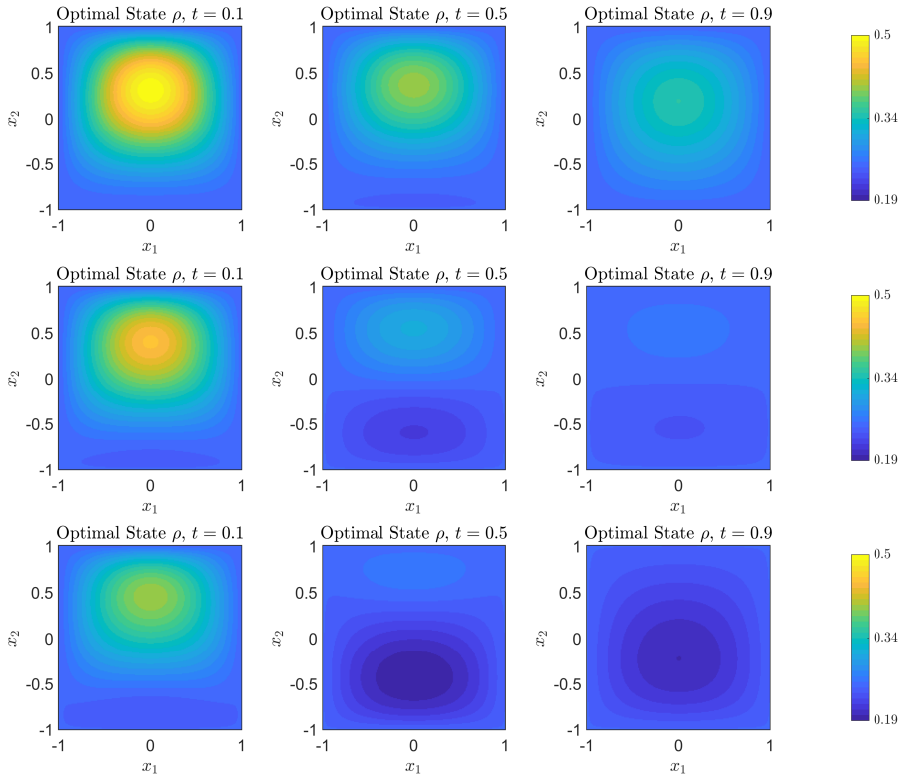

Figure 5.14: Source control, Dirichlet: Snapshots of the optimal ρ for $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom), for $\beta = 10^1$. Here, since not much control enters the system, one can see the effect of V_1 (see Figure 5.13) as well as of the different interaction strengths.

5.5.3 Solving Optimal Control Problems in Three Dimensions

We now consider three-dimensional problems, which are extremely hard to resolve numerically, due to the 'curse of dimensionality'. A bilinear flow control problem (5.8) with no-flux boundary conditions (5.10) is considered. This has been chosen as an illustrative example in three dimensions since it is both the most challenging combination of control type and boundary conditions and also the most physically relevant for applications of interest. The domain is given by $(0,T)\times\Omega = (0,1)\times(-1,1)^3$. The numbers of spatial points are $N_1 = N_2 = N_3 = 20$ and the number of time points is $n = 11$. All other parameters are chosen as in the two-dimensional case.

The chosen inputs for a first example are

$$\begin{aligned}\rho_0 &= \frac{1}{8}, \quad \widehat{\rho} = \frac{1}{8}(1-t) + t\left(\frac{\pi}{4}\right)^3 \cos\left(\frac{\pi x_1}{2}\right)\cos\left(\frac{\pi x_2}{2}\right)\cos\left(\frac{\pi x_3}{2}\right),\\ V_1 &= \left((x_1+0.3)^2-1\right)\left((x_1-0.4)^2-0.5\right)\left((x_2+0.3)^2-1\right)\left((x_2-0.4)^2-0.5\right)\\ &\quad \left((x_3+0.3)^2-1\right)\left((x_3-0.4)^2-0.5\right).\end{aligned}$$

This example is only run for $\beta = 10^{-3}$, due to a running time between 30 and 38 hours per problem, caused by the already mentioned 'curse of dimensionality'. Convergence within 6 iterations can be observed for this example, as displayed in Figure 5.17 for this choice of β only. We obtain, for $\kappa = 0$, $\mathcal{J}_c = 0.0078$. This can be compared to $\mathcal{J}_{uc} = 0.0185$ from the computed forward problem with $\vec{w} = \vec{0}$. For $\kappa = 1$ we obtain $\mathcal{J}_c = 0.0099$, compared to $\mathcal{J}_{uc} = 0.0222$ in the uncontrolled case, and for $\kappa = -1$ we have $\mathcal{J}_c = 0.0059$, with $\mathcal{J}_{uc} = 0.0149$. As expected, the optimal control leads to a cost which is significantly lower than in the uncontrolled case.

The desired state $\widehat{\rho}$ in this example is selected in order to cluster the particles in the middle of the domain, starting from a uniform distribution. The effects of the different interaction strengths and V_1 are clearly displayed in Figure 5.15, and are particularly obvious at earlier times of the particle evolution. For $t = 0.1$, the particles cluster in potential valleys. For $\kappa = -1$, the particles accumulate more in the middle of the domain, while for $\kappa = 1$ they cluster in the bottom corners. The top corners are less preferred due to the shape of V_1. It is evident in Figure 5.16 that attractive particles enhance the control in pushing the density into a cluster in the middle of the domain, as prescribed by the desired state $\widehat{\rho}$, while more control is needed for a similar effect in the repulsive setup. This is highlighted by the colour and size of the arrows symbolizing the control.

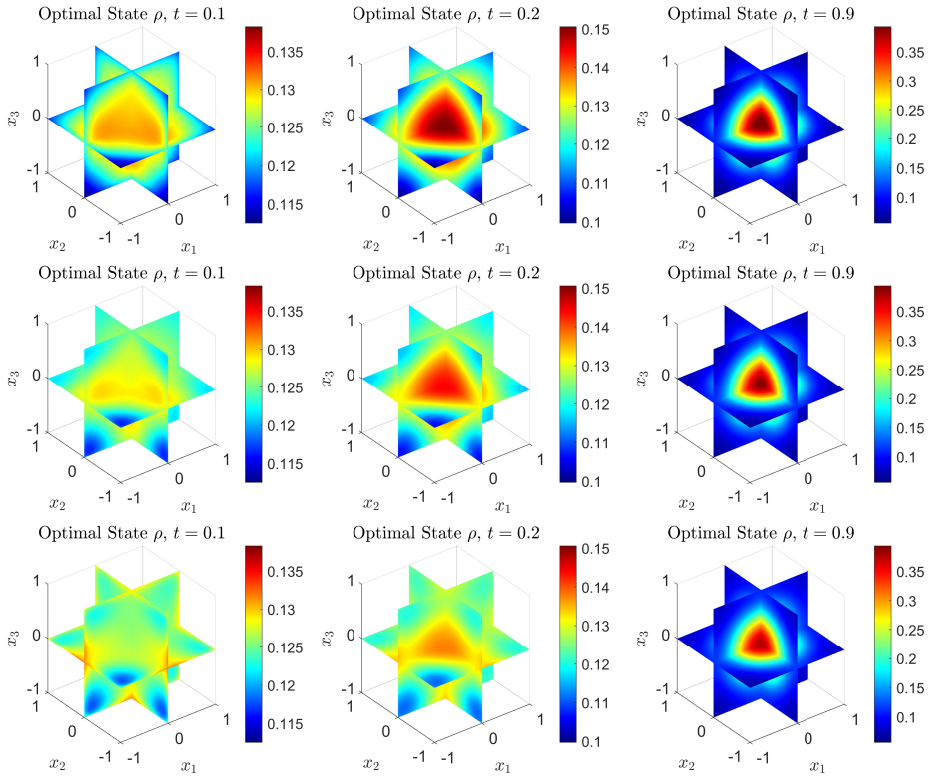

Figure 5.15: 3D flow control, no-flux: Evolution of the optimal state ρ for $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom), for $\beta = 10^{-3}$.

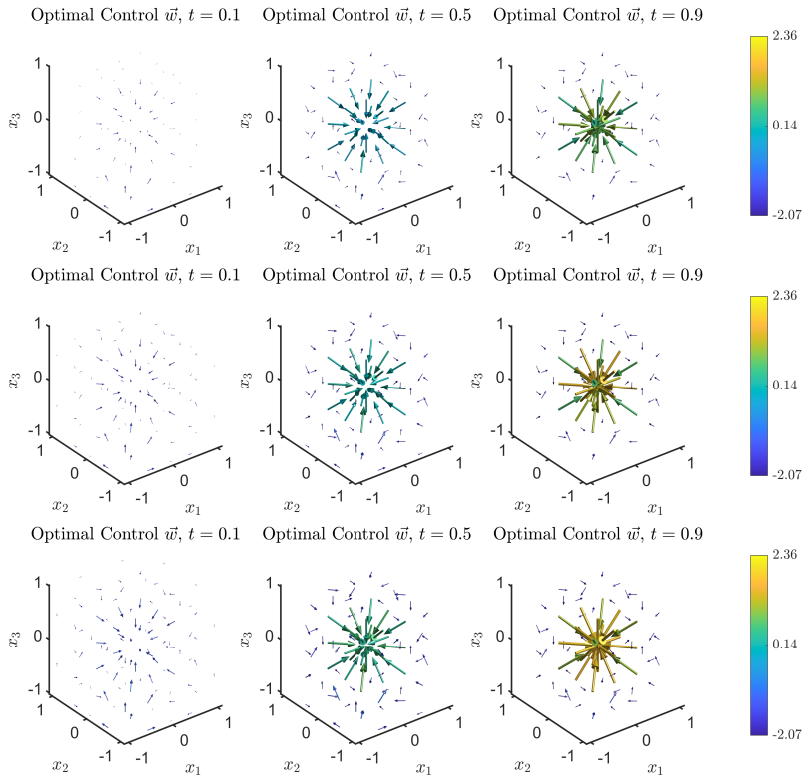

Figure 5.16: 3D flow control, no-flux: Evolution of the optimal control $\vec{w}$ for $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom), for $\beta = 10^{-3}$. Note that the colour and size of the arrows indicate the strength of the control.

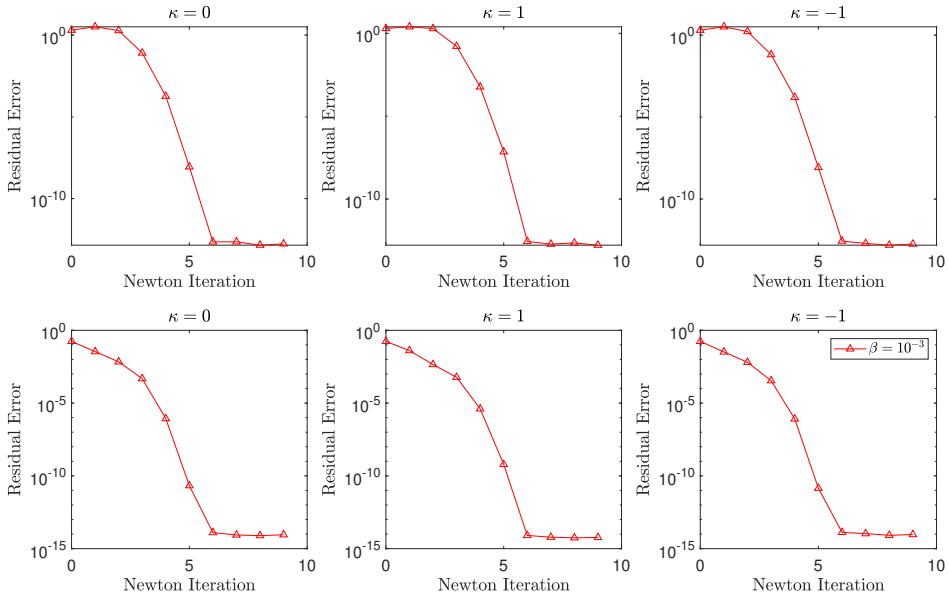

Figure 5.17: 3D flow control, no-flux: Convergence of the Newton–Krylov algorithm for $\beta = 10^{-3}$. Top row: convergence in the state variable for different κ. Bottom row: convergence in the adjoint variable. Convergence is measured using the residual error. Note the expected rapid convergence in all cases.

For a second three-dimensional example, included for illustration, the input choices are

$$\rho_0 = \frac{1}{8}, \quad V_1 = \frac{\pi}{16}^2 \sin(\pi x_1) \cos\left(\frac{\pi x_2}{2}\right) \cos\left(\frac{\pi x_3}{2}\right),$$
$$\widehat{\rho} = \frac{1}{8}(1-t) + t\left(\frac{\pi}{32}^2 \sin(\pi x_1)^2 \cos\left(\frac{\pi x_2}{2}\right) \cos\left(\frac{\pi x_3}{2}\right) + \frac{\pi^3}{128} \cos\left(\frac{\pi x_1}{2}\right) \cos\left(\frac{\pi x_2}{2}\right) \cos\left(\frac{\pi x_3}{2}\right)\right).$$

We get, for $\kappa = 0$, $\mathcal{J}_c = 0.0100$. This can be compared to $\mathcal{J}_{uc} = 0.0183$ from the computed forward problem with $\vec{w} = \vec{0}$. For $\kappa = 1$ we obtain $\mathcal{J}_c = 0.0118$, compared to $\mathcal{J}_{uc} = 0.0213$ in the uncontrolled case, and for $\kappa = -1$ we have $\mathcal{J}_c = 0.0084$, with $\mathcal{J}_{uc} = 0.0156$. As expected, the optimal control leads to a cost which is significantly lower than in the uncontrolled case. The resulting optimal state can be seen in Figure 5.18 for the different interaction strengths. It can be observed that the particles cluster more in the volume $x_1 < 0$, since there is a potential valley in this location. However, $\widehat{\rho}$ requires the particles to accumulate in the centre of the volume $x_1 > 0$ as well. It is evident that this is most successful for attractive particles, since the attractive force supports the desired state. By contrast, repulsive particles counteract the prescribed desired state, and hence the cluster in the volume $x_1 > 0$ is smaller in this case. The optimal control acts to drive the particles to the desired state, as displayed in Figure 5.19. Most control is applied in the volume $x_1 > 0$, since the external potential is steep in this area.

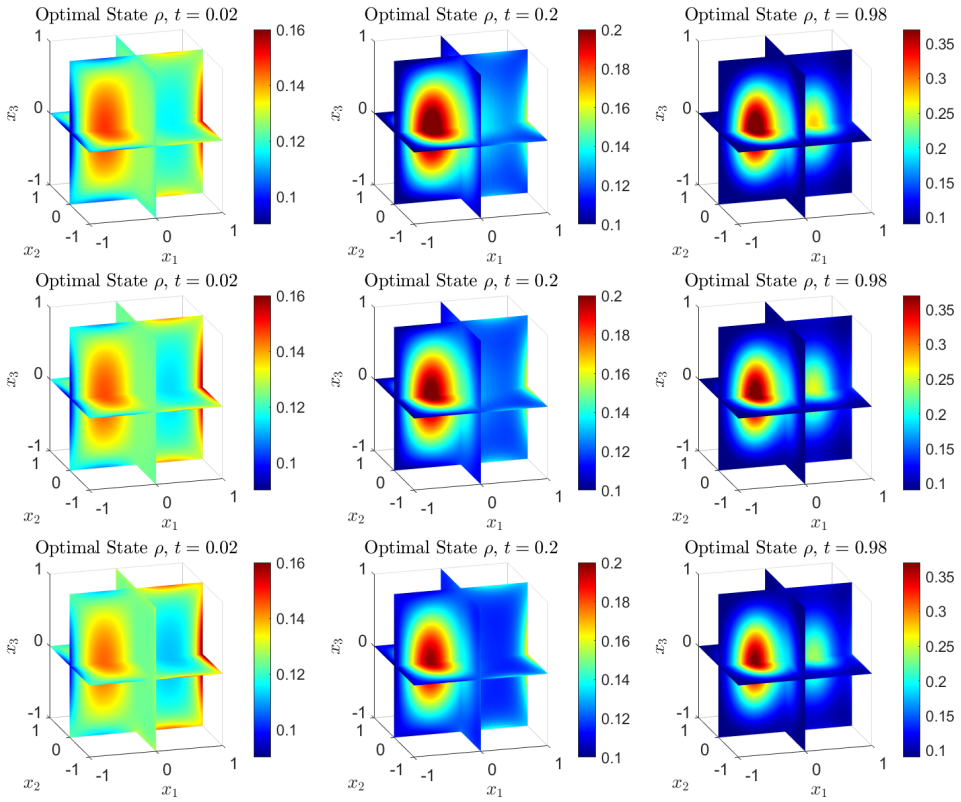

Figure 5.18: 3D flow control, no-flux: Evolution of the optimal state ρ for $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom), for $\beta = 10^{-3}$.

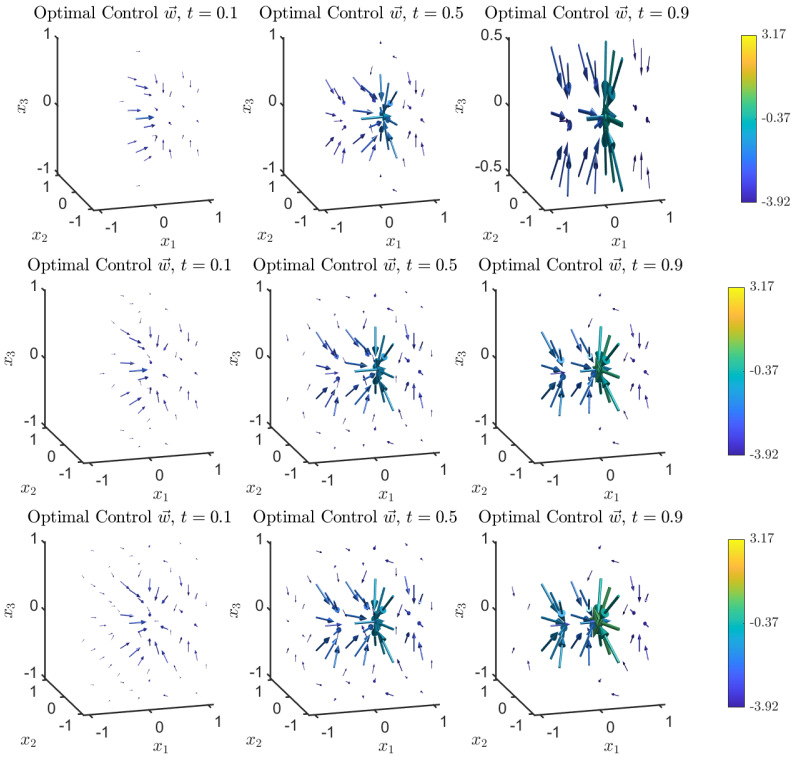

Figure 5.19: 3D flow control, no-flux: Evolution of the optimal control $\vec{w}$ for $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom), for $\beta = 10^{-3}$. Note that the colour and size of the arrows indicate the strength of the control.

Chapter 6

MultiShape: A Spectral Element Method

Many problems in the applied sciences, including biology, chemical engineering, and physics can be described by (integro-)PDE models. These include wide-ranging applications of industrial relevance, including those in drug delivery [186], manufacturing [187, Chapter 2], and the food industry [188]. In the previous two chapters we have developed a numerical method to solve such models, and introduced a corresponding optimal control framework. However, many of the applications mentioned above cannot be captured accurately by solutions on simple rectangular or periodic domains, but need to be solved on complicated domains, to include relevant features of the problem setup. Examples of such applications are brewing, microfluidics [189], and milling processes in the pharmaceutical industry [190]. Therefore, efficient numerical methods for solving integro-PDEs on complicated domains are of ever-increasing interest to academic and industrial communities [191, 192, 193, 194, 195, 196].

In this chapter, we introduce an extension to the numerical method in order to incorporate the solution of integro-PDEs and optimal control problems on complicated domains. This is done using a Spectral Element Method, which we call *MultiShape*. The work in this chapter is part of the preprint [2]. The chapter is organized as follows: In Section 6.1 we review relevant literature for the method, in Section 6.2, necessary shapes for the method are introduced, namely a quadrilateral and a wedge, which is a section of an annulus. Section 6.3 is concerned with the implementation of the spectral element method, while Section 6.4 provides relevant validation tests. Finally, Section 6.5 demonstrates the success of the spectral element implementation in solving DDFT models and optimal control problems.

6.1 Survey of Related Work

Perhaps the most popular numerical scheme for solving PDE problems on complicated domains is the finite element method (FEM) [103], which is both an active area of applied mathematics research and frequently used in a wide range of applications [191], with several commercial and open-source software solutions available, such as FEniCS [197, 198] and Abaqus [199]. However, for a large class of PDE models, there is another effective numerical method—the pseudospectral method—which, in particular for modelling problems that involve nonlocal phenomena, can be more efficient and accurate than the FEM for a comparable computational cost. We are particularly interested in applications in which nonlocal phenomena enter the model through, for example, convolution terms. For these models, FEM becomes computationally expensive because one of its key advantages, the frequently obtained sparse matrix structures, is generally lost. Pseudospectral methods, however, do not rely on sparsity and nonlocal terms can be treated without significant additional computational cost. This has been discussed in detail in Chapter 3.

A natural extension of pseudospectral methods is the spectral element method (SEM), which can be seen as a combination of a pseudospectral method with a higher-order finite element method. The basis functions used by SEM, such as Chebyshev or Lagrange polynomials, are typically of higher order than those used in FEM. Additionally, SEM generally uses Chebyshev–Lobatto collocation grids on each element, as compared to a more standard equispaced grid in FEM. At the intersections between the elements, continuity is enforced by imposing two matching conditions, usually continuity of the solution and its first derivative normal to the intersection, for PDE problems that are second order in space; see [93, Chapter 22]. There are a number of different, but related, spectral element methods. One can consider the strong form of the PDE and apply matching conditions on the intersections of the elements directly. This is called the patching method, and was first introduced by Orszag [200]. The Galerkin spectral element method solves the weak form of a PDE model, and was pioneered by Patera [201] using Chebyshev polynomials as basis functions, and later adapted to Lagrange polynomials by Komatitsch and Vilotte [202], which is now the standard choice [93, Chapter 22]. As shown in [203], for elliptic PDEs this approach is essentially equivalent to the patching formulation. A third method for solving a PDE using SEM is based on overlapping elements and was introduced by Morchoisne [204].

Spectral element methods have been applied to solve a range of problems. Some of the earlier applications were in (classical) fluid dynamics (see [95, Chapter 13] and [192, Chapter 5]) and geosciences [202, 205]. Further applications of SEM include fractional PDEs [206], relativity [207, 208], financial modelling [196, 209, 210], and optimal control problems with elliptic PDE constraints [211, 212, 213]. Often SEM is applied as a higher-order finite element method, using a large number of elements with low degree polynomials. However, some work has been done on domain decomposition and 'multidomain' methods, e.g., [207], that actively apply SEM to describe domains that are more complicated. Examples of these include modelling blood flow in veins, aircraft engines, or flows around airfoils [192, Chapter 5], flows around obstacles [193], flows in petroleum reservoirs [214], seismic waves [202, 205], and sound propagation [215]. While most existing work does not include nonlocal integral terms, some advances have been made, see e.g., [193, 195, 196, 216]. These works consider different, mostly one-dimensional, integral terms, which are tackled in the context of the (weak form) Galerkin SEM formalism, using either Gaussian quadrature or Fast Fourier Transforms. Additionally, the chosen boundary conditions do not involve convolution integral terms, unlike those for the problems considered here.

The present work develops a multidomain spectral element method, called MultiShape[1], including an open-source software package [217], which is able to solve various (integro-)PDE problems on complicated domains. This implementation includes the application of complicated boundary conditions, such as (nonlinear, nonlocal) no-flux conditions, as well as the efficient solution of integro-PDEs, involving the evaluation of convolution integrals. For the problems considered in the present work, this is a significant improvement to the approaches taken in the previous works mentioned above. The work follows from the pseudospectral code library 2DChebClass [84], extending this framework to a spectral element method, with minimal additional effort for the user. Since, in this thesis, we are interested in solving DDFT models and optimal control problems involving such models, we note that in DDFT, complicated domains are often modelled by imposing steep potential gradients in parts of the domain, e.g., to model flow through a constriction [218], or around obstacles [66, 219], which can be computationally expensive due to large gradients in the system. Here we demonstrate that spectral element methods are a valuable alternative to this method and solutions to DDFT problems on complicated domains using SEM are illustrated in Section 6.5.

6.2 The MultiShape Elements

In this section we introduce two additional shapes to the ones presented in Section 3.3, which will serve as elements for the spectral element method. These are chosen so that a combination

[1] We apply the convention that 'MultiShape', with capital letters, refers to our methodology as a whole, while a lower-case 'multishape' refers to an individual shape (i.e., domain).

of these elements can model a range of complicated domains, and, in particular, industrially relevant domains, such as pipes, funnels, and vessels.

6.2.1 The Quadrilateral

Most of the approach is similar to the construction of a two-dimensional box introduced in Section 3.3.3. In contrast to the box discretization, the set of vertices $\{Y_i\} = \{(y_1^i, y_2^i)\}$, $i = 1, 2, 3, 4$, has to be provided, as well as the number of discretization points in each direction, N_1 and N_2. The computational domain is $[-1, 1]^2$, which can be mapped to an arbitrary quadrilateral $[a, b] \times [c, d]$, $a, b, c, d \in \mathbb{R}$. An example of a quadrilateral domain can be seen in Figure 6.1.

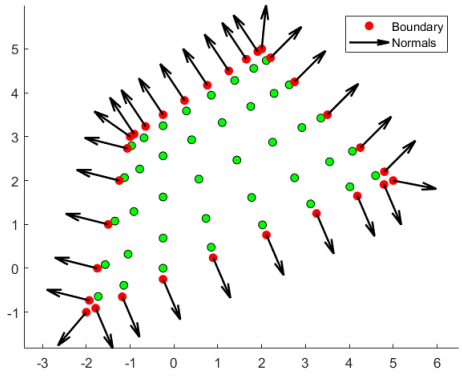

Figure 6.1: Quadrilateral discretization for $N_1 = 10$ and $N_2 = 7$ discretization points. The red points symbolize the domain boundary, while the arrows indicate outward normals on the boundary.

Discretization Points and Domains

As in the case of the box discretization, one has to construct the coordinates of the two-dimensional grid using Kronecker products. This gives a set of all coordinates $(\mathbf{x}_1^K, \mathbf{x}_2^K)$, see (3.72). The points $(\mathbf{x}_1^K, \mathbf{x}_2^K)$ in the computational domain $[-1, 1]^2$ can be mapped to an arbitrary quadrilateral shape $[a, b] \times [c, d]$, discretized via $(\mathbf{y}_1^K, \mathbf{y}_2^K)$. We use the superscript k to indicate that $x^k \in \mathbf{x}^K$, for $k = 0, 1, ..., (N_1 - 1) \times (N_2 - 1)$. In contrast to the box, we cannot use the linear maps from the 1D line, in each coordinate, but have to use a *bilinear map* as follows. We first apply a linear map from $[-1, 1]^2$ to $[0, 1]^2$

$$x_1^k = \frac{x_1^k + 1}{2}, \quad x_2^k = \frac{x_2^k + 1}{2}.$$

Then the bilinear maps

$$\begin{aligned} y_1^k &= \alpha_1 + \alpha_2 x_1^k + \alpha_3 x_2^k + \alpha_4 x_1^k x_2^k, \\ y_2^k &= \beta_1 + \beta_2 x_1^k + \beta_3 x_2^k + \beta_4 x_1^k x_2^k, \end{aligned} \tag{6.1}$$

take the points on the computational domain to the ones on the physical domain. The α_i, β_i are determined by mapping the coordinates of the four corners of the computational domain in a cyclic order onto the corners of the quadrilateral. The Jacobian of this bilinear map is also stored by the code in this step, as well as the Hessians with respect to both variables.

Most of the time we are given the coordinates of the quadrilateral in physical space, so that we have to map back to the computational domain. In order to do this, we again need to first find the values of the parameters α_i and β_i and then invert the bilinear map to solve for the

set of points on the computational domain. We define a matrix B, with $A = BY$, such that $A_i = [\alpha_i, \beta_i]$, where $i = 1, 2, 3, 4$, based on the bilinear maps (6.1). The matrix is defined as

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}.$$

Once we know the values of the parameters, we can solve the inverse map

$$x_1^k = \frac{y_1^k - \alpha_1 - \alpha_3 x_2^k}{\alpha_2 + \alpha_4 x_2^k}$$

and the quadratic

$$a\left(x_2^k\right)^2 + b x_2^k + c = 0,$$

where

$$\begin{aligned} a &= \alpha_4\beta_3 - \alpha_3\beta_4, \\ b &= \alpha_4\beta_1 - \alpha_1\beta - 4 + \alpha_2\beta_3 - \alpha_3\beta_2 + y_1^k\beta_4 - y_2^k\alpha_2, \\ c &= \alpha_2\beta_1 - \alpha_1\beta_2 + y_1^k\beta_2 - y_2^k\alpha_2. \end{aligned}$$

Doing this provides the values for the x_1^k and x_2^k on $[0, 1]^2$. In order to map to the computational domain $[-1, 1]^2$, we apply a final linear map

$$x_1^k = 2x_1^k - 1, \quad x_2^k = 2x_2^k - 1.$$

One can furthermore compute the Jacobian of the transformation between the physical and computational space. It is defined as

$$J = \begin{bmatrix} \frac{\partial \mathbf{y}_1^K}{\partial \mathbf{x}_1^K} & \frac{\partial \mathbf{y}_1^K}{\partial \mathbf{x}_2^K} \\ \frac{\partial \mathbf{y}_2^K}{\partial \mathbf{x}_1^K} & \frac{\partial \mathbf{y}_2^K}{\partial \mathbf{x}_2^K} \end{bmatrix}, \tag{6.2}$$

where each of these entries are vectors, so that J is a three-dimensional array.

Boundaries and Normals

Boundaries are found in the same way as for the box discretization. Normals are also constructed in an equivalent way. The only aspect that has to be taken into consideration is how to effectively sort the four corner indices of the quadrilaterals, so that the definitions of left, right, top and bottom make sense, and, as a result, that the normal vectors are defined as outward facing normals. The issue with not doing this is demonstrated in Figure 6.2. It compares the sorting of indices according to the code for a box, described in Section 3.3.3, with the new procedure described below. One can observe that the sorting procedure for a box is not adequate for the given example. The four corners are identified wrongly, and not in a clockwise order, which results in a discretized domain that is not convex and overlaps itself. With the new sorting procedure for sorting the corner indices, described below, this issue is rectified. In order to sort quadrilateral indices effectively, the vertices $\{Y_i\} = \{(y_1^i, y_2^i)\}$, $i = 1, 2, 3, 4$, are sorted according to y_2^i and the smallest of those is chosen as a reference point. If there are two smallest y_2 values, the leftmost is chosen, by consulting the corresponding y_1 value. Then the angles between the minimum y_2 value $y_2^{\min}$ and the other vertices is calculated as

$$\theta_i = \frac{180}{\pi} \arctan(y_2^i - y_2^{\min}, y_1^i - y_1^{\min}).$$

This angle satisfies $0 \leq \theta_i < 180$, for $i = 1, 2, 3, 4$. Once the indices are sorted, the normal vectors are defined as described in Section 3.3.3.

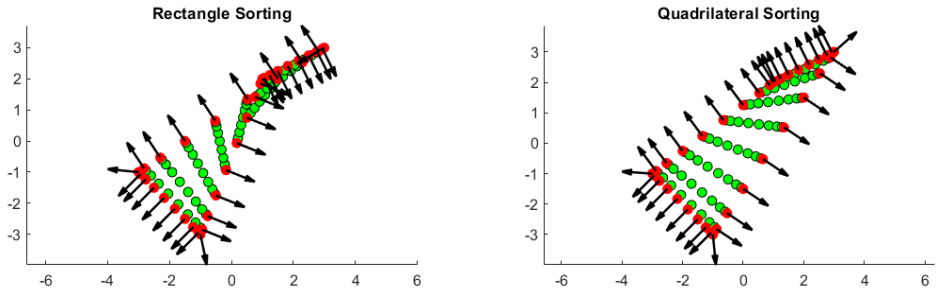

Figure 6.2: Demonstration of an issue arising when sorting quadrilateral indices in the same way as rectangle indices. On the left the result of sorting using the process described in Section 3.3.3 is displayed, while the right plot shows the result of the new sorting strategy.

Interpolation

Interpolation in two dimensions is based on the idea of Kronecker tensor products. The interpolation matrices, which we define as Interp_1 and Interp_2, are computed for each set of points $\{x_1^j\}$ on the x_1-axis and $\{x_2^i\}$ in the x_2 direction, as explained in Section 3.3.1. The Kronecker product of these two matrices is taken, resulting in the two-dimensional interpolation matrix. For an example with $i = j = 0, 1, 2$, this is the block matrix of size 9×9, compare to (3.76).

