Doctoral theses at NTNU, 2013:214

Karl Yngve Lervåg

Calculation of interface curvatures with the level-set method for two-phase flow simulations and a second-order diffuse-domain method for elliptic problems in complex geometries

Doctoral Thesis

Karl Yngve Lervåg

ISBN 978-82-471-4544-9 (printed version)
ISBN 978-82-471-4545-6 (electronic version)
ISSN 1503-8181

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology
Thesis for the degree of Philosophiae Doctor
Faculty of Engineering Science & Technology
Department of Energy and Process Engineering

Doctoral theses at NTNU, 2013:214

NTNU

NTNU – Trondheim
Norwegian University of
Science and Technology

Karl Yngve Lervåg

Calculation of interface curvatures with the level-set method for two-phase flow simulations and a second-order diffuse-domain method for elliptic problems in complex geometries

Thesis for the degree of Philosophiae Doctor

Trondheim, July 2013

Norwegian University of Science and Technology
Faculty of Engineering Science & Technology
Department of Energy and Process Engineering

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Engineering Science & Technology
Department of Energy and Process Engineering

ISBN 978-82-471-4544-9 (printed version)
ISBN 978-82-471-4545-6 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2013:214

Printed by Skipnes Kommunikasjon as

Dedicated to Jon Vegard Lervåg (1979–2011)

Abstract

This thesis considers in the first part the mathematical modelling of incompressible two-phase flow, in particular the calculation of interface curvatures and normal vectors with the level-set method. The main contribution is the development of two new numerical methods that enable a more robust calculation of the curvature and normal vectors in areas where the gradient of the level-set method is discontinuous.

Incompressible two-phase flow is in this thesis modelled by the Navier-Stokes equations with a singular source term at the interface between the phases. The singular source term leads to a set of interface jump conditions. These jump conditions are used in the ghost-fluid method to solve two-phase flow in a sharp manner. The interface position is captured and evolved in time with the level-set method. The Navier-Stokes equations for two-phase flow are solved with projection methods and discretized by finite differences in space and Runge-Kutta methods in time. The advective terms in the governing equations are discretized by a weighted essentially non-oscillatory scheme.

In the second part, the thesis considers the more general problem of solving partial-differential equations (PDEs) in complex geometries. An extension of a diffuse-domain method is presented, where the accuracy is improved by adding a correction term. The extension is derived for elliptic problems with Neumann and Robin boundary conditions. One of the advantages of the diffuse-domain methods is that they allow the use of standard tools and methods because they are based on solving PDEs reformulated in larger and regular domains.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for partial fulfilment of the requirements for the degree of philosophiae doctor. The doctoral work has been performed at the Department of Energy and Process Engineering, NTNU, Trondheim, with Professor Bernhard Müller as main supervisor and with Svend Tollak Munkejord, chief scientist at SINTEF Energy Research, as co-supervisor. The work was carried out in the period from September 2010 to June 2013.

The project was financed through the research project "Enabling low emission LNG systems", performed under the Petromaks program and coordinated by SINTEF Energy Research. I gratefully acknowledge the support from the project partners: Statoil, GDF SUEZ, and the Research Council of Norway (contract number 193062/S60).

I am very thankful to both of my supervisors, Bernhard Müller and Svend Tollak Munkejord. In the regular meetings throughout the PhD project they have given me very helpful and encouraging comments and feedback on my work. They have allowed me freedom to pursue my own ideas, but at the same time they have ensured that I was on track so that I finished my PhD work on time.

I am also indebted to Professor John Lowengrub for inviting me to stay at the University of Irvine, California. My stay at UC Irvine was a very enlightening and enjoyable experience, both on a personal and a professional level. I also want to thank Esteban Meca at Lowengrub's lab for many helpful and inspiring discussions.

I am very grateful to the Fulbright Foundation, both for the financial support for the stay in Irvine and for the invaluable aid in the practical matters of living abroad. I am particularly grateful to Ann Kerr for organising several very interesting seminars and events in the Los Angeles area that allowed both me and my wife to meet a lot of wonderful people.

I would like to thank all my colleagues and friends from the Department of Energy and Process Engineering, NTNU. In particular, I would like to thank Claudio Walker for helpful discussions about the modelling of two-phase flow with the level-set method. Further, I would like to thank Åsmund Ervik for many fruitful meetings and discussions, both with regard to our paper about the LOLEX method, and about the intricacies and complications with our numerical code. Last, but not least, I thank my office mate Halvor Lund. It has been a great pleasure to share the office with him these last three years.

I also want to thank Halvor Lund, Frode Bjørdal, and Lars Eivind Lervåg for proofreading my manuscript.

Finally, I extend my deepest gratitude to my wife for her unconditional love and support.

Trondheim, June 2013
Karl Yngve Lervåg

Contents

"The scientific man does not aim at an immediate result. He does not expect that his advanced ideas will be readily taken up. His work is like that of the planter – for the future. His duty is to lay the foundation for those who are to come, and point the way."

— Nikola Tesla (1856–1943)

1 Introduction

This thesis considers the mathematical modelling and numerical computation of two-phase flows. It focuses on developing more robust numerical methods to calculate the curvature and the normal vector of the interface between the two phases. In addition it considers a diffuse-domain method for solving partial differential equations in complex geometries, and it derives an asymptotically second-order method for elliptic problems.

1.1. Background and motivation

Two-phase flows are particular examples of multiphase flows of gas and liquid with an interface that separates the two phases. In the pedantic sense, two-phase flow is flow of a single fluid that occurs as two different phases, for example steam and water. However, it is common to be more general, and in this thesis we use the term two-phase flow also for immiscible mixtures of different fluids, such as water and oil.

Two-phase flows are crucial to a large amount of processes, both in nature and in industry. Examples range from weather phenomena, such as rain drops falling through air, to industrial processes, for instance the separation of water from oil. In general, multiphase flow phenomena

influence any process where liquids and gases are involved. In the oil and gas industry, most processes are two-phase or multiphase flow processes. Needless to say, the understanding of these phenomena is fundamental in the development of new or improved processes.

Consider the international trade of liquefied natural gas (LNG), which is a particular branch of the oil and gas industry that has undergone an exceptional growth in the last decades. There is a strong focus both in Norway and internationally on producing LNG on large floaters (FLNG)*. There are a number of both environmental and economic advantages of such FLNG facilities. In particular, FLNG facilities would remove the need of long pipelines from the gas fields to the shore, there would be no requirement for compression units to pump the gas to the shore, and one would not need to construct onshore production facilities. This would significantly reduce the environmental footprint, and would help preserve marine and coastal environments. Since an FLNG facility can be moved to a new location when a field has been depleted, it would make it economically viable to open up new business opportunities to develop offshore gas fields that would otherwise remain stranded.

Moving the LNG production to an offshore facility presents a demanding set of challenges. A particular challenge is that the elements of a conventional LNG facility need to fit into an area roughly one quarter the original size. Heat exchangers are among the main challenges in the design and operation of LNG plants [24]. Compact and efficient heat exchangers are needed to obtain an energy efficient plant with low emissions. More optimized designs require more accurate tools for design and operation. Such tools can only come as a result of an improved physical understanding of the complex two-phase flows occurring in the heat exchangers. This can be achieved by more detailed mathematical modelling, together with dedicated laboratory observations, cf. [49].

*FLNG facilities do not yet exist, although a facility is under development by Royal Dutch Shell [68]. The construction of this facility was started in 2012, and the first drilling is stated to commence in 2013 [60].

1.2. Overview of methods for two-phase flow simulations

In the following, a brief overview is given of different methods for modelling two-phase flows where the interface location is known. In particular, we focus on methods that handle the interface evolution. For more in-depth reviews, see for instance [14, 57, 59].

Interface propagation methods comprise a range of methods that are often categorised as either front tracking or front capturing. Front-tracking methods use Lagrangian particles to track the interface explicitly [69], while front-capturing methods use an Eulerian approach to capture the interface implicitly. Examples of the latter are volume-of-fluid methods [57, 70], phase-field methods [4, 26, 42], and level-set methods [53].

The main advantages of front-tracking methods are their inherent accuracy and that topological changes do not occur without explicit action. Hence, unphysical numerical reconnection does not occur. This means that if front tracking is used for the simulations of two drops that collide, these drops will not coalesce due to numerical reconnection or merging. However, the handling of topological changes are challenging, in particular in three dimensions [61]. Also, there are some issues of numerical instabilities, as discussed by Sethian [58] and Osher and Sethian [52].