Differentiation

The differentiation matrices in computational space are constructed as in the box discretization. The only difference that has to be taken into account for a quadrilateral is the mapping to the physical domain, since this is now defined by a bilinear map. This is done by applying the Jacobian (6.2) to the differentiation matrix. We compute $\tilde{J} = \left(J^\top\right)^{-1}$ and obtain the gradient operator

$$\text{Grad} = \left[\text{diag}\left(\tilde{J}_{11}\right) D_{x_1} + \text{diag}\left(\tilde{J}_{12}\right) D_{x_2}, \text{diag}\left(\tilde{J}_{21}\right) D_{x_1} + \text{diag}\left(\tilde{J}_{22}\right) D_{x_2}\right]^\top.$$

The divergence operator is defined in an equivalent way, and the construction of the Laplacian follows in a similar manner. The computational implementation of the Laplacian is more complicated than for the gradient and divergence operations, involving the Hessian matrices of the bilinear map.

Integration and Convolution

Integration and convolution are defined in the same way as for the box discretization. The only difference in creating the integration vector in physical space is to apply the determinant of the Jacobian (6.2), since the integration vectors are computed on the computational grid.

6.2.2 The Wedge

A wedge is a domain defined in polar coordinates, which can be thought of as a section of an annulus defined by two angles and two radii. Therefore, to initialize the construction of it in the code, inner and outer radii, maximum and minimum angles, as well as the origin of the wedge have to be given. Furthermore, the number of discretization points for each direction has to be specified. The key idea for this domain is that it can be mapped conformally to the square $[-1, 1]^2$, which is exactly the computational domain that we consider for two-dimensional discretizations. A wedge discretization is illustrated in Figure 6.3.

Discretization Points and Domains

A wedge is defined in polar coordinates by an inner and outer radius r as well as a minimum and maximum angle θ_1 and θ_2. While θ_1 should be chosen to lie in $[0, 2\pi]$, the coordinate θ_2 is mapped into this principal domain by computing

$$\theta_2 = \theta_1 + d, \quad \text{where} \quad d = \theta_2 - \theta_1 \mod 2\pi.$$

The wedge can be mapped conformally to a computational domain $[-1, 1]^2$ in polar coordinates. For this the Kronecker vectors (3.72) for the sets of computational points $\{x_1^k\}$ and $\{x_2^k\}$ are taken. The Kronecker points $\mathbf{x}_1^K$ and $\mathbf{x}_2^K$ can then be mapped to the physical domain in r and θ using the linear map (3.62) and then back to the computational domain using the map (3.63) from the one-dimensional code. The derivatives of the map (3.62) for both variables are stored in the Jacobian matrix J.

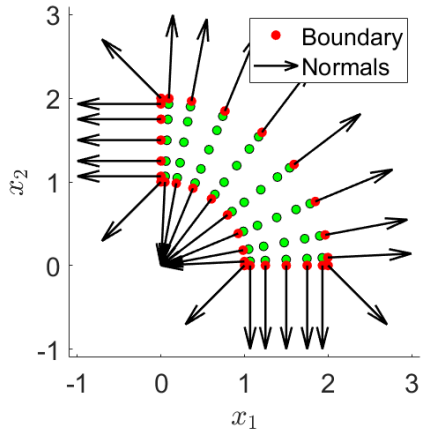

Figure 6.3: Wedge discretization for $N_1 = 7$ and $N_2 = 10$ discretization points. The red points symbolize the domain boundary, while the arrows indicate outward normals on the boundary.

Boundaries and Normals

The boundaries of this shape are found in an identical manner to the quadrilateral case, since this is based on the computational grid. The definition of the normals in polar coordinates is straightforward, since they are simply defined in the radial and angular directions. On the side $\theta = \theta_2$ we have $\vec{n} = [1, 0]$ and for the side $r = r_2$ we have $\vec{n} = [0, 1]$. On the boundaries where $\theta = \theta_1$ and $r = r_1$ we obtain $\vec{n} = [-1, 0]$ and $\vec{n} = [0, -1]$, respectively.

It can be useful to compute the normals in Cartesian coordinates, for example for plotting or for applications on a multishape domain, where Cartesian and polar shapes are considered simultaneously. For this we have to consider the four faces of the shape, the inner and outer radii r_1, r_2 and inner and outer angle θ_1 and θ_2. For the outer radius, we get $m_{r_2,1} = I$ and $m_{r_2,2} = \text{diag}(\theta)$, where $\vec{n}_{r_2} = [m_{r_2,1}, m_{r_2,2}]$ denotes the normal vector on the face on which r_2 is constant. Equivalently, for the side where θ_2 is constant, we have $m_{\theta_2,1} = I$ and $m_{\theta_2,2} = \text{diag}(\theta) + \text{diag}(\pi/2)$. For the inner radius we obtain $m_{r_1,1} = I$ and $m_{r_1,2} = \text{diag}(\theta) + \text{diag}(\pi)$. For the side where θ_1 is constant we have $m_{\theta_1,1} = I$ and $m_{\theta_1,2} = \text{diag}(\theta) + \text{diag}(3\pi/2)$. For each face we have $\vec{n}_{(\cdot)} = [m_{(\cdot),1}, m_{(\cdot),2}]$, where $(\cdot) = r_1, r_2, \theta_1, \theta_2$. The first component, which corresponds to the radial variable, of each normal $m_{r_2,1}$, $m_{r_1,1}$, $m_{\theta_2,1}$, $m_{\theta_1,1}$ is taken, to define a set n_1. Any points that do not lie on the boundary are deleted. For the second component of the normals, extra care has to be taken at the corners of the shape to fix the angles. The computation for each pair of angles is explained in Algorithm 3. Then all components $m_{r_2,2}$, $m_{r_1,2}$, $m_{\theta_2,2}$, $m_{\theta_1,2}$, with the update at the corners are stacked to give n_2. We define $\vec{n}_{\text{Polar}} = [n_1, n_2] = [n_{r_1}; n_{\theta_1}; n_{r_2}; n_{\theta_2}]$,

and apply the standard transformation from polar to Cartesian coordinates to obtain the normal vectors $\vec{n}$ for the wedge in Cartesian coordinates.

Algorithm 3: Determining angles of normals

Given two angles θ_1 and θ_2 at the corner of the wedge, we normalize to obtain a resulting angle θ_c. Set $\phi = \theta_2 - \theta_1$
if $|\phi| \leq \pi$ **then**
  $\theta_c = \theta_1 + \phi/2$
else if $\pi < \phi$ *and* $\phi < (3/2)\pi$ **then**
  $\theta_c = \theta_1 + (2\pi - \phi)/2$
else
  $\theta_c = \theta_1 - (2\pi - \phi)/2$
end
Set $\theta_c = \theta_c \mod 2\pi$.

Interpolation and Differentiation

Since interpolation is done on the computational grid this is equivalent to the discussion in Section 3.3.3.

In order to derive the differentiation matrices for the wedge, we start with the one-dimensional differentiation matrices described in Section 3.3.1. Since these are derived on the computational domain, we have to map them to the physical domain. This is done for each coordinate individually so that we have

$$D_r = J_1 D \otimes I, \quad D_\theta = I \otimes J_2 D,$$

where D is the differentiation matrix for the one-dimensional domain and J_1 and J_2 are the corresponding Jacobians of the mappings from the computational to the physical domain. For each discretization point $j = 0, ..., N-1$ we have $J_1^j = \frac{dr^j}{dx_1^j}$ and $J_2^j = \frac{d\theta^j}{dx_2^j}$. However, this will result in differentiation matrices with respect to the coordinates r and θ. The standard definitions of gradient, divergence, and Laplacian in polar coordinates have to be used to derive their discretized counterparts. For $r \neq 0$ we obtain

$$\begin{aligned}
\text{Grad} &= \left[D_r, \frac{1}{r} D_\theta \right]^\top, \\
\text{Div} &= \left[D_r + \frac{1}{r}, \frac{1}{r} D_\theta \right], \\
\text{Lap} &= D_{rr} + \frac{1}{r} D_r + \frac{1}{r^2} D_{\theta\theta}.
\end{aligned}$$

At $r = 0$, we need to treat these operators more carefully, since they involve terms of the form $\frac{1}{r}$. The derivatives $\frac{\partial^j}{\partial \theta^j}$, $j = 1, 2, ...$, applied to a function f are required to satisfy $\left.\frac{\partial^j f}{\partial \theta^j}\right|_{r=0} = 0$, so that the solution is well defined at the origin. Then, the term $\frac{1}{r}\frac{\partial f}{\partial \theta}$ has indeterminate form, since both the top and bottom of the fraction tend to zero as $r \to 0$. Therefore, L'Hôpital's rule can be applied, so that $\lim_{r\to 0} \frac{1}{r}\frac{\partial f}{\partial \theta} = \lim_{r\to 0} \frac{\partial^2 f}{\partial r \partial \theta}$ and we hence obtain Grad $\approx [D_r, D_{r\theta}]^\top$ at $r = 0$. Such an argument has been made in, e.g., [220]. In order to deal with terms involving $\frac{1}{r}$ in the divergence operator, one has to show that such terms are of indeterminate form, so that L'Hôpital's rule can be applied. In [220] it is argued, without proof, that this condition must be satisfied, since otherwise terms involving $\frac{1}{r}$ would blow up as $r \to 0$. In [221, 222], it is then shown explicitly that the divergence operator in polar coordinates indeed satisfies this condition. Therefore, at $r = 0$, we obtain Div $\approx [2D_r, D_{r\theta}]$. Finally, we have to consider the Laplacian

operator. If one follows the heuristic argument in [220], one obtains Lap $\approx 2D_{rr} + \frac{1}{2}D_{\theta\theta}D_{rr}$ at $r = 0$. One can instead combine the results for gradient and divergence operators derived above to obtain Lap $\approx 2D_{rr} + D_{\theta\theta}D_{rr}$ at $r = 0$, which is presumably more accurate. However, since we assumed that the derivatives with respect to θ are zero at $r = 0$, we can conclude that the two approaches are equivalent.

Integration and Convolution

The integration vector is found in the same way as discussed in Section 3.3.3. However, since this is an integral in polar coordinates, in a last step the integration vector has to be multiplied pointwise by r to satisfy the polar integration formula $\int\int f(r,\theta)rdrd\theta$.

The convolution matrix is computed in the same way as in Section 3.3.3, now using the polar version of the integration vector. Note that the convolution we are computing is

$$(\rho \star \tilde{\chi})(r,\theta) = \int\int \tilde{\chi}(r-r',\theta-\theta')\rho(r',\theta')r'dr'd\theta',$$

where $\tilde{\chi}$ is the polar version of the weight function χ.

6.3 Implementation

Now that the building blocks, or elements, are introduced, we can discuss the implementation of the spectral element method in MATLAB. First, since MultiShape is a class in MATLAB, the class methods are introduced. Then the user interface is explained. Note that the computational complexity going from one to two shapes doubles from computing a problem of dimension M to $2M$. This is in contrast to increasing the dimension of the problem, in which we multiply by a factor of M. If a one-dimensional problem has size M, then a two-dimensional problem is M^2, and a three-dimensional problem is M^3, which illustrates the curse of dimensionality mentioned previously.

We solve the strong form of (3.59) given by

$$\frac{\partial\rho}{\partial t} = \nabla^2\rho + \nabla\cdot(\rho\nabla V_1) - \nabla\cdot(\rho\vec{w}) + \nabla\cdot\int_\Omega \rho(t,\vec{x})\rho(t,\vec{x}')\nabla V_2(|\vec{x}-\vec{x}'|)d\vec{x}' \quad \text{in } (0,T)\times\Omega, \tag{6.3}$$

with associated boundary conditions, on each of the elements. Matching conditions (patching) are applied at the intersection between two elements, and the PDE boundary conditions are applied on the multishape domain boundary, see Figure 6.4 for reference. The matching conditions between elements are

$$\begin{aligned} \rho_i(t,\vec{x}) &= \rho_j(t,\vec{x}) && \text{on } (0,T)\times\partial\Omega_{i,j}, \\ -\vec{\mathrm{j}}_i\cdot\vec{n}_i &= \vec{\mathrm{j}}_j\cdot\vec{n}_j && \text{on } (0,T)\times\partial\Omega_{i,j}, \end{aligned} \tag{6.4}$$

where $\partial\Omega_{i,j}$ denotes the intersection boundary between elements e_i and e_j, $i,j = 1,...,n_e$, with n_e the number of elements. Quantities of the form ρ_i denote the value of ρ on element i. In line with work on SEM for (classical) fluid dynamics problems, e.g., [192, Chapter 5], and since the system (6.3) is of interest in complex fluid dynamics, it is a natural choice to match the flux $\vec{\mathrm{j}}(\rho)$ at the intersection of two elements, as compared to directly matching the first derivative of ρ. For problems of the form (3.59) these two approaches are essentially equivalent. The minus sign in the the flux matching condition arises due to the opposite orientation of (outward) normals for adjoining elements.

In order to solve optimal control problems of the form (5.2) or (5.8), introduced in Chapter 5 on a multishape, matching conditions have to be applied to the state equation, as described in (6.4). We furthermore need to apply matching conditions for the corresponding adjoint

equation. These are given by

$$\begin{aligned} q_i(t,\vec{x}) &= q_j(t,\vec{x}) && \text{on } (0,T)\times\partial\Omega_{i,j}, \\ -\nabla q_i\cdot\vec{n}_i &= \nabla q_j\cdot\vec{n}_j && \text{on } (0,T)\times\partial\Omega_{i,j}, \end{aligned}$$

irrespective of the control type and boundary conditions chosen.

6.3.1 Class Methods

In this section, technical aspects of the MultiShape implementation, as a class in MATLAB, are discussed, such as the construction of differentiation and interpolation matrices, integration vectors, convolution operators, and the application of intersection conditions, all of which are defined as class methods. While many quantities can be computed on the individual elements first, a key part of the implementation is that all operators can be applied to the whole multishape domain simultaneously, which is described in the following. Note that, since quadrilateral and wedge elements are defined in two different coordinate systems, Cartesian and polar, the corresponding operators are defined in terms of two different coordinate systems as well.

In line with the philosophy of the MultiShape code, it draws on the 2DChebClass implementation for single shapes discussed in Section 3.3. Discretizing a function ρ on a multishape domain is done by discretizing ρ on each element individually through 2DChebClass, and stacking the resulting vectors into a vector of size $M\times 1$. Here $M=\sum_{i=1}^{n_e} N_1^{(i)}N_2^{(i)}$, where $N_1^{(i)}$, $N_2^{(i)}$ are the number of points for element e_i in the the x_1 and x_2 directions, respectively, and n_e the number of elements in the multishape. The resulting discretized function is

$$\boldsymbol{\rho}_{M\times 1}=\begin{bmatrix}\boldsymbol{\rho}_1\\ \boldsymbol{\rho}_2\\ \vdots\\ \boldsymbol{\rho}_{n_e}\end{bmatrix},$$

where $\boldsymbol{\rho}_i$ is a $N_1^{(i)}N_2^{(i)}\times 1$ vector, for $i=1,...,n_e$. The same approach is taken when constructing the integration vector for the multishape, which is a row vector of size $1\times M$.

The gradient, divergence, and Laplacian operators for a multishape are constructed by computing the operators on each individual element, using 2DChebClass, and arranging them in a block diagonal matrix, as demonstrated below for an operator D acting on a function (vector) $\boldsymbol{\rho}$

$$D\boldsymbol{\rho}=\begin{bmatrix}D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_{n_e}\end{bmatrix}\begin{bmatrix}\boldsymbol{\rho}_1\\ \boldsymbol{\rho}_2\\ \vdots\\ \boldsymbol{\rho}_{n_e}\end{bmatrix}.$$

Note that empty blocks in the above matrix denote zeros. Computations can then be carried out directly on the multishape, instead of on each element individually, since the stacking ensures that the order of elements is preserved. A matrix D_i corresponding to element e_i is applied only to $\boldsymbol{\rho}_i$, corresponding to the same element, within the matrix–vector multiplication. Note that D_i and D_j could be defined in two different coordinate systems, depending on whether, for example, e_i and e_j are quadrilateral or wedge elements.

Interpolating a function ρ from one grid onto another is another crucial function of the MultiShape implementation. Accurate interpolation is relevant for plotting of results, computation of convolutions with finite support, and evaluation of quantities on subdomains, as well as comparing results evaluated on different grids. Many of these applications only require the interpolation matrices in computational space $[-1,1]^2$ for each element individually, which can

simply be stacked, as illustrated above for the matrix D, and applied to a vector $\boldsymbol{\rho}$. However, mapping a function ρ from a set of points $\mathbf{y}_{\text{in}}$ onto another arbitrary set of points $\mathbf{y}_{\text{out}}$, which is defined in physical (not computational) space, is more complicated than stacking the individual interpolation matrices and applying these to $\boldsymbol{\rho}(\mathbf{y}_{\text{in}})$. The reason for this is that it is not clear a-priori in which element of the multishape the different points within $\mathbf{y}_{\text{out}}$ lie. The code is equipped to determine this automatically for the user, by identifying which of the $\mathbf{y}_{\text{out}}$ lie in which shape. Once this is established, the interpolation matrix for each element e_i can be used to interpolate onto the set of points in that element, $\mathbf{y}_{i,\text{out}}$.

In order to apply boundary and matching conditions we need to identify the boundaries of the multishape and the intersections between individual elements. To aid illustration, Figure 6.4 displays a multishape, consisting of two elements, with marked intersection boundary, multishape boundary, and normal vectors to the boundary. 2DChebClass is able to automatically compute the boundaries of a single element e_i, $i = 1, ..., n_e$. It furthermore distinguishes the four parts, or faces, of the boundary, i.e., 'top', 'bottom', 'left', and 'right' for a quadrilateral; inner and outer radial and maximum and minimum angular boundaries for a wedge. We denote these faces by $f_{i,k}$, $k = 1, 2, 3, 4$. In order to find the intersections between two elements, we have to check whether the points on face $f_{i,k}$ of element e_i are equal to those of face $f_{j,l}$ of element e_j for each possible combination of elements and faces. The ordering of the points lying on face $f_{i,k}$ can also be the reverse of the order of points on $f_{j,l}$, which MultiShape checks automatically. Then, the multishape boundary can be found by subtracting those intersection boundaries from the union of boundaries of the individual elements, leaving only those parts of the element boundaries that are not adjacent to another element. In this way, the patching method is implemented – note that, in principle, it is possible to modify this approach and admit the intersecting faces of two elements to have different numbers of points, then use MultiShape's interpolation function to match these. However, since a similar number of points is generally expected in neighbouring elements, this additional complexity is omitted in the current implementation.

Constructing the normal vectors on the boundary of the multishape, which are needed to compute no-flux boundary conditions and intra-element matching conditions, is a little more complicated. The starting point is again to consider the normals for each individual element, provided by 2DChebClass, and in particular those lying on the boundary of the multishape. However, at the intersection corners between two elements, these normal vectors are not defined consistently. In this case the code considers the normal vectors on the multishape boundary to both sides of the intersection corner and averages them. In order to do this correctly, all normals first have to be converted into Cartesian coordinates, averaged, and converted back into polar coordinates where appropriate. When the discretization of the multishape is more complicated, taking the average of adjacent normals may not be sufficient, and the resulting outward normal may not be sensible. Since this can be very specific to the discretization at hand, the option to override any of the normal vectors manually is given. Such a case is demonstrated in Figure 6.5.

The computation of convolution integrals is the final part of the MultiShape implementation that we discuss here. Since computing a convolution integral is a global operation, we cannot rely on reusing the computations on individual elements. However, the general steps are equivalent to the approach for a single element given by (3.77), using the integration vector of the multishape. All vectors $\mathbf{y}_1^K$, $\mathbf{y}_2^K$ are defined in Cartesian coordinates for this computation.

6.3.2 User Interface

This section illustrates how a multishape is set up by the user and describes the information that needs to be provided in order to solve PDE problem (3.59) on the multishape using the DAE solver `ode15s` in MATLAB [120, 121]. In order to set up a multishape, each element has to be defined, as well as an order in which they are arranged, from 1 to n_e, although this ordering is essentially arbitrary. For each element the number of discretization points, N_1 in the x_1-direction, N_2 in the x_2-direction, needs to be specified. To uniquely determine a shape, for a quadrilateral element it suffices to specify the corners; a wedge element is defined in polar

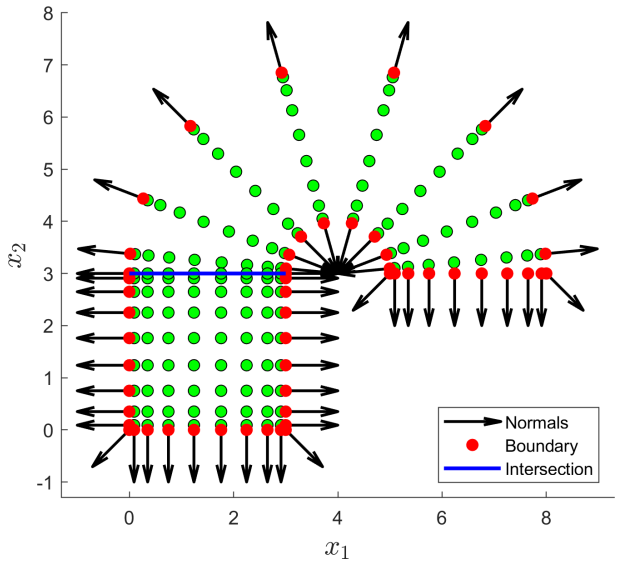

Figure 6.4: A multishape consisting of a quadrilateral and a wedge element. Displayed are collocation points, highlighted boundary points, the intersection boundary between the two shapes, and outward normal vectors.

coordinates, by specifying an inner and outer radius, a minimum and maximum angle, and the location of the origin.

In PDE problems such as (3.59), matching conditions are applied at the intersections between two elements e_i and e_j. However other conditions can be applied at the intersection boundary between two elements. For example, the MultiShape library additionally supports the application of no-flux boundary conditions between elements, modelling a hard wall, and the inclusion of other matching options into the code library is straightforward. Which condition to apply on each intersection between elements e_i and e_j, can be specified simply by setting a flag. The code demonstrating how to set up a multishape, and provide the necessary geometrical and numerical information, is found in Listing 6.1. The resulting multishape can be seen in Figure 6.4. Note that, in order to create a functioning multishape, the number of points of the neighbouring faces of two elements have to match, as discussed in Section 6.3.1. Once the multishape is set up, required operators and properties, such as differentiation matrices or the location of boundary points, can be simply extracted from the multishape structure, as demonstrated in Listing 3.1.

Using MultiShape to discretize a PDE and apply boundary and intersection conditions is straightforward. A discretized version of problem (3.59) can be defined by

$$M\boldsymbol{\rho}'(t) = \mathbf{f}(\boldsymbol{\rho}(t), t),$$

where the mass matrix M for a collocation method is diagonal, and $\mathbf{f}$ is the discretization of the right hand side of the PDE. This can be compared to the approach in Section 3.3.6. The boundary and matching conditions are applied by setting the relevant (diagonal) entries in the mass matrix to zero. Discretizing the right-hand side of the PDE, $\mathbf{f}(\boldsymbol{\rho}(t), t)$, is illustrated in Listing 3.2. The differentiation and convolution matrices are constructed automatically by the MultiShape library and can be applied to a vector $\boldsymbol{\rho}$ using matrix–vector multiplication. The intersection conditions are applied via an inbuilt function, which uses the intersection boundaries that are automatically identified during construction of the multishape. The quantities to be provided are the solution, $\boldsymbol{\rho}$, and the flux, $\mathrm{j}(\boldsymbol{\rho})$, that are to be matched. Similarly, the code automatically identifies the boundary of the multishape, so that the boundary conditions of the PDE can be applied in the same way as for single shapes in 2DChebClass. Note that the nonlocal no-flux boundary conditions of problem (3.59) are applied in Listing 3.2 in a single line of

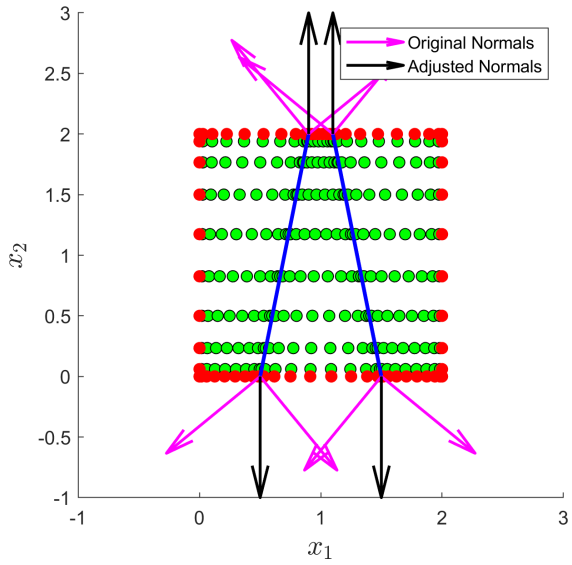

Figure 6.5: Overriding the normal vectors at intersection corners, which are automatically constructed by MultiShape, to ensure they are defined in the normal direction to the multishape face on which the intersection corner is located.

code (line 21), demonstrating the ease of implementation of complicated boundary conditions. Additionally, this line can be replaced to apply other boundary conditions, such as Dirichlet or Robin conditions. Note that several 'helper functions' are defined within MultiShape, which support straightforward implementation. These include masks to access individual elements of the multishape, computation of inner products between two vectors, and the application of intersection boundary conditions, as discussed in Section 6.3.1.

In order to solve the model problem (3.59) using the MATLAB DAE solver `ode15s` [120, 121], one needs to specify the right hand side of the discretized PDE as illustrated above, an initial condition, a time interval, and a mass matrix, as well as the solver tolerances. This is illustrated in Listing 6.4. Note that the codes in Listings 6.2, 6.3, and 6.4 are independent of the specific multishape defined in Listing 6.1. If one were to compute the same PDE system on a different multishape, the only part of the code that needed to be amended would be in Listing 6.1, demonstrating the modularity of the code. Note that in order to use the Newton–Krylov method introduced in Section 5.4 with the MultiShape methodology, one only needs to apply the matching conditions between the elements in the functions given to the solver and swap out the spatial discretization of one shape with the MultiShape discretization. These features are not demonstrated in a listing, in the interest of brevity.

Listing 6.1: Setting up a multishape

```
1   % Number of points (note N2x1 = N1x1 since the numbers of points need
2   % to match at intersection boundaries)
3   N1x1 = 12; N1x2 = 15; N2x1 = N1x1; N2x2 = 15;
4
5   % For a quadrilateral:
6   % - specify corners and number of points
7   x1Min = 0; x2Min = 0; x1Max = 3; x2Max = 3;
8   geom1.Y = [x1Min,x2Min;x1Min,x2Max;x1Max,x2Max;x1Max,x2Min];
9   geom1.N = [N1x1;N1x2];
10  % - assign the geometry to be shape 1; specify that it is a quadrilateral
11  shapes(1).geom = geom1; shapes(1).shape = 'Quadrilateral';
12
13  % For a wedge:
14  % - specify inner/outer radius, minimum/maximum angle, and origin
15  rMin = 1; rMax = 4; thetaMin = 0; thetaMax = pi;
16  geom2.R_in = rMin; geom2.R_out = rMax;
```

```
    geom2.th1 = thetaMin; geom2.th2 = thetaMax;
    geom2.Origin = [4,3];
    geom2.N = [N2x1;N2x2];
    % - assign the geometry to be shape 2; specify that it is a wedge
    shapes(2).geom = geom2; shapes(2).shape = 'Wedge';

    % Match intersections between shape 1 and shape 2
    BCs = struct; BCs(1,2).function = 'BCmatch';
    % Construct multishape, given the shapes
    MS = MultiShape(shapes);
```

Listing 6.2: Extracting operators

```
    % Extract pre-computed operators from multishape MS: this improves
    % performance of the ODE solve, since accessing structures in MATLAB
    % is comparatively slow
    grad = MS.Diff.grad; div = MS.Diff.div; Lap = MS.Diff.Lap; Int = MS.Int;
    bound = MS.Ind.bound; normal = MS.Ind.normal;
    intersections = MS.Ind.intersections;
    x1 = MS.Pts.y1_kv; x2 = MS.Pts.y2_kv;

    % Compute convolution matrix, for a kernel defined by a function V2Fun
    Conv = MS.ComputeConvolutionMatrix(@V2Fun,true);
```

Listing 6.3: Discretizing the right-hand side of the PDE system

```
    % Right-hand side function, including boundary & intersection conditions
    function drhodt = drho_dt(t,rhoab)
    % Extract the individual species
    rhoa = rhoab(1:end/2,:); rhob = rhoab(end/2+1:end,:);

    % Define discretized PDE using known external potential vector
    % (Vext), differentiation matrices (grad, div), and convolution
    % matrices between species a and b (Convaa, Convab, Convbb, Convba)
    rhoaflux = Flux(rhoa,rhob,Convaa,Convab);
    drhoadt  = -div*rhoaflux;
    rhobflux = Flux(rhob,rhoa,Convbb,Convba);
    drhobdt  = -div*rhobflux;

    % Apply matching conditions at intersections
    dataa.rho = rhoa; dataa.flux = rhoaflux;
    drhoadt = MS1.ApplyIntersectionBCs(drhoadt,dataa,BCs);
    datab.rho = rhob; datab.flux = rhobflux;
    drhobdt = MS1.ApplyIntersectionBCs(drhobdt,datab,BCs);

    % Apply no flux boundary conditions at boundary
    drhoadt(bound) = normal*rhoaflux; drhobdt(bound) = normal*rhobflux;
    drhodt = [drhoadt;drhobdt];
    end
    function rhoFlux = Flux(rhoa,rhob,Convaa,Convab)
    % Stack rho - this is now a size 2M x 1 vector
    rhoa2 = MS.MakeVector(rhoa,rhoa); rhob2 = MS.MakeVector(rhob,rhob);

    rhoFlux = -(grad*rhoa + rhoa2.*(grad*Vext) + rhoa2.*(grad*(Convaa*rhoa)) + ...
        rhoa2.*(grad*(Convab*rhob)));
    end
```

Listing 6.4: Solving a PDE system on a multishape using MATLAB's DAE solver

```
    % Solving a system of PDEs using MATLAB's inbuilt DAE solver
    % Set the mass matrix
    mM = ones(size(x1));   % vector of ones, of size M x 1
    % M = length of discretized spatial domain
    mM(bound) = 0;         % set to zero at boundaries
    mM(intersections) = 0; % set to zero at intersections
    MIn = diag([mM;mM]);   % diagonal matrix to account for system of 2 PDEs
```

```
8
9       % Set options for ODE solver: relative & absolute tolerance, mass matrix
10      opts = odeset('RelTol',10^-9,'AbsTol',10^-9,'Mass',MIn);
11      % Solve ODE, given the following:
12      % - vector of time points comp_times
13      % - initial condition rhoab_ic
14      % - right-hand side function 'drho_dt', computed in Listing 3
15      [outTimes, rhoab_t] = ode15s(@drho_dt,comp_times,rhoab_ic,opts);
```

6.4 Validation of the MultiShape Methodology

This section contains validation tests for the MultiShape implementation, investigating the accuracy of the differentiation, integration, interpolation, and convolution operations. We further validate the methodology by computing the solution to a Poisson equation with Dirichlet boundary conditions. All of these tests have been carried out by comparing the numerical results to analytic solutions. In all tests exponential convergence is observed, until a certain threshold is reached, which is mostly dictated by machine precision, and to a lesser extent by compounding precision or interpolation errors.

Most errors in this section are calculated using a relative numerical L^2 norm defined as

$$\mathcal{E} = \frac{||f_{\text{Num}} - f_{\text{Ex}}||_{L^2}}{||f_{\text{Ex}}||_{L^2} + 10^{-10}}, \tag{6.5}$$

where f_{Num} and f_{Ex} correspond to numerical and exact solutions, and the additional (arbitrary) small term in the denominator is added to avoid the possibility of division by zero – the results do not change significantly for other reasonable values. The interpolation error is calculated in the same way but replacing the L^2 norm by a (vector) l_∞ norm. This is because the error is calculated on a uniform grid and not a Chebyshev grid. Therefore, the computation of the L^2 norm would require numerical integration in uniform space, which would introduce larger integration errors than the Clenshaw–Curtis integration formula [119] implemented in MultiShape does. Since integration results in a single number, for such computations, the L^2 norm in (6.5) is replaced by taking absolute values, with f_{Num} and f_{Ex} the numerical and exact integrals, respectively. All tests in this section are carried out on Dell PowerEdge R430 running Scientific Linux 7, four Intel Xeon E5-2680 v3 2.5GHz, 30M Cache, 9.6 GT/s QPI 192 GB RAM.