The volume-of-fluid method utilizes a volume-fraction function whose values represent the characteristic function of one of the fluid domains. Its values are zero or one, except in those cells cut by the interface. A considerable advantage of the volume-of-fluid method is that it conserves the mass of both fluids well. However, the reconstruction of the interface from volume fractions is not simple, and computation of geometric quantities such as the interface curvature is not straightforward. Also, spurious bubbles and drops may be created, cf. [36].

The phase-field methods treat the interface in a diffuse manner, where the fluid properties, such as density and viscosity, change rapidly but smoothly across the interface. These methods typically solve the coupled Cahn-Hilliard/Navier-Stokes equations, where the Cahn-Hilliard equation is based on the free energy of an interface [10]. Through this energy formulation, one can model more advanced interface physics such as van der Waals interactions, electrostatic forces, and fluids with varying miscibilities. However, the phase-field methods require that one resolves

very small length scales at the interfaces. This poses a severe restriction on the applicability of phase-field methods for two-phase flow, where the length scales of the flow are generally much larger than those of the interface.

In this thesis, we have used the level-set method [52], which implicitly captures the interface as an isocontour of a function defined in the entire domain. The main motivation of this choice is both that the level-set method handles topological changes of evolving interfaces automatically, cf. Sethian and Smereka [59], and that it is relatively straightforward to implement.

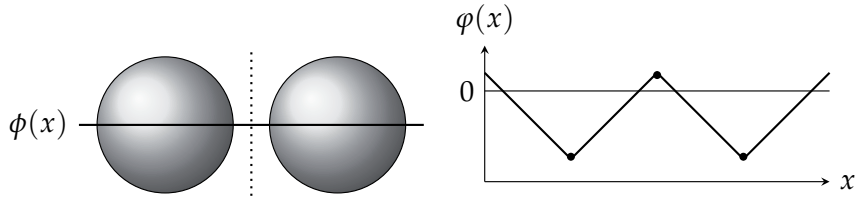
It should be noted that the automatic handling of topological changes is not based on physical principles. For instance, when two interfaces approach each other and their distance becomes less than the spacing of the grid, the level-set method can no longer resolve both interfaces and so they are merged. An important consequence is that the level-set method does not model the physics involved in the coalescence process, and in particular in the smaller scales of the film-drainage process. This process involves a wide range of length scales, varying from the nanometer scale where van der Waals force are active to the length scales of the external flow. Some effort has been made to include the effects of the smaller scales in front-capturing methods, cf. [48, 71]. Recently, Kwakkel et al. [35] presented a level-set/volume-of-fluid method that is coupled with a film-drainage model that predicts if and when two colliding droplets will coalesce. To prevent the numerical coalescence, each droplet has its own locally defined level-set function.

The level-set method has been used to model several diverse phenomena, such as tumour growth [43, 44, 45], propagation of wildland fire [46], and computer RAM production [47]. For a good and thorough introduction to the level-set method, see [53].

A weakness of the level-set method is that it does not conserve the mass of the two fluids, in particular in areas of low resolution and/or high curvature. Different approaches have been developed to overcome this disadvantage, for example the conservative level-set method [50, 51], the particle level-set method [16], or the coupled level-set/volume-of-fluid method [65].

When we use the level-set method to capture the interface for immiscible and incompressible two-phase flow simulations, there will be a sharp

change in the density and viscosity across the interface. In this thesis we use the ghost-fluid method [17], which is a sharp-interface method where the jumps are included in the spatial discretizations in a sharp manner. An alternative method is the continuous surface-force method introduced by Brackbill et al. [8], where the density and the viscosity are smeared out across the interface through a smoothed Heaviside function.



(a) Drops in near contact **(b)** Slice of the level-set function

Figure 1.1.: (a) Two drops in near contact. The dotted line marks a region where the derivative of the level-set function, $\varphi(x)$, is not defined. (b) A one-dimensional slice of the level-set function. The dots mark points where the derivative of $\varphi(x)$ is discontinuous.

1.3. Goal and contribution of the present thesis

The main goal of the PhD project has been to develop fundamental knowledge of two-phase flow phenomenon that are relevant for compact heat exchangers.

In order to do detailed theoretical studies of phenomena that are relevant for compact heat exchangers, we need to consider two-phase flows with mass and heat transfer in confined and complex geometries. One such relevant phenomenon is the drop-film collision process [76]. Even when restricted to isothermal and immiscible flows, this simple phenomenon remains a challenge. When the level-set method is used to capture the interface, one must be particularly careful about how one calculates the interface curvature and normal vector. For instance, when two drops are in near collision there is a kink region in the level-set function between the drops, where the gradient is discontinuous, see Figure 1.1. This discontinuity may lead to large errors in the curvature and the normal vector if it is not taken into account in the discretization stencils.

As a step towards computing two-phase flow simulations in confined geometries, the thesis has considered the diffuse-domain method [40]. This is a method where partial-differential equations in complex domains are extended into larger, regular domains with the use of diffuse approximations of the physical boundaries. The approximations converge asymptotically to the original problem when the width of the diffuse

boundary is reduced. With this method one can use standard numerical methods to solve equations that incorporate complex boundaries.

The main contributions of the present thesis are two new methods to calculate the curvature and normal vector in a robust manner with the level-set method. These methods are shown to yield more accurate calculations of drop-film and drop-drop collisions. The methods are compared with standard methods and with other methods from the literature.

In addition, the thesis presents an extension of a diffuse-domain method by a high-order correction term for the solution of elliptic problems in complex geometries. New analysis provided in the thesis improves the understanding of the diffuse-domain method, and the derived method is shown to be more accurate than the existing diffuse-domain method.

1.4. Outline of the thesis

The thesis is organised as follows: Chapter 2 gives a brief overview of the derivation of the governing equations for two-phase flow. It includes a consideration of the fluid-fluid interface conditions and the derivation of a simplified jump tensor for the viscous term at the interface.

Chapter 3 gives a detailed description of the numerical methods that are used to solve the two-phase flow equations. In particular, it describes the level-set method, which is used to capture the interfaces, projection methods that are used to solve the Navier-Stokes equations, and the spatial and temporal discretization schemes. At the end of the chapter, an overview of the novel discretization methods for the curvature and the normal vector is given.

Chapter 4 gives a short introduction to the diffuse-domain method for an elliptic problem with Neumann boundary conditions. It introduces the high-order correction term, and shows that the new method converges asymptotically with second-order to the original problem. The chapter presents new analysis that shows that the correction term is not necessary for second-order convergence.

In Chapter 5 the main results of the contributed papers A–E are summarized, and the author's contributions are highlighted. Finally,

Chapter 6 gives concluding remarks and provides an outlook for future work.

Full-text versions of the research papers A–E are provided in the Appendices at the end of the thesis.

"But it is just this characteristic of simplicity in the laws of nature hitherto discovered which it would be fallacious to generalize, for it is obvious that simplicity has been a part cause of their discovery, and can, therefore, give no ground for the supposition that other undiscovered laws are equally simple."

— Bertrand Russel (1872–1970)

2

Governing equations for two-phase flow

In this chapter we will give a brief overview of the derivation of the governing equations for two-phase flow. We begin with a consideration of the Navier-Stokes equations for single-phase flow. We then introduce a singular surface-force term and derive the Navier-Stokes equations for two-phase flow. Finally, we use the Navier-Stokes equations for two-phase flow to derive jump conditions at the interface between the phases.

2.1. The Navier-Stokes equations for single-phase flow

The following is a brief derivation of the Navier-Stokes equations for single-phase flows. For a more thorough derivation of these equations, see for instance Aris [5, §4 and §5] or White [73, Chapter 4].

We consider a single-phase, viscous flow in some domain Ω with boundary $\partial\Omega$. When temperature effects are neglected the flow is described by the Cauchy equation

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}\right) = \boldsymbol{\nabla}\cdot\boldsymbol{T} + \rho\boldsymbol{f}, \tag{2.1}$$

and the mass conservation equation,

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{u}) = 0. \tag{2.2}$$

Here ρ is the fluid density, $\boldsymbol{u}$ is the flow velocity, t is time, $\boldsymbol{T}$ is the stress tensor, $\boldsymbol{f}$ denotes body forces, and $\boldsymbol{\nabla}$ is used to denote the gradient and divergence operators. The stress tensor for Newtonian fluids with zero bulk viscosity is

$$\boldsymbol{T} = -p\boldsymbol{I} + 2\mu\boldsymbol{D} - \frac{2}{3}\mu(\operatorname{tr}\boldsymbol{D})\boldsymbol{I}, \tag{2.3}$$

where p is the pressure, $\boldsymbol{I}$ is the identity tensor, μ is the dynamic viscosity, and $\operatorname{tr}\boldsymbol{D}$ denotes the trace of the strain-rate tensor $\boldsymbol{D}$,

$$\boldsymbol{D} = \frac{1}{2}\left(\boldsymbol{\nabla}\boldsymbol{u} + (\boldsymbol{\nabla}\boldsymbol{u})^T\right). \tag{2.4}$$

For incompressible flow, the mass-conservation equation reduces to

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0, \tag{2.5}$$

that is, the velocity field must be divergence free. If we further assume that the viscosity is constant, then it follows that the divergence of the stress tensor reduces to

$$\boldsymbol{\nabla} \cdot \boldsymbol{T} = -\boldsymbol{\nabla} p + \mu \boldsymbol{\Delta} \boldsymbol{u}, \tag{2.6}$$

and so the Cauchy equation (2.1) becomes

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla}\boldsymbol{u}\right) = -\boldsymbol{\nabla} p + \mu \boldsymbol{\Delta}\boldsymbol{u} + \rho \boldsymbol{f}. \tag{2.7}$$

Equations (2.5) and (2.7) are the incompressible Navier-Stokes equations for single-phase flow with constant viscosity.

2.2. The Navier-Stokes equations for two-phase flow

We now consider an immiscible two-phase flow of two Newtonian fluids, each with its own viscosity and density. We let Ω_1 and Ω_2 denote the domains occupied by fluid 1 and fluid 2, respectively, and let the

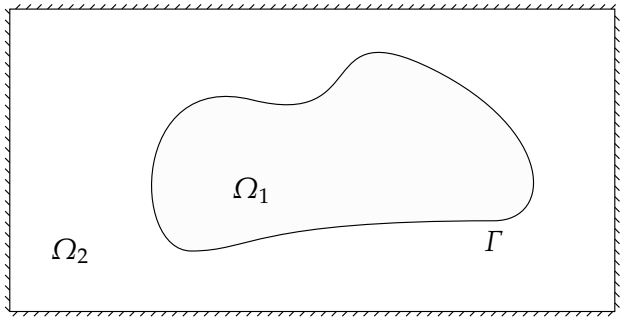


Figure 2.1.: An illustration of a two-phase flow domain. The interface Γ separates the two phases Ω_1 and Ω_2.

interface between the fluids be denoted by Γ. Then $\Omega = \Omega_1 \cup \Omega_2$ and $\partial\Omega = (\partial\Omega_1 \cup \partial\Omega_2) \setminus \Gamma$ are the fluid domain and its boundary, respectively. See Figure 2.1 for an illustration of a two-phase flow domain.

The extension of the single-phase model to account for two fluids can be made by adding a singular surface-force term to represent the effects of surface tension between the fluids. We assume that the surface tension is constant, in which case the singular surface force can be defined as

$$\boldsymbol{f}_s(\boldsymbol{x}, t) = \int_\Gamma \sigma\kappa\boldsymbol{n}\delta(\boldsymbol{x} - \boldsymbol{x}_I(\boldsymbol{s}))\,\mathrm{d}\boldsymbol{s}, \tag{2.8}$$

where σ is the surface tension, κ is the local curvature, $\boldsymbol{n}$ is the normal vector, δ is the Dirac delta function, and $\boldsymbol{x}_I(\boldsymbol{s})$ is a parametrisation of the interface. Thus the Navier-Stokes equations for immiscible and incompressible two-phase flow are

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0, \tag{2.9}$$

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla}\boldsymbol{u}\right) = -\boldsymbol{\nabla} p + \mu\boldsymbol{\Delta}\boldsymbol{u} + \rho\boldsymbol{f} + \boldsymbol{f}_s. \tag{2.10}$$

2.3. Fluid-fluid interface conditions

The surface-tension force and the jump in viscosity and density across the interface Γ lead to a set of interface conditions that must be satis-

fied along the interface Γ. The following is a brief derivation of these conditions.

First, we only consider flows where there is no mass transfer, which implies that

$$[\![\boldsymbol{u}]\!] \cdot \boldsymbol{n} = 0, \tag{2.11}$$

where $[\![\cdot]\!]$ denotes the jump at the interface, for instance $[\![\mu]\!] = \mu_2 - \mu_1$. Further, for viscous flows there is no slip at the interface, and thus the tangential velocity component of the two fluids must be equal at the interface,

$$[\![\boldsymbol{u}]\!] \cdot \boldsymbol{t} = 0. \tag{2.12}$$

It follows that

$$[\![\boldsymbol{u}]\!] = 0, \tag{2.13}$$

$$[\![\boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{t} = 0. \tag{2.14}$$

The latter is a direct consequence of the former. Both fluids are incompressible, which gives the trivial identity $[\![\boldsymbol{\nabla} \cdot \boldsymbol{u}]\!] = 0$. If we use the identity

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = \boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{n} + \boldsymbol{t} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t} \tag{2.15}$$

together with (2.14) we get

$$\boldsymbol{n} \cdot [\![\boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{n} = 0, \tag{2.16}$$

which means that the normal component of the normal derivative of the velocity field is continuous across the interface. Note that in (2.15) and in similar expressions in the following, the nabla operator is only applied to $\boldsymbol{u}$, not the normal and tangential vectors that follow.

Next, we consider the conservation of momentum. We define a control volume

$$\Omega_\epsilon = \left\{ \boldsymbol{x} \in \Omega \, : \, \min_{\boldsymbol{x}_I \in \Gamma} |\boldsymbol{x} - \boldsymbol{x}_I| \leq \epsilon \right\}, \tag{2.17}$$

where $\epsilon > 0$, see Figure 2.2. We then integrate the Cauchy equation (2.1) with the added singular surface-force term over the domain Ω_ϵ,

$$\int_{\Omega_\epsilon} \rho \frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t} \, \mathrm{d}\boldsymbol{x} = \int_{\Omega_\epsilon} \boldsymbol{\nabla} \cdot \boldsymbol{T} \, \mathrm{d}\boldsymbol{x} + \int_{\Omega_\epsilon} \rho \boldsymbol{f} \, \mathrm{d}\boldsymbol{x} + \int_{\Omega_\epsilon} \int_\Gamma \sigma \kappa \boldsymbol{n} \delta(\boldsymbol{x} - \boldsymbol{x}_I(\boldsymbol{s})) \, \mathrm{d}\boldsymbol{s} \, \mathrm{d}\boldsymbol{x}. \tag{2.18}$$

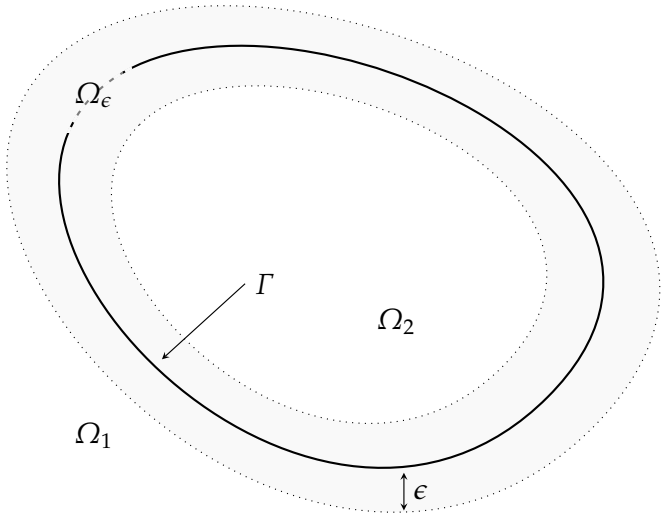


Figure 2.2.: A sketch of the control volume Ω_ϵ that covers the interface, Γ.