6.4.1 Validation of Discretized Operators

The multishapes considered in this validation section are discretizations of boxes and wedges. This is so that comparisons can be made with the 2DChebClass implementation for a single shape. The different discretizations considered for a variety of test cases can be seen in Figures 6.6 and 6.7. The four discretizations of a box in Figure 6.6 include a reference multishape, discretizations with varying number of elements, uneven splits of the domain, as well as a discretization that contains an interior point, at which four elements intersect, (d). The wedge discretizations in Figure 6.7 include a reference multishape, a multishape that is discretized in the radial direction as well as one in the angular direction, and a discretization containing three elements.

The three differentiation operators, that is gradient, divergence, and Laplacian, are validated by comparing against an exact solution. The test function considered is

$$g_1(x_1, x_2) = \exp\left(-\frac{(x_1x_2 - 1)^2}{20}\right)\sin\left(\frac{x_1x_2}{4}\right),$$

which is also used to test the accuracy of the interpolation matrix. This is tested by interpolating $g_1(x_1, x_2)$ onto a (uniform) target grid defined by $x_{1,U}$, $x_{2,U}$ and comparing the result to g_1

evaluated directly on the target grid. The choice of g_1 is largely arbitrary; similar results were obtained for a wide range of functions. This is also the case with the functions chosen below for other validation tests. For integration we choose the (Gaussian) test function

$$g_2(x_1, x_2) = \exp\left(-x_1^2 - x_2^2\right).$$

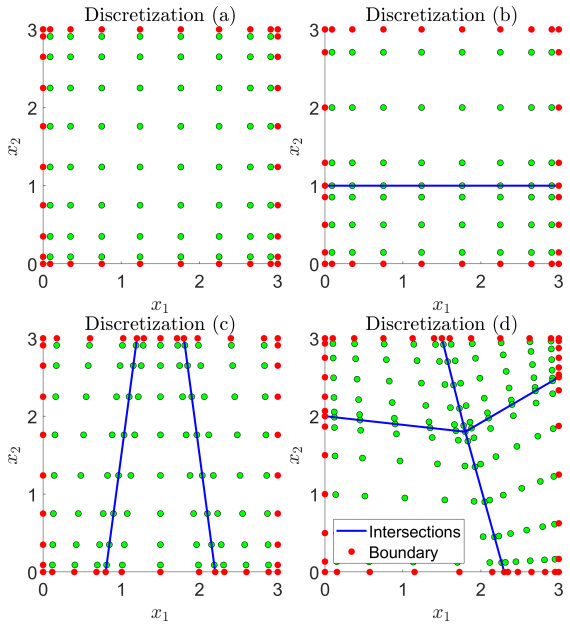

Figure 6.6: Different multishape discretizations of a box (a–d).

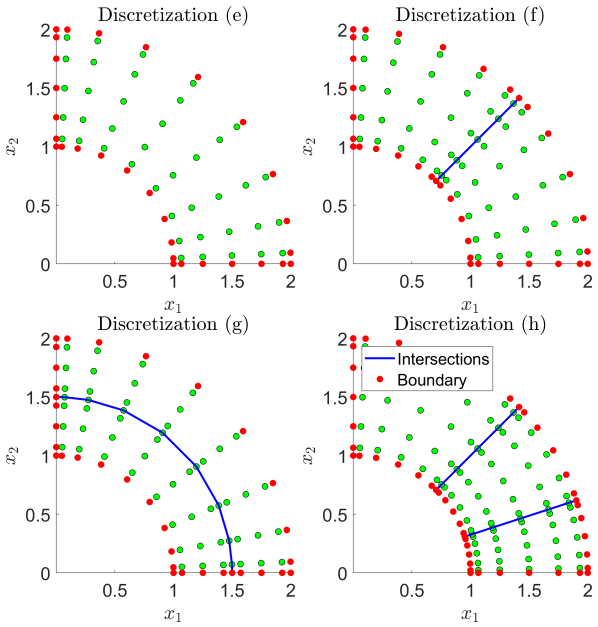

Figure 6.7: Different multishape discretizations of a wedge (e–h).

To test the convolution operator we have two different test functions

$$\chi_C(\vec{x}) = \exp\left(\frac{x_1^2 - x_2^2}{10}\right), \qquad n_C(\vec{z}) = z_1^2 + z_1 z_2,$$
$$\chi_P(\vec{x}) = \exp(x_1 + x_2), \qquad n_P(\vec{z}) = \exp\left(-z_1^2 - z_2^2 + z_1 + z_2\right),$$

where χ_C, n_C are considered for Cartesian discretizations and χ_P, n_P are the test functions on polar multishapes. This is to facilitate the analytical computation of the convolutions.
The convergence resulting from computations with the Laplacian, divergence, gradient, and interpolation matrices on different discretizations can be seen in Figure 6.8. As expected, convergence to the exact solution is observed as the number of discretization points is increased. For each discretization, the total number of points in each of the x_1 and x_2 directions (over the whole multishape), denoted by N_Σ, remains the same. Since the overall number of points for each multishape is the same, the computational cost for each discretization is fixed and so the performance of each discretization choice can be compared. For each multishape, the N_Σ points are equally distributed among the different elements. However, as the following comparison demonstrates, a minimum number of points has to be allocated in each the x_1 and x_2 directions and on each element of the multishape, in order to achieve optimal convergence. Multishape (c) in Figure 6.6 is discretized into three elements in such a way that the number of points in the x_1 coordinate is split three ways, resulting in each element only having $N_\Sigma/3$ points in the x_1 coordinate, and subsequently a poorer overall convergence than, for example, in multishape (d). While multishape (d) is discretized into four shapes, the x_1 and x_2 coordinate are only divided once, so that each element has $N_\Sigma/2$ points in each direction, improving convergence considerably.

6.4.2 The Poisson Equation

Having validated the convergence of different operators on different multishapes, for the next test we solve a Poisson equation. This demonstrates that applying the intersection matching conditions between the elements of a multishape works as expected, without adding further complexities, such as PDE solvers or integral terms. We solve the Poisson equation

$$\begin{aligned} \nabla^2 \rho(\vec{x}) &= f(\vec{x}) \qquad \text{in } \Omega, \\ \rho(\vec{x}) &= u(\vec{x}) \qquad \text{on } \partial\Omega. \end{aligned} \tag{6.6}$$

The right-hand side of the PDE,

$$f(\vec{x}) = \left(\left(x_1^2 - x_1 - 0.75\right) + \left(x_2^2 - x_2 - 0.75\right)\right) \exp\left(-0.5(x_1 - 0.5)^2 - 0.5(x_2 - 0.5)^2\right),$$

is chosen such that the Poisson equation on an infinite domain

$$\begin{aligned} \nabla^2 u(\vec{x}) &= f(\vec{x}) \qquad \text{in } \mathbb{R}^2, \\ u(\vec{x}) &\to 0 \qquad \text{as } |\vec{x}| \to \infty. \end{aligned}$$

has an exact solution, $u(\vec{x}) = \exp(-0.5(x_1 - 0.5)^2 - 0.5(x_2 - 0.5)^2)$. This can then be used to test the accuracy for the numerical solution to the Poisson problem (6.6), by enforcing the non-constant Dirichlet boundary conditions $\rho(\vec{x}) = u(\vec{x})$ on the multishape boundary. As in Section 6.4.1, we test the convergence on the different discretizations of the box (Figure 6.6) and wedge (Figure 6.7), for varying number of discretization points, as displayed in Figure 6.9. Similarly, the solution to the Poisson equation on a multishape, which combines a quadrilateral and a wedge element, as in Figure 6.4, is computed. Intersection matching conditions are applied between the two elements as illustrated in Section 6.3.1. Note that the two elements are not defined in the same coordinate system, which MultiShape automatically takes into account when applying the intersection condition. The three test cases considered involve taking equal numbers of points in both coordinate directions, half the number of points in

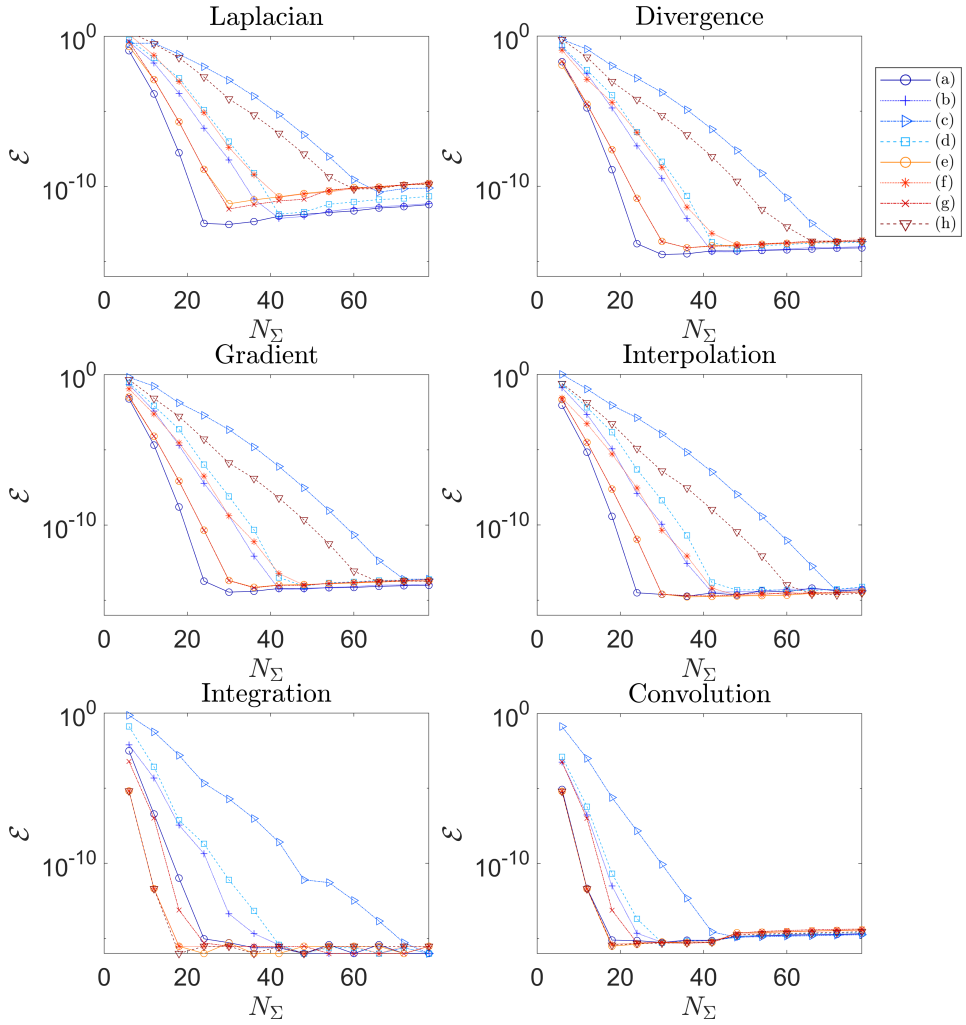

Figure 6.8: The error, as defined by (6.5), in the numerical differentiation, integration, interpolation, and convolution operators. Errors are computed for varying numbers of discretization points N_Σ and measured against an exact solution.

the first coordinate direction (x_1 and r, respectively), and half the number of points in the second coordinate direction (x_2 and θ, respectively). We compare the error in the numerical solution to the Poisson problem (6.6) with the resulting computational timings. Figure 6.10 showcases the convergence of the numerical result to the exact solution with increasing time, which correlates with increasing total numbers of discretization points. The results in Figure 6.10 demonstrate that the distribution of points among the coordinate directions does not result in a large difference in computation time taken for comparable accuracy in the resulting solution. However, the most efficient discretization for the particular choice of multishape and Poisson problem appears to be distributing double the points in the x_2 and angular directions than in the x_1 and radial directions.

6.5 Numerical Examples

In this section, we provide numerical examples computed using the MultiShape package. This includes solutions to integro-PDE models, as well as to optimal control problems. All model problems are computed with an ODE tolerance of 10^{-8}, while all optimal control problems are computed with a tolerance of 10^{-16}, 10 Newton iterations, and 200 GMRES iterations.

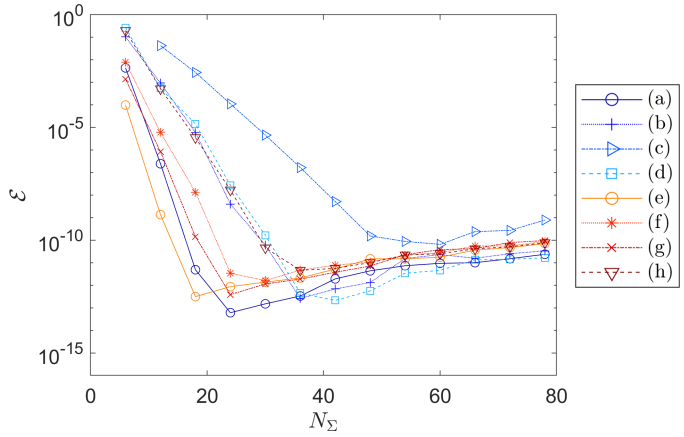

Figure 6.9: The error, as defined by (6.5), in computing the solution to the Poisson equation on different discretizations of a box and a wedge. Errors are computed for varying numbers of discretization points N_Σ and measured against an exact solution. [Note that for (c), with $N_\Sigma = 6$ no interior points in x_1 are present, so this data point is omitted.]

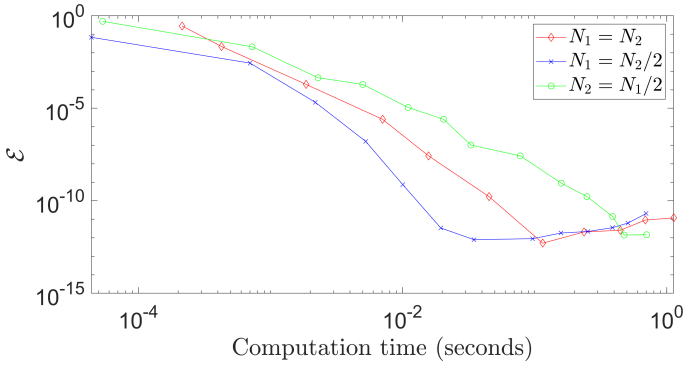

Figure 6.10: The error, defined by (6.5), in solving the Poisson equation on a multishape, against computation time, for varying numbers of discretization points in the two coordinate directions, N_1 and N_2, for each element.

6.5.1 Solving DDFT Models

Comparison of Model Solutions Computed on Domains Defined Using External Potentials and MultiShape

It is common practice in DDFT computations to model complicated domains by steep external potentials that prevent particles from accessing certain parts of the domain and therefore effectively modelling walls. One such example is given in [218]. The authors model flow of a colloidal fluid through a channel that presents a constriction in the middle. The boundary of this channel is given by the external potential

$$V_1 = V_0 \left[1 - 0.5\mathrm{erf}((y + g(x_1))/\sqrt{2}w) + 0.5\mathrm{erf}((y - g(x_1))/\sqrt{2}w)\right], \tag{6.7}$$

where

$$g(x_1) = \begin{cases} L_{x_2}/2 - \alpha[1 + \cos(2\pi(x_1 - x_{1,0})/L_c)] & \text{for} \quad |x_1 - x_{1,0}| < L_c/2, \\ L_{x_2}/2 & \text{otherwise,} \end{cases}$$
$$\alpha = (L_{x_2}/4)(1 - b).$$

Here we choose $V_0 = 10$, $w = 0.25$, $b = 0.5$, $L_{x_2} = 6\sqrt{\frac{\sqrt{3}}{2}}$, $L_c = 2.686$ and $x_{1,0} = 0$, in view of the parameters in [218]. While we do not aim to replicate the results in [218], we would like

to investigate the difference in modelling flow through a constriction using the domain shaped by V_1 and a similar domain created by the MultiShape package. In the absence of particle interactions and background flow, the equilibrium of ρ is given by

$$\rho(0, \vec{x}) = Z^{-1} e^{-V_1},$$

with normalization constant Z. This is chosen as initial condition for the example in which V_1 imposes the domain boundary. For the MultiShape example, we choose a uniform initial condition for ρ over the multishape. Both initial conditions are normalized by Z, so that the average number of particles, i.e., the mass, inside the channel is 10 and an advective field $\vec{w} = [0.5, 0]^\top$ is imposed. The time horizon for the simulation is $(0, 1)$. We model a problem of the form (3.59) with no-flux boundary conditions (3.60).

For the MultiShape implementation the domain is constructed using four quadrilateral shapes and $N_1 = N_2 = 20$ points are chosen on each element, resulting in a total of $M = 1600$ spatial discretization points. For the implementation using V_1, we use a single square domain, but with $N_1 = N_2 = 80$ points, i.e., $M = 6400$. The reason for the increased number of discretization points is that the external potential modelling the domain boundaries causes steep particle gradients, which are difficult to resolve numerically. This in turn requires more discretization points. In order to make this more feasible, one would have to utilize adaptive meshing strategies, to resolve steep gradients more accurately. Combining such an approach with a pseudospectral discretization is highly non-trivial and problem-specific, see, e.g., [93, Chapter 16], [223], [224]. One could instead apply an adaptive-grid finite element method [103, Chapter 9]. This, however, would mean forfeiting the superior convergence properties of spectral methods for the problems of interest in this thesis. We therefore conclude that, in this case, the MultiShape approach is computationally advantageous.

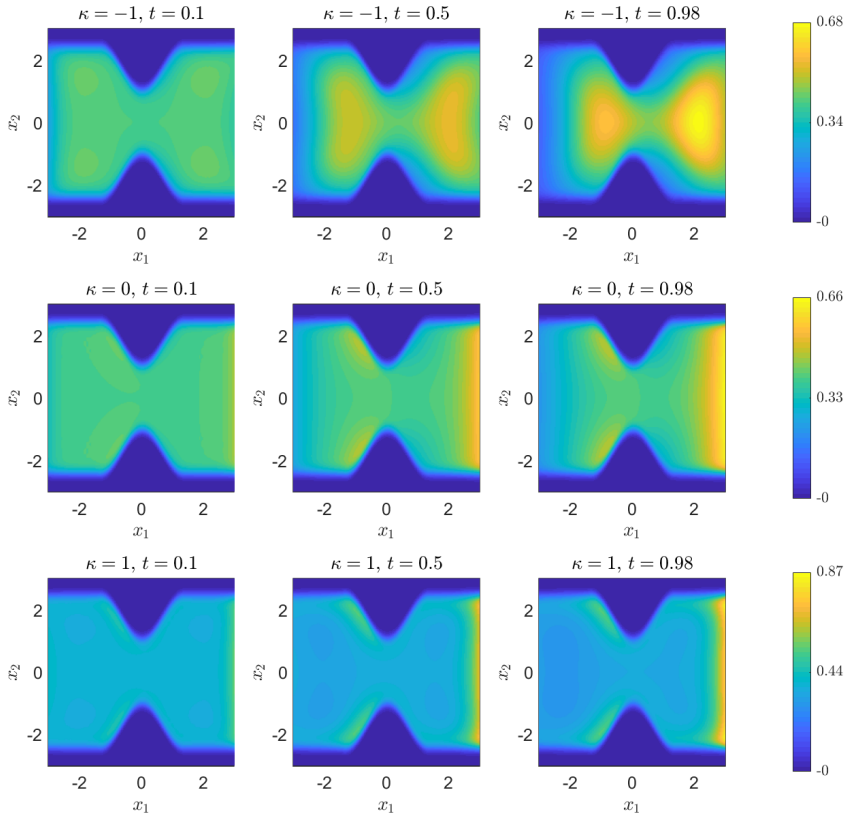

Figure 6.11: Evolution of ρ for flow through a constriction at different interaction strengths, $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom). Here, the domain is modelled via the external potential V_1 given by (6.7).

The results for the domain modelled by V_1 are presented in Figure 6.11, while the MultiShape domain results are displayed in Figure 6.12. We can observe that the particle dynamics are comparable for the two approaches, for each choice of interaction strength. In all examples the particle mass is carried to the right of the domain by the advective field. This causes particles to accumulate in front of the constriction. Attractive particles accumulate in the middle, so that a 'sand-clock' dynamic can be observed. Repulsive particles accumulate on both sides of the constriction, so that the particles build a layer on the domain boundary on the left of the constriction, as well as on the narrowest point of the channel.

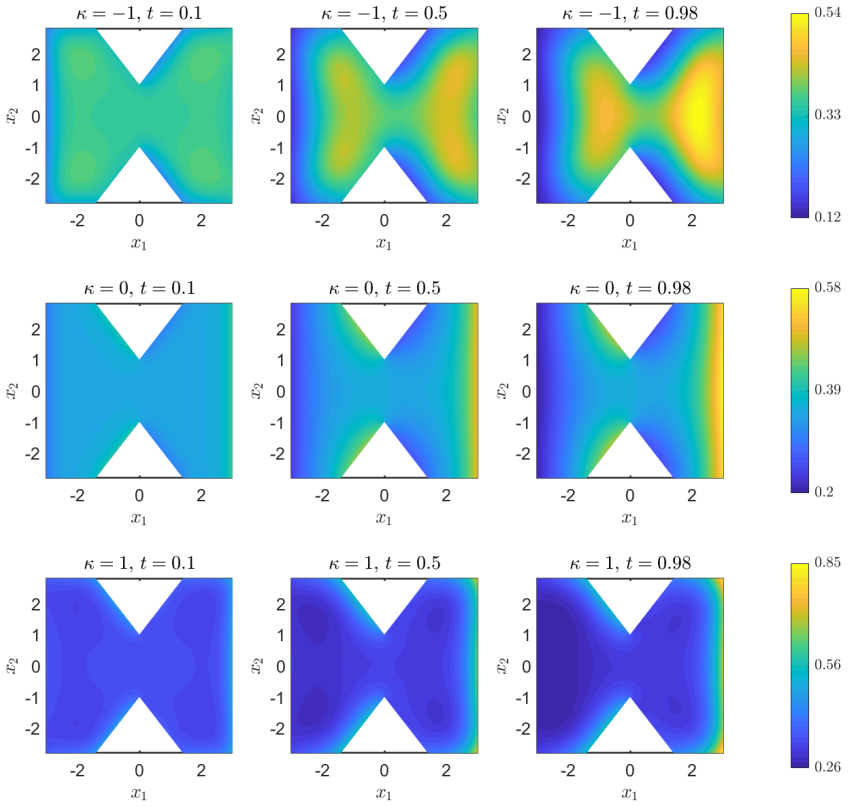

Figure 6.12: Evolution of ρ for flow through a constriction at different interaction strengths, $\kappa = -1$, $\kappa = 0$, and $\kappa = 1$ (top to bottom). Here, the domain is constructed using the MultiShape package.

While both modelling approaches show similar dynamics, there are a few shortcomings of the approach modelling boundaries using V_1, as displayed in Figure 6.11. One issue is that the potential essentially models a repulsive wall. Due to this, one can see that a layer of particles forms at the walls created by V_1. Moreover, despite the higher computational effort of this method, the potential is not steep enough to model rigid walls, so that the domain boundary appears blurred and mass is presumably not conserved on the inside of the channel. In contrast, when using the MultiShape domain composition, as done in Figure 6.12, one can see more details of the particle dynamics for each of the interaction strengths, the domain boundaries are clearly defined and no 'repulsive' wall is modelled. Moreover, in this setup, mass is conserved. However, in the MultiShape setup, one has to be aware of corner singularities, which may affect the accuracy of the numerical solution. Results on the regularity of PDE solutions were discussed in Chapter 3, and in particular Theorem 3.1.9 for elliptic PDEs shows that there is a clear distinction between interior regularity and regularity on the domain boundary. While one may not have the same level of regularity on the boundary of the domain as in the interior, globally the inaccuracies of corner singularities do not significantly impact the numerical accuracy, see [93, Chapter 2]. This can be explained by rounding the sharp corners slightly, resulting in the

boundary being representable by a smooth curve [93, Chapter 2].

Models of Pipe Flow

In this section, again problems of the form (3.59) with no-flux boundary conditions (3.60) are solved. To illustrate the versatility of the MultiShape method, we introduce two examples which model particles flowing through pipes. The first example is constructed using a combination of four wedge and quadrilateral shapes with each containing $N_1 = N_2 = 20$ spatial discretization points. The initial condition for this model problem is $\rho(0, \vec{x}) = Z^{-1} \exp(-0.1(x_1 + 5.5)^2 - 0.1(x_2 + 1.1)^2)$, where Z normalizes $\rho(0, \vec{x})$ to mass 3, and the time horizon for the simulation is $(0, 6)$. The result of this simulation is presented in Figure 6.13 for attractive and repulsive particles. One can see a qualitative difference between the two cases, with the attractive particles forming a cluster that moves slightly slower through the pipe than the repulsive particles that spread out along the sides of the channel.

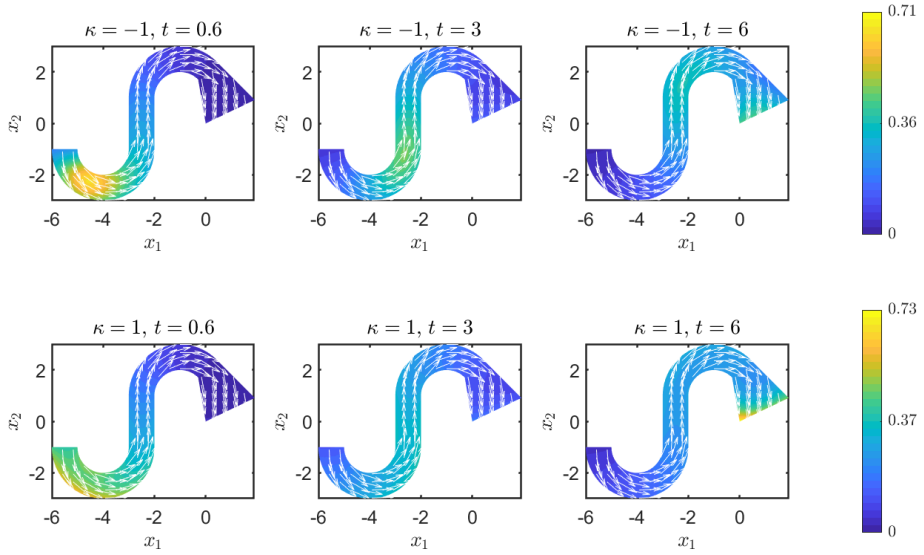

Figure 6.13: Evolution of ρ for flow through a pipe with superimposed advective field. Results are displayed for $\kappa = -1$ (top) and $\kappa = 1$ (bottom).

A second pipe flow example is constructed, using a combination of three quadrilateral and wedge elements, with $N_1 = N_2 = 20$ points for each element. The chosen time horizon for this problem is $(0, 2)$. The initial condition is $\rho(0, \vec{x}) = Z^{-1} \exp(-0.5((x_1 - 0.75)^2) - 0.5((x_2 - 3.3)^2))$, where Z^{-1} normalizes the mass of ρ to one. Here, the dynamics is driven by an external potential modelling gravity: $V_1 = 0.5x_2$. The resulting dynamics are displayed in Figure 6.14 for attractive and repulsive particles. It is evident that the repulsive particles sediment to the bottom of the pipe quicker than the attractive particles. The attractive forces cause particles to build clumps in the middle of the pipe, which counteracts the effect of the gravitational forces.

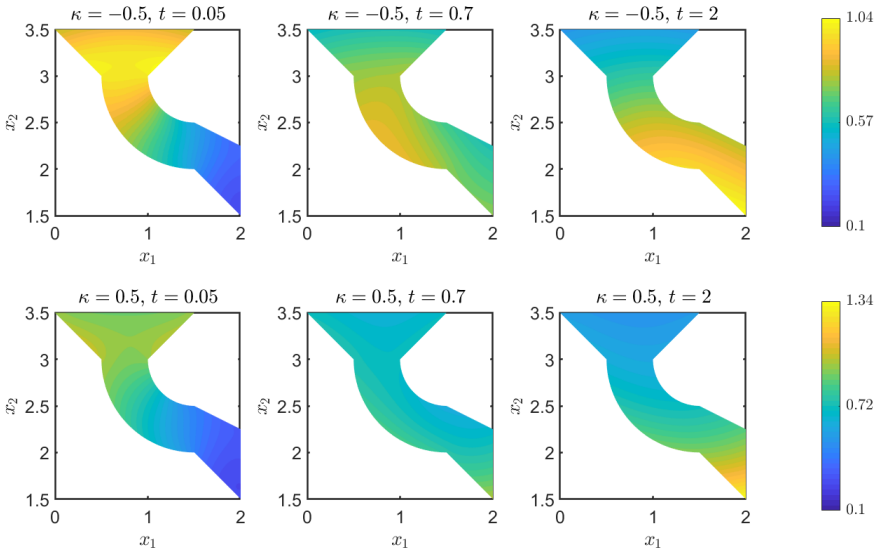

Figure 6.14: Evolution of ρ for flow through a pipe under gravity. Results are displayed for $\kappa = -0.5$ (top) and $\kappa = 0.5$ (bottom).

6.5.2 Solving an Optimal Control Problem

In this section, we solve a flow control problem (5.8) with no-flux boundary conditions (5.10). Inspired by the DDFT example in Figure 6.14, we pose the following optimal control problem: We choose the same domain as in the forward example, with $N_1 = N_2 = 20$ and $n = 11$, the same initial condition $\rho(0, \vec{x}) = Z^{-1} \exp(-0.5((x_1 - 0.75)^2) - 0.5((x_2 - 3.3)^2))$, and the same external potential modelling gravity: $V_1 = 0.5x_2$. The time horizon for this model problem is $(0, 2)$. The desired state is given by the forward problem in Figure 6.14 with $\kappa = 0.5$, while the optimal control model contains attractive particles ($\kappa = -0.5$). The initial guess for the Newton–Krylov solver is $\rho(t, \vec{x}) = \rho(0, \vec{x})$ and $q(t, \vec{x}) = 0$, for all t.

Without imposing any control, the optimal solution would therefore reduce to the forward problem with $\kappa = -0.5$, while imposing control will push the optimal solution closer to the desired state given by the forward problem with $\kappa = 0.5$. We impose flow control through an advective field. The resulting optimal control problem for $\beta = 10^{-1}$ can be seen in Figure 6.15. The β value in this figure is chosen to illustrate that the resulting particle distribution lies between the attractive and repulsive forward models in Figure 6.14, as expected. At earlier times, the control acts to direct the particles towards the bottom of the domain, since more repulsive particles accumulate there than attractive particles do. At later times, the control is concerned with the spread of particles to counteract the attractive forces between them.

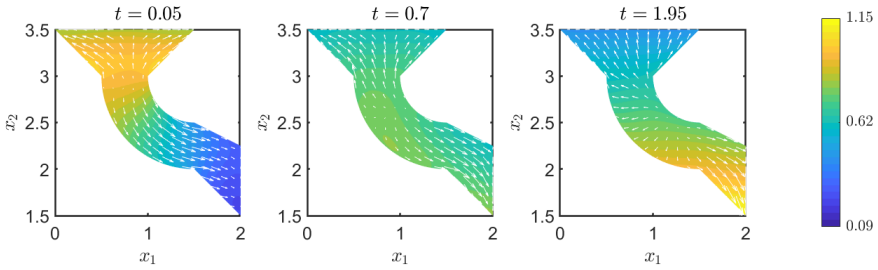

Figure 6.15: Evolution of the optimal ρ under gravity for $\beta = 10^{-1}$. The advective control field $\vec{w}$ is superimposed.

	$\beta = 10^{-4}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{0}$
$\mathcal{J}_{uc}$	5.90×10^{-3}	5.90×10^{-3}	5.90×10^{-3}	5.90×10^{-3}
$\mathcal{J}_c$	2.91×10^{-5}	1.96×10^{-4}	3.60×10^{-3}	5.56×10^{-3}

Table 6.1: MultiShape flow control problem with no-flux boundary conditions: Cost $\mathcal{J}_{uc}$ of applying no control (i.e., $\vec{w} = \vec{0}$) and optimal control cost $\mathcal{J}_c$ for a range of values of the regularization parameter β.

In Table 6.1 the values of the cost functionals for $\vec{w} = \vec{0}$, the uncontrolled system, are compared with the values of the cost functional for the optimal solutions at different choices of β. As in previous optimal control examples, $\mathcal{J}_c$ decreases with decreasing β and approaches $\mathcal{J}_{uc}$ with increasing β.

In Chapter 3, a numerical method was introduced for solving PDEs on simple domains. In Chapter 5, a method for the numerical solution of PDE-constrained optimization problems was discussed. These methods were successfully extended in the current chapter to the computation of (integro-)PDE models and PDE-constrained optimization problems on complicated domains, called multishapes. This enables the solution of problems in which the domain is an important part of the process of interest, as is the case in many real-world applications. Examples of such domains are pipes, vessels, funnels and channels. In the next two chapters this numerical framework is applied to problems involving extended DDFT models which capture different physical effects.

Chapter 7

Multiple Species of Particles

Now that we have established a framework for the optimal control of particle dynamics and extended the numerical method to be able to compute problems on complicated domains, the methodology can be used for other DDFT models and optimal control problems involving such model extensions. In this section, we will apply the framework to mixtures of different kinds of particles, i.e., multiple particle species, and in the following chapters we will introduce further model extensions from DDFT that allow us to capture different physical effects. The work in this chapter is described in the preprint [2].

The idea of DDFT for particle mixtures was already introduced in [225]. Derivations of this DDFT, including hydrodynamic interactions, were covered in [226, 227]. This model has been applied in many settings, such as for modelling tumour and cell growth [228, 229], chemical reactions [230, 231, 232], phase separation [46, 55, 233], and one-component systems in two states [234, 235]. Other applications were illustrated in the review paper [5]. In the following, we will introduce the model and the associated optimal control problem, and apply the numerical framework developed in previous sections to solve both on complicated domains.

7.1 The DDFT Model

So far, we have been interested in solving models of the form

$$\frac{\partial \rho}{\partial t} = \nabla \cdot \left(\rho \nabla \frac{\delta \mathcal{F}[\rho]}{\delta \rho} \right),$$

with suitable initial and boundary conditions. This model describes how a particle density ρ evolves under the effects of the free energy functional $\mathcal{F}$. In this thesis this has included diffusion, external forces such as gravity, and particle–particle interactions. This is now extended to multiple species of particles with densities $\rho_1, \rho_2, ..., \rho_{n_s}$, where $\{\rho_a\}$ denotes the set of all species $a = 1, 2, ..., n_s$ and n_s is the total number of species. The key idea is that the given free energy of the system depends on all species of particles in the system, which is denoted by $\mathcal{F}[\{\rho_b\}]$, $b = 1, 2, ..., n_s$. The resulting DDFT model is a system of n_s equations of the form

$$\frac{\partial \rho_a}{\partial t} = \nabla \cdot \left(\rho_a \nabla \frac{\delta \mathcal{F}[\{\rho_b\}]}{\delta \rho_a} \right). \tag{7.1}$$

The mean-field free energy functional (1.2) that has been used throughout this thesis can be generalized to

$$\mathcal{F}[\{\rho_b\}] = \sum_{a=1}^{n_s} \left[\int_\Omega \rho_a (\ln \rho_a - 1) d\vec{x} + \int_\Omega \rho_a V_1 d\vec{x} + \frac{1}{2} \sum_{b=1}^{n_s} \int_\Omega \int_\Omega \rho_a(\vec{x}) \rho_b(\vec{x}') V_{a,b}(|\vec{x} - \vec{x}'|) d\vec{x}' d\vec{x} \right].$$

Here, we change notation from the interaction potential being V_2 to $V_{a,b}$ to denote the interactions between species a and b. In the following, this interaction potential is given by

$$V_{a,b}(x) = \kappa_{a,b} \exp\left(-(x/\sigma_{a,b})^2\right), \tag{7.2}$$

where $\sigma_{a,b}$ represents half of the distance between the centres of particle a and particle b. In the case of $\sigma_{a,a}$ this can be thought of as the particle radius. Other choices of $V_{a,b}$ can be made and implemented in the numerical method without significant effort.

After inserting the free energy into (7.1) a system of PDEs of the form

$$\frac{\partial \rho_a}{\partial t} = \nabla^2 \rho_a + \nabla \cdot (\rho_a \nabla V_1) + \sum_{b=1}^{n_s} \nabla_{\vec{x}} \cdot \int_\Omega \rho_a(\vec{x}) \rho_b(\vec{x}') \nabla_{\vec{x}'} V_{a,b}(|\vec{x} - \vec{x}'|) d\vec{x}' \tag{7.3}$$

is obtained. This can be solved either with Dirichlet boundary conditions, or with no-flux conditions, which, generalized to more than one species, read

$$\rho_a \nabla \frac{\delta \mathcal{F}}{\delta \rho_a} \cdot \vec{n} = \frac{\partial \rho_a}{\partial n} + \rho_a \frac{\partial V_1}{\partial n} + \sum_{b=1}^{n_s} \int_\Omega \rho_a(\vec{x}) \rho_b(\vec{x}') \frac{\partial V_{a,b}}{\partial n}(|\vec{x} - \vec{x}'|) d\vec{x}' = 0.$$

This system can be solved using MATLAB's differential–algebraic equation solver `ode15s` [120, 121] as done for the one species model in the previous chapters.

7.1.1 Equilibrium Solution of a DDFT Model

Often one is interested not (only) in the time-dependent solution to the DDFT equations, but (also) in the long-time, equilibrium solution. We would like to find such a solution on multishape domains. There are two equivalent approaches to finding these steady states; investigating the behaviour of (7.1) for large t, such that $\frac{\partial \rho_a}{\partial t} \approx 0$, or minimizing $\mathcal{F}$ with respect to ρ_a. Focusing on the latter approach, we solve the system of equations $\frac{\delta \mathcal{F}}{\delta \rho_a} = 0$, resulting in the nonlinear equation

$$\rho_a = Z^{-1} \exp\left(-V_1 - \sum_{b=1}^{n_s} \int_\Omega \rho_b(\vec{x}') V_{a,b}(|\vec{x} - \vec{x}'|) d\vec{x}'\right), \tag{7.4}$$

where Z is a normalization constant, enforcing that the mass of, and therefore the number of particles in, the system remains constant.

This system of equations is of the form $\rho_a = G_a(\rho_1, ..., \rho_{n_s})$ and can be solved self-consistently using, for example, iterative methods. In this work, we are considering a system of two species, which demonstrates all relevant features of the numerical solution, but this can easily be extended to systems with more species. In order to find the solution to the self-consistent system of equations (7.4), we use an iterative scheme, known as Picard iteration in the DDFT community [20]. Algorithm 4 illustrates this scheme, given initial guesses for the particle distributions $\rho_{1,\text{ig}}$, $\rho_{2,\text{ig}}$, an error tolerance to terminate the algorithm, and a mixing parameter λ for the Picard step. The Picard step takes a linear combination of the previous and new guesses for ρ_1, ρ_2, which is designed to stabilize the algorithm. The error measure in this algorithm is defined as the maximum of the error across all species, individually computed using (6.5). A numerical example of this formulation is presented in Section 7.4.1. One could, in principle, implement more sophisticated numerical methods for solving (7.4). However, this method is efficient and robust for a variety of mixing parameters and problems. In future work, it could be interesting to apply higher order methods, such as those discussed in Chapter 4, to more complicated problems, to ensure convergence and accuracy in such cases as well.

Note that problems in equilibrium Density Functional Theory (DFT) also require finding a minimum of a free energy $\mathcal{F}$, and can therefore be tackled by such variational approaches. An

implementation is discussed in [20], while overviews of DFT are given in, e.g., [14, 16, 236, 237].

Algorithm 4: Picard Equilibrium Solver

Set $\rho_{1,\text{ig}}$, $\rho_{2,\text{ig}}$, Tol, and λ.
Set $\rho_{1,\text{old}} = \rho_{1,\text{ig}}$, $\rho_{2,\text{old}} = \rho_{2,\text{ig}}$.
while $\mathcal{E} <$ Tol **do**
  Compute $\rho_{1,\text{new}} = G_1(\rho_{1,\text{old}}, \rho_{2,\text{old}})$, $\rho_{2,\text{new}} = G_2(\rho_{1,\text{old}}, \rho_{2,\text{old}})$.
  Compute error $\mathcal{E}$, defined in (6.5).
  if $\mathcal{E} \geq$ Tol **then**
    Set $\rho_{1,\text{update}} = (1-\lambda)\rho_{1,\text{old}} + \lambda\rho_{1,\text{new}}$, $\rho_{2,\text{update}} = (1-\lambda)\rho_{2,\text{old}} + \lambda\rho_{2,\text{new}}$.
    Update $\rho_{1,\text{old}} = \rho_{1,\text{update}}$, $\rho_{2,\text{old}} = \rho_{2,\text{update}}$.
  end if
end while

7.2 The Optimal Control Problem

We can formulate an optimal control problem involving a system of n_s interacting species of particles, such as in (7.3). One such optimal control problem reads

$$\min_{\{\rho_a\},\vec{w}} \mathcal{J}(\{\rho_a\},\vec{w}) := \frac{1}{2}\sum_{a=1}^{n_s} \|\rho_a - \widehat{\rho}_a\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2}\|\vec{w}\|^2_{L^2((0,T)\times\Omega)}, \tag{7.5}$$

subject to the following system of n_s PDE constraints, for $a = 1, ..., n_s$:

$$\begin{aligned}
\frac{\partial \rho_a}{\partial t} &= \nabla^2\rho_a - \nabla\cdot(\rho_a\vec{w}) + \nabla\cdot(\rho_a\nabla V_1) \\
&\quad + \sum_{b=1}^{n_s} \nabla_{\vec{x}}\cdot\int_\Omega \rho_a(\vec{x})\rho_b(\vec{x}')\nabla_{\vec{x}'}V_{a,b}(|\vec{x}-\vec{x}'|)d\vec{x}' \quad \text{in } (0,T)\times\Omega, \\
\frac{\partial \rho_a}{\partial n} - \rho_a\vec{w}\cdot\vec{n} + \rho_a\frac{\partial V_1}{\partial n} &+ \sum_{b=1}^{n_s}\int_\Omega \rho_a(\vec{x})\rho_b(\vec{x}')\frac{\partial V_{a,b}}{\partial n}(|\vec{x}-\vec{x}'|)d\vec{x}' = 0 \quad \text{on } (0,T)\times\partial\Omega, \\
\rho_a(0,\vec{x}) &= \rho_{a,0}(\vec{x}) \quad \text{at } \{t=0\}\times\Omega.
\end{aligned}$$

Here $\{\rho_a\}$ denotes the set of all particle densities for species $a = 1, ..., n_s$. Note that the difference between the system of PDE constraints above and the system (7.3) is the additional term involving $\vec{w}$. This term models how a particle distribution ρ_a may be influenced by a background flow $\vec{w}$. In this optimal control problem, the aim is to drive each of the particle distributions ρ_a closer to the given desired states $\widehat{\rho}_a$, which can be achieved by modifying $\vec{w}$, the control variable. Note that all species of particles are influenced by the same control field. In view of (5.8) for single species, this problem can be cast in the notation of the previous chapters:

$$\min_{\{\rho_a\},\vec{w}} \mathcal{J}(\{\rho_a\},\vec{w}) := \frac{1}{2}\sum_{a=1}^{n_s} \|\rho_a - \widehat{\rho}_a\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2}\|\vec{w}\|^2_{L^2((0,T)\times\Omega)}$$

subject to

$$\begin{aligned}
\frac{\partial \rho_a}{\partial t} &= -\nabla\cdot\big(\mathcal{D}(\rho_a) + \mathcal{I}(\{\rho_b\}) + \mathcal{C}(\rho_a,\vec{w})\big) + f_a \qquad \text{in } (0,T)\times\Omega, \\
0 &= \big(\mathcal{D}(\rho_a) + \mathcal{I}(\{\rho_b\}) + \mathcal{C}(\rho_a,\vec{w})\big)\cdot\vec{n} \qquad \text{on } (0,T)\times\partial\Omega, \\
\rho_a(0,\vec{x}) &= \rho_{a,0}(\vec{x}) \qquad \text{at } \{t=0\}\times\Omega,
\end{aligned}$$

where $\{\rho_b\}$ denotes the set of all particle densities for species $b = 1, ..., n_s$, and

$$\mathcal{D}(\rho_a) = -\nabla\rho_a - \rho_a\nabla V_1 + \rho_a\vec{f}, \quad C(\rho_a, \vec{w}) = \rho_a\vec{w},$$
$$\mathcal{I}(\{\rho_b\}) = -\sum_{b=1}^{n_s} \nabla_{\vec{x}} \cdot \int_\Omega \rho_a(\vec{x})\rho_b(\vec{x}')\nabla_{\vec{x}'} V_{a,b}(|\vec{x} - \vec{x}'|)d\vec{x}',$$

in line with the definitions in Section 5.1. It is evident how other boundary conditions or controls can be applied for this problem by replacing the relevant terms, as done in Section 5.1.

7.2.1 Derivation of First-Order Optimality System

Given the optimal control formulation, the corresponding first-order optimality system can be derived, as done in Section 5.2. The full Lagrangian for this problem is

$$\begin{aligned}\mathcal{L}(\{\rho_a\}, \vec{w}, \{q_a\}, \{q_{a,\partial\Omega}\}) = \mathcal{J}(\{\rho_a\}, w) \\ - \sum_{a=1}^{n_s} \int_\Omega \int_0^T \left(\frac{\partial \rho_a}{\partial t} + \nabla \cdot \mathcal{D}(\rho_a) + \nabla \cdot \mathcal{I}(\{\rho_b\}) + \nabla \cdot C(\rho_a, \vec{w}) - f \right) q_a d\vec{x}dt \\ - \sum_{a=1}^{n_s} \int_0^T \int_{\partial\Omega} \left(\mathcal{D}(\rho_a) + \mathcal{I}(\{\rho_b\}) + \mathcal{C}(\rho_a, \vec{w})\right) \cdot \vec{n} q_{a,\partial\Omega} d\vec{x}dt.\end{aligned} \tag{7.6}$$

Deriving the adjoint equations is done by computing the equations $\mathcal{L}_{\rho_a}(\{\bar{\rho}_a\}, \bar{w}, \{q_a\}, \{q_{a,\partial\Omega}\})h = 0$, compare with Section 5.2. We note that the terms $\mathcal{D}(\rho_a)$ and $C(\rho_a, \vec{w})$ involve only one particle species. Therefore, when deriving the adjoint equation, we can take the definitions for $\mathcal{D}^*(q_a)$ and $\mathcal{C}^*(q_a, \vec{w})$ from (5.13) and (5.18) directly.

The term that requires separate treatment is $\nabla \cdot \mathcal{I}(\{\rho_b\})$, since it depends on more than one particle species. Since the terms in the Lagrangian are additive, it can be treated separately and added to the terms depending only on ρ_a, as has been done for the integral term in Section 5.2. Hence, we consider

$$\begin{aligned}T(\{\rho_b\}, \{q_b\}) :&= -\sum_{a=1}^{n_s} \int_0^T \int_\Omega \nabla \cdot \mathcal{I}(\{\rho_b\}) q_a d\vec{x}dt \\ &= \sum_{a=1}^{n_s} \left[\int_0^T \int_\Omega q_a(t, \vec{x}) \nabla \cdot \left(\rho_a(t, \vec{x}) \sum_{b=1}^{n_s} \int_\Omega \rho_b(t, \vec{x}') \nabla_{\vec{x}} V_{a,b}(|\vec{x} - \vec{x}'|) d\vec{x}' \right) d\vec{x}dt \right].\end{aligned}$$

We can take the derivative of this term with respect to each ρ_a. Since $T(\{\rho_b\}, \{q_b\})$ involves a double sum over all particle species, only some terms will enter the derivative with respect to ρ_a in direction h: those where one or both indices are equal to a. The terms that need to be considered when differentiating are:

$$\begin{aligned}\int_0^T \int_\Omega \Bigg[& q_a(t, \vec{x}) \nabla \cdot \left(\rho_a(t, \vec{x}) \int_\Omega \rho_a(t, \vec{x}') \nabla_{\vec{x}} V_{a,a}(|\vec{x} - \vec{x}'|) d\vec{x}' \right) \\ & + \sum_{\substack{b=1 \\ b \neq a}}^{n_s} q_a(t, \vec{x}) \nabla \cdot \left(\rho_a(t, \vec{x}) \int_\Omega \rho_b(t, \vec{x}') \nabla_{\vec{x}} V_{a,b}(|\vec{x} - \vec{x}'|) d\vec{x}' \right) \\ & + \sum_{\substack{b=1 \\ b \neq a}}^{n_s} q_b(t, \vec{x}) \nabla \cdot \left(\rho_b(t, \vec{x}) \int_\Omega \rho_a(t, \vec{x}') \nabla_{\vec{x}} V_{b,a}(|\vec{x} - \vec{x}'|) d\vec{x}' \right) \Bigg] d\vec{x}dt.\end{aligned}$$

Then, using the product rule on the first term, we obtain

$$\begin{aligned}
T_{\rho_a}(\{\rho_b\},\{q_b\})h = &\int_0^T \int_\Omega q_a(t,\vec{x})\nabla\cdot\left(h(t,\vec{x})\int_\Omega \rho_a(t,\vec{x}')\nabla_{\vec{x}}V_{a,a}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)d\vec{x}dt \\
&+\int_0^T \int_\Omega q_a(t,\vec{x})\nabla\cdot\left(\rho_a(t,\vec{x})\int_\Omega h(t,\vec{x}')\nabla_{\vec{x}}V_{a,a}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)d\vec{x}dt \\
&+\sum_{\substack{b=1\\b\neq a}}^{n_s}\int_0^T \int_\Omega q_a(t,\vec{x})\nabla\cdot\left(h(t,\vec{x})\int_\Omega \rho_b(t,\vec{x}')\nabla_{\vec{x}}V_{a,b}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)d\vec{x}dt \\
&+\sum_{\substack{b=1\\b\neq a}}^{n_s}\int_0^T \int_\Omega q_b(t,\vec{x})\nabla\cdot\left(\rho_b(t,\vec{x})\int_\Omega h(t,\vec{x}')\nabla_{\vec{x}}V_{b,a}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)d\vec{x}dt.
\end{aligned}$$

Each term is integrated by parts and regrouped to find

$$\begin{aligned}
T_{\rho_a}(\{\rho_b\},\{q_b\})h = &\int_0^T \int_{\partial\Omega} q_a(t,\vec{x})\left(\int_\Omega (h(t,\vec{x})\rho_a(t,\vec{x}')+\rho_a(t,\vec{x})h(t,\vec{x}'))\frac{\partial V_{a,a}(|\vec{x}-\vec{x}'|)}{\partial n}d\vec{x}'\right)d\vec{x}dt \\
&+\sum_{\substack{b=1\\b\neq a}}^{n_s}\int_0^T \int_{\partial\Omega}\left[q_a(t,\vec{x})\int_\Omega h(t,\vec{x})\rho_b(t,\vec{x}')\frac{\partial V_{a,b}(|\vec{x}-\vec{x}'|)}{\partial n}d\vec{x}'\right. \\
&\qquad\left.+q_b(t,\vec{x})\int_\Omega \rho_b(t,\vec{x})h(t,\vec{x}')\frac{\partial V_{b,a}(|\vec{x}-\vec{x}'|)}{\partial n}d\vec{x}'\right]d\vec{x}dt \\
&-\int_0^T \int_\Omega \nabla q_a(t,\vec{x})h(t,\vec{x})\cdot\left(\int_\Omega \rho_a(t,\vec{x}')\nabla_{\vec{x}}V_{a,a}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)d\vec{x}dt \\
&-\int_0^T \int_\Omega \nabla q_a(t,\vec{x})\rho_a(t,\vec{x})\cdot\left(\int_\Omega h(t,\vec{x}')\nabla_{\vec{x}}V_{a,a}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)d\vec{x}dt \\
&-\sum_{\substack{b=1\\b\neq a}}^{n_s}\int_0^T \int_\Omega \nabla q_a(t,\vec{x})h(t,\vec{x})\cdot\left(\int_\Omega \rho_b(t,\vec{x}')\nabla_{\vec{x}}V_{a,b}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)d\vec{x}dt \\
&-\sum_{\substack{b=1\\b\neq a}}^{n_s}\int_0^T \int_\Omega \nabla q_b(t,\vec{x})\rho_b(t,\vec{x})\cdot\left(\int_\Omega h(t,\vec{x}')\nabla_{\vec{x}}V_{b,a}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)d\vec{x}dt.
\end{aligned}$$

In each term in which h depends on $\vec{x}'$ the inner and outer integrals are swapped by relabelling $\vec{x}\to\vec{x}'$. This results in

$$\begin{aligned}
T_{\rho_a}(\{\rho_b\},\{q_b\})h = &\int_0^T \int_{\partial\Omega} q_a(t,\vec{x})\left(\int_\Omega (h(t,\vec{x})\rho_a(t,\vec{x}')+\rho_a(t,\vec{x})h(t,\vec{x}'))\frac{\partial V_{a,a}(|\vec{x}-\vec{x}'|)}{\partial n}d\vec{x}'\right)d\vec{x}dt \\
&+\sum_{\substack{b=1\\b\neq a}}^{n_s}\int_0^T \int_{\partial\Omega}\left[q_a(t,\vec{x})\int_\Omega h(t,\vec{x})\rho_b(t,\vec{x}')\frac{\partial V_{a,b}(|\vec{x}-\vec{x}'|)}{\partial n}d\vec{x}'\right. \\
&\qquad\left.+q_b(t,\vec{x})\int_\Omega \rho_b(t,\vec{x})h(t,\vec{x}')\frac{\partial V_{b,a}(|\vec{x}-\vec{x}'|)}{\partial n}d\vec{x}'\right]d\vec{x}dt \\
&-\int_0^T \int_\Omega \nabla q_a(t,\vec{x})h(t,\vec{x})\cdot\left(\int_\Omega \rho_a(t,\vec{x}')\nabla_{\vec{x}}V_{a,a}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)d\vec{x}dt \\
&-\int_0^T \int_\Omega h(t,\vec{x})\left(\int_\Omega \nabla_{\vec{x}'}q_a(t,\vec{x}')\rho_a(t,\vec{x}')\cdot\nabla_{\vec{x}}V_{a,a}(|\vec{x}'-\vec{x}|)d\vec{x}'\right)d\vec{x}dt \\
&-\sum_{\substack{b=1\\b\neq a}}^{n_s}\int_0^T \int_\Omega \nabla q_a(t,\vec{x})h(t,\vec{x})\cdot\left(\int_\Omega \rho_b(t,\vec{x}')\nabla_{\vec{x}}V_{a,b}(|\vec{x}-\vec{x}'|)d\vec{x}'\right)d\vec{x}dt
\end{aligned}$$

$$- \sum_{\substack{b=1 \\ b \neq a}}^{n_s} \int_0^T \int_\Omega h(t,\vec{x}) \left(\int_\Omega \nabla_{\vec{x}'} q_b(t,\vec{x}') \rho_b(t,\vec{x}') \cdot \nabla_{\vec{x}} V_{b,a}(|\vec{x}' - \vec{x}|) d\vec{x}' \right) d\vec{x} dt,$$

so that h is outside of the inner integral.

The notation $\nabla_{\vec{x}'}$ denotes the gradient with respect to $\vec{x}'$. Finally, as assumed in Section 5.2, $V_{a,b}(|x|)$ is an even function, so that $\nabla V_{a,b}$ is an odd function. Moreover, we assume that $V_{a,b} = V_{b,a}$. Therefore, $\nabla V_{b,a}(|\vec{x}' - \vec{x}|) = -\nabla V_{a,b}(|\vec{x} - \vec{x}'|)$ and combining these observations, we obtain

$$\begin{aligned}
T_{\rho_a}(\{\rho_b\},\{q_b\})h = & \int_0^T \int_{\partial\Omega} q_a(t,\vec{x}) \left(\int_\Omega (h(t,\vec{x})\rho_a(t,\vec{x}') + \rho_a(t,\vec{x})h(t,\vec{x}')) \frac{\partial V_{a,a}(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' \right) d\vec{x} dt \\
& + \sum_{\substack{b=1 \\ b \neq a}}^{n_s} \int_0^T \int_{\partial\Omega} \left[q_a(t,\vec{x}) \int_\Omega h(t,\vec{x}) \rho_b(t,\vec{x}') \frac{\partial V_{a,b}(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' \right. \\
& \qquad \left. + q_b(t,\vec{x}) \int_\Omega \rho_b(t,\vec{x}) h(t,\vec{x}') \frac{\partial V_{b,a}(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' \right] d\vec{x} dt \\
& - \int_0^T \int_\Omega h(t,\vec{x}) \left[\int_\Omega \left(\nabla_{\vec{x}} q_a(t,\vec{x}) - \nabla_{\vec{x}'} q_a(t,\vec{x}') \right) \cdot \rho_a(t,\vec{x}') \nabla_{\vec{x}} V_{a,a}(|\vec{x} - \vec{x}'|) d\vec{x}' \right. \\
& \qquad + \sum_{\substack{b=1 \\ b \neq a}}^{n_s} \int_\Omega \nabla_{\vec{x}} q_a(t,\vec{x}) \cdot \rho_b(t,\vec{x}') \nabla_{\vec{x}} V_{a,b}(|\vec{x} - \vec{x}'|) d\vec{x}' \\
& \qquad \left. - \sum_{\substack{b=1 \\ b \neq a}}^{n_s} \int_\Omega \nabla_{\vec{x}'} q_b(t,\vec{x}') \rho_b(t,\vec{x}') \cdot \nabla_{\vec{x}} V_{a,b}(|\vec{x} - \vec{x}'|) d\vec{x}' \right] d\vec{x} dt.
\end{aligned}$$

Then the interior terms can be denoted by

$$\begin{aligned}
\mathcal{I}^*(\{\rho_b\},\{q_b\}) := & \int_\Omega \left(\nabla_{\vec{x}} q_a(t,\vec{x}) - \nabla_{\vec{x}'} q_a(t,\vec{x}') \right) \cdot \rho_a(t,\vec{x}') \nabla_{\vec{x}} V_{a,a}(|\vec{x} - \vec{x}'|) d\vec{x}' \\
& + \sum_{\substack{b=1 \\ b \neq a}}^{n_s} \int_\Omega \nabla_{\vec{x}} q_a(t,\vec{x}) \cdot \rho_b(t,\vec{x}') \nabla_{\vec{x}} V_{a,b}(|\vec{x} - \vec{x}'|) d\vec{x}' \\
& - \sum_{\substack{b=1 \\ b \neq a}}^{n_s} \int_\Omega \nabla_{\vec{x}'} q_b(t,\vec{x}') \rho_b(t,\vec{x}') \cdot \nabla_{\vec{x}} V_{a,b}(|\vec{x} - \vec{x}'|) d\vec{x}' \\
= & \sum_{b=1}^{n_s} \int_\Omega \left(\nabla_{\vec{x}} q_a(t,\vec{x}) - \nabla_{\vec{x}'} q_b(t,\vec{x}') \right) \rho_b(t,\vec{x}') \cdot \nabla_{\vec{x}} V_{a,b}(|\vec{x} - \vec{x}'|) d\vec{x}'.
\end{aligned}$$

The last equality follows by noting that the integral involving only a is a special case of the sum with $b = a$. Applying the same observation to the boundary terms, we have

$$\begin{aligned}
T_{\rho_a}(\{\rho_b\},\{q_b\})h = & \sum_{b=1}^{n_s} \int_0^T \int_{\partial\Omega} \left[q_a(t,\vec{x}) \int_\Omega h(t,\vec{x}) \rho_b(t,\vec{x}') \frac{\partial V_{a,b}(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' \right. \\
& \qquad \left. + q_b(t,\vec{x}) \int_\Omega \rho_b(t,\vec{x}) h(t,\vec{x}') \frac{\partial V_{b,a}(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' \right] d\vec{x} dt \\
& - \int_0^T \int_\Omega h(t,\vec{x}) \mathcal{I}^*(\{\rho_b\},\{q_b\}) d\vec{x} dt.
\end{aligned} \tag{7.7}$$

Considering the derivation in Section 5.2.2 for a single particle species, it is evident that the boundary terms vanish. This can be seen when applying the single particle derivation to each

term in the sum of boundary terms in (7.7). Therefore, the boundary condition of the adjoint equation for a single species ρ_a remains unaffected. Then, for each species a, the following PDE for the adjoint variable q_a arises:

$$\begin{aligned}\frac{\partial q_a}{\partial t} =& -\nabla^2 q_a - \vec{w}\cdot\nabla q_a + \nabla V_1\cdot\nabla q_a + \widehat{\rho}_a - \rho_a \\ &+ \sum_{b=1}^{n_s}\int_\Omega \rho_b(t,\vec{x}')\left(\nabla_{\vec{x}} q_a(t,\vec{x}) - \nabla_{\vec{x}'} q_b(t,\vec{x}')\right)\cdot\nabla V_{a,b}(|\vec{x}-\vec{x}'|)d\vec{x}',\end{aligned}$$

with boundary condition $\frac{\partial q_a}{\partial n} = 0$ and final time condition $q_a(T,\vec{x}) = 0$.

The final part of the first-order optimality system is to derive the gradient equation by taking the derivative of the Lagrangian (7.6) with respect to $\vec{w}$ and setting it to zero. This becomes

$$\mathcal{L}_{\vec{w}}(\{\bar{\rho}_b\},\bar{w},\{q_b\},\{q_{a,\partial\Omega}\})\vec{h} = \int_0^T\int_\Omega\left[\beta\vec{w}\cdot\vec{h} - \sum_{a=1}^{n_s}\nabla\cdot\left(\rho_a\vec{h}\right)q_a\right]d\vec{x}dt = \vec{0}.$$

Integrating by parts and noting that the boundary terms vanish due to $q_a = 0$ on $\partial\Omega$ results in

$$\mathcal{L}_{\vec{w}}(\{\bar{\rho}_b\},\bar{w},\{q_b\},\{q_{a,\partial\Omega}\})\vec{h} = \int_0^T\int_\Omega\left[\beta\vec{w}\cdot\vec{h} + \sum_{a=1}^{n_s}\rho_a\vec{h}\cdot\nabla q_a\right]d\vec{x}dt = \vec{0}.$$

Since this holds for all permissible $\vec{h}$, the gradient equation is

$$\beta\vec{w} + \sum_{a=1}^{n_s}\rho_a\nabla q_a = \vec{0}.$$

Finally, the first-order optimality system for a multiple species flow control problem with no-flux boundary conditions consists of n_s systems of the form

$$\begin{cases}
\text{State equations} & \\
\quad \frac{\partial\rho_a}{\partial t} = -\nabla\cdot\left(\mathcal{D}(\rho_a) + \mathcal{I}(\{\rho_b\}) + \mathcal{C}(\rho_a,\vec{w})\right) + f_a & \text{in } (0,T)\times\Omega, \\
\quad 0 = \left(\mathcal{D}(\rho_a) + \mathcal{I}(\{\rho_b\}) + \mathcal{C}(\rho_a,\vec{w})\right)\cdot\vec{n} & \text{on } (0,T)\times\partial\Omega, \\
\rho_a(0,\vec{x}) = \rho_{a,0}(\vec{x}) & \text{at } \{t=0\}\times\partial\Omega, \\
\\
\text{Adjoint equations} & \\
\quad \frac{\partial q_a}{\partial t} = \mathcal{D}^*(q_a) + \mathcal{I}^*(\{\rho_b\},\{q_b\}) + \mathcal{C}^*(q_a,\vec{w}) - \rho_a + \widehat{\rho}_a & \text{in } (0,T)\times\Omega, \\
\quad \frac{\partial q_a}{\partial n} = 0 & \text{on } (0,T)\times\partial\Omega, \\
q_a(T,\vec{x}) = 0 & \text{at } \{t=T\}\times\partial\Omega, \\
\\
\text{Gradient/control equations} & \\
\quad \vec{w} = -\frac{1}{\beta}\sum_{a=1}^{n_s}\rho_a\nabla q_a & \text{in } (0,T)\times\Omega,
\end{cases} \tag{7.8}$$

which can be compared to the single species system (5.21). This system of PDEs is solved by combining the MultiShape methodology with the Newton–Krylov algorithm described in Chapters 5 and 6.

7.3 Validation Tests

In order to test the numerical method for multiple species, validation strategies are developed in this section. Note that the increase in computational complexity from one to two species is

equivalent to considering two elements in the MultiShape implementation instead of one: the size of the problem doubles from M for a one-species problem to $2M$ for a two-species problem, which is cheap when compared to, for example, adding a spatial dimension.