Here $\mathrm{D}\boldsymbol{u}/\mathrm{D}t = \partial\boldsymbol{u}/\partial t + \boldsymbol{u}\cdot\boldsymbol{\nabla}\boldsymbol{u}$ denotes the convective derivative. Now we apply the Gauss theorem and change the order of integration in the last term to obtain

$$\int_{\Omega_\epsilon} \rho\frac{\mathrm{D}\boldsymbol{u}}{\mathrm{D}t}\,\mathrm{d}\boldsymbol{x} = \int_{\partial\Omega_\epsilon} \boldsymbol{T}\cdot\boldsymbol{n}\,\mathrm{d}s + \int_{\Omega_\epsilon}\rho\boldsymbol{f}\,\mathrm{d}\boldsymbol{x} + \int_\Gamma \sigma\kappa\boldsymbol{n}\,\mathrm{d}s. \tag{2.19}$$

If we let ϵ go to zero, the left-hand side and the second term on the right-hand side vanish, and we get

$$0 = \int_\Gamma (\llbracket\boldsymbol{T}\rrbracket\cdot\boldsymbol{n} + \sigma\kappa\boldsymbol{n})\,\mathrm{d}s. \tag{2.20}$$

Since the derivation above is also valid for any subset of Ω_ϵ containing a part of Γ, (2.20) must hold for any part of Γ. Therefore

$$0 = \llbracket\boldsymbol{T}\rrbracket\cdot\boldsymbol{n} + \sigma\kappa\boldsymbol{n}. \tag{2.21}$$

With $\boldsymbol{T} = -p\boldsymbol{I} + 2\mu\boldsymbol{D}$ we finally get the interface condition for the stresses,

$$\llbracket p\rrbracket\boldsymbol{n} - \llbracket 2\mu\boldsymbol{D}\rrbracket\cdot\boldsymbol{n} = \sigma\kappa\boldsymbol{n}. \tag{2.22}$$

The surface tension force is seen to introduce a discontinuity in the normal stresses across the interface. The tangential stresses are continuous.

2.4. Interface conditions for the pressure and the viscous term

The previous section gave a brief derivation of the interface conditions for immiscible and incompressible two-phase flow without mass transfer. In order to use these conditions for the discretization of the Navier-Stokes equations (2.9) and (2.10), we need to rewrite them into a more suitable form. In particular, we want to find explicit expressions for the jump in the pressure $[\![p]\!]$ and the viscous term $[\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!]$. The following derivation is based on [22] and [30].

First, a jump condition for the pressure is obtained by taking the inner product of (2.22) with the normal vector $\boldsymbol{n}$,

$$[\![p]\!] = \boldsymbol{n} \cdot [\![2\mu \boldsymbol{D}]\!] \cdot \boldsymbol{n} + \sigma\kappa = 2[\![\mu]\!] \boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{n} + \sigma\kappa, \tag{2.23}$$

where (2.16) was used for the second equality.

To find the jump in the viscous term, it is first decomposed into an interface normal coordinate system as

$$\begin{aligned} [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] = {} & (\boldsymbol{n} \cdot [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{n})\, \boldsymbol{n} \otimes \boldsymbol{n} + (\boldsymbol{t} \cdot [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{t})\, \boldsymbol{t} \otimes \boldsymbol{t} \\ & + (\boldsymbol{n} \cdot [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{t})\, \boldsymbol{n} \otimes \boldsymbol{t} + (\boldsymbol{t} \cdot [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{n})\, \boldsymbol{t} \otimes \boldsymbol{n}, \end{aligned} \tag{2.24}$$

where $\otimes$ denotes the dyadic product. We already showed that $[\![\boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{t} = 0$ and $\boldsymbol{n} \cdot [\![\boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{n} = 0$, cf. (2.14) and (2.16), which gives

$$\boldsymbol{n} \cdot [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{n} = [\![\mu]\!]\, \boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{n}, \tag{2.25}$$

$$\boldsymbol{t} \cdot [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{t} = [\![\mu]\!]\, \boldsymbol{t} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t}, \tag{2.26}$$

$$\boldsymbol{n} \cdot [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{t} = [\![\mu]\!]\, \boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t}. \tag{2.27}$$

We then take the inner product of (2.22) with $\boldsymbol{t}$, which gives

$$\boldsymbol{t} \cdot \left[\!\!\left[\mu \left(\boldsymbol{\nabla} \boldsymbol{u} + (\boldsymbol{\nabla} \boldsymbol{u})^T \right) \right]\!\!\right] \cdot \boldsymbol{n} = \boldsymbol{t} \cdot [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{n} + [\![\mu]\!] \boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t} = 0, \tag{2.28}$$

or

$$\boldsymbol{t} \cdot [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] \cdot \boldsymbol{n} = -[\![\mu]\!] \boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t}. \tag{2.29}$$

The jump in the viscous term becomes

$$\begin{aligned} [\![\mu \boldsymbol{\nabla} \boldsymbol{u}]\!] = [\![\mu]\!] \Big(& (\boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{n}) \boldsymbol{n} \otimes \boldsymbol{n} + (\boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t}) \boldsymbol{n} \otimes \boldsymbol{t} \\ & - (\boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t}) \boldsymbol{t} \otimes \boldsymbol{n} + (\boldsymbol{t} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t}) \boldsymbol{t} \otimes \boldsymbol{t} \Big). \end{aligned} \tag{2.30}$$

This can be simplified further by noting that

$$(\boldsymbol{\nabla u} \cdot t) \otimes \boldsymbol{t} = (\boldsymbol{n} \cdot \boldsymbol{\nabla u} \cdot \boldsymbol{t})\boldsymbol{n} \otimes \boldsymbol{t} + (\boldsymbol{t} \cdot \boldsymbol{\nabla u} \cdot \boldsymbol{t})\boldsymbol{t} \otimes \boldsymbol{t}. \tag{2.31}$$

Thus we obtain the expression for the jump in the viscous term that has been used in the present work,

$$[\![\mu \boldsymbol{\nabla u}]\!] = [\![\mu]\!]\Big((\boldsymbol{n} \cdot \boldsymbol{\nabla u} \cdot \boldsymbol{n})\boldsymbol{n} \otimes \boldsymbol{n} - (\boldsymbol{n} \cdot \boldsymbol{\nabla u} \cdot \boldsymbol{t})\boldsymbol{t} \otimes \boldsymbol{n} + (\boldsymbol{\nabla u} \cdot \boldsymbol{t}) \otimes \boldsymbol{t}\Big). \tag{2.32}$$

2.5. Summary

This chapter has given a brief derivation of the immiscible and incompressible Navier-Stokes equations for two-phase flow (2.9) and (2.10), where the viscosity is assumed constant in each phase. The effect of surface tension is included as a singular source term.

Fluid-fluid interface conditions have been discussed, and it has been shown how the singular source term and a jump in viscosity lead to a set of interface jump conditions for the pressure (2.23) and the viscous term (2.32).

"As long as you recognize your sinful ways, correct your behaviour and adopt the right method and the right step size, your misdemeanours will be forgiven and your solution will prosper."

— Arieh Iserles (1947)

3 Numerical methods

This chapter describes the numerical methods that have been used in this thesis to solve the Navier-Stokes equations for two-phase flow. First, a brief introduction to the level-set method is given, which is followed by an outline of the two projection methods that have been used to solve the Navier-Stokes equations. The spatial and temporal discretization methods are then summarized, and at the end of the chapter the new methods to calculate the curvature and the normal vectors are presented.

3.1. The level-set method

In order to solve the Navier-Stokes equations for two-phase flow, we need to know the location of the interface. The level-set method proposed by Osher and Sethian [52] allows us to capture the interface location as the zero level set of the level-set function $\varphi(\boldsymbol{x}, t)$. The level-set function is typically defined as a signed-distance function,

$$\varphi(\boldsymbol{x}, t) = \begin{cases} d(\boldsymbol{x}, t) & \text{if } \boldsymbol{x} \in \Omega_2, \\ -d(\boldsymbol{x}, t) & \text{if } \boldsymbol{x} \in \Omega_1, \end{cases} \tag{3.1}$$

where $d(\boldsymbol{x}, t)$ is the shortest distance to the interface Γ,

$$d(\boldsymbol{x}, t) = \min_{\boldsymbol{x}_I \in \Gamma} |\boldsymbol{x} - \boldsymbol{x}_I|. \tag{3.2}$$

Thus the interface can be defined implicitly as

$$\Gamma(t) = \{\boldsymbol{x} \in \Omega \,:\, \varphi(\boldsymbol{x}, t) = 0\}, \quad t \in \mathbb{R}^+. \tag{3.3}$$

The position of the interface is updated by solving an advection equation for φ,

$$\frac{\partial \varphi}{\partial t} + \hat{\boldsymbol{u}} \cdot \boldsymbol{\nabla} \varphi = 0, \tag{3.4}$$

where $\hat{\boldsymbol{u}}$ is the velocity at the interface extended to the entire domain. We extend the interface velocity through solving a velocity-extrapolation equation,

$$\frac{\partial \hat{\boldsymbol{u}}}{\partial \tau} + S(\varphi)\boldsymbol{n} \cdot \boldsymbol{\nabla} \hat{\boldsymbol{u}} = 0, \quad \hat{\boldsymbol{u}}_{\tau=0} = \boldsymbol{u}, \tag{3.5}$$

to steady state, cf. [2, 78]. Here τ is a pseudo-time and S is a smeared sign function which is equal to zero at the interface,

$$S(\varphi) = \frac{\varphi}{\sqrt{\varphi^2 + 2\Delta x^2}}. \tag{3.6}$$

When we solve the level-set equation (3.4), the non-uniform advection will distort the signed-distance property of the level-set function, and numerical dissipation error adds to this distortion. The level-set function is therefore reinitialized regularly by solving

$$\begin{aligned} \frac{\partial \varphi}{\partial \tau} + S(\varphi_0)(|\boldsymbol{\nabla} \varphi| - 1) &= 0, \\ \varphi(\boldsymbol{x}, 0) &= \varphi_0(\boldsymbol{x}), \end{aligned} \tag{3.7}$$

to steady state as proposed by Sussman et al. [64]. The level-set function just before initialization is used as the initial condition φ_0.