7.3.1 Validation Setup

The validation tests presented here are devised for the optimal control problem, since this is the most complex model we consider. In order to test the implementation, we make the following observation: If we set $\rho = \rho_1 + \rho_2$, i.e., if we split a single particle density into two densities that add up to the original one, then we can test whether the system of two species 1 and 2 produces the same result as the implementation for a single species. In terms of the PDE problem, this is easy to see, considering model (7.3):

$$\begin{aligned}
\frac{\partial \rho}{\partial t} = \frac{\partial \rho_1}{\partial t} + \frac{\partial \rho_2}{\partial t} &= \nabla^2 \rho_1 + \nabla \cdot (\rho_1 \nabla V_1) + \sum_{b=1}^{2} \nabla_{\vec{x}} \cdot \int_\Omega \rho_1(\vec{x}) \rho_b(\vec{x}') \nabla_{\vec{x}'} V_{1,b}(|\vec{x} - \vec{x}'|) d\vec{x}' \\
&\quad + \nabla^2 \rho_2 + \nabla \cdot (\rho_2 \nabla V_1) + \sum_{b=1}^{2} \nabla_{\vec{x}} \cdot \int_\Omega \rho_2(\vec{x}) \rho_b(\vec{x}') \nabla_{\vec{x}'} V_{2,b}(|\vec{x} - \vec{x}'|) d\vec{x}' \\
&= \nabla^2 (\rho_1 + \rho_2) + \nabla \cdot \big((\rho_1 + \rho_2) \nabla V_1 \big) \\
&\quad + \nabla_{\vec{x}} \cdot \int_\Omega \rho_1(\vec{x}) \rho_1(\vec{x}') \nabla_{\vec{x}'} V_{1,1}(|\vec{x} - \vec{x}'|) d\vec{x}' + \nabla_{\vec{x}} \cdot \int_\Omega \rho_1(\vec{x}) \rho_2(\vec{x}') \nabla_{\vec{x}'} V_{1,2}(|\vec{x} - \vec{x}'|) d\vec{x}' \\
&\quad + \nabla_{\vec{x}} \cdot \int_\Omega \rho_2(\vec{x}) \rho_1(\vec{x}') \nabla_{\vec{x}'} V_{2,1}(|\vec{x} - \vec{x}'|) d\vec{x}' + \nabla_{\vec{x}} \cdot \int_\Omega \rho_2(\vec{x}) \rho_2(\vec{x}') \nabla_{\vec{x}'} V_{1,2}(|\vec{x} - \vec{x}'|) d\vec{x}'.
\end{aligned}$$

Now, if $\rho_1 + \rho_2 = \rho$ and if we assume that $V_{1,1} = V_{1,2} = V_{2,1} = V_{2,2}$, then we have

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= \nabla^2 \rho + \nabla \cdot (\rho \nabla V_1) \\
&\quad + \nabla_{\vec{x}} \cdot \int_\Omega \big(\rho_1(\vec{x}) \rho_1(\vec{x}') + \rho_1(\vec{x}) \rho_2(\vec{x}') + \rho_2(\vec{x}) \rho_1(\vec{x}') + \rho_2(\vec{x}) \rho_2(\vec{x}') \big) \nabla_{\vec{x}'} V_{1,2}(|\vec{x} - \vec{x}'|) d\vec{x}'.
\end{aligned}$$

Considering the factors of ρ_1 and ρ_2 in the integral term, we can rewrite them as

$$\begin{aligned}
&\rho_1(\vec{x}) \rho_1(\vec{x}') + \rho_1(\vec{x}) \rho_2(\vec{x}') + \rho_2(\vec{x}) \rho_1(\vec{x}') + \rho_2(\vec{x}) \rho_2(\vec{x}') \\
&= \rho_1(\vec{x}) \big(\rho_1(\vec{x}') + \rho_2(\vec{x}') \big) + \rho_2(\vec{x}) \big(\rho_1(\vec{x}') + \rho_2(\vec{x}') \big) = \rho_1(\vec{x}) \rho(\vec{x}') + \rho_2(\vec{x}) \rho(\vec{x}') \\
&= \big(\rho_1(\vec{x}) + \rho_2(\vec{x}) \big) \rho(\vec{x}') = \rho(\vec{x}) \rho(\vec{x}'),
\end{aligned}$$

and so

$$\frac{\partial \rho}{\partial t} = \nabla^2 \rho + \nabla \cdot (\rho \nabla V_1) + \nabla_{\vec{x}} \cdot \int_\Omega \rho(\vec{x}) \rho(\vec{x}') \nabla_{\vec{x}'} V_{1,2}(|\vec{x} - \vec{x}'|) d\vec{x}',$$

as required. A similar argument can be used for the optimal control problem (7.5). However, considering the given cost functional

$$\min_{\{\rho_a\}, \vec{w}} \mathcal{J}(\{\rho_a\}, \vec{w}) := \frac{1}{2} \sum_{a=1}^{2} \|\rho_a - \widehat{\rho}_a\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2} \|\vec{w}\|^2_{L^2((0,T)\times\Omega)},$$

we can observe that the single species term

$$\begin{aligned}
\frac{1}{2} \|\rho - \widehat{\rho}\|^2_{L^2((0,T)\times\Omega)} &= \frac{1}{2} \|\rho_1 + \rho_2 - \widehat{\rho}_1 - \widehat{\rho}_2\|^2_{L^2((0,T)\times\Omega)} = \frac{1}{2} \int_0^T \int_\Omega (\rho_1 + \rho_2 - \widehat{\rho}_1 - \widehat{\rho}_2)^2 \, d\vec{x} dt \\
&= \frac{1}{2} \int_0^T \int_\Omega (\rho_1 - \widehat{\rho}_1)^2 \, d\vec{x} dt + \frac{1}{2} \int_0^T \int_\Omega (\rho_2 - \widehat{\rho}_2)^2 \, d\vec{x} dt \\
&\quad + \int_0^T \int_\Omega (\rho_1 - \widehat{\rho}_1)(\rho_2 - \widehat{\rho}_2) \, d\vec{x} dt
\end{aligned}$$

$$= \frac{1}{2}\sum_{a=1}^{2} \|\rho_a - \widehat{\rho}_a\|^2_{L^2((0,T)\times\Omega)} + \int_0^T \int_\Omega (\rho_1 - \widehat{\rho}_1)\,(\rho_2 - \widehat{\rho}_2)\, d\vec{x}dt$$
$$\neq \frac{1}{2}\sum_{a=1}^{2} \|\rho_a - \widehat{\rho}_a\|^2_{L^2((0,T)\times\Omega)}.$$

Therefore, in order to test the optimal control problem for multiple species against that for single species, we need to temporarily add the cross term $\int_0^T \int_\Omega (\rho_1 - \widehat{\rho}_1)\,(\rho_2 - \widehat{\rho}_2)\, d\vec{x}dt$ arising from the cost functional of the single species into the optimality system. This only modifies the adjoint equations, which become

$$\frac{\partial q_a}{\partial t} = \mathcal{D}^*(q_a) + \mathcal{I}^*(\{\rho_b\},\{q_b\}) + \mathcal{C}^*(q_a,\vec{w}) - \rho_a + \widehat{\rho}_a - \rho_b + \widehat{\rho}_b. \tag{7.9}$$

If $a = 1$ and $b = 2$, we see that $\rho - \widehat{\rho} = \rho_1 - \widehat{\rho}_1 + \rho_2 - \widehat{\rho}_2$, so that the adjoint source term for each species equals the one in the single species case. This modified cost functional is only used for the validation of the code in this section. The additional terms are turned off in the implementation, using a flag, for all other examples.

The errors in this section are measured by the norm

$$\mathcal{E} = \max\left(\frac{||f_{\text{Num}} - f_{\text{Ref}}||_{L^2}}{||f_{\text{Ref}}||_{L^2} + 10^{-10}}\right), \tag{7.10}$$

compare with Section 6.4. Here f_{Ref} is either an exact solution, or the numerical solution for a one-species particle system.

7.3.2 Validation Examples

First, as in the single species case, an exact solution can be constructed to test the optimal control problem (7.8) with the modified adjoint equation (7.9) and without particle interactions, i.e., $\kappa = 0$. This is done in an equivalent fashion to the process in Section 5.5.1. We choose

$$\rho_{1,\text{ex}} = \frac{c\beta^{1/2}}{4} e^t(\cos(\pi x_1)+1)(\cos(\pi x_2)+1),$$
$$\rho_{2,\text{ex}} = \frac{(1-c)\beta^{1/2}}{4} e^t(\cos(\pi x_1)+1)(\cos(\pi x_2)+1),$$
$$q_{1,\text{ex}} = q_{2,\text{ex}} = \frac{\beta^{1/2}}{4}\left(e^T - e^t\right)(\cos(\pi x_1)+1)(\cos(\pi x_2)+1),$$

with $c = 0.3$. The considered domain is $(0,1)\times[-1,1]^2$, with $N_1 = N_2 = 20$ and $n = 11$ and the initial guesses for the Newton–Krylov algorithm are $\rho_1(t,\vec{x}) = \rho_{1,\text{ex}}(0,\vec{x})$, $\rho_2(t,\vec{x}) = \rho_{2,\text{ex}}(0,\vec{x})$ and $q_1(t,\vec{x}) = q_2(t,\vec{x}) = 0$. As in the previous chapters, the solver tolerance is 10^{-16}, the number of Newton iterations is 10, and the maximum number of GMRES iterations is 200. In Table 7.1, the numerical accuracy for different choices of β is presented. It is evident that the state and adjoint solution converge to high accuracy.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_{\rho_1}$	1.65×10^{-13}	1.65×10^{-13}	1.64×10^{-13}	1.65×10^{-13}	1.65×10^{-13}
$\mathcal{E}_{\rho_2}$	1.64×10^{-13}	1.63×10^{-13}	1.65×10^{-13}	1.64×10^{-13}	1.65×10^{-13}
$\mathcal{E}_{q_1} = \mathcal{E}_{q_2}$	3.89×10^{-13}	3.89×10^{-13}	3.90×10^{-13}	3.90×10^{-13}	3.90×10^{-13}

Table 7.1: Accuracy of the Newton–Krylov algorithm for a two-species flow control problem with no-flux boundary conditions. The error is measured against an exact solution in the norm (7.10).

Then we turn our attention to some problems in which an exact solution is not known, with the representative choice $\kappa = -1$. First, we compare the one species implementation, for a species with density ρ, with the multiple species implementation with two species of density ρ_1 and ρ_2, respectively. We set $\rho_1 = c\rho$ and $\rho_2 = (1-c)\rho$, with $c = 0.3$, so that $\rho = \rho_1 + \rho_2$. These problems for one and for two species are computed on the domain $(0,1) \times [-1,1]^2$ with $N_1 = N_2 = 20$ and $n = 11$. The initial condition for ρ is $\rho(0,\vec{x}) = \frac{1}{4}$. We write

$$\widehat{\rho} = \exp\left(-0.5(x_1 - 0.5)^2 - 0.5(x_2 - 0.5)^2\right)$$

and define the desired states for ρ_1 and ρ_2 by $\widehat{\rho}_1 = c\widehat{\rho}$ and $\widehat{\rho}_2 = (1-c)\widehat{\rho}$. The error between the one and two species implementation is measured in the norm (7.10) with $f_{\text{Num}} = \rho_1 + \rho_2$ and $f_{\text{Ref}} = \rho$. In Table 7.2 the resulting error for a range of β values is displayed.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_\rho$	3.49×10^{-15}	2.36×10^{-15}	6.80×10^{-15}	5.23×10^{-15}	4.09×10^{-15}
$\mathcal{E}_q$	2.06×10^{-11}	2.42×10^{-13}	7.61×10^{-14}	4.29×10^{-14}	1.87×10^{-14}

Table 7.2: The Newton–Krylov solution of a one-species flow control problem with no-flux boundary conditions is compared to the same problem computed with species 1 and 2 satisfying $\rho_1 = c\rho$ and $\rho_2 = (1-c)\rho$, $c = 0.3$. The error between ρ and $\rho_1 + \rho_2$, as well as $q = q_1 = q_b$, is measured in the norm (7.10).

In a second test, we split the two species not according to a constant c but to a function c_{Fun}. This is given by

$$c_{\text{Fun}} = \exp\left(x_1^2 + x_2^2\right).$$

Then $\rho_1 = c_{\text{Fun}}\rho$ and $\rho_2 = (1 - c_{\text{Fun}})\rho$. Adding an extra level of complexity, we now consider the same canonical domain, but constructed from two rectangular elements, $[-1,1] \times [-1,0]$ and $[-1,1] \times [0,1]$, in order to test that the multiple species implementation is compatible with the MultiShape implementation. Again, the error of the method is measured by computing (7.10) with $f_{\text{Num}} = \rho_1 + \rho_2$ and $f_{\text{Ref}} = \rho$. Table 7.3 displays these errors for different choices of β. It is evident that splitting ρ in a non-constant way does not impact the accuracy of the solution.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_\rho$	1.14×10^{-14}	9.99×10^{-15}	1.30×10^{-14}	1.01×10^{-14}	1.30×10^{-14}
$\mathcal{E}_q$	4.53×10^{-11}	6.85×10^{-13}	1.24×10^{-13}	2.58×10^{-14}	5.00×10^{-14}

Table 7.3: The Newton–Krylov solution of a one-species flow control problem with no-flux boundary conditions is compared to the same problem computed with species 1 and 2 satisfying $\rho_1 = c_{\text{Fun}}\rho$ and $\rho_2 = (1 - c_{\text{Fun}})\rho$. The error between ρ and $\rho_1 + \rho_2$, as well as $q = q_1 = q_b$, is measured in the norm (7.10).

Note that many further validation tests have been carried out for this implementation, but due to their repetitiveness and in the interest of brevity, these are omitted here.

7.4 Numerical Examples

In this section, problems involving multiple species models are solved numerically on a multishape, with appropriate intersection conditions applied between multishape elements. Throughout this section, initial conditions for ρ_a, $a = 1, 2$, are of the form

$$\rho_{a,\text{ic}} = \frac{c_M f_{a,\text{ic}}}{\int_\Omega f_{a,\text{ic}} d\vec{x}}. \tag{7.11}$$

Note that the model problems considered in this section are presented in order of increasing computational complexity instead of in the order in which the models were introduced in sections 7.1 and 7.2. All tests are carried out on Dell PowerEdge R430 running Scientific Linux 7, four Intel Xeon E5-2680 v3 2.5GHz, 30M Cache, 9.6 GT/s QPI 192 GB RAM and are tested using Matlab Versions 2020a–2021b.

7.4.1 Equilibrium Solution of a DDFT Model

We first solve an example involving the model equations introduced in Section 7.1.1, describing the equilibrium properties of a system with two interacting particle species. For this example, a multishape is defined using three elements: two quadrilaterals and one wedge, see Figure 7.1. The number of points on each shape is $N_1 = N_2 = 20$, chosen as a balance between accuracy and computation time, as informed by the validation tests presented in Section 6.4. We choose the parameters of the problem as follows. Interaction strengths, for an interaction potential of the form (7.2), are $\kappa_{1,1} = -7$, $\kappa_{2,2} = -3$, $\kappa_{1,2} = \kappa_{2,1} = 2$, with $\sigma_{1,1} = 0.1$, $\sigma_{2,2} = 1$, $\sigma_{1,2} = \sigma_{2,1} = 0.55$. This simulates species 1 being significantly smaller than species 2, with intra-species attractive potentials and inter-species repulsion. We furthermore impose an external potential $V_1 = 0.1x_2$, simulating gravitational forces acting on the particles. While the equilibrium results are robust for different choices of initial conditions, we choose $\rho_{1,\text{ic}}$, $\rho_{2,\text{ic}}$ of the form (7.11), with $c_M = 1$ and $f_{1,\text{ic}} = \exp\left(-0.5(x_1-1)^2 - 0.5(x_2-3.3)^2\right)$, $f_{2,\text{ic}} = \exp\left(-0.3(x_1-1.8)^2 - 0.3(x_2-2)^2\right)$, since they are relatively far from both the equilibrium and each other. With this choice of initial conditions, Algorithm 4, with $\lambda = 0.5$ and a termination tolerance of 10^{-8}, converges to an equilibrium state within 266 iterations, taking 6 seconds. The initial conditions, the equilibrium solution, as well as the convergence of the free energy of the system, are displayed in Figure 7.1. Note that, while one can expect that an equilibrium solution exists, due to the assumption of a minimum free energy of the system, one cannot draw any conclusions about uniqueness. As shown in [170], even for one-species particle dynamics models on the torus, equilibrium solutions are not necessarily unique. In the

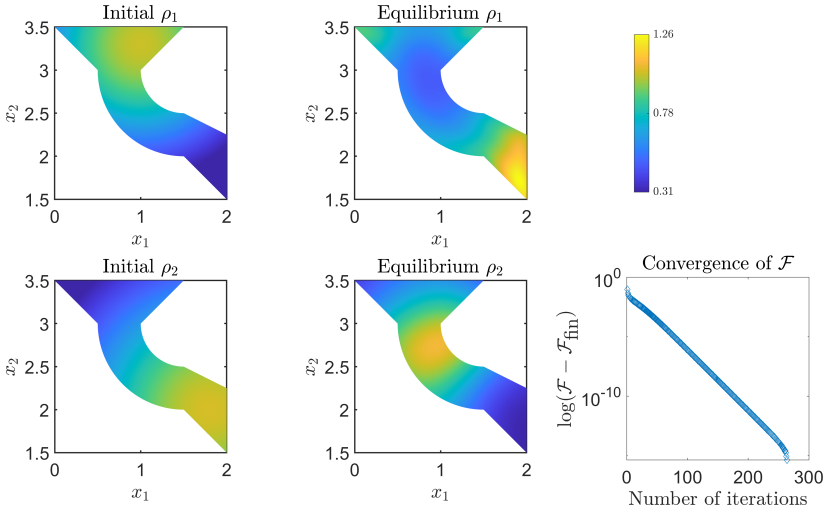

Figure 7.1: Equilibrium solution for a two species system on a multishape. Bottom right: Convergence of the free energy $\mathcal{F}$, measuring the log change in $\mathcal{F}$ shifted by the final value of the free energy $\mathcal{F}_{\text{fin}}$ in order to compare the differences in free energy rather than absolute values.

steady-state solution shown in Figure 7.1, the interplay between the gravitational forces and particle interactions can be observed. Since species 1 and 2 repel each other, they are found to separate within the domain. Furthermore, since species 1 is smaller in size than species 2, it clusters at the boundary of the domain, while species 2 accumulates in the middle. The effect of the gravitational potential is evident when considering the proportion of species 1 at the top and bottom of the domain. Since gravity is acting on the particles, it is more natural for the

particles to accumulate near the bottom of the domain. Species 2 is also subject to gravitational forces, since without gravity it would be located in the top part of the domain. Increasing the gravitational force, while keeping the interactions the same, forces species 2 downwards towards species 1, while causing a steeper accumulation of particles of species 1 in the bottom corner. However, this is not shown in Figure 7.1 in the interest of brevity.

7.4.2 Time-Dependent Solution of a DDFT Model

We now consider an example of the dynamic model (7.3), which models a funnel through which particles of different sizes are passing. There are two types of particles in the system, one species being larger than the other. We expect that smaller particles should pass through the narrow part of the funnel more easily than the larger particles. We introduce an external potential $V_g = c_g x_2$, modelling gravity, with $c_g = 0.15$. The left and right walls of the funnel are repulsive; a simple way to approximate volume exclusion effects. Since the two species are of different sizes, the repulsion is of different strengths. The external potential, modelling the repulsive walls, is defined as $V_{\text{rep}} = \epsilon \left(\exp\left(-(d_L/\alpha)^2\right) + \exp\left(-(d_R/\alpha)^2\right) \right)$, where d_L and d_R are the shortest (Euclidean) distances to the left and right boundary of the multishape respectively. We choose $\epsilon = 0.6$, $\alpha_1 = 0.5$ for species 1, $\alpha_2 = 2$ for species 2. The interactions between particles are again described by the potential (7.2), with $\kappa_{1,1} = \kappa_{2,2} = \kappa_{1,2} = \kappa_{2,1} = 0.1$ and corresponding $\sigma_{1,1} = 0.5$, $\sigma_{2,2} = 2$, $\sigma_{1,2} = \sigma_{2,1} = 1.25$. The choice of $\sigma_{2,2}$ is motivated by the fact that the narrow channel in the multishape has width 2. The initial conditions for each species are of the form (7.11), with $f_{1,\text{ic}} = f_{2,\text{ic}} = x_1 + 5$ and $c_M = 20$. The time interval for this simulation is $t \in [0, 20]$, and each element in the multishape is discretized using $N_1 = N_2 = 20$ points. The DAE solver `ode15s` [120, 121], with absolute and relative tolerances set to 10^{-9}, takes around 20 seconds to solve this problem and the result is displayed in Figure 7.2. It is evident that the behaviour of the two species is qualitatively different. Due to their size, the larger particles gather in the middle of the funnel, while the smaller particles gather at the boundaries. Both species experience the repulsion from the funnel boundary, but species 2 accumulates noticeably further away from the walls than species 1, due to their larger size. This arrangement causes the larger particles to be blocked by the smaller particles in entering the narrow part of the funnel, exacerbated by the fact that the particles repel each other. Both species are affected by gravitational forces, so that sedimentation can be observed and, over time, more particles accumulate in the bottom part of the domain. However, since the first part of the narrow channel is curved, some particles accumulate in this section of the domain before travelling downwards.

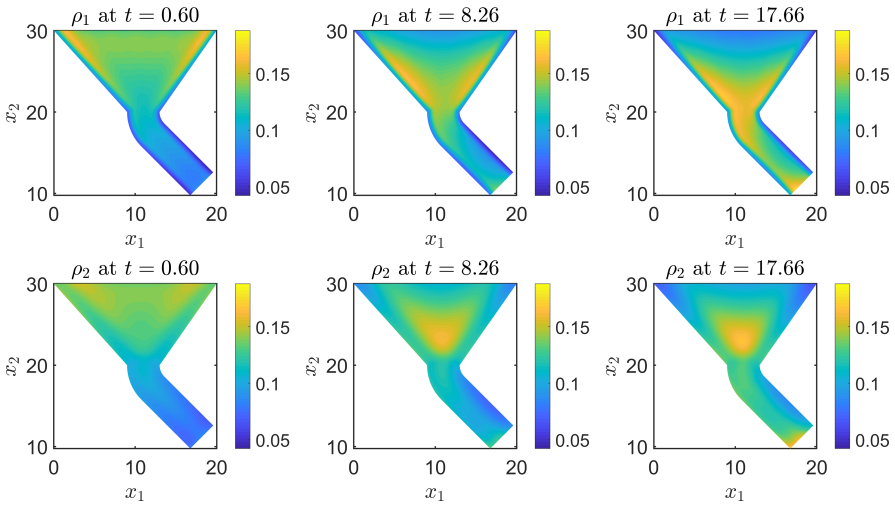

Figure 7.2: Dynamics of two interacting species of different size in a funnel under the influence of gravitational forces.

7.4.3 Solving an Optimal Control Problem

Finally, we compute results for a flow control problem of the form introduced in Section 7.2 on four elements, with $N_1 = N_2 = 14$ discretization points in space on each element and $n = 11$ points in the time interval $t \in [0, 5]$. Note that the normal vector at the intersection of the four elements (see Figure 7.3 for reference) has to be correctly defined in a manual fashion, as illustrated for a simpler test case in Figure 6.5. We choose two interacting particle species with $\sigma_{1,1} = 0.5$, $\sigma_{2,2} = 1$, with $\kappa_{1,1} = -0.8$, $\kappa_{1,2} = 0.6$, $\kappa_{2,1} = 0.2$, $\kappa_{2,2} = -0.3$. External potentials are imposed for each species: $V_1^1 = 0.05\left(\exp\left(-0.1(x_1-2)^2 - 0.1(x_2-1)^2\right) - x_2\right)$ and $V_1^2 = 0.1\left(\exp\left(-0.3(x_1-3)^2 - 0.3(x_2-4)^2\right) + 2x_2\right)$. The initial conditions for ρ_1 and ρ_2 are defined by (7.11), where $f_{1,\text{ic}} = f_{2,\text{ic}} = \exp\left(-0.15(x_1+0.5)^2 - 0.15(x_2-5)^2\right)$ and $c_M = 1$. As in previous examples, the initial guesses for the Newton–Krylov solver are $\rho_1(t, \vec{x}) = \rho_1(0, \vec{x})$, $\rho_2(t, \vec{x}) = \rho_2(0, \vec{x})$ and $q_1(t, \vec{x}) = q_2(t, \vec{x}) = 0$. We choose $\beta = 10^{-3}$, allowing a significant amount of control to be introduced into the system, so that we expect to get reasonably close to the target states $\widehat{\rho}_1$ and $\widehat{\rho}_2$. The targets are defined by a forward problem with the same initial conditions and external potential, with a background flow of strength 0.1 in the x_1 direction (along the channel in the wedge case). Without imposing control (i.e., with $\vec{w} = \vec{0}$), we can evaluate the cost functional (7.5), which results in $\mathcal{J}_{uc} = 0.0038$ ('uc' referring to 'uncontrolled'). The value of the cost functional for the optimized system is $\mathcal{J}_c = 1.7571 \times 10^{-4}$, showcasing that the overall cost obtained is significantly reduced when the optimal control is introduced. We can observe that both species of particles are carried along by the background flow field (optimal control) over time. Since species 1 experiences an upward force, it accumulates in the top-right part of the channel system at later times. Equivalently, since species 2 experiences a downward acting force, it accumulates in the bent part of the channel on the bottom right at later times of the simulation. It takes around 3 hours to solve this optimal control problem, which converges within 10 iterations of the Newton–Krylov algorithm, to a residual error of order 10^{-13} and 10^{-15} for the state and adjoint variable, respectively.

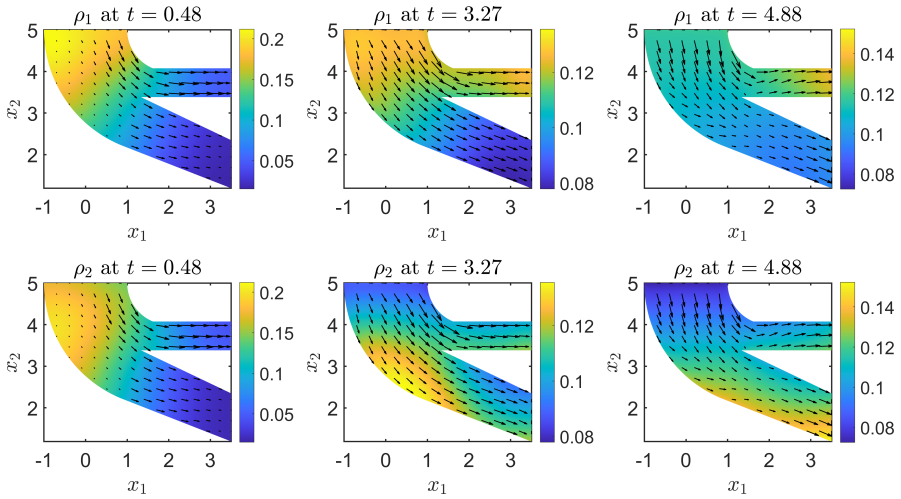

Figure 7.3: Optimal states ρ_1 and ρ_2 with superimposed optimal control $\vec{w}$.

This chapter demonstrated how the numerical framework introduced in this thesis can be extended to particle mixtures. We have introduced the DDFT model for particle mixtures and derived a corresponding optimal control problem. Validation tests for the numerical method applied to mixtures have been discussed, before presenting solutions to selected model problems. This is a first demonstration of the versatility of the method developed in this thesis, which enables the description of processes that involve different kinds of particles on complicated domains, as well as their optimal control. In the next chapter we will make a further generalization to the method by extending the DDFT model to include further physical effects.

Chapter 8

Sedimentation of Particles

Up to this point, the DDFT model considered in this thesis has described *soft particles*. These can be deformed and overlap. However, in many applications it is of importance to capture volume exclusion effects, i.e., to model *hard particles* that cannot overlap. In this way, fewer particles can occupy a given volume than if soft particles are considered, which is a more realistic account of many real-world particle systems. There exist real-world applications of both soft and hard particle systems. An example of soft particles are polymers that are able to overlap and deform, while cells are examples of particles that are deformable but that exclude volume, so that they cannot overlap. Many real-world particle systems, however, are (approximately) hard sphere systems. Such particles neither deform nor overlap, such as silver nanoparticles used in drug delivery mechanisms, micro-beads in cosmetics, or sand. Pedestrians, cars, and animals can also be thought of as hard particles. Volume exclusion effects can also be important in accurately modelling sedimentation processes. The reason for this is that, without volume exclusion, particles would form a very thin layer at the bottom of the domain. This is not physically accurate for hard particle fluids, such as sand suspended in water. The sand particles cannot overlap and create a bigger layer at the bottom of the domain than a model that does not describe volume exclusion would predict.

In this chapter we introduce one way of modelling these volume exclusion effects, demonstrate the difference between the model with and without such effects, then derive and solve an optimal control formulation involving this extended model.

8.1 The DDFT Model

In Section 2.2.3, different approximations to the excess free energy functional have been discussed. So far, in this thesis we have only considered one of the simplest such approximations: the mean-field approximation given by

$$\mathcal{F}_{\mathrm{MF}}[\rho] = \frac{1}{2}\int_\Omega\int_\Omega \rho(\vec{x})\rho(\vec{x}')V_2(|\vec{x}-\vec{x}'|)d\vec{x}'d\vec{x}.$$

This version of $\mathcal{F}_{\mathrm{exc}}$ models soft particles and cannot capture volume exclusion effects, i.e., non-overlapping particles. A successful approach of modelling hard particles is by applying Fundamental Measure Theory (FMT), see Section 2.2.3. However, this is computationally expensive, and becomes highly non-trivial when deriving optimal control problems involving such models. Instead, in this thesis, we consider a simplification of FMT, discussed in more detail in Chapter 2 and leave more complicated approximations for future work. The approximation of interest is

$$\mathcal{F}_{\mathrm{VE}}[\rho] = \int_\Omega \left(-\rho\ln(1-a\rho) + \frac{a\rho^2}{1-a\rho}\right)d\vec{x},$$

where $a = \pi R^2$ and R the particle radius, compare to (2.10). Note that we require $\rho < \frac{1}{a}$. The approximations $\mathcal{F}_{\text{MF}}$ and $\mathcal{F}_{\text{VE}}$ for the free energy can be combined to model both volume exclusion and attractive or repulsive soft interactions, which has been done in [44]. Then the Helmholtz free energy functional becomes

$$\mathcal{F}[\rho] = \int_\Omega \Big[\rho(\vec{x}) \ln\big(\rho(\vec{x}) - 1\big) + \rho(\vec{x}) V_1(\vec{x})\Big]\, d\vec{x} + \mathcal{F}_{\text{MF}}[\rho] + \mathcal{F}_{\text{VE}}[\rho]. \tag{8.1}$$

In order to write the general DDFT problem

$$\frac{\partial \rho}{\partial t} = \nabla \cdot \left(\rho \nabla \frac{\delta \mathcal{F}[\rho]}{\delta \rho}\right)$$

in terms of (8.1), we need to compute $\nabla \cdot \left(\rho \nabla \frac{\delta \mathcal{F}_{\text{VE}}[\rho]}{\delta \rho}\right)$, as done in Section 2.3.1 for the other terms in $\mathcal{F}$. This first requires taking the functional derivative of $\mathcal{F}_{\text{VE}}$:

$$\begin{aligned}
\frac{\delta \mathcal{F}_{\text{VE}}[\rho]}{\delta \rho} &= -1 - \ln(1 - a\rho) + \frac{a\rho}{1 - a\rho} + \frac{1}{1 - a\rho} + \frac{a\rho}{(1 - a\rho)^2} \\
&= -1 - \ln(1 - a\rho) + \frac{1}{(a\rho - 1)^2} - \frac{1}{a\rho - 1} - 1 \\
&= -2 - \ln(1 - a\rho) - \frac{a\rho - 2}{(a\rho - 1)^2}.
\end{aligned}$$

Taking the gradient results in

$$\begin{aligned}
\nabla \frac{\delta \mathcal{F}_{\text{VE}}[\rho]}{\delta \rho} &= -\nabla \ln(1 - a\rho) - \nabla \frac{a\rho - 2}{(a\rho - 1)^2} \\
&= -\frac{\nabla(1 - a\rho)}{1 - a\rho} - \nabla \frac{a\rho - 2}{(a\rho - 1)^2} = \frac{a\nabla\rho}{1 - a\rho} - \nabla \frac{a\rho - 2}{(a\rho - 1)^2},
\end{aligned}$$

where we have used the identity $f \nabla \ln f = \nabla f$ and noted that $1 - a\rho \neq 0$. Multiplying by ρ results in

$$\mathcal{S}(\rho) := -\rho \nabla \frac{\delta \mathcal{F}_{\text{VE}}[\rho]}{\delta \rho} = -\frac{a\rho\nabla\rho}{1 - a\rho} + \rho \nabla \frac{a\rho - 2}{(a\rho - 1)^2}. \tag{8.2}$$

Finally we take the divergence

$$\nabla \cdot \mathcal{S}(\rho) := -\nabla \cdot \left(\rho \nabla \frac{\delta \mathcal{F}_{\text{VE}}[\rho]}{\delta \rho}\right) = -\nabla \cdot \left(\frac{a\rho\nabla\rho}{1 - a\rho}\right) + \nabla \cdot \left(\rho \nabla \frac{a\rho - 2}{(a\rho - 1)^2}\right). \tag{8.3}$$

Then, these terms are added to the standard DDFT model, resulting in

$$\begin{aligned}
\frac{\partial \rho}{\partial t} = \nabla^2 \rho + \nabla \cdot \left(\frac{a\rho\nabla\rho}{1 - a\rho}\right) - \nabla \cdot \left(\rho \nabla \frac{a\rho - 2}{(a\rho - 1)^2}\right) + \nabla \cdot (\rho \nabla V_1) \\
+ \nabla \cdot \int_\Omega \rho(t, \vec{x}) \rho(t, \vec{x}') \nabla V_2(|\vec{x} - \vec{x}'|) d\vec{x}' \qquad \text{in } (0, T) \times \Omega.
\end{aligned} \tag{8.4}$$

The additional term for no-flux boundary conditions is (8.2), so that the boundary conditions become

$$\frac{\partial \rho}{\partial n} + \frac{a\rho\nabla\rho}{1 - a\rho} \cdot \vec{n} - \rho \nabla \frac{a\rho - 2}{(a\rho - 1)^2} \cdot \vec{n} + \rho \frac{\partial V_1}{\partial n} + \int_\Omega \rho(t, \vec{x}) \rho(t, \vec{x}') \frac{\partial V_2(|\vec{x} - \vec{x}'|)}{\partial n} d\vec{x}' = 0$$
$$\text{on } (0, T) \times \partial\Omega.$$

This integro-PDE can be solved on complicated domains using the MultiShape package, as described in Chapter 6, which will be demonstrated in Section 8.4.