The level-set equations (3.4), (3.5), and (3.7) are discretized in time and space as described in the following sections. The method presented by Adalsteinsson and Sethian [1] is used to improve the computational speed. The method is often called the narrow-band method, since the

level-set function is only updated in a narrow band across the interface at each time step.

One of the advantages of the level-set method is that normal vectors and curvatures can be readily calculated from the level-set function, that is,

$$\boldsymbol{n} = \frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}, \tag{3.8}$$

$$\kappa = \boldsymbol{\nabla} \cdot \left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|} \right). \tag{3.9}$$

3.2. Projection methods

We have employed two different projection methods in this thesis to solve the incompressible Navier-Stokes equations for two-phase flow (2.9) and (2.10). Papers A, B, and C used the direct projection scheme 1 (DP1) [22], and Paper D used the more standard Chorin projection method [12].

The projection methods are a family of methods for the solution of the incompressible Navier-Stokes equations that are based on the Helmholtz-Hodge theorem. This theorem states that an arbitrary vector field can be decomposed into a divergence-free part and a rotation-free part. That is, any vector field $\boldsymbol{a}$ can be written as

$$\boldsymbol{a} = \boldsymbol{a}' + \boldsymbol{\nabla}\psi, \tag{3.10}$$

where $\boldsymbol{a}'$ is a vector with $\boldsymbol{\nabla} \cdot \boldsymbol{a}' = 0$ and ψ is a scalar potential. The proof of this theorem can be found in for instance [5, §3.44] or [11].

3.2.1. The direct projection scheme 1

The DP1 was developed by Hansen [22] and is based on a direct application of the Helmholtz-Hodge theorem. We assume that the velocity $\boldsymbol{u}$ is sufficiently smooth, and then rewrite (2.7) to get

$$\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{w} - \frac{\boldsymbol{\nabla} p}{\rho}, \tag{3.11}$$

where

$$\boldsymbol{w} = -(\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} + \nu \boldsymbol{\nabla}^2 \boldsymbol{u} + \boldsymbol{f}. \tag{3.12}$$

The divergence of (3.11) yields a Poisson equation for the pressure,

$$\boldsymbol{\nabla} \cdot \left(\frac{\boldsymbol{\nabla} p}{\rho} \right) = \boldsymbol{\nabla} \cdot \boldsymbol{w}. \tag{3.13}$$

The DP1 scheme follows from a direct numerical discretization of the above equations. First, $\boldsymbol{w}$ is calculated with (3.12). Then the pressure is found by solving the Poisson equation (3.13). Finally, an Euler step is used to solve (3.11), that is,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t \left(\boldsymbol{w} - \frac{\boldsymbol{\nabla} p}{\rho} \right). \tag{3.14}$$

3.2.2. The Chorin projection method

The Chorin projection method was presented by Chorin [12], and is today one of the standard methods for solving the Navier-Stokes equations, e.g. [72]. The method is briefly presented in the following.

Let Δt be the time step, and consider the discretization of the momentum equation (2.7) with the forward Euler method,

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} = -\boldsymbol{u}^n \cdot \boldsymbol{\nabla} \boldsymbol{u}^n - \frac{\boldsymbol{\nabla} p^{n+1}}{\rho} + \frac{\mu}{\rho} \boldsymbol{\Delta} \boldsymbol{u}^n + \boldsymbol{f}, \tag{3.15}$$

where $\boldsymbol{u}^n \equiv \boldsymbol{u}(\boldsymbol{x}, n\Delta t)$ and $p^n \equiv p(\boldsymbol{x}, n\Delta t)$ are assumed known at time level n. Note that the pressure gradient is evaluated at time level $n+1$. Next, to solve (3.15) in two steps, we introduce the intermediate velocity field $\boldsymbol{u}^\star$,

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^\star + \boldsymbol{u}^\star - \boldsymbol{u}^n}{\Delta t} = -\boldsymbol{u}^n \cdot \boldsymbol{\nabla} \boldsymbol{u}^n - \frac{\boldsymbol{\nabla} p^{n+1}}{\rho} + \frac{\mu}{\rho} \boldsymbol{\Delta} \boldsymbol{u}^n + \boldsymbol{f}, \tag{3.16}$$

which is chosen such that

$$\frac{\boldsymbol{u}^\star - \boldsymbol{u}^n}{\Delta t} = -\boldsymbol{u}^n \cdot \boldsymbol{\nabla} \boldsymbol{u}^n + \frac{\mu}{\rho} \boldsymbol{\Delta} \boldsymbol{u}^n + \boldsymbol{f}, \tag{3.17}$$

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^\star}{\Delta t} = -\frac{\boldsymbol{\nabla} p^{n+1}}{\rho}. \tag{3.18}$$

From (3.17) we get an explicit expression for $\boldsymbol{u}^\star$,

$$\boldsymbol{u}^\star = \boldsymbol{u}^n + \Delta t \left(-\boldsymbol{u}^n \cdot \boldsymbol{\nabla} \boldsymbol{u}^n + \frac{\mu}{\rho} \boldsymbol{\Delta} \boldsymbol{u}^n + \boldsymbol{f} \right). \tag{3.19}$$

Next, the divergence of (3.18) and $\boldsymbol{\nabla} \cdot \boldsymbol{u}^{n+1} = 0$ yields a Poisson equation for the pressure,

$$\boldsymbol{\nabla} \cdot \left(\frac{\boldsymbol{\nabla} p^{n+1}}{\rho} \right) = \frac{\boldsymbol{\nabla} \cdot \boldsymbol{u}^\star}{\Delta t}. \tag{3.20}$$

Finally, we obtain for (3.18)

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^\star - \Delta t \frac{\boldsymbol{\nabla} p^{n+1}}{\rho}. \tag{3.21}$$

3.3. Spatial discretization

The governing equations (2.9), (2.10), (3.4), (3.5), and (3.7) are discretized on a staggered grid [23], where the scalar values are located at the cell centres and the vector values are located at the cell edges, see Figure 3.1. The domain boundary coincides with cell edges, and the fixed grid spacing is Δx in the x direction and Δy in the y direction.

The x- and y-derivatives, divergence, and Laplacian operators are discretized by the second-order central-difference scheme,

$$G_x p\big|_{i+\frac{1}{2},j} = \frac{p_{i+1,j} - p_{i,j}}{\Delta x}, \tag{3.22}$$

$$G_y p\big|_{i,j+\frac{1}{2}} = \frac{p_{i,j+1} - p_{i,j}}{\Delta y}, \tag{3.23}$$

$$\boldsymbol{D} \cdot \boldsymbol{u}\big|_{i,j} = \frac{u_{i+1/2,j} - u_{i-1/2,j}}{\Delta x} + \frac{v_{i,j+1/2} - v_{i,j-1/2}}{\Delta y}, \tag{3.24}$$

$$Lp\big|_{i,j} = \frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(\Delta x)^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(\Delta y)^2}, \tag{3.25}$$

respectively. Here $\boldsymbol{u} = (u, v)$ is a vector and p is a scalar. Note that the gradient of a scalar is a vector and has values located at the cell edges. These definitions are consistent with the staggered grid, since the gradient returns a vector with components defined on the cell faces, while the divergence and the Laplacian returns a scalar defined at the cell centres.

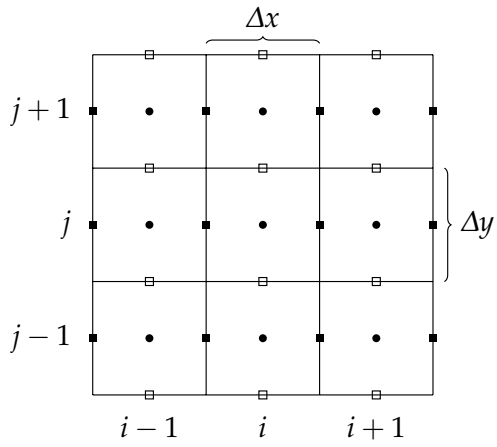


Figure 3.1.: An illustration of a small part of a uniform staggered grid, where the fixed grid spacings Δx and Δy are indicated. The scalar values are stored at the cell centres (filled circles), the x-component of the vector values is stored at the cell edges $(i+\frac{1}{2}, j)$ (filled squares), and the y-component of the vector values is stored at the cell edges $(i, j+\frac{1}{2})$ (open squares).

3.3.1. Advective terms

The advective terms, $\boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u}$ in the momentum equation (2.10) and $\hat{\boldsymbol{u}} \cdot \boldsymbol{\nabla} \varphi$ in the level-set equation (3.4), and the normal derivative $\boldsymbol{n} \cdot \boldsymbol{\nabla} \hat{\boldsymbol{u}}$ in the velocity-extrapolation equation (3.5) are discretized with the weighted essentially non-oscillatory (WENO) scheme, cf. [17] and [28]. The WENO scheme is a high-order upwind scheme that is fifth-order accurate in smooth regions. In nonsmooth regions the accuracy is reduced to a minimum of third order. The following is a brief outline of the WENO scheme.

First, when we calculate the advective term for the velocity, the vector components are required on all cell edges. That is, we must interpolate the x-component of the velocity to the location of the y-component of

the velocity and vice versa. To do this we use a linear interpolation,

$$u_{i,j+\frac{1}{2}} = \frac{1}{4}\left(u_{i-\frac{1}{2},j} + u_{i-\frac{1}{2},j+1} + u_{i+\frac{1}{2},j+1} + u_{i+\frac{1}{2},j}\right), \tag{3.26}$$

$$v_{i+\frac{1}{2},j} = \frac{1}{4}\left(v_{i,j-\frac{1}{2}} + v_{i+1,j-\frac{1}{2}} + v_{i+1,j+\frac{1}{2}} + v_{i,j+\frac{1}{2}}\right). \tag{3.27}$$

Similarly, when we calculate the advective term for the level-set function φ, we first interpolate the velocity to the scalar grid,

$$u_{i,j} = \frac{1}{2}\left(u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j}\right), \tag{3.28}$$

$$v_{i,j} = \frac{1}{2}\left(v_{i,j+\frac{1}{2}} + v_{i,j-\frac{1}{2}}\right). \tag{3.29}$$

We now consider the WENO scheme for the advective operator in 1D at the point $x_{i+\frac{1}{2}}$. The scheme extends naturally to higher dimensions. First, if $u_{i+\frac{1}{2}} = 0$, then

$$u\frac{\partial u}{\partial x}\bigg|_{i+\frac{1}{2}} = 0. \tag{3.30}$$

Otherwise we need to calculate a set of five differences that depend on the upwind direction. The differences are denoted Δv_k for $k = 1, \ldots, 5$. If $u_{i+\frac{1}{2}} > 0$, then we calculate

$$\Delta v_k = \frac{u_{i+\frac{2k-5}{2}} - u_{i+\frac{2k-7}{2}}}{\Delta x}. \tag{3.31}$$

Otherwise if $u_{i+\frac{1}{2}} < 0$, we calculate

$$\Delta v_k = \frac{u_{i-\frac{2k-9}{2}} - u_{i-\frac{2k-7}{2}}}{\Delta x}. \tag{3.32}$$

Next we calculate expressions for the smoothness of three substencils,

$$S_1 = \frac{13}{12}(\Delta v_1 - 2\Delta v_2 + \Delta v_3)^2 + \frac{1}{4}(\Delta v_1 - 4\Delta v_2 + 3\Delta v_3)^2, \tag{3.33}$$

$$S_2 = \frac{13}{12}(\Delta v_2 - 2\Delta v_3 + \Delta v_4)^2 + \frac{1}{4}(\Delta v_2 - \Delta v_4)^2, \tag{3.34}$$

$$S_3 = \frac{13}{12}(\Delta v_3 - 2\Delta v_4 + \Delta v_5)^2 + \frac{1}{4}(3\Delta v_3 - 4\Delta v_4 + \Delta v_5)^2, \tag{3.35}$$

where a small S indicates a smooth substencil. These smoothness factors are then used to compute weights for the substencils,

$$w_k = \frac{b_k}{b_1 + b_2 + b_3}, \tag{3.36}$$

for $k = 1, 2, 3$, where

$$b_1 = \frac{1}{10}\frac{1}{(\epsilon + S_1)^2}, \quad b_2 = \frac{6}{10}\frac{1}{(\epsilon + S_2)^2}, \quad b_3 = \frac{3}{10}\frac{1}{(\epsilon + S_3)^2}. \tag{3.37}$$

Here ϵ is a regularization parameter that is used to avoid division by zero. We have used $\epsilon = 10^{-6}$ in this work. Finally, the WENO scheme for the gradient becomes

$$\begin{aligned}\left.\frac{\partial u}{\partial x}\right|_{i+\frac{1}{2}} \simeq\ & w_1\left(\frac{1}{3}\Delta v_1 - \frac{7}{6}\Delta v_2 + \frac{11}{6}\Delta v_3\right) \\ & + w_2\left(-\frac{1}{6}\Delta v_2 + \frac{5}{6}\Delta v_3 + \frac{1}{3}\Delta v_4\right) \\ & + w_3\left(\frac{1}{3}\Delta v_3 + \frac{5}{6}\Delta v_4 - \frac{1}{6}\Delta v_5\right).\end{aligned} \tag{3.38}$$

3.3.2. The viscous term

The viscous term $\mu\boldsymbol{\Delta u}$ in the Navier-Stokes equations is discretized by standard second-order central differences using the ghost-fluid method (GFM) [17, 30, 41]. This method includes the jump in the viscous term at the interface (2.32) in the discretization stencil in a sharp manner.

In the following we present the GFM scheme in the one-dimensional case. The extension to higher dimensions is straightforward. In order to simplify the notation, we omit the half indices and let $k \equiv i + \frac{1}{2}$. Further, we consider a general case where

$$[\![u]\!] = a_\Gamma, \qquad \left[\!\!\left[\mu\tfrac{\partial u}{\partial x}\right]\!\!\right] = b_\Gamma. \tag{3.39}$$

For the viscous term, $a_\Gamma = 0$, cf. (2.13), and b_Γ is given by the jump tensor (2.32). The jump tensor is calculated at the cell centres near the interface with the second-order central-difference scheme. It is then

interpolated linearly to the cell edges, from where it is again interpolated linearly to the interface when needed.

If the interface does not cross the stencil, then the GFM scheme reduces to the standard second-order central-difference stencil. Else there are four stencils, depending on the location of the interface. One such interface configuration is sketched in Figure 3.2. In all the stencils, θ is defined as the relative distance from the interface to the node on the left, for instance

$$\theta = \frac{|\varphi_k|}{|\varphi_k| + |\varphi_{k+1}|}, \tag{3.40}$$

where φ_k and φ_{k+1} are the level-set function values linearly interpolated to the vector grid. The four stencils are given below, where $h \equiv \Delta x$.