8.2 The Optimal Control Problem

Next, the modified forward problem is applied as PDE constraint in an optimal control problem. In order to do this, we refer back to Chapter 5. Modifying (5.8) by adding (8.3), we obtain the optimal control problem

$$\begin{aligned}
&\min_{\rho,\vec{w}} \mathcal{J}(\rho,\vec{w}) := \frac{1}{2}\|\rho-\widehat{\rho}\|^2_{L^2((0,T)\times\Omega)} + \frac{\beta}{2}\|\vec{w}\|^2_{L^2((0,T)\times\Omega)} \qquad (8.5)\\
&\text{subject to}\\
&\frac{\partial\rho}{\partial t} = -\nabla\cdot\big(\mathcal{D}(\rho)+\mathcal{S}(\rho)+\mathcal{I}(\rho)+\mathcal{C}(\rho,\vec{w})\big)+f \quad \text{in } (0,T)\times\Omega,\\
&\rho(0,\vec{x}) = \rho_0(\vec{x}) \qquad\qquad\qquad\qquad\qquad\qquad \text{at } \{t=0\}\times\Omega,
\end{aligned}$$

where $\mathcal{D}$, $\mathcal{S}$, $\mathcal{I}$, and $\mathcal{C}$ are defined in (5.3), (8.2), (5.4), and (5.9), respectively. The PDE constraint is equal to (8.4) with additional terms $\mathcal{C}(\rho,\vec{w})$ and f.

Modifying the no-flux conditions (5.10) by adding (8.2) results in

$$\big(\mathcal{D}(\rho)+\mathcal{S}(\rho)+\mathcal{I}(\rho)+\mathcal{C}(\rho,\vec{w})\big)\cdot\vec{n} = 0 \quad \text{on } (0,T)\times\partial\Omega.$$

Note that one can, as before, swap the flow control term for a source control term and no-flux boundary conditions for Dirichlet conditions without much additional work.

8.3 Derivation of First-Order Optimality System

In order to derive the first-order optimality system, one has to investigate the effect of the additional term $\nabla\cdot\mathcal{S}(\rho)$. The modified Lagrangian is

$$\begin{aligned}
\mathcal{L}(\rho,\vec{w},q,q_{\partial\Omega}) = \mathcal{J}(\rho,\vec{w}) &- \int_0^T\int_\Omega \left(\frac{\partial\rho}{\partial t}+\nabla\cdot\big(\mathcal{D}(\rho)+\mathcal{S}(\rho)+\mathcal{I}(\rho)+\mathcal{C}(\rho,\vec{w})\big)-f\right) q\,d\vec{x}dt\\
&- \int_0^T\int_{\partial\Omega}\big(\mathcal{D}(\rho)+\mathcal{S}(\rho)+\mathcal{I}(\rho)+\mathcal{C}(\rho,\vec{w})\big)\cdot\vec{n}\,q_{\partial\Omega}\,d\vec{x}dt.
\end{aligned}$$

We consider the contribution of $\mathcal{S}$ to the adjoint equation and compute $\mathcal{L}_\rho(\bar{\rho},\bar{w},q,q_{\partial\Omega})h = 0$. Since we know the contribution of all other terms, we consider only the term

$$-\int_0^T\int_\Omega \nabla\cdot S(\rho)q\,d\vec{x}dt = \int_0^T\int_\Omega\left[\nabla\cdot\left(\frac{a\rho\nabla\rho}{1-a\rho}\right)-\nabla\cdot\left(\rho\nabla\frac{a\rho-2}{(a\rho-1)^2}\right)\right]q\,d\vec{x}dt =: T_1+T_2.$$

Following the approach in 5.2, taking derivatives with respect to ρ in direction h gives

$$\begin{aligned}
\left(T_1\right)_\rho(\rho,q)h &= \int_0^T\int_\Omega\left[\nabla\cdot\left(\nabla h\frac{a\rho}{1-a\rho}+h\frac{a\nabla\rho}{(1-a\rho)^2}\right)\right]q\,d\vec{x}dt,\\
\left(T_2\right)_\rho(\rho,q)h &= \int_0^T\int_\Omega\left[-\nabla\cdot\left(h\nabla\frac{a\rho-2}{(a\rho-1)^2}+\rho\nabla\frac{ha(3-a\rho)}{(a\rho-1)^3}\right)\right]q\,d\vec{x}dt,
\end{aligned}$$

resulting in the contribution to the Lagrangian derivative

$$\begin{aligned}
\left(T_1\right)_\rho(\rho,q)h+\left(T_2\right)_\rho(\rho,q)h = \int_0^T\int_\Omega\Bigg[&\nabla\cdot\left(\nabla h\frac{a\rho}{1-a\rho}+h\frac{a\nabla\rho}{(1-a\rho)^2}\right)\\
&-\nabla\cdot\left(h\nabla\frac{a\rho-2}{(a\rho-1)^2}+\rho\nabla\frac{ha(3-a\rho)}{(a\rho-1)^3}\right)\Bigg]q\,d\vec{x}dt.
\end{aligned}$$

Rearranging in terms of h and its derivatives, and using the product rule when necessary gives

$$(T_1)_\rho(\rho,q)h + (T_2)_\rho(\rho,q)h = \int_0^T \int_\Omega \left[\nabla \cdot \left(\nabla h \left(\frac{a\rho}{1-a\rho} - \frac{a\rho(3-a\rho)}{(a\rho-1)^3} \right) \right) \right.$$
$$\left. + \nabla \cdot \left(h \left(\frac{a\nabla\rho}{(1-a\rho)^2} - \nabla \frac{a\rho-2}{(a\rho-1)^2} - \rho\nabla \frac{a(3-a\rho)}{(a\rho-1)^3} \right) \right) \right] q d\vec{x} dt.$$

We can define

$$\begin{aligned} A :=& \frac{a\rho}{1-a\rho} - \frac{a\rho(3-a\rho)}{(a\rho-1)^3} \\ \vec{B} :=& \frac{a\nabla\rho}{(1-a\rho)^2} - \nabla\left(\frac{a\rho-2}{(a\rho-1)^2} \right) - \rho\nabla\left(\frac{a(3-a\rho)}{(a\rho-1)^3} \right), \end{aligned} \tag{8.6}$$

and note that $\nabla A = \vec{B}$. Then

$$(T_1)_\rho(\rho,q)h + (T_2)_\rho(\rho,q)h = \int_0^T \int_\Omega q \left[\nabla \cdot (A\nabla h) + \nabla \cdot \left(h\vec{B} \right) \right] d\vec{x} dt.$$

Integrating this term by parts results in

$$\begin{aligned} (T_1)_\rho(\rho,q)h + (T_2)_\rho(\rho,q)h =& \int_0^T \int_{\partial\Omega} \left(qh\vec{B} + qA\nabla h - hA\nabla q \right) \cdot \vec{n} d\vec{x} dt \\ & - \int_0^T \int_\Omega h \left[\vec{B} \cdot \nabla q - \nabla \cdot (A\nabla q) \right] d\vec{x} dt. \end{aligned} \tag{8.7}$$

As done in Chapter 5, we restrict h so that $h = 0$, $\frac{\partial h}{\partial n} = 0$ on $(0,T) \times \partial\Omega$. Then, a density argument that was applied to (5.12) in Chapter 5 is used here to conclude that the contribution from $\mathcal{S}$ to the adjoint equation is

$$\mathcal{S}^*(q,\rho) := \vec{B} \cdot \nabla q - \nabla \cdot (A\nabla q). \tag{8.8}$$

The modified adjoint equation is

$$\frac{\partial q}{\partial t} = \mathcal{D}^*(q) + \mathcal{S}^*(q,\rho) + \mathcal{I}^*(q,\rho) + \mathcal{C}^*(q,\vec{w}) - \rho + \widehat{\rho},$$

where the $\mathcal{D}^*$, $\mathcal{I}^*$, and C^* are defined in (5.13), (5.17), and (5.18). Moreover, we have, after some algebraic simplifications,

$$-\nabla^2 q + \mathcal{S}^*(q,\rho) = \frac{1+a\rho}{(a\rho-1)^3} \nabla^2 q. \tag{8.9}$$

8.3.1 Boundary Terms

Next, the effect of $\mathcal{S}$ on the boundary conditions of the adjoint equation is considered. The relevant term in the Lagrangian is

$$T_3(\rho, q_{\partial\Omega}) := -\int_0^T \int_{\partial\Omega} \mathcal{S}(\rho) \cdot \vec{n} q_{\partial\Omega} d\vec{x} dt = \int_0^T \int_{\partial\Omega} \left(\frac{a\rho\nabla\rho}{1-a\rho} - \rho\nabla \frac{a\rho-2}{(a\rho-1)^2} \right) \cdot \vec{n} q_{\partial\Omega} d\vec{x} dt.$$

Taking derivatives with respect to ρ, we obtain

$$\begin{aligned}
T_{3,\rho}(\rho, q_{\partial\Omega})h &= \int_0^T \int_{\partial\Omega} \left[\nabla h \frac{a\rho}{1-a\rho} + h\frac{a\nabla\rho}{(1-a\rho)^2} - h\nabla\left(\frac{a\rho-2}{(a\rho-1)^2}\right) \right. \\
&\qquad \left. -\rho\nabla\left(h\frac{a(3-a\rho)}{(a\rho-1)^3}\right)\right] \cdot \vec{n} q_{\partial\Omega} d\vec{x} dt \\
&= \int_0^T \int_{\partial\Omega} \left[\nabla h \frac{a\rho}{1-a\rho} + h\frac{a\nabla\rho}{(1-a\rho)^2} - h\nabla\left(\frac{a\rho-2}{(a\rho-1)^2}\right) \right. \\
&\qquad \left. -\nabla h\frac{a\rho(3-a\rho)}{(a\rho-1)^3} - h\rho\nabla\frac{a(3-a\rho)}{(a\rho-1)^3}\right] \cdot \vec{n} q_{\partial\Omega} d\vec{x} dt \\
&= \int_0^T \int_{\partial\Omega} \left(\nabla h A + h\vec{B}\right) \cdot \vec{n} q_{\partial\Omega} d\vec{x} dt,
\end{aligned}$$

where we used A and $\vec{B}$ defined in (8.6). Combining this with the boundary terms in (8.7) gives

$$\int_0^T \int_{\partial\Omega} \left(qh\vec{B} + qA\nabla h - hA\nabla q\right) \cdot \vec{n} dr dt + \int_0^T \int_{\partial\Omega} \left(\nabla h A + h\vec{B}\right) \cdot \vec{n} q_{\partial\Omega} d\vec{x} dt.$$

Collecting terms in h and ∇h results in

$$\int_0^T \int_{\partial\Omega} \left(-hA\nabla q + h\vec{B}(q + q_{\partial\Omega}) + \nabla h A(q + q_{\partial\Omega})\right) \cdot \vec{n} d\vec{x} dt.$$

Restricting h such that $h = 0$ on $(0,T) \times \partial\Omega$, we obtain

$$\int_0^T \int_{\partial\Omega} \nabla h A(q + q_{\partial\Omega}) \cdot \vec{n} dr dt,$$

which results in $q = -q_{\partial\Omega}$. Then, loosening the restrictions on h results in

$$\int_0^T \int_{\partial\Omega} h\left(-A\nabla q + \vec{B}(q + q_{\partial\Omega})\right) \cdot \vec{n} d\vec{x} dt,$$

so that the additional contribution to the adjoint boundary condition is

$$-A\nabla q \cdot \vec{n}.$$

Combining this with (5.20) results in the modified boundary condition

$$-\frac{1+a\rho}{(a\rho-1)^3}\nabla q \cdot \vec{n} = 0,$$

which can be compared with (8.9).

The term $\mathcal{S}$ does not depend on $\vec{w}$ so that it does not impact the gradient equation. The full optimality system therefore reads

$$\begin{cases} \text{State equation} \\ \quad \frac{\partial \rho}{\partial t} = -\nabla \cdot \big(\mathcal{D}(\rho) + \mathcal{S}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho, \vec{w})\big) + f & \text{in } (0,T)\times\Omega, \\ \quad 0 = \big(\mathcal{D}(\rho) + \mathcal{S}(\rho) + \mathcal{I}(\rho) + \mathcal{C}(\rho, \vec{w})\big) \cdot \vec{n} & \text{on } (0,T)\times\partial\Omega, \\ \rho(0,\vec{x}) = \rho_0(\vec{x}) & \text{at } \{t=0\}\times\partial\Omega, \\ \\ \text{Adjoint equation} \\ \quad \frac{\partial q}{\partial t} = \mathcal{D}^*(q) + \mathcal{S}^*(q,\rho) + \mathcal{I}^*(q,\rho) + \mathcal{C}^*(q,\vec{w}) - \rho + \widehat{\rho} & \text{in } (0,T)\times\Omega, \\ \quad 0 = \dfrac{1+a\rho}{(a\rho-1)^3}\dfrac{\partial q}{\partial n} & \text{on } (0,T)\times\partial\Omega, \\ q(T,\vec{x}) = 0 & \text{at } \{t=T\}\times\partial\Omega, \\ \\ \text{Gradient/control equation} \\ \quad \vec{w} = -\frac{1}{\beta}\rho\nabla q & \text{in } (0,T)\times\Omega, \end{cases} \tag{8.10}$$

compare to (5.21). The operators are defined in (5.3), (8.2) (5.4), (5.9), (5.13), (8.8), (5.17), and (5.18). This can be solved with the Newton–Krylov algorithm introduced in Chapter 5, in combination with the MultiShape package described in Chapter 6, and examples of this are presented in the following.

8.4 Numerical Examples

In this section, we present numerical examples for the modified DDFT model introduced in this chapter and the corresponding optimal control problem. First, the optimal control solution is compared to exact solutions, as done in the previous chapters, to validate the numerical implementation. Then, the new DDFT model is investigated by comparing it to the DDFT model that does not approximate volume exclusion (i.e., $\mathcal{S} = 0$), see (3.59), and to existing results published in [44]. The optimal control problem is investigated in a next step. First, it is compared to the optimal control model with $\mathcal{S} = 0$. Then, two optimal control examples on MultiShapes are presented. In all examples $a = \pi/4$, $\sigma = \frac{1}{2}$, unless stated otherwise. As before, the initial guesses for the Newton–Krylov solver are $\rho(t,\vec{x}) = \rho(0,\vec{x})$ and $q(t,\vec{x}) = 0$, for all t. Moreover, the ODE solver tolerance is set to 10^{-7}, and the inputs for the Newton–Krylov algorithm are the tolerance 10^{-16}, 10 Newton iterations, and a maximum of 200 GMRES iterations. All tests are computed on Dell PowerEdge R430 running Scientific Linux 7, four Intel Xeon E5-2680 v3 2.5GHz, 30M Cache, 9.6 GT/s QPI 192 GB RAM, using Matlab Versions 2020a–2021b.

8.4.1 Validation Tests

As in the previous chapters, we compare the numerical solution of the optimal control problem to an exact solution. As in Chapter 7, we omit the tests for the forward problem in the interest of brevity. Following the approach in Section 5.5, an exact solution can be constructed by choosing

$$\begin{aligned} \rho_{\text{ex}}(t,x) &= \frac{c}{10}\beta^{1/2}e^t\cos(\pi x_1)\cos(\pi x_2), \\ q_{\text{ex}}(t,x) &= \frac{c}{10}\beta^{1/2}\left(e^T - e^t\right)\cos(\pi x_1)\cos(\pi x_2). \end{aligned}$$

The additional constant c has to be defined differently for different choices of β. Recalling that the factor $\beta^{1/2}$ was introduced in previous sections to avoid advection dominated problems, we are now faced with an additional challenge. If we choose $c = 1$ for all β, and β is large, then this causes issues with the numerical solution. The reason is that if $1 - a\rho = 0$ at any point

$(t, \vec{x})$, the denominator of $\mathcal{S}$ is zero. To avoid such a situation, we define $c = 1$ for $\beta \leq 1$ and $c = \frac{1}{\beta}$ for $\beta > 1$. The two cases are necessary, since if $\beta < 1$, $\frac{1}{\beta}$ becomes large, causing the same problem as described above for large β.

The advective field $\vec{w}_{\text{ex}}$ is constructed using the choices for ρ and q and we define

$$f_{\text{ex}} = \frac{\partial \rho_{\text{ex}}}{\partial t} + \nabla \cdot \left(\mathcal{D}(\rho_{\text{ex}}) + \mathcal{S}(\rho_{\text{ex}}) + \mathcal{C}(\rho_{\text{ex}}, \vec{w}_{\text{ex}})\right),$$
$$\widehat{\rho}_{\text{ex}} = \frac{\partial q_{\text{ex}}}{\partial t} - \mathcal{D}^*(q_{\text{ex}}) - \mathcal{S}^*(q_{\text{ex}}, \rho_{\text{ex}}) - \mathcal{C}^*(q_{\text{ex}}, \vec{w}_{\text{ex}}) + \rho_{\text{ex}},$$

to satisfy the PDE system (8.10), compare to (5.28). In Table 8.1 the numerical error for different values of β is shown. Here, the chosen domain is $(0, 1) \times [-1, 1]^2$. The spatial MultiShape domain is divided into two quadrilateral elements, $[-1, 1] \times [-1, 0]$ and $[-1, 1] \times [0, 1]$, with each element containing $N_1 = N_2 = 30$ spatial points, and $n = 11$ is the number of time points. The error measure is again given by (7.10). It is evident that, as in the previous chapters, the numerical solution converges to the exact solution to high orders of accuracy.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$	$\beta = 10^{3}$
$\mathcal{E}_\rho$	1.38×10^{-13}	1.43×10^{-13}	1.49×10^{-11}	1.48×10^{-11}	1.42×10^{-13}
$\mathcal{E}_q$	4.44×10^{-14}	3.34×10^{-14}	1.96×10^{-11}	1.97×10^{-11}	3.37×10^{-14}

Table 8.1: Numerical accuracy of optimal control problem (8.5) with no-flux boundary conditions for different choices of β. The error is measured against an exact solution in the norm (7.10).

8.4.2 Solving DDFT Models

Comparison with the Previous Model

In this chapter, we have added the term $\mathcal{S}$ to the DDFT model (3.59) to capture volume exclusion effects. This means that particles cannot overlap and therefore we expect that they cannot be packed as densely as in the model with $\mathcal{S} = 0$. We choose weakly attractive particles, with interaction strength $\kappa = -0.5$, that experience a gravitational force given by the potential $V_1 = 0.1x_2$. The temporal–spatial domain is $(0, 10) \times [0, 20] \times [0, 15]$. The spatial MultiShape domain is split into two quadrilateral elements, $[0, 10] \times [0, 15]$ and $[10, 20] \times [0, 15]$, with $N_1 = N_2 = 20$ discretization points each. The initial condition for ρ is given by

$$\rho(0, \vec{x}) = \bar{\rho},$$

where $\bar{\rho}$ is a constant, denoting the average particle density.

We study the solution to the models with $\mathcal{S} = 0$, (3.59), and $\mathcal{S} \neq 0$, (8.4), for varying $\bar{\rho}$. In Figure 8.1, this is presented for $\bar{\rho} = 0.1, 0.3, 0.5$. It is evident that in all cases the model with $\mathcal{S} \neq 0$ permits less particle clustering than the model with $\mathcal{S} = 0$, as expected. For lower densities, the two models exhibit similar results, while for higher densities, the two different models produce very different results. This demonstrates that volume exclusion can be an important feature when modelling particle dynamics, such as in sedimentation phenomena, as discussed above.

Comparison to Existing Results

As the chosen model is inspired by the work [44], we are interested in whether we are able to reproduce some of the results in this paper, and study the effects of different boundary conditions. In [44], it is investigated how attractive particles impact sedimentation processes. The authors choose a domain $(0, 300) \times [0, 64] \times [0, 43.5]$. The particles experience the effect of

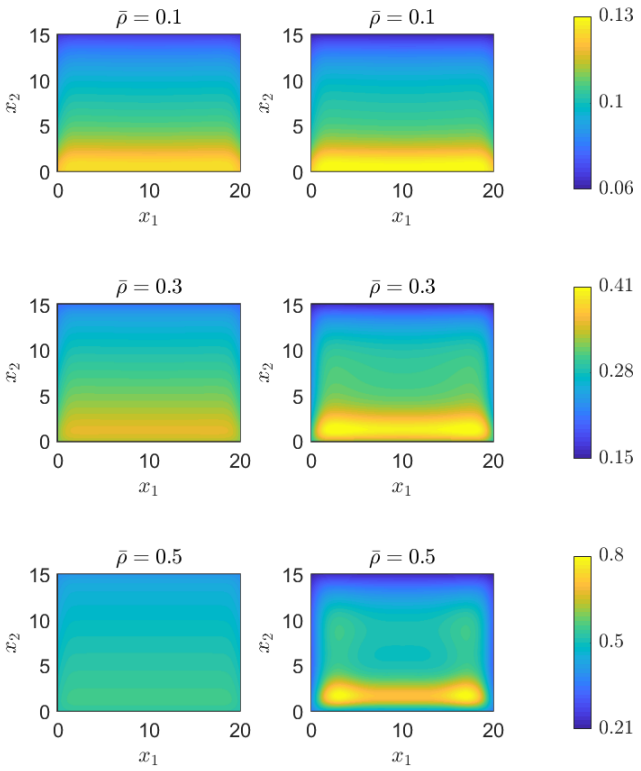

Figure 8.1: Comparison of a model problem with $\mathcal{S} \neq 0$ (left) and $\mathcal{S} = 0$ (right) for different average particle densities $\bar{\rho}$ at $t = 10$.

the external potential $V_1 = 0.1x_2$ and particle–particle interactions given by the potential

$$V_2(\|x\|) = \kappa e^{-\|x\|/\sigma},$$

where $\sigma = 1$ is the particle radius, as before, and $\kappa = -3.5$.

Two DDFT solutions are presented in [44], both computed in a periodic box, with the left and right boundaries being periodic and the top and bottom boundaries modelled as hard walls, using an infinite external potential for $x_2 < 0$ and $x_2 > 43.5$. The first result is computed with an average density $\bar{\rho} = 0.072$, while the second model has average density $\bar{\rho} = 0.2$. The initial condition is given by

$$\rho(0, \vec{x}) = \bar{\rho} \pm \frac{\xi}{20}\bar{\rho},$$

where ξ is a uniform random variable in $[0, 1]$. In both examples phase separation occurs and the particles form a small number of clusters. However, the authors in [44] find that for lower densities, at later times, the particles form a finite number of clusters on the bottom of the domain, while for the higher density example the particles form a single layer of particles. This is explained by the fact that the particles minimize the free energy in the system, which in the lower density regime causes the individual clusters.

In this thesis, these results are replicated with one small adjustment, namely that the hard walls are not imposed by an external potential but by the no-flux boundary conditions of our software. We choose $N_1 = N_2 = 100$ computational points, since the phase separation is hard

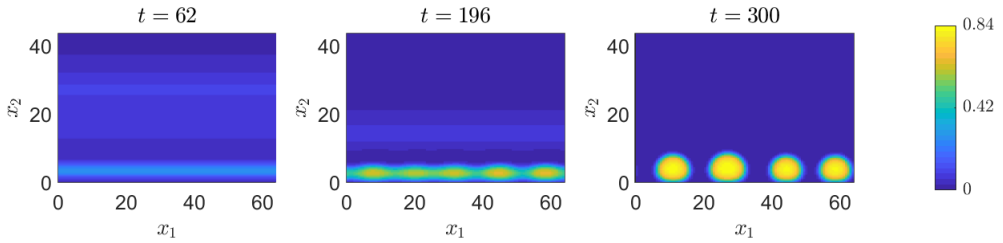

Figure 8.2: Sedimentation of attractive particles in a periodic box with an average density of $\bar{\rho} = 0.072$.

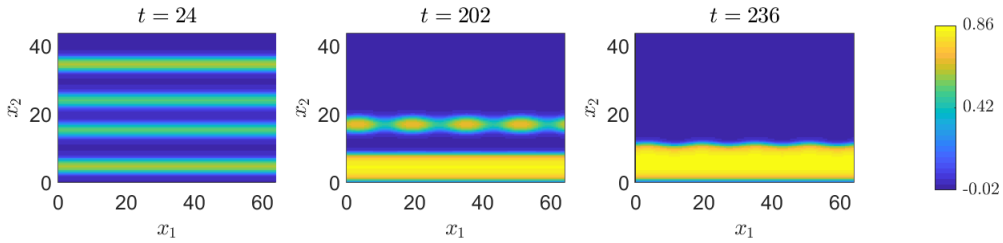

Figure 8.3: Sedimentation of attractive particles in a periodic box with an average density of $\bar{\rho} = 0.2$.

to resolve numerically. The computational domain is a periodic box. The computational time for the problem with $\bar{\rho} = 0.072$ is 3 hours, while the time to compute the problem with $\bar{\rho} = 0.2$ is 11 hours. This demonstrates the increased difficulty of the problem with increased mass in the system. Figures 8.2 and 8.3 show the resulting density profiles at different times. These are similar to those reported in [44], and the predicted clustering for $\bar{\rho} = 0.072$, as well as the uniform layer of particles for $\bar{\rho} = 0.2$ at later times is found. There is a difference in the particle evolution for the model with $\bar{\rho} = 0.2$ when compared to the result in [44]. In the paper, individual clusters form throughout the domain before the layer of particles forms at later times. In our simulation, several layers of particles form, then some small clusters of particles emerge, before the characteristic single layer of particles is created. These differences in result may be caused by the different initial noise, or by the application of no-flux conditions at the top and bottom walls, instead of a confining external potential modelling hard walls.

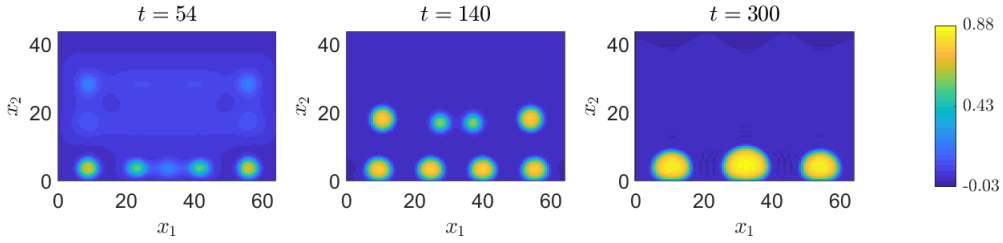

Figure 8.4: Sedimentation of attractive particles in a no-flux box with an average density of $\bar{\rho} = 0.072$.

Next, the same experiments are carried out, but with no-flux conditions on all four boundaries, see Figures 8.4 and 8.5. The chosen computational domain is a box. The computational times here are 12 hours for the problem with $\bar{\rho} = 0.072$ and 16 hours for the problem with $\bar{\rho} = 0.2$. This demonstrates that the problem with no-flux boundary conditions is harder to solve numerically than the problem with periodic conditions. Comparing the results in Figure 8.2 and 8.4, it is evident that while the particles form a small number of clusters in both cases, they do exhibit differences. Since the periodic problem does not have a boundary in x_1, first the particles only arrange in lines, before the particle clusters arrange approximately evenly

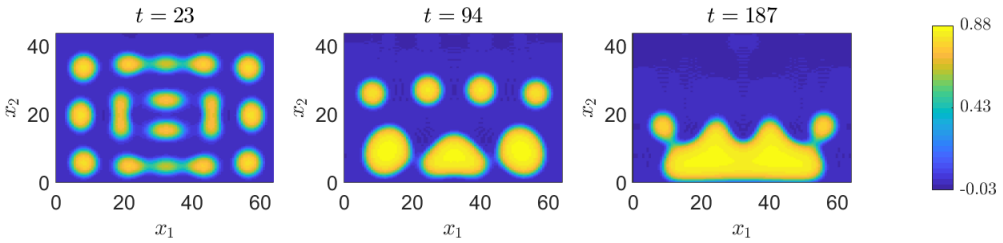

Figure 8.5: Sedimentation of attractive particles in a no-flux box with an average density of $\bar{\rho} = 0.2$.

spaced in x_1. In the example with no-flux boundary conditions, there is now a boundary in the x_1 direction and particles cluster earlier in the simulation and closer to the middle of the domain. Therefore, at later times, the middle cluster is larger than the other clusters of particles. A similar effect can be observed for the simulation with $\bar{\rho} = 0.2$ displayed in Figure 8.5. The particles form clusters, rather than lines, due to the symmetry breaking in x_1 caused by the no-flux boundary conditions. Over time, differently sized, larger clusters form, before one uniform clump of particles is created at later times. This resembles the single particle layer found for the periodic domain, but the no-flux condition prevents the spread of particles to the walls at $x_1 = 0$ and $x_1 = 64$.

8.4.3 Solving Optimal Control Problems

We are now in the position to consider some optimal control problems of the form (8.5). We first compare the two models with $\mathcal{S} = 0$, (5.8), and $\mathcal{S} \neq 0$, (8.5), as done for the forward problems, before presenting two optimal control problems on a MultiShape with $\mathcal{S} \neq 0$. The first-order optimality systems to be solved are (5.21), for $\mathcal{S} = 0$, and (8.10), for $\mathcal{S} \neq 0$.

Comparison Between the Two Models

As for the forward problem, we would like to investigate the effect of the additional term $\mathcal{S}$ on the model, and consequently on a corresponding optimal control problem. For this, we consider the example presented in Section 8.4.2 with $\bar{\rho} = 0.4$ and interaction strength $\kappa = -0.3$. The desired state $\widehat{\rho}$ is created by a forward problem with $\mathcal{S} \neq 0$ and $V_1 = 0.1x_2$. The optimal control problems are computed with $V_1 = 0$, so that the control has to act as a gravitational force. We choose a time horizon $(0, 10)$, and a MultiShape domain consisting of the two quadrilaterals $[-1, 1] \times [-1, 0]$ and $[-1, 1] \times [0, 1]$, with each $N_1 = N_2 = 20$ spatial points, and $n = 11$ points in time. Computing the optimal control problem with $\mathcal{S} \neq 0$, for $\beta = 10^{-3}$, the optimal cost is $\mathcal{J}_c = 0.0142$, as compared to $\mathcal{J}_{uc} = 1.0244$. Computing the optimal control problem with $\mathcal{S} = 0$, for $\beta = 10^{-3}$, the optimal cost is $\mathcal{J}_c = 0.0108$, as compared to $\mathcal{J}_{uc} = 1.2870$. In Figure 8.6, the solutions for both models are presented. It is evident that while the particle distributions are similar in both cases, due to the choice of β, the optimal controls for the two models act in different ways. While the optimal control for $\mathcal{S} \neq 0$ acts downward, in lieu of the gravitational force, the optimal control of the model with $\mathcal{S} = 0$ invests more control to spread the particles out, since in this model the particles are allowed to cluster more tightly than in the target state, created with the volume excluding ($\mathcal{S} \neq 0$) model.

Note that we do not chose a $\widehat{\rho}$ created from a DDFT model with $\mathcal{S} = 0$. The reason for this is that the optimal control for the problem with $\mathcal{S} \neq 0$ would attempt to push the particles to cluster more tightly than the volume exclusion property of this model allows. However, this would cause numerical issues, since $\rho \to \frac{1}{a}$ in this case, causing the term $\mathcal{S}$ to become very large.