1. Phase 1 is to the left and interface lies between k and $k+1$:

$$\begin{aligned} \frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right)\bigg|_{x_k} &= \frac{\hat{\mu}\,(u_{k+1}-u_k) - \mu_1\,(u_k - u_{k-1})}{h^2} \\ &\quad - \frac{\hat{\mu} a_\Gamma}{h^2} - \frac{\hat{\mu} b_\Gamma (1-\theta)}{\mu_2 h}, \end{aligned} \tag{3.41}$$

$$\hat{\mu} = \frac{\mu_1 \mu_2}{\theta\mu_2 + (1-\theta)\mu_1}. \tag{3.42}$$

2. Phase 1 is to the left and interface lies between $k-1$ and k:

$$\begin{aligned} \frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right)\bigg|_{x_k} &= \frac{\mu_2\,(u_{k+1}-u_k) - \hat{\mu}\,(u_k - u_{k-1})}{h^2} \\ &\quad + \frac{\hat{\mu} a_\Gamma}{h^2} - \frac{\hat{\mu} b_\Gamma \theta}{\mu_1 h}, \end{aligned} \tag{3.43}$$

$$\hat{\mu} = \frac{\mu_1 \mu_2}{\theta\mu_2 + (1-\theta)\mu_1}. \tag{3.44}$$

3. Phase 2 is to the left and interface lies between k and $k+1$:

$$\begin{aligned} \frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right)\bigg|_{x_k} &= \frac{\hat{\mu}\,(u_{k+1}-u_k) - \mu_2\,(u_k - u_{k-1})}{h^2} \\ &\quad + \frac{\hat{\mu} a_\Gamma}{h^2} + \frac{\hat{\mu} b_\Gamma (1-\theta)}{\mu_1 h}, \end{aligned} \tag{3.45}$$

$$\hat{\mu} = \frac{\mu_1 \mu_2}{\theta\mu_1 + (1-\theta)\mu_2}. \tag{3.46}$$

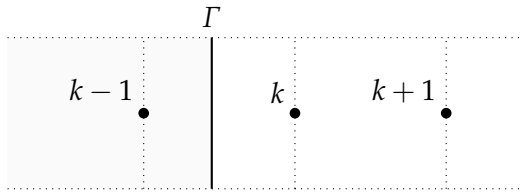


Figure 3.2.: One-dimensional case where the interface Γ separates fluid 1 on the left from fluid 2 on the right.

4. Phase 2 is to the left and interface lies between $k-1$ and k:

$$\left.\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right)\right|_{x_k} = \frac{\mu_1\left(u_{k+1}-u_k\right)-\hat{\mu}\left(u_k-u_{k-1}\right)}{h^2} - \frac{\hat{\mu}a_\Gamma}{h^2}+\frac{\hat{\mu}b_\Gamma\theta}{\mu_2 h}, \tag{3.47}$$

$$\hat{\mu} = \frac{\mu_1\mu_2}{\theta\mu_1+(1-\theta)\mu_2}. \tag{3.48}$$

3.3.3. Pressure Poisson equation

The Laplace operator in the Poisson equations (3.13) and (3.20) is also discretized by standard second-order central differences using the GFM [30, 41]. The GFM was discussed in the previous section, where stencils were given for the viscous term in the one-dimensional case. For the pressure, the stencils are the same except we use p instead of u, and $1/\rho$ and $1/\hat{\rho}$ instead of μ and $\hat{\mu}$. To find the jump in the pressure at the interface, $[\![p]\!] = a_\Gamma$, the jump condition (2.23) is calculated at the cell centres and then interpolated to the interface. In addition, we use that $b_\Gamma = 0$, which is justified by Kang et al. [30, Section 3.7].

The resulting linear system for the pressure is solved with a solver from the Portable, Extensible Toolkit for Scientific Computation (PETSc) [6]. PETSc makes available a large selection of solvers. In most cases, we used the direct solver based on LU factorisation or the conjugate gradient method with incomplete Cholesky factorisation. In some cases, we used the GMRES solver with a preconditioner, either incomplete LU factorisation or an algebraic multigrid method. See [6] for more details

and a list of available methods, and see for example [25] or [55] for an introduction to linear solvers.

Finally, the gradient of the pressure, used in (3.11) and (3.21), is also calculated with the GFM. In one dimension, the stencil is

$$\left.\frac{1}{\rho}\frac{\partial p}{\partial x}\right|_{i+\frac{1}{2}} = \frac{1}{\hat{\rho}}\frac{(p_{i+1} - a_\Gamma) - p_i}{h}, \tag{3.49}$$

$$\hat{\rho} = \theta\rho_1 + (1-\theta)\rho_2, \tag{3.50}$$

if $\phi_i \leq 0$ and $\phi_{i+1} > 0$, or else

$$\left.\frac{1}{\rho}\frac{\partial p}{\partial x}\right|_{i+\frac{1}{2}} = \frac{1}{\hat{\rho}}\frac{(p_{i+1} + a_\Gamma) - p_i}{h}, \tag{3.51}$$

$$\hat{\rho} = \theta\rho_2 + (1-\theta)\rho_1, \tag{3.52}$$

where

$$\theta = \frac{|\varphi_i|}{|\varphi_i| + |\varphi_{i+1}|}. \tag{3.53}$$

3.4. Temporal discretization

The temporal discretization is done with the explicit strong stability-preserving Runge-Kutta (SSP-RK) schemes, see [31, 62]. The idea behind the SSP-RK schemes is to preserve the stability of a low-order method when it is extended to higher order. It is argued in [21] that if one extends the forward Euler method, which is total variation diminishing (TVD) for a suitable discretization of a scalar conservation law, to a non-SSP higher-order method, then overshoots may occur at discontinuities.

An SSP-RK method can be written as a convex sum of explicit Euler steps with time step Δt,

$$\mathcal{E}\left(x^n\right) = x^n + \Delta t\mathcal{F}(x^n, t^n). \tag{3.54}$$

where $\mathcal{F}$ is the residual of the PDE to be solved. In this thesis we have

used two methods: The third-order three-stage SSP-RK method [62],

$$\begin{aligned} x^{(1)} &= \mathcal{E}\left(x^n\right), \\ x^{(2)} &= \frac{3}{4}x^n + \frac{1}{4}\mathcal{E}\left(x^{(1)}\right), \\ x^{n+1} &= \frac{1}{3}x^n + \frac{2}{3}\mathcal{E}\left(x^{(2)}\right), \end{aligned} \tag{3.55}$$

and the third-order four-stage SSP-RK method [33],

$$\begin{aligned} x^{(1)} &= \frac{1}{2}x^n + \frac{1}{2}\mathcal{E}\left(x^n\right), \\ x^{(2)} &= \frac{1}{2}x^{(1)} + \frac{1}{2}\mathcal{E}\left(x^{(1)}\right), \\ x^{(3)} &= \frac{2}{3}x^n + \frac{1}{6}x^{(2)} + \frac{1}{6}\mathcal{E}\left(x^{(2)}\right), \\ x^{n+1} &= \frac{1}{2}x^{(3)} + \frac{1}{2}\mathcal{E}\left(x^{(3)}\right). \end{aligned} \tag{3.56}$$

The semi-discretized velocity-extrapolation equation (3.5) and level-set reinitialization equation (3.7) can be written as systems of ordinary differential equations (ODEs) of the form

$$\frac{\mathrm{d}\boldsymbol{\psi}}{\mathrm{d}\tau} = \boldsymbol{F}(\boldsymbol{\psi}, \tau), \tag{3.57}$$

where $\boldsymbol{\psi}$ is a vector containing the discrete variables and $\boldsymbol{F}$ contains the spatially discretized terms. The explicit Euler step becomes

$$\mathcal{E}(\boldsymbol{\psi}^n) = \boldsymbol{\psi}^n + \Delta\tau \boldsymbol{F}(\boldsymbol{\psi}^n, \tau^n). \tag{3.58}$$

In this thesis, these equations are solved with the third-order four-stage SSP-RK method (3.56). The four-stage method is used because it is more accurate than the three-stage method. We wanted to make sure that the error is mainly dominated by the spatial discretization.

The semi-discretized Navier-Stokes equations (2.5) and (2.7) are solved together with the level-set advection equation (3.4) with the third-order three-stage SSP-RK method (3.55). Here an explicit Euler step $\mathcal{E}(\boldsymbol{u}^n, \varphi^n)$ consists of the following steps:

1. Calculate the curvature and the normal vectors from φ^n with one of the methods described in Section 3.7.

2. Calculate the intermediate velocity field, either $\boldsymbol{w}$ (3.12) or $\boldsymbol{u}^\star$ (3.17).

3. Solve the Poisson equation (3.13) or (3.20) for the pressure as described in Section 3.3.3.

4. Correct the velocity field with either (3.14) or (3.21).

5. Advect the level-set function φ^n with (3.4).

3.5. Time step restriction

We employ the Courant-Friedrich-Lewy (CFL) condition to allow adaptive time stepping and to enforce stability. The CFL condition that is used in this thesis is

$$\Delta t = \frac{C}{\frac{C_c+C_v}{2} + \sqrt{(C_c+C_v)^2 + 4C_g^2 + 4C_s^2}}, \tag{3.59}$$

where the CFL restriction $C < 1$, and C_c, C_v, C_s, and C_g represent the contributions from the convective term, the viscous stresses, the surface tension, and the gravity, respectively,

$$C_c = \frac{\max_{i,j} |u_{i,j}|}{\Delta x} + \frac{\max_{i,j} |v_{i,j}|}{\Delta y}, \tag{3.60}$$

$$C_v = 2 \max\left(\frac{\mu_1}{\rho_1}, \frac{\mu_2}{\rho_2}\right)\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right), \tag{3.61}$$

$$C_s = \sqrt{\frac{\sigma \max_{i,j} |\kappa_{i,j}|}{\max(\rho_1, \rho_2) \min(\Delta x^2, \Delta y^2)}}, \tag{3.62}$$

$$C_g = \sqrt{\frac{|g_x|}{\Delta x} + \frac{|g_y|}{\Delta y}}. \tag{3.63}$$

This CFL condition is discussed in more detail by Lervåg [38], and it is based on the condition by Kang et al. [30].