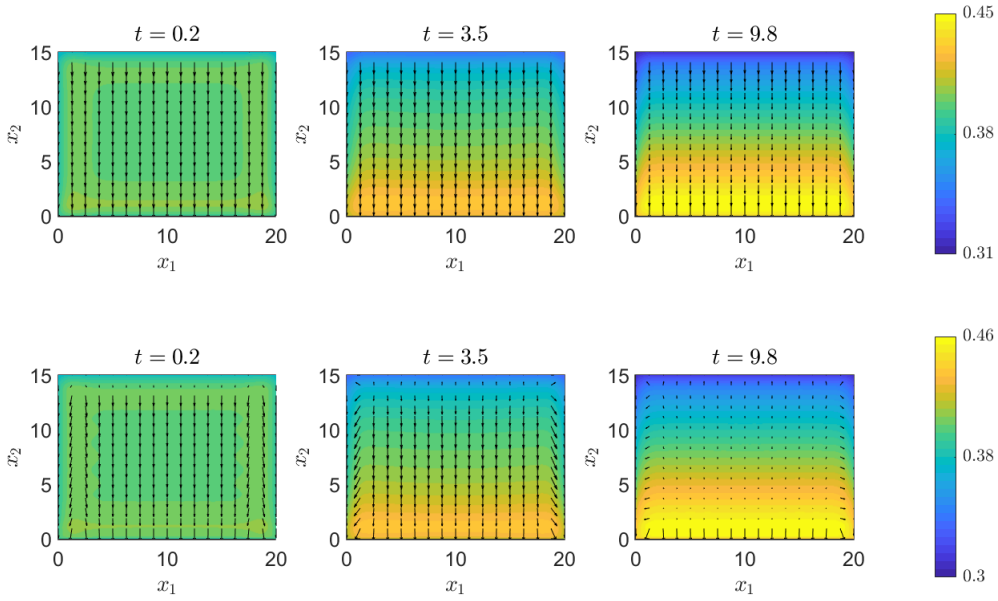

Figure 8.6: Comparison between the optimal solution ρ and control $\vec{w}$ for $\mathcal{S} \neq 0$ (top) and $\mathcal{S} = 0$ (bottom) for $\beta = 10^{-3}$.

Two Examples on MultiShapes

We now compute two optimal control examples with $\mathcal{S} \neq 0$ on a MultiShape domain. First, we consider an example constructed from two quadrilateral shapes, inspired by the shape of a brewing vessel. We choose the time horizon $(0, 1)$ and for each element $N_1 = N_2 = 20$ and $n = 11$. The initial condition for ρ is chosen to be $\rho(0, \vec{x}) = Z^{-1} \exp(-0.5(x_1 - 0.75)^2 - 0.5(x_2 - 2)^2)$, where Z^{-1} is such that $\bar{\rho} = 0.2$. The particle–particle interaction strength is $\kappa = -2$. The desired state is created by computing a forward problem of the above specification with an additional external potential of the form $V_1 = 0.5x_2$ acting on the particles. Moreover, an advective field $\vec{w} = 0.1[1, 1]^\top$ is imposed. Since the external potential is acting downwards and the advective field directs the particles to the top right, we expect the optimal control to behave like a combination of these two forces. Since the gravitational force is stronger than the advective field, we expect the optimal control to push particles towards the bottom right corner of the domain. In Figure 8.7 the result is displayed for $\beta = 10^{-5}$. It is evident that the control $\vec{w}$ is acting as expected, so that, over time, the particles accumulate in the bottom right corner of the domain. In Table 8.2 the cost functional of the optimal result is compared to that of the uncontrolled problem, for different values of β.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$	$\beta = 10^{1}$
$\mathcal{J}_{uc}$	1.46×10^{-4}	1.46×10^{-4}	1.46×10^{-4}	1.46×10^{-4}
$\mathcal{J}_c$	2.32×10^{-6}	8.15×10^{-5}	1.45×10^{-4}	1.46×10^{-4}

Table 8.2: Optimal control example with no-flux boundary conditions and volume exclusion ($\mathcal{S} \neq 0$). The value of the cost functional of the uncontrolled problem with $\vec{w} = \vec{0}$, denoted by $\mathcal{J}_{uc}$ is compared to the optimal cost $\mathcal{J}_c$.

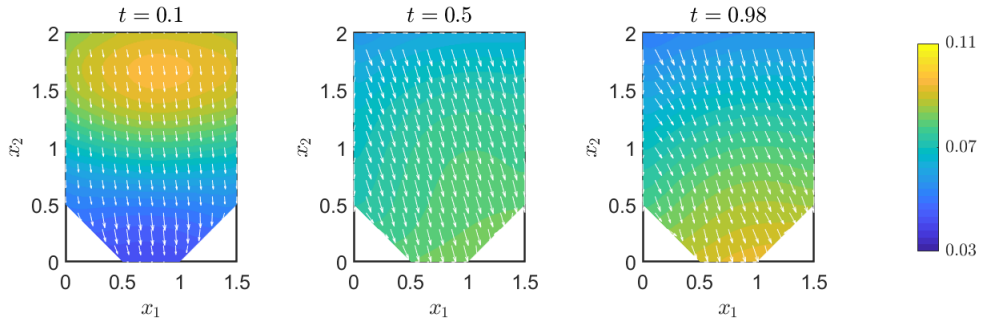

Figure 8.7: An optimal control example: Displayed is the optimal solution ρ for $\beta = 10^{-5}$ with superimposed optimal control $\vec{w}$ for a problem with volume exclusion effects ($\mathcal{S} \neq 0$).

A second example is computed on a MultiShape consisting of one quadrilateral and one wedge element. Here the time horizon is $(0, 2)$, while the choice of discretization points remains the same as in the previous example. The initial condition for ρ is

$$\rho(0, \vec{x}) = Z^{-1} \exp(-0.5(x_1 - 0.75)^2 - 0.5(x_2 - 3.3)^2),$$

where Z is a normalization constant, which results in an average density of $\bar{\rho} = 0.1$. The initial guesses for the optimization solver are given, as above, by $\rho(t, \vec{x}) = \rho(0, \vec{x})$ and $q(t, \vec{x}) = q(T, \vec{x}) = 0$. The particles in this problem do not experience any particle–particle interactions (i.e., $\kappa = 0$), and no external potential. However, the desired state $\widehat{\rho}$ is designed by computing the model problem with $\kappa = -1$ and $V_1 = 0.5x_2$. Therefore, we expect the control $\vec{w}$ to simulate attractive particle–particle, and gravitational forces. The result for $\beta = 10^{-3}$ is displayed in Figure 8.8. It is evident that the control acts by forcing the particles together and down the channel. In Table 8.3, the resulting values of the cost functional for different choices of β are compared to those of the uncontrolled problem.

	$\beta = 10^{-5}$	$\beta = 10^{-3}$	$\beta = 10^{-1}$
$\mathcal{J}_{uc}$	2.07×10^{-4}	2.07×10^{-4}	2.07×10^{-4}
$\mathcal{J}_c$	3.13×10^{-6}	8.99×10^{-5}	2.05×10^{-4}

Table 8.3: A second optimal control example with no-flux boundary conditions and volume exclusion ($\mathcal{S} \neq 0$). The value of the cost functional of the uncontrolled problem with $\vec{w} = \vec{0}$, denoted by $\mathcal{J}_{uc}$, is compared to the optimal cost $\mathcal{J}_c$.

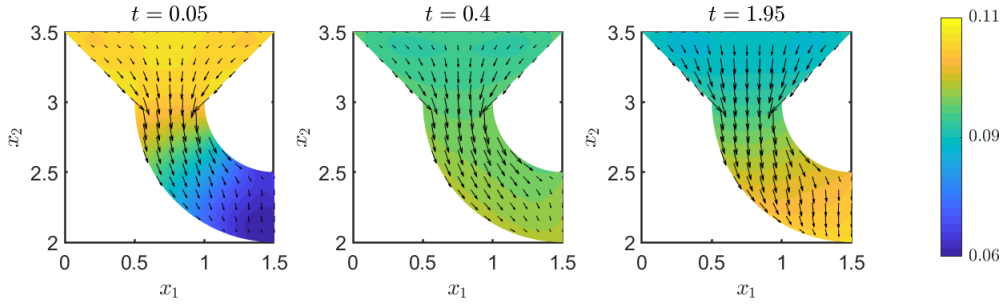

Figure 8.8: A second optimal control example: Displayed is the optimal solution ρ with superimposed optimal control $\vec{w}$ for a problem with volume exclusion effects ($\mathcal{S} \neq 0$).

8.4.4 Summary

This chapter illustrated that extensions of the DDFT model can be tackled with the method developed in this thesis. While this was successfully done, it is evident that this extended model is numerically more challenging to solve than the standard DDFT, due to the terms that prescribe the volume exclusion. In order to solve a model of this form, more care has to be taken in ensuring that ρ does not become too large in any part of the domain, at any time, and in particular that $0 < \rho < \frac{1}{a}$. This extended DDFT model furthermore raises new theoretical questions about existence, uniqueness, and regularity of solutions, due to the form of the term $\mathcal{S}$. Since $\mathcal{S}$ becomes unbounded as $\rho \to \frac{1}{a}$, such questions require careful examination in future work.

The solution of a corresponding optimal control problem posed additional challenges that had to be addressed. First, the derivation of the first-order optimality system was significantly more involved than that for the standard and multiple species DDFTs. The successful numerical solution of the optimal control problem involving this extended model depends on both the choice of ρ and $\widehat{\rho}$. While this is the case for any optimal control problem, for the present model the state ρ is required to satisfy $0 < \rho < \frac{1}{a}$. Moreover, the choice of $\widehat{\rho}$ should also satisfy these bounds, since otherwise the control may act to push ρ outside its prescribed bounds towards $\widehat{\rho}$. In order to ensure these conditions on ρ are met, one could extend the optimal control method to include state constraints in future work.

Despite these open questions, the implemented extension enables the modelling of volume exclusion effects that are important for capturing many real-world processes accurately. It furthermore demonstrates that the method introduced in this thesis is, in principle, applicable to more complicated extensions of DDFT, some of which were discussed in detail in Section 2.3.7.

Chapter 9

Conclusion and Outlook

In this chapter we summarize the results that were presented in this thesis, and highlight the contributions made to the research communities involved. Then, several avenues for further work are discussed, including extensions to the model problem, different optimal control set-ups, and real-world applications.

9.1 Summary

The aim of this thesis was to address the key challenge of optimizing real-world engineering and industrial processes. This required a careful combination of model development, derivation of PDE-constrained optimization frameworks, and sophisticated numerical implementations. The relevant integro-PDE models, originating from DDFT for colloidal fluids, describe the dynamics of a 'particle' density within a fluid bath, under the influence of diffusion, external forces, and nonlocal particle–particle interactions. Such interactions, which also propagate into nonlocal PDE boundary conditions, are crucial in applications, but provide significant challenges in both the derivation and numerical implementation of optimal control approaches. This thesis made key contributions to these areas. This includes the derivation of optimality conditions for models that incorporate essential physical effects such as multiple interacting species and volume exclusion effects. In a key part of the work, we have developed an associated numerical method, which is a novel combination of pseudospectral and spectral element schemes with a state-of-the-art Newton–Krylov optimization algorithm. The open-source implementation tackles problems in up to three dimensions, and on complex domains.

Chapters 2, 3, and 4 introduced the necessary knowledge in particle physics, PDEs, optimal control, and associated numerical methods, for the work presented in this thesis. Then, in Chapter 5, a method for solving PDE-constrained optimization problems for interacting particle systems was introduced, laying the foundation for the following chapters. This included the derivation of optimality conditions for optimal control problems with nonlinear, nonlocal PDE constraints, and the introduction of the numerical method used to solve the resulting nonlocally coupled system of equations. Given the key motivation of solving industrially-relevant problems, in Chapter 6 the numerical method was extended to include computations on complicated domains. This enabled the solution of DDFT models, and associated optimal control problems, on shapes that are relevant in applications, such as funnels, vessels, or pipes.

Once this framework for solving models and optimal control problems efficiently on complicated domains was established, extensions to the model could be introduced. Depending on the application of interest, different physical effects may become relevant. Therefore, Chapters 7 and 8 were concerned with different extensions to the DDFT model, addressing some such effects. First, in Chapter 7, a model for particle mixtures was introduced to capture processes that involve two or more types of particles. Then, in Chapter 8, the DDFT model was extended to describe volume exclusion effects. This is important when modelling hard-sphere particles that cannot overlap, which covers a wide range of real-world problems.

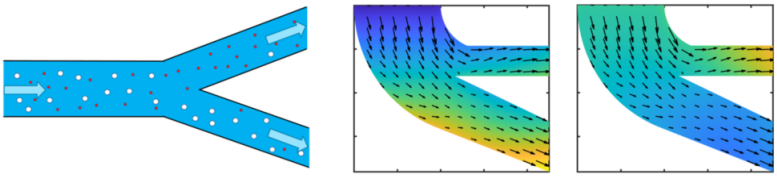

Figure 9.1: Separation of a particle mixture, compare to Figure 1.2 and Figure 7.3.

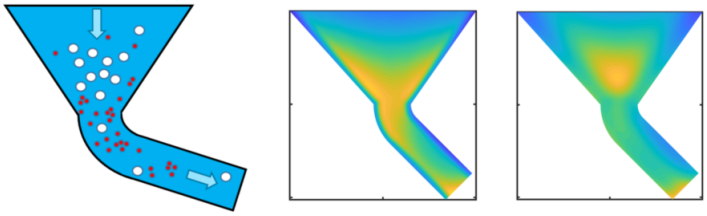

Figure 9.2: A particle mixture is filtered through a funnel, compare to Figure 1.3 and Figure 7.2.

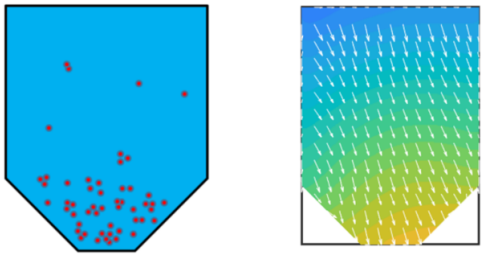

Figure 9.3: Sedimentation of colloids, compare to Figure 1.1 and Figure 8.7.

In order to summarize the versatility of the methods introduced in this thesis, we revisit the example problems introduced in Chapter 1, Figures 1.1, 1.2, and 1.3, and demonstrate how these have been addressed throughout this thesis. In Figure 9.1, the solution of an optimal control problem, solved in Chapter 7, in which a mixture of particles is advected through a complicated geometry and separates at the end of the channel, is shown. Figure 9.2 displays a DDFT solution from Chapter 7 (Figure 7.2), modelling how a mixture of particles passes through a funnel. Finally, in Figure 9.3, an example of sedimenting particles in a vessel is shown. An optimal control problem of this form was solved in Chapter 8 for volume excluding particles, see Figure 8.7. Many other interesting problems have been discussed in this thesis. However, there are further extensions to the framework developed in this thesis that can be addressed in future work.

9.2 Outlook

While the work presented in this thesis can be used to solve many interesting problems and capture different physical effects, there are countless avenues for further work. This includes extensions of the model, alternative optimal control problems, and areas of application. In the following, we discuss each of these aspects in turn, illustrating some interesting additional research questions.

9.2.1 DDFT Models

As discussed in Chapter 2, there are several extensions to the standard DDFT model. In this thesis, most work has been focused on the standard model, with some extensions introduced in the later chapters. However, there are several additional models that would be interesting to investigate. One rather straightforward extension would be to include reaction terms within the DDFT model, to describe chemical reactions, chemotaxis, or the spread of diseases. It would furthermore be of interest to model nonspherical or active particles, which would increase the dimension of the model, but not necessarily the complexity of its implementation. This would enable the description of particles with an orientation, such as plate-shaped blood cells, magnetic particles, or bacteria.

Another avenue of research would be the implementation of a DDFT model that describes inertial effects. Such a model has been introduced in Section 2.3.5. It consists of two equations, one for the particle density ρ and another for the particle velocity field $\vec{v}$, and can describe underdamped particle dynamics. While the implementations of this model in one dimension and in two dimensions on a rectangular domain are straightforward, there are challenges in the correct application of boundary conditions for more complicated domains, and in the application of matching conditions at intersection boundaries for MultiShape domains, that have to be overcome. Since underdamped particle dynamics are considered in this model, which can be hard to resolve numerically, a smoothing term may have to be introduced in the velocity equation to aid numerical computations, see [54]. Further challenges arise for associated optimal control problems, both in the derivation of a first-order optimality system and in its numerical solution. The optimality system consists of five equations instead of three, two of which are vector equations, making the optimality system more computationally expensive to solve. Moreover, this optimal control problem is sensitive to the choices of initial conditions for ρ and $\vec{v}$, as well as to choices of desired states $\widehat{\rho}$ and $\widehat{v}$. The reason for this is that the underdamped dynamics can favour strong advective fields prescribed by the control, which are difficult to resolve numerically. Nevertheless, implementing this DDFT on MultiShape domains and solving associated optimal control problems would be interesting to investigate, since many real-world processes are better captured by underdamped than overdamped dynamics.

Two further extensions that would be more challenging to implement are the inclusion of hydrodynamic interactions or hard sphere particles modelled by Fundamental Measure Theory (FMT). The key challenge with these models is the computation of additional convolution integrals that arise. These DDFTs could be implemented for the forward model on simple domains. However, FMT requires the computation of a convolution integral at each position $\vec{x}$ over a disc-shaped subdomain centred at $\vec{x}$. This requires accurate interpolation and becomes challenging to compute numerically for domains that are finite. At the boundary of a rectangular domain, for example, the required subdomains for the FMT convolutions are intersections between discs and parts of the boundary. These subdomains have to be defined for each $\vec{x}$ in the vicinity of the boundary, the number of which depends on the number of discretization points. Furthermore, weak formulations, and consequently optimality conditions for PDE-constrained optimization problems, are harder to derive for models involving hydrodynamic contributions or convolution terms arising from FMT. The additional terms from hydrodynamic interactions and from FMT involve more multiplicative factors of ρ, compared to the models discussed in this thesis. Introducing further factors of ρ in Chapter 8 already resulted in a complicated weak formulation. FMT and hydrodynamics would introduce such terms at higher order, making their computation even more involved.

Lastly, it would be interesting to consider models that include superadiabatic effects. This can be done by applying Power Functional Theory, which was introduced in Section 2.3.7. A PFT model could, in principle, be implemented using the methods described in this thesis. The complexity of such an implementation depends on the chosen approximation for the excess dissipation $\mathcal{P}_{\text{exc}}$. For example, (2.45) is similar in structure to the mean-field interaction term considered throughout this thesis and would therefore be a good choice for a first implementation. A challenge would be the time dependence of $\mathcal{P}_{\text{exc}}$. Numerically, this requires an update

of the convolution kernel at each time step, so an efficient time integration scheme would have to be incorporated into the numerical method. By contrast, in the numerical method presented in this thesis, the (time-independent) interaction kernel is computed only once during the whole algorithm. Similar complications would have to be overcome when adapting the optimal control solver.

9.2.2 Optimal Control Problems

Depending on the problem of interest, there are several changes that can be made to the optimal control problem. The form of the cost functional $\mathcal{J}$ considered in this thesis models *distributed control*, meaning that the control variable acts on the entire domain. There are also important *boundary control* and *subdomain control* problems, which involve control being applied only on the boundary or a subdomain of Ω. Both would be interesting alternatives for different applications. In many processes, one can only control, for example, the particle influx on the boundary of the domain as opposed to the advective field in the entire domain. This could be the case in the example illustrated in Figure 9.1, so that boundary control could be an interesting alternative to the distributed control approach taken in this thesis. Subdomain control could be used in cases in which only a part of the domain is accessible to the experimentalist. An example of this could be a setup in which a magnetic field can only be applied to the particles in a certain part of the domain. An additional control type is *time-independent control*, which is relevant in processes that can be optimally setup but not controlled over the whole time interval. This is the case for many industry processes which are set up once, during the design phase, and not modified after. Time-independent control is the time-averaged best control strategy and is often less 'optimal' when compared to a time-dependent strategy.

In this thesis, we have mostly considered flow control and no additional restrictions were placed on the advective control field $\vec{w}$. This meant that the control term was not part of the original DDFT formulation, since it could not necessarily be written in terms of the free energy $\mathcal{F}$. This becomes a problem when the activity of the fluid becomes too high, and therefore the particle correlations are far away from the equilibrium correlation, so that the adiabatic approximation breaks down. One way of circumventing such issues in the forward model is by prescribing that $\vec{w}$ is conservative, i.e., that it is the gradient of a potential $\vec{w} = \nabla V$. Another option would be to extend the optimal control setup to include models from Power Functional Theory, which are able to describe non-conservative flow fields, as discussed in Section 2.3.7. The challenges associated with adapting the current methods to PFT models were discussed in Section 9.2.1.

Another interesting avenue of research in this context would be the inclusion of state or control constraints. The choice of constraints depends largely on the application at hand. It could for example be impossible to set up certain control scenarios in practice, or, as discussed above, one may require the control to be a conservative force. We have already seen in Chapter 8 that one may need to enforce a condition on ρ, to ensure that the model is well defined. Moreover, since ρ is a density, one should also require that $\rho > 0$ in $(0, T) \times \Omega$. These would be examples of state constraints that could be incorporated in the optimal control setup.

Further possible adjustments to the optimal control problem relate to the target $\widehat{\rho}$. In this thesis, the target was prescribed over the whole domain. However, in many applications, one is mostly interested in the target at the final time $\widehat{\rho}(T, \vec{x})$. For example, if the aim is that particles should sediment to the bottom of the domain by time T, one could prescribe $\widehat{\rho}$ at this time only. Similar in spirit is the *subdomain observation*, in which the target is only prescribed in a certain part of the domain. For example, if the model is a nano-filtration device (see Figure 9.1), one may only be interested in the particle distribution of the right part of the domain, in which particles of different types should be separated. Moreover, instead of prescribing a target $\widehat{\rho}$, one could also use other quantities as targets. For example, we could aim to prescribe the particle flux j, or the average position $\vec{x}$.

There are also questions about which norm one should use to measure the value of $\mathcal{J}$. In this thesis, the L^2 norm is used, as it is standard in the optimal control community. However,

since ρ is a distribution, there are arguments in the literature that an appropriate norm to use for measuring the cost functional would be the Wasserstein distance [168, 171].

9.2.3 Boundary Conditions and Domains

It would be interesting to investigate models and corresponding optimal control problems with alternative choices of boundary conditions to the ones discussed in this thesis. One of the most interesting cases would be to model in- and outflow for pipes, so that the two ends of the pipe do not have no-flux boundary conditions, but density-dependent Dirichlet conditions. These would be able to describe how particles are added to and removed from a system, which is relevant to many real-world processes. This would especially be interesting in connection with the MultiShape package, so that complicated channels, funnels, or vessels with in- and outflow vents could be modelled. Figures 9.1 and 9.2 illustrate cases in which such an approach could be beneficial.

Another interesting extension would be to incorporate semi-infinite and periodic domains into the MultiShape package. If the MultiShape package would include semi-infinite boxes, one could model, for example, a river that traces a complicated path in the considered domain, but that is approximately straight outside of the given area of interest. Including periodic boundaries within the MultiShape implementation would allow for comparison with models that use periodic conditions in combination with complicated domains modelled using external potentials. Moreover, one could describe phenomena that do have a periodic component to them, such as pattern formation in a large domain, so that boundary conditions would not affect the overall model outcome significantly.

9.2.4 Applications

There are many potential applications that could be worked on using the framework introduced in this thesis. Many industrial processes can be described by interacting particle systems. One such example is modelling the sedimentation of yeast in beer during brewing. Yeast flocculates, i.e., forms clusters, which sediment under gravity. This can, in principle, be described by DDFT. In brewing, considerable time is lost in this process, which can take up to several weeks. Therefore, mathematical modelling can capture the process accurately, and optimization methods can describe how to improve it by changing the setup of the system. This can inform how to speed up this process and how resources such as time and money can be saved.

Another application is nano-filtration, in which a mixture of different particles is separated within a nano-filtration device. Such devices can be used in biomedical applications, when separating red and white blood cells, for example, as well as in environmental setups, in which algae and toxic materials can be filtered out of a fluid. Such processes can be described by DDFT for particle mixtures, and optimal control solutions can inform the optimal inflow rates of particle mixtures so that optimal separation occurs. Other applications that can be tackled using this framework include (optimal) drug delivery and cancer treatments, disease spread and prevention, opinion dynamics, and animal migration. In fact, the list of possible application areas is long. As illustrated throughout this thesis, many DDFT models can be developed, and optimal control problems with DDFT constraints can be derived and solved, using this framework or an extension thereof.

In any real-world application it is of paramount importance to ensure that the model solution matches experimental data. Efficient techniques for incorporating data into the model therefore form an important avenue of research for the problems introduced in this thesis. Parameters in the model, such as the diffusivity, temperature, and interaction strength have to be estimated from data, which opens an entirely different direction of research. Such investigations could utilize tools from optimal control for parameter estimation, Bayesian optimization, and machine learning. As illustrated, there are many exciting application areas of the work presented in this thesis, which could make important contributions in various industrial, medical, and social sectors.

Bibliography

[1] M. Aduamoah, B. D. Goddard, J. W. Pearson, and J. C. Roden. Pseudospectral methods and iterative solvers for optimization problems from multiscale particle dynamics. *BIT Numerical Mathematics*, p. 1703–1743, 2022.

[2] J. C. Roden, R. D. Mills-Williams, J. W. Pearson, and B. D. Goddard. MultiShape: A spectral element method, with applications to dynamic density functional theory and PDE-constrained optimization. *arXiv preprint arXiv:2207.05589*, 2022.

[3] H. Yin, *Mathematical modelling of droplets and bubbles on surfaces*. PhD thesis, University of Loughborough, DOI: 10.26174/thesis.lboro.12129231.v1, 2019.

[4] R. Evans, Density functional theory for inhomogeneous fluids I: Simple fluids in equilibrium [Lecture notes] 2009.

[5] M. te Vrugt, H. Löwen, and R. Wittkowski. Classical dynamical density functional theory: From fundamentals to applications. *Advances in Physics*, 69(2):121–247, 2020.

[6] P. Olla, *An Introduction to Thermodynamics and Statistical Physics*. Cham: Springer, 2015.

[7] F. Schwabl, *Statistical Mechanics*. Berlin: Springer, 2nd edition, 2006.

[8] R. Pathria and P. D. Beale, *Statistical Mechanics*. Amsterdam: Academic Press, 4th edition, 2021.

[9] K. Huang, *Statistical Mechanics*. New York: John Wiley & Sons, 2nd edition, 1987.

[10] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Physical Review*, 136(3B):B864, 1964.

[11] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Physical Review*, 140(4A):A1133, 1965.

[12] C. Ebner, W. Saam, and D. Stroud. Density-functional theory of simple classical fluids. I. Surfaces. *Physical Review A*, 14(6):2264, 1976.

[13] W. Saam and C. Ebner. Density-functional theory of classical systems. *Physical Review A*, 15(6):2566, 1977.

[14] R. Evans. The nature of the liquid-vapour interface and other topics in the statistical mechanics of non-uniform, classical fluids. *Advances in Physics*, 28(2):143–200, 1979.

[15] J.-P. Hansen, *Theory of Simple Liquids: With Applications to Soft Matter*. Amsterdam: Academic Press, 4th edition, 2013.

[16] H. Löwen. Melting, freezing and colloidal suspensions. *Physics Reports*, 237(5):249–324, 1994.

[17] N. D. Mermin. Thermal properties of the inhomogeneous electron gas. *Physical Review*, 137(5A):A1441, 1965.

[18] J. Chayes and L. Chayes. On the validity of the inverse conjecture in classical density functional theory. *Journal of Statistical Physics*, 36(3):471–488, 1984.

[19] U. Thiele, T. Frohoff-Hülsmann, S. Engelnkemper, E. Knobloch, and A. J. Archer. First order phase transitions and the thermodynamic limit. *New Journal of Physics*, 21(12):123021, 2019.

[20] R. Roth. Fundamental measure theory for hard-sphere mixtures: A review. *Journal of Physics: Condensed Matter*, 22(6):063102, 2010.

[21] M. Rex and H. Löwen. Dynamical density functional theory for colloidal dispersions including hydrodynamic interactions. *The European Physical Journal E, Soft Matter*, 28(2):139–146, 2009.

[22] S. M. Tschopp and J. M. Brader. Fundamental measure theory of inhomogeneous two-body correlation functions. *Physical Review E*, 103(4):042103, 2021.

[23] S. M. Tschopp, F. Sammüller, S. Hermann, M. Schmidt, and J. M. Brader. Force density functional theory in- and out-of-equilibrium. *Physical Review E*, 106(1):014115, 2022.

[24] S. M. Tschopp and J. M. Brader. First-principles superadiabatic theory for the dynamics of inhomogeneous fluids. *arXiv preprint arXiv:2209.11586*, 2022.

[25] J. K. Percus. Equilibrium state of a classical fluid of hard rods in an external field. *Journal of Statistical Physics*, 15(6):505–511, 1976.

[26] T. K. Vanderlick, H. T. Davis, and J. K. Percus. The statistical mechanics of inhomogeneous hard rod mixtures. *The Journal of Chemical Physics*, 91(11):7136–7145, 1989.

[27] Y. Rosenfeld. Free-energy model for the inhomogeneous hard-sphere fluid mixture and density-functional theory of freezing. *Physical Review Letters*, 63(9):980–983, 1989.

[28] Y. Rosenfeld. Free-energy model for the inhomogeneous hard-sphere fluid in D dimensions: Structure factors for the hard-disk ($D = 2$) mixtures in simple explicit form. *Physical Review A*, 42(10):5978–5989, 1990.

[29] H. Reiss, H. L. Frisch, and J. L. Lebowitz. Statistical mechanics of rigid spheres. *The Journal of Chemical Physics*, 31(2):369–380, 1959.

[30] H. Reiss, H. L. Frisch, E. Helfand, and J. L. Lebowitz. Aspects of the statistical thermodynamics of real fluids. *The Journal of Chemical Physics*, 32(1):119–124, 1960.

[31] E. Helfand, H. L. Frisch, and J. L. Lebowitz. Theory of the two- and one-dimensional rigid sphere fluids. *The Journal of Chemical Physics*, 34(3):1037–1042, 1961.

[32] J. K. Percus and G. J. Yevick. Analysis of classical statistical mechanics by means of collective coordinates. *Physical Review*, 110(1):1–13, 1958.

[33] Y. Rosenfeld. Scaled field particle theory of the structure and the thermodynamics of isotropic hard particle fluids. *The Journal of Chemical Physics*, 89(7):4272–4287, 1988.

[34] J. Winkelmann. The liquid-vapour interface of pure fluids and mixtures: Application of computer simulation and density functional theory. *Journal of Physics: Condensed Matter*, 13(21):4739–4768, 2001.

[35] A. L. Thorneywork, S. K. Schnyder, D. G. A. L. Aarts, J. Horbach, R. Roth, and R. P. A. Dullens. Structure factors in a two-dimensional binary colloidal hard sphere system. *Molecular Physics*, 116(21-22):3245–3257, 2018.

[36] R. Roth. Fluid of discs with competing interactions. *Molecular Physics*, 109(23-24):2897–2905, 2011.

[37] A. González, J. A. White, and R. Evans. Density functional theory for hard-sphere fluids: A generating function approach. *Journal of Physics: Condensed Matter*, 9(11):2375–2398, 1997.

[38] Y. Martínez-Ratón, J. A. Capitán, and J. A. Cuesta. Fundamental-measure density functional for mixtures of parallel hard cylinders. *Physical Review E*, 77(5):051205, 2008.

[39] M. Schmidt, H. Löwen, J. M. Brader, and R. Evans. Density functional theory for a model colloid polymer mixture: Bulk fluid phases. *Journal of Physics: Condensed Matter*, 14(40):9353–9382, 2002.

[40] H. Graf and H. Löwen. Phase diagram of tobacco mosaic virus solutions. *Physical Review E*, 59(2):1932–1942, 1999.

[41] S. DuBois and A. Perera. Entropy driven demixing in fluids of rigidly ordered particles. *The Journal of Chemical Physics*, 116(14):6354–6367, 2002.

[42] A. Chamoux and A. Perera. Direct correlation functions in two-dimensional anisotropic fluids. *Physical Review E*, 58(2):1933–1947, 1998.

[43] A. Chamoux and A. Perera. Approximations for the direct correlation function in multicomponent molecular fluids. *The Journal of Chemical Physics*, 104(4):1493–1505, 1996.

[44] A. J. Archer and A. Malijevský. On the interplay between sedimentation and phase separation phenomena in two-dimensional colloidal fluids. *Molecular Physics*, 109(7-10):1087–1099, 2011.

[45] A. J. Archer. Two-dimensional fluid with competing interactions exhibiting microphase separation: Theory for bulk and interfacial properties. *Physical Review E*, 78(3):031402, 2008.

[46] O. Zvyagolskaya, A. J. Archer, and C. Bechinger. Criticality and phase separation in a two-dimensional binary colloidal fluid induced by the solvent critical behavior. *EPL (Europhysics Letters)*, 96(2):28005, 2011.

[47] A. Malijevský and A. J. Archer. Sedimentation of a two-dimensional colloidal mixture exhibiting liquid-liquid and gas-liquid phase separation: A dynamical density functional theory study. *The Journal of Chemical Physics*, 139(14):144901, 2013.

[48] T. Ramakrishnan and M. Yussouff. First-principles order-parameter theory of freezing. *Physical Review B*, 19(5):2775, 1979.

[49] A. J. Archer and R. Evans. Dynamical density functional theory and its application to spinodal decomposition. *The Journal of Chemical Physics*, 121(9):4246–4254, 2004.

[50] T. Munakata. Time-dependent density-functional theory with H theorems. *Physical Review E*, 50(3):2347–2350, 1994.

[51] U. M. B. Marconi and P. Tarazona. Dynamic density functional theory of fluids. *The Journal of Chemical Physics*, 110(16):8032–8044, 1999.

[52] G. K.-L. Chan and R. Finken. Time-dependent density functional theory of classical fluids. *Physical Review Letters*, 94(18):183001, 2005.

[53] D. de las Heras, J. M. Brader, A. Fortini, and M. Schmidt. Particle conservation in dynamical density functional theory. *Journal of Physics: Condensed Matter*, 28(24):244024, 2016.

[54] A. J. Archer. Dynamical density functional theory for molecular and colloidal fluids: A microscopic approach to fluid mechanics. *The Journal of Chemical Physics*, 130(1):014509, 2009.

[55] A. J. Archer. Dynamical density functional theory: Binary phase-separating colloidal fluid in a cavity. *Journal of Physics: Condensed Matter*, 17(10):1405, 2005.

[56] G. A. Pavliotis, *Stochastic Processes and Applications: Diffusion Processes, the Fokker–Planck and Langevin Equations*. New York: Springer, 2014.

[57] J. F. Lutsko. A dynamical theory of nucleation for colloids and macromolecules. *The Journal of Chemical Physics*, 136(3):034509, 2012.

[58] B. D. Goddard, G. A. Pavliotis, and S. Kalliadasis. The overdamped limit of dynamic density functional theory: Rigorous results. *Multiscale Modeling & Simulation*, 10(2):633–663, 2012.

[59] A. Nold, *From the Nano- to the Macroscale – Bridging Scales for the Moving Contact Line Problem*. PhD thesis, Imperial College London, DOI: doi.org/10.25560/34933, 2016.

[60] J. Rotne and S. Prager. Variational treatment of hydrodynamic interaction in polymers. *The Journal of Chemical Physics*, 50(11):4831–4837, 1969.

[61] M. Stimson and G. B. Jeffery. The motion of two spheres in a viscous fluid. *Proceedings of the Royal Society of London. Series A*, 111(757):110–116, 1926.

[62] B. Goddard, A. Nold, N. Savva, P. Yatsyshin, and S. Kalliadasis. Unification of dynamic density functional theory for colloidal fluids to include inertia and hydrodynamic interactions: Derivation and numerical experiments. *Journal of Physics: Condensed Matter*, 25(3):035101, 2012.

[63] C. P. Royall, J. Dzubiella, M. Schmidt, and A. van Blaaderen. Nonequilibrium sedimentation of colloids on the particle scale. *Physical Review Letters*, 98(18):188304, 2007.

[64] M. Rex and H. Löwen. Dynamical density functional theory with hydrodynamic interactions and colloids in unstable traps. *Physical Review Letters*, 101(14):148302, 2008.

[65] A. M. Menzel, A. Saha, C. Hoell, and H. Löwen. Dynamical density functional theory for microswimmers. *The Journal of Chemical Physics*, 144(2):024115, 2016.

[66] M. Rauscher, A. Domínguez, M. Krüger, and F. Penna. A dynamic density functional theory for particles in a flowing solvent. *The Journal of Chemical Physics*, 127(24):244906, 2007.

[67] J. M. Brader and M. Krüger. Density profiles of a colloidal liquid at a wall under shear flow. *Molecular Physics*, 109(7-10):1029–1041, 2011.

[68] S. Mondal, S. Mukherjee, and B. Bagchi. Origin of diverse time scales in the protein hydration layer solvation dynamics: A simulation study. *The Journal of Chemical Physics*, 147(15):154901, 2017.

[69] M. Bier and R. van Roij. Relaxation dynamics in fluids of platelike colloidal particles. *Physical Review E*, 76(2):021405, 2007.

[70] M. Bier, R. van Roij, M. Dijkstra, and P. van der Schoot. Self-diffusion of particles in complex fluids: Temporary cages and permanent barriers. *Physical Review Letters*, 101(21):215901, 2008.

[71] E. Grelet, M. P. Lettinga, M. Bier, R. Van Roij, and P. Van der Schoot. Dynamical and structural insights into the smectic phase of rod-like particles. *Journal of Physics: Condensed Matter*, 20(49):494213, 2008.

[72] R. Wittkowski and H. Löwen. Dynamical density functional theory for colloidal particles with arbitrary shape. *Molecular Physics*, 109(23-24):2935–2943, 2011.

[73] M. A. Durán-Olivencia, B. D. Goddard, and S. Kalliadasis. Dynamical density functional theory for orientable colloids including inertia and hydrodynamic interactions. *Journal of Statistical Physics*, 164(4):785–809, 2016.

[74] B. D. Goddard, T. Hurst, and R. Ocone. Modelling inelastic granular media using dynamical density functional theory. *Journal of Statistical Physics*, 183(1):1–22, 2021.

[75] M. Schmidt. Statics and dynamics of inhomogeneous liquids via the internal-energy functional. *Physical Review E*, 84(5):051203, 2011.

[76] K. Fuchizaki and K. Kawasaki. Dynamical density functional theory for glassy behaviour. *Journal of Physics: Condensed Matter*, 14(46):12203, 2002.

[77] D. S. Dean. Langevin equation for the density of a system of interacting Langevin processes. *Journal of Physics A: Mathematical and General*, 29(24):L613, 1996.

[78] M. Schmidt. Power functional theory for many-body dynamics. *Reviews of Modern Physics*, 94(1):015007, 2022.

[79] M. Schmidt and J. M. Brader. Power functional theory for Brownian dynamics. *The Journal of Chemical Physics*, 138(21):214101, 2013.

[80] D. de Las Heras and M. Schmidt. Velocity gradient power functional for Brownian dynamics. *Physical Review Letters*, 120(2):028001, 2018.

[81] F. Sammüller, D. d. l. Heras, and M. Schmidt. Inhomogeneous steady shear dynamics of a three-body colloidal gel former. *arXiv preprint arXiv:2210.07679*, 2022.

[82] N. C. Stuhlmüller, T. Eckert, D. de Las Heras, and M. Schmidt. Structural nonequilibrium forces in driven colloidal systems. *Physical Review Letters*, 121(9):098002, 2018.

[83] D. de las Heras and M. Schmidt. Flow and structure in nonequilibrium Brownian many-body systems. *Physical Review Letters*, 125(1):018001, 2020.

[84] A. Nold, B. D. Goddard, P. Yatsyshin, N. Savva, and S. Kalliadasis. Pseudospectral methods for density functional theory in bounded and unbounded domains. *Journal of Computational Physics*, 334:639–664, 2017.

[85] L. C. Evans, *Partial Differential Equations*. Providence, Rhode Island: American Mathematical Society, 2nd edition, 2010.

[86] F. Tröltzsch, *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*. Providence: American Mathematical Society, 2010.

[87] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE Constraints*. Dordrecht: Springer Netherlands, 2009.

[88] C. Lanczos. Trigonometric interpolation of empirical and analytical functions. *Journal of Mathematics and Physics*, 17(1-4):123–199, 1938.

[89] S. A. Orszag. Comparison of pseudospectral and spectral approximation. *Studies in Applied Mathematics*, 51(3):253–259, 1972.

[90] H.-O. Kreiss and J. Oliger. Comparison of accurate methods for the integration of hyperbolic equations. *Tellus*, 24(3):199–215, 1972.

[91] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*. Philadelphia: SIAM, 1977.

[92] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods: Fundamentals in Single Domains*. Berlin: Springer, 2006.

[93] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*. Mineola, New York: Dover Publications, Inc., 2nd edition, 2000.

[94] L. N. Trefethen, *Spectral Methods in MATLAB*. Philadelphia: SIAM, 2000.

[95] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*. Berlin: Springer, 1988.

[96] R. Kosloff. Propagation methods for quantum molecular dynamics. *Annual Review of Physical Chemistry*, 45(1):145–178, 1994.

[97] L. Philipp and B. D. Shizgal. A pseudospectral solution of a bistable Fokker–Planck equation that models protein folding. *Physica A: Statistical Mechanics and its Applications*, 522:158–166, 2019.

[98] S. Jalali and M. Heshmati. Vibration analysis of tapered circular poroelastic plates with radially graded porosity using pseudo-spectral method. *Mechanics of Materials*, 140:103240, 2020.

[99] F. Soleymani. Pricing multi-asset option problems: A Chebyshev pseudo-spectral method. *BIT Numerical Mathematics*, 59(1):243–270, 2019.

[100] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004.

[101] P. D. Lax and R. D. Richtmyer. Survey of the stability of linear finite difference equations. *Communications on Pure and Applied Mathematics*, 9(2):267–293, 1956.

[102] J. Sanz-Serna and C. Palencia. A general equivalence theorem in the theory of discretization methods. *Mathematics of Computation*, 45(171):143–152, 1985.

[103] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*. New York: Springer, 3rd edition, 2008.

[104] S. Mazumder, *Numerical Methods for Partial Differential Equations: Finite Difference and Finite Volume Methods*. London: Academic Press, 2016.

[105] L. J. D. Frink and A. G. Salinger. Two- and three-dimensional nonlocal density functional theory for inhomogeneous fluids: I. Algorithms and parallelization. *Journal of Computational Physics*, 159(2):407–424, 2000.

[106] L. D. Frink, A. Salinger, M. Sears, J. Weinhold, and A. Frischknecht. Numerical challenges in the application of density functional theory to biology and nanotechnology. *Journal of Physics: Condensed Matter*, 14(46):12167, 2002.

[107] D. Zhou, J. Mi, and C. Zhong. Three-dimensional density functional study of heterogeneous nucleation of droplets on solid surfaces. *The Journal of Physical Chemistry B*, 116(48):14100–14106, 2012.

[108] M. G. Knepley, D. A. Karpeev, S. Davidovits, R. S. Eisenberg, and D. Gillespie. An efficient algorithm for classical density functional theory in three dimensions: Ionic solutions. *The Journal of Chemical Physics*, 132(12):124101, 2010.

[109] M. P. Sears and L. J. Frink. A new efficient method for density functional theory calculations of inhomogeneous fluids. *Journal of Computational Physics*, 190(1):184–200, 2003.

[110] J. De La Torre, P. Español, and A. Donev. Finite element discretization of nonlinear diffusion equations with thermal fluctuations. *The Journal of Chemical Physics*, 142(9):094115, 2015.

[111] C. Chalmers, R. Smith, and A. J. Archer. Dynamical density functional theory for the evaporation of droplets of nanoparticle suspension. *Langmuir*, 33(50):14490–14501, 2017.

[112] J. A. Carrillo, M. J. Castro, S. Kalliadasis, and S. P. Perez. High-order well-balanced finite-volume schemes for hydrodynamic equations with nonlocal free energy. *SIAM Journal on Scientific Computing*, 43(2):A828–A858, 2021.

[113] J. A. Carrillo, A. Chertock, and Y. Huang. A finite-volume method for nonlinear nonlocal equations with a gradient flow structure. *Communications in Computational Physics*, 17(1):233–258, 2015.

[114] J. A. Carrillo, S. Kalliadasis, S. P. Perez, and C.-W. Shu. Well-balanced finite-volume schemes for hydrodynamic equations with general free energy. *Multiscale Modeling & Simulation*, 18(1):502–541, 2020.

[115] J. Mendes, A. Russo, S. P. Perez, and S. Kalliadasis. A finite-volume scheme for gradient-flow equations with non-homogeneous diffusion. *Computers & Mathematics with Applications*, 89:150–162, 2021.

[116] A. Russo, M. A. Durán-Olivencia, P. Yatsyshin, and S. Kalliadasis. Memory effects in fluctuating dynamic density-functional theory: Theory and simulations. *Journal of Physics A: Mathematical and Theoretical*, 53(44):445007, 2020.

[117] A. Russo, S. P. Perez, M. A. Durán-Olivencia, P. Yatsyshin, J. A. Carrillo, and S. Kalliadasis. A finite-volume method for fluctuating dynamical density functional theory. *Journal of Computational Physics*, 428:109796, 2021.

[118] B. D. Goddard, A. Nold, and S. Kalliadasis, "2DChebClass [Software]." http://dx.doi.org/10.7488/ds/1991, 2017.

[119] C. W. Clenshaw and A. R. Curtis. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2(1):197–205, 1960.

[120] L. F. Shampine and M. W. Reichelt. The MATLAB ODE suite. *SIAM journal on scientific computing*, 18(1):1–22, 1997.

[121] L. F. Shampine, M. W. Reichelt, and J. A. Kierzenka. Solving index-1 DAEs in MATLAB and Simulink. *SIAM Review*, 41(3):538–552, 1999.

[122] P. C. Hansen, *Discrete Inverse Problems*. Philadelphia: SIAM, 2010.

[123] H. Antil, D. P. Kouri, M.-D. Lacasse, and D. Ridzal, *Frontiers in PDE-Constrained Optimization*. New York: Springer, 1st edition, 2018.

[124] J. C. de los Reyes, *Numerical PDE-Constrained Optimization*. Cham: Springer, 2015.

[125] E. A. Gonzalez-Velasco. The product rule for Fréchet derivatives. *International Journal of Mathematical Education in Science and Technology*, 17(1):79–83, 1986.

[126] S. Lenhart. Optimal control of a convective-diffusive fluid problem. *Mathematical Models and Methods in Applied Sciences*, 5(2):225–237, 1995.

[127] H. R. Joshi. Optimal control of the convective velocity coefficient in a parabolic problem. *Nonlinear Analysis: Theory, Methods & Applications*, 63(5-7):e1383–e1390, 2005.

[128] R. Glowinski, Y. Song, X. Yuan, and H. Yue. Bilinear optimal control of an advection-reaction-diffusion system. *SIAM Review*, 64(2):392–421, 2022.

[129] A. Fleig and R. Guglielmi. Bilinear optimal control of the Fokker–Planck equation. *IFAC-PapersOnLine*, 49(8):254–259, 2016.

[130] A. Fleig and R. Guglielmi. Optimal control of the Fokker–Planck equation with space-dependent controls. *Journal of Optimization Theory and Applications*, 174(2):408–427, 2017.

[131] R. Herzog and K. Kunisch. Algorithms for PDE-constrained optimization. *GAMM-Mitteilungen*, 33(2):163–176, 2010.

[132] A. Manzoni, A. Quarteroni, and S. Salsa, *Optimal Control of Partial Differential Equations: Analysis, Approximation, and Applications*. Cham: Springer, 2021.

[133] S. S. Collis and M. Heinkenschloss, Analysis of the streamline upwind/Petrov Galerkin method applied to the solution of optimal control problems, Tech. Rep. 02-01, Rice University, Houston, 2002.

[134] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer, 2nd edition, 2006.

[135] K. Chen, *Matrix Preconditioning Techniques and Applications*. Cambridge: Cambridge University Press, 2005.

[136] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[137] A. J. Wathen. Preconditioning. *Acta Numerica*, 24:329–376, 2015.

[138] J. W. Pearson and J. Pestana. Preconditioners for Krylov subspace methods: An overview. *GAMM-Mitteilungen*, 43(4):e202000015, 2020.

[139] R. Herzog, "Algorithms and preconditioning in PDE-constrained optimization [Lecture notes]."

[140] T. Rusten and R. Winther. A preconditioned iterative method for saddlepoint problems. *SIAM Journal on Matrix Analysis and Applications*, 13(3):887–18, 1992.

[141] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409, 1952.

[142] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[143] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[144] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics*. Oxford: Oxford University Press, 2005.

[145] A. Greenbaum, *Iterative Methods for Solving Linear Systems*. Philadelphia: SIAM, 1997.

[146] J. W. Pearson, *Fast Iterative Solvers for PDE-Constrained Optimization Problems*. PhD thesis, University of Oxford, 2013.

[147] M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[148] Y. A. Kuznetsov. Efficient iterative solvers for elliptic finite element problems on non-matching grids. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 10(3):187–211, 1995.

[149] M. F. Murphy, G. H. Golub, and A. J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972, 2000.

[150] D. Silvester and A. Wathen. Fast iterative solution of stabilised Stokes systems part II: Using general block preconditioners. *SIAM Journal on Numerical Analysis*, 31(5):1352–1367, 1994.

[151] D. Kay, D. Loghin, and A. Wathen. A preconditioner for the steady-state Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 24(1):237–256, 2002.

[152] J. W. Pearson and A. J. Wathen. A new approximation of the Schur complement in preconditioners for PDE-constrained optimization. *Numerical Linear Algebra with Applications*, 19(5):816–829, 2012.

[153] J. W. Pearson. Preconditioned iterative methods for Navier–Stokes control problems. *Journal of Computational Physics*, 292:194–207, 2015.

[154] J. Liu and J. W. Pearson. Parameter-robust preconditioning for the optimal control of the wave equation. *Numerical Algorithms*, 83(3):1171–1203, 2020.

[155] W. Zulehner. Nonstandard norms and robust estimates for saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 32(2):536–560, 2011.

[156] K.-A. Mardal, B. F. Nielsen, and M. Nordaas. Robust preconditioners for PDE-constrained optimization with limited observations. *BIT Numerical Mathematics*, 57(2):405–431, 2017.

[157] I. C. Ipsen. A note on preconditioning nonsymmetric matrices. *SIAM Journal on Scientific Computing*, 23(3):1050–1051, 2001.

[158] F. Cucker and S. Smale. Emergent behavior in flocks. *IEEE Transactions on Automatic Control*, 52(2):852–862, 2007.

[159] F. Cucker and S. Smale. On the mathematics of emergence. *Japanese Journal of Mathematics*, 2(1):197–227, 2007.

[160] M. Fornasier and F. Solombrino. Mean-field optimal control. *ESAIM: Control, Optimisation and Calculus of Variations*, 20(4):1123–1152, 2014.

[161] M. Fornasier, B. Piccoli, and F. Rossi. Mean-field sparse optimal control. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2028):20130400, 2014.

[162] M. Fornasier, S. Lisini, C. Orrieri, and G. Savaré. Mean-field optimal control as Gamma-limit of finite agent controls. *European Journal of Applied Mathematics*, 30(6):1153–1186, 2019.

[163] B. Piccoli, F. Rossi, and E. Trélat. Control to flocking of the kinetic Cucker–Smale model. *SIAM Journal on Mathematical Analysis*, 47(6):4685–4719, 2014.

[164] M. Fornasier, Learning and sparse control of multiagent systems, in *European Congress of Mathematics*, volume 7, 2016.

[165] M. Burger, R. Pinnau, C. Totzeck, and O. Tse. Mean-field optimal control and optimality conditions in the space of probability measures. *SIAM Journal on Control and Optimization*, 59(2):977–1006, 2021.

[166] G. Albi, Y.-P. Choi, M. Fornasier, and D. Kalise. Mean field control hierarchy. *Applied Mathematics and Optimization*, 76(1):93–135, 2016.

[167] J. A. Carrillo, E. A. Pimentel, and V. K. Voskanyan. On a mean field optimal control problem. *Nonlinear Analysis*, 199:112039, 2020.

[168] R. Pinnau, C. Totzeck, O. Tse, and S. Martin. A consensus-based model for global optimization and its mean-field limit. *Mathematical Models and Methods in Applied Sciences*, 27(1):183–204, 2017.

[169] J. Carrillo, Y.-P. Choi, C. Totzeck, and O. Tse. An analytical framework for consensus-based global optimization method. *Mathematical Models and Methods in Applied Sciences*, 28(6):1037–1066, 2018.

[170] J. A. Carrillo, R. S. Gvalani, G. A. Pavliotis, and A. Schlichting. Long-time behaviour and phase transitions for the McKean–Vlasov equation on the torus. *Archive for Rational Mechanics and Analysis*, 235(1):635–690, 2020.

[171] M. Burger, R. Pinnau, C. Totzeck, O. Tse, and A. Roth. Instantaneous control of interacting particle systems in the mean-field limit. *Journal of Computational Physics*, 405:109181, 2020.

[172] M. Burger, R. Pinnau, A. Roth, C. Totzeck, and O. Tse. Controlling a self-organizing system of individuals guided by a few external agents–particle description and mean-field limit. *arXiv preprint arXiv:1610.01325*, 2016.

[173] C. Cheng and G. Knorr. The integration of the Vlasov equation in configuration space. *Journal of Computational Physics*, 22(3):330–351, 1976.

[174] G. Albi and L. Pareschi. Selective model-predictive control for flocking systems. *Communications in Applied and Industrial Mathematics*, 9(2):4–21, 2018.

[175] G. Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968.

[176] G. Albi, L. Pareschi, and M. Zanella. Boltzmann-type control of opinion consensus through leaders. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2028):20140138, 2014.

[177] G. Albi, M. Herty, and L. Pareschi. Kinetic description of optimal control problems and applications to opinion consensus. *Communications in Mathematical Sciences*, 13(6):1407–1429, 2014.

[178] M. Burger, M. Di Francesco, P. A. Markowich, and M.-T. Wolfram. Mean field games with nonlinear mobilities in pedestrian dynamics. *Discrete and Continuous Dynamical Systems - B*, 19(5):1311–1333, 2014.

[179] S. Güttel and J. W. Pearson. A spectral-in-time Newton–Krylov method for nonlinear PDE-constrained optimization. *IMA Journal of Numerical Analysis*, 42(2):1478–1499, 2022.

[180] M. A. Woodbury. Inverting modified matrices. Memorandum Report, 42. *Statistical Research Group, Princeton University*, 1950.

[181] W. E. Roth. On direct product matrices. *Bulletin of the American Mathematical Society*, 40(6):461–468, 1934.

[182] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*. Philadelphia: SIAM, 2003.

[183] M. Aduamoah, B. D. Goddard, J. W. Pearson, and J. C. Roden, "2DChebClassPDECO [Software]." https://bitbucket.org/bdgoddard/2dchebclasspdecopublic/, 2020.

[184] S. Güttel and J. W. Pearson, "PDEOptim [Software]." https://github.com/nla-group/pdeoptim/, 2020.

[185] S. Güttel and J. W. Pearson. A rational deferred correction approach to parabolic optimal control problems. *IMA Journal of Numerical Analysis*, 38(4):1861–1892, 2018.

[186] N. A. Peppas and B. Narasimhan. Mathematical models in drug delivery: How modeling has shaped the way we design new drug delivery systems. *Journal of Controlled Release*, 190:75–81, 2014.

[187] Y. Jaluria, *Advanced Materials Processing and Manufacturing*. Cham: Springer, 2018.

[188] F. Erdogdu, F. Sarghini, and F. Marra. Mathematical modeling for virtualization in food processing. *Food Engineering Reviews*, 9(4):295–313, 2017.

[189] H. Bridle, B. Miller, and M. P. Desmulliez. Application of microfluidics in waterborne pathogen monitoring: A review. *Water Research*, 55:256–271, 2014.

[190] K. D. Seibert, P. C. Collins, C. V. Luciani, and E. S. Fisher, Milling operations in the pharmaceutical industry, in *Chemical Engineering in the Pharmaceutical Industry: Active Pharmaceutical Ingredients* (D. J. am Ende and M. T. am Ende, eds.), pp. 861–879, Hoboken, NJ: Wiley Online Library, 2019.

[191] S. S. Rao, *The Finite Element Method in Engineering*. Oxford: Butterworth-Heinemann, 5th edition, 2010.

[192] C. Canuto, A. Quarteroni, M. Y. Hussaini, and T. A. Zang, *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics*. Berlin: Springer, 2007.

[193] A. Lozinski, C. Chauvière, J. Fang, and R. G. Owens. Fokker–Planck simulations of fast flows of melts and concentrated polymer solutions in complex geometries. *Journal of Rheology*, 47(2):535–561, 2003.

[194] A. Lozinski and C. Chauviere. A fast solver for Fokker–Planck equation applied to viscoelastic flows calculations: 2D FENE model. *Journal of Computational Physics*, 189(2):607–625, 2003.

[195] F. Fakhar-Izadi and M. Dehghan. Space–time spectral method for a weakly singular parabolic partial integro-differential equation on irregular domains. *Computers & Mathematics with Applications*, 67(10):1884–1904, 2014.

[196] W. Zhu and D. A. Kopriva. A spectral element method to price European options. I. Single asset with and without jump diffusion. *Journal of Scientific Computing*, 39(2):222–243, 2009.

[197] M. S. Alnaes, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. The FEniCS project Version 1.5. *Archive of Numerical Software*, 3(100), 2015.

[198] A. Logg, K.-A. Mardal, and G. N. Wells, eds., *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*. Berlin: Springer, 2012.

[199] M. Smith, *ABAQUS/Standard User's Manual, Version 6.9*. United States: Dassault Systèmes Simulia Corp, 2009.

[200] S. A. Orszag. Spectral methods for problems in complex geometries. *Journal of Computational Physics*, 37(1):70–92, 1980.

[201] A. T. Patera. A spectral element method for fluid dynamics: Laminar flow in a channel expansion. *Journal of Computational Physics*, 54(3):468–488, 1984.

[202] D. Komatitsch and J.-P. Vilotte. The spectral element method: An efficient tool to simulate the seismic response of 2D and 3D geological structures. *Bulletin of the Seismological Society of America*, 88(2):368–392, 1998.

[203] D. Funaro. A multidomain spectral approximation of elliptic equations. *Numerical Methods for Partial Differential Equations*, 2(3):187–205, 1986.

[204] Y. Morchoisne. Inhomogeneous flow calculations by spectral methods: Mono-domain and multi-domain techniques. *SIAM-CBMS*, 181, 1984.

[205] A. Fichtner, H. Igel, H.-P. Bunge, and B. L. Kennett. Simulation and inversion of seismic wave propagation on continental scales based on a spectral-element method. *Journal of Numerical Analysis, Industrial and Applied Mathematics*, 4(1-2):11–22, 2009.

[206] A. Lischke, M. Zayernouri, and Z. Zhang, Spectral and spectral element methods for fractional advection–diffusion–reaction equations, in *Handbook of Fractional Calculus with Applications. Volume 3, Numerical Methods* (G. E. Karniadakis, ed.), pp. 157–184, Berlin: De Gruyter, 2019.

[207] H. P. Pfeiffer, L. E. Kidder, M. A. Scheel, and S. A. Teukolsky. A multidomain spectral method for solving elliptic equations. *Computer Physics Communications*, 152(3):253–273, 2003.

[208] P. Grandclément and J. Novak. Spectral methods for numerical relativity. *Living Reviews in Relativity*, 12(1):1–103, 2009.

[209] W. Zhu and D. A. Kopriva. A spectral element approximation to price European options with one asset and stochastic volatility. *Journal of Scientific Computing*, 42(3):426–446, 2010.

[210] W. Zhu and D. A. Kopriva. A spectral element approximation to price European options. II. The Black-Scholes model with two underlying assets. *Journal of Scientific Computing*, 39(3):323–339, 2009.

[211] Y. Chen and F. Huang. Spectral method approximation of flow optimal control problems with H^1-norm state constraint. *Numerical Mathematics: Theory, Methods and Applications*, 10(3):614–638, 2017.

[212] J. Zhou and D. Yang. Spectral mixed Galerkin method for state constrained optimal control problem governed by the first bi-harmonic equation. *International Journal of Computer Mathematics*, 88(14):2988–3011, 2011.

[213] Y. Chen, N. Yi, and W. Liu. A Legendre–Galerkin spectral method for optimal control problems governed by elliptic equations. *SIAM Journal on Numerical Analysis*, 46(5):2254–2275, 2008.

[214] A. Taneja and J. Higdon. A fully-coupled discontinuous Galerkin spectral element method for two-phase flow in petroleum reservoirs. *Journal of Computational Physics*, 352:341–372, 2018.

[215] W. H. Lin. A least-squares spectral element method for sound propagation in acoustic ducts. *The Journal of the Acoustical Society of America*, 104(5):3111–3114, 1998.

[216] F. Fakhar-Izadi and M. Dehghan. A spectral element method using the modal basis and its application in solving second-order nonlinear partial differential equations. *Mathematical Methods in the Applied Sciences*, 38(3):478–504, 2015.

[217] J. C. Roden, R. D. Mills-Williams, J. W. Pearson, and B. D. Goddard, "MultiShape [Software]." https://bitbucket.org/bdgoddard/multishapepublic/src/master/, 2022.

[218] U. Zimmermann, F. Smallenburg, and H. Löwen. Flow of colloidal solids and fluids through constrictions: Dynamical density functional theory versus simulation. *Journal of Physics: Condensed Matter*, 28(24):244019, 2016.

[219] L. Almenar and M. Rauscher. Dynamics of colloids in confined geometries. *Journal of Physics: Condensed Matter*, 23(18):184115, 2011.

[220] M. D. Griffin, E. Jones, and J. D. Anderson Jr. A computational fluid dynamic technique valid at the centerline for non-axisymmetric problems in cylindrical coordinates. *Journal of Computational Physics*, 30(3):352–360, 1979.

[221] G. S. Constantinescu and S. K. Lele. A highly accurate technique for the treatment of flow equations at the polar axis in cylindrical coordinates using series expansions. *Journal of Computational Physics*, 183(1):165–186, 2002.

[222] E. Tavakoli, B. Lessani, and R. Hosseini. High-order pole-treatment in cylindrical coordinates for incompressible flow simulations with finite-difference collocated schemes. *Journal of Computational Physics*, 296:1–24, 2015.

[223] T. W. Tee and L. N. Trefethen. A rational spectral collocation method with adaptively transformed Chebyshev grid points. *SIAM Journal on Scientific Computing*, 28(5):1798–1811, 2006.

[224] H. Jafari-Varzaneh and S. M. Hosseini. A new map for the Chebyshev pseudospectral solution of differential equations with large gradients. *Numerical Algorithms*, 69(1):95–108, 2015.

[225] T. Munakata. A dynamical extension of the density functional theory. *Journal of the Physical Society of Japan*, 58(7):2434–2438, 1989.

[226] R. Wittkowski, H. Löwen, and H. R. Brand. Extended dynamical density functional theory for colloidal mixtures with temperature gradients. *The Journal of Chemical Physics*, 137(22):224904, 2012.

[227] B. Goddard, A. Nold, and S. Kalliadasis. Multi-species dynamical density functional theory. *The Journal of Chemical Physics*, 138(14):144904, 2013.

[228] A. Chauviere, H. Hatzikirou, I. G. Kevrekidis, J. S. Lowengrub, and V. Cristini. Dynamic density functional theory of solid tumor growth: Preliminary models. *AIP Advances*, 2(1):011210, 2012.

[229] H. M. Al-Saedi, A. J. Archer, and J. Ward. Dynamical density-functional-theory-based modeling of tissue dynamics: Application to tumor growth. *Physical Review E*, 98(2):022407, 2018.

[230] J. F. Lutsko. Mechanism for the stabilization of protein clusters above the solubility curve: The role of non-ideal chemical reactions. *Journal of Physics: Condensed Matter*, 28(24):244020, 2016.

[231] J. F. Lutsko and G. Nicolis. Mechanism for the stabilization of protein clusters above the solubility curve. *Soft Matter*, 12(1):93–98, 2016.

[232] R. Roa, T. Siegl, W. K. Kim, and J. Dzubiella. Product interactions and feedback in diffusion-controlled reactions. *The Journal of Chemical Physics*, 148(6):064705, 2018.

[233] A. J. Archer. Dynamical density functional theory: Phase separation in a cavity and the influence of symmetry. *Journal of Physics: Condensed Matter*, 17(45):S3253, 2005.

[234] A. Moncho-Jordá and J. Dzubiella. Controlling the microstructure and phase behavior of confined soft colloids by active interaction switching. *Physical Review Letters*, 125(7):078001, 2020.

[235] R. Zakine, J.-B. Fournier, and F. Van Wijland. Field-embedded particles driven by active flips. *Physical Review Letters*, 121(2):028001, 2018.

[236] J. Wu. Density functional theory for chemical engineering: From capillarity to soft materials. *AIChE Journal*, 52(3):1169–1193, 2006.

[237] J. Wu and Z. Li. Density-functional theory for complex fluids. *Annual Review of Physical Chemistry*, 58:85–112, 2007.