3.6. Axisymmetry

For axisymmetric flow, the governing equations (2.9) and (2.10) become

$$\frac{1}{r}\frac{\partial}{\partial r}(ru) + \frac{\partial v}{\partial z} = 0, \tag{3.64}$$

$$\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial r} + v\frac{\partial u}{\partial z}\right) = -\frac{\partial p}{\partial r} + \mu\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{\partial^2 u}{\partial z^2} - \frac{u}{r^2}\right) + \rho f_r + f_{s_r}, \tag{3.65}$$

and

$$\rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial r} + v\frac{\partial v}{\partial z}\right) = -\frac{\partial p}{\partial z} + \mu\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial v}{\partial r}\right) + \frac{\partial^2 v}{\partial z^2}\right) + \rho f_z + f_{s_z}. \tag{3.66}$$

Here u and v are the radial and axial velocity components, f_r and f_z are the radial and axial body-force components, and f_{s_r} and f_{s_z} are the radial and axial components of the singular surface force, respectively. The equations are solved as explained in the previous sections.

3.7. Discretization of the curvature and the normal vector

As stated in Section 3.1, the normal vector (3.8) and the curvature (3.9) can be calculated from the level-set function as

$$\boldsymbol{n} = \frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|},$$
$$\kappa = \boldsymbol{\nabla}\cdot\left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}\right).$$

They are typically discretized with the standard second-order central-difference scheme, cf. [30, 59, 75]. The normal vector is calculated at the cell edges, and the curvature is calculated at the grid nodes. The

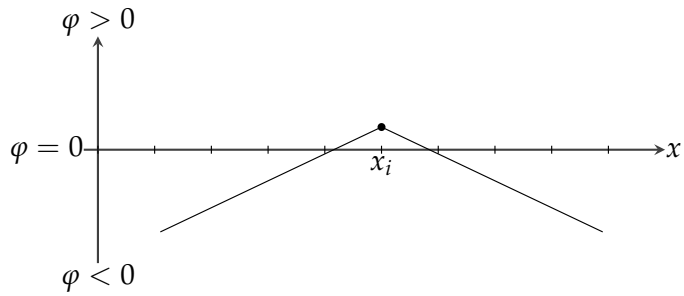


Figure 3.3.: A level-set function with a gradient that is discontinuous at x_i.

curvature is then interpolated to the interface where needed with linear interpolation, for instance with

$$\kappa_\Gamma = \frac{|\varphi_{i,j}|\kappa_{i+1,j} + |\varphi_{i+1,k}|\kappa_{i,j}}{|\varphi_{i,j}| + |\varphi_{i+1,j}|}. \tag{3.67}$$

If the level-set method is used to capture non-trivial geometries, then it will contain kink regions, that is, areas where the gradient of the level-set function is discontinuous. Figure 3.3 shows a simple example of such a kink region for a level-set function in a one-dimensional domain that captures two interfaces, one on each side of x_i. The kink at x_i may lead to large errors both for the curvature and the normal vector if one is not careful. Errors in the curvature lead to errors in the surface tension force and in the pressure, which in turn lead to errors in the interface evolution and in the two-phase flow. Errors in the normal vector affect both the calculation of the viscous jump condition and the advection of the interface. If the level-set method is used to study for example coalescence and breakup of drops, these errors may severely affect the simulations.

This problem was to our knowledge first described by Smereka [63], who increase the numerical smoothing in the curvature discretization to lessen the effect. Several non-smearing approaches have subsequently been developed. Macklin and Lowengrub [44] used the level-set method to study tumor growth, and they present a one-sided direction-difference scheme for the discretization of the normal vector and the curvature. Later, Macklin and Lowengrub [43] presented an improved geometry-

aware curvature discretization, where the curvature is calculated based on a local least-squares parametrisation of the interface.

A different approach to avoid the kinks was presented by Salac and Lu [56]. They used a level-set extraction technique, where an extraction algorithm was used to reconstruct separate level-set functions for each distinct *body*. The term *body* is used here to denote a subset of a given phase or fluid. For example, in the case of two drops of water colliding in air, the water drops would make two distinct bodies. One can also avoid the extraction algorithm altogether by use of multiple marker functions for different bodies, see for instance [13, 34]. Note, however, that the latter approach means that the different bodies will not coalesce unless explicit action is made, cf. [35]. Also, both of the approaches mentioned here fails to handle the problem of kinks from a single body. That is, there may still be kink areas due to deformed bodies, for instance bodies with thin filaments or tails, or bodies shaped like horse shoes.

In the following, we first present the direction-difference scheme [44]. We then describe the curve-fitting discretization method and the local level-set extraction method.

3.7.1. Direction-difference scheme

The direction-difference scheme (DDS) was introduced by Macklin and Lowengrub [44]. It uses a quality function to ensure that the difference stencils never cross any kink regions. The DDS is used in Papers A–C to calculate the normal vectors. However, in Paper C it is shown that the DDS does not always yield an accurate approximation of the normal vector.

The basic strategy is to use a combination of central differences and one-sided differences based on the values of a quality function,

$$Q(\boldsymbol{x}) = |1 - |\boldsymbol{\nabla}\varphi(\boldsymbol{x})|| \,. \tag{3.68}$$

The quality function is approximated with central differences, and is used to detect the areas where the level-set function differs from the signed-distance function. In the following, let $Q_{i,j} \equiv Q(\boldsymbol{x}_{i,j})$ and define a parameter $\eta > 0$. This threshold parameter is tuned such that the quality function will detect all the kinks.

The quality function is used to define a direction function,

$$\boldsymbol{D}(\boldsymbol{x}_{i,j}) = (D_x(\boldsymbol{x}_{i,j}), D_y(\boldsymbol{x}_{i,j})), \tag{3.69}$$

where

$$D_x(\boldsymbol{x}_{i,j}) = \begin{cases} -1 & \text{if } Q_{i-1,j} < \eta \text{ and } Q_{i+1,j} \geq \eta, \\ 1 & \text{if } Q_{i-1,j} \geq \eta \text{ and } Q_{i+1,j} < \eta, \\ 0 & \text{if } Q_{i-1,j} < \eta \text{ and } Q_{i,j} < \eta \text{ and } Q_{i+1,j} < \eta, \\ 0 & \text{if } Q_{i-1,j} \geq \eta \text{ and } Q_{i,j} \geq \eta \text{ and } Q_{i+1,j} \geq \eta, \\ 4 & \text{otherwise.} \end{cases} \tag{3.70}$$

$D_y(\boldsymbol{x}_{i,j})$ is defined in a similar manner. If $D_x(\boldsymbol{x}_{i,j}) + D_y(\boldsymbol{x}_{i,j}) > 2$, then $\boldsymbol{D}(\boldsymbol{x}_{i,j})$ is chosen as the vector normal to $\boldsymbol{\nabla}\varphi(\boldsymbol{x}_{i,j})$. It is normalized, and the sign is chosen such that it points in the direction of the best quality. See [44] for more details.

The DDS is then defined as

$$\partial_x f_{i,j} = \begin{cases} \frac{f_{i,j} - f_{i-1,j}}{\Delta x} & \text{if } D_x(x_i, y_j) = -1, \\ \frac{f_{i+1,j} - f_{i,j}}{\Delta x} & \text{if } D_x(x_i, y_j) = 1, \\ \frac{f_{i+1,j} - f_{i-1,j}}{2\Delta x} & \text{if } D_x(x_i, y_j) = 0, \end{cases} \tag{3.71}$$

and similarly for $\partial_y f_{i,j}$, where $f_{i,j}$ is a piecewise smooth function. The DDS is equivalent to using central differences in smooth areas and one-sided differences in areas close to the kinks.

3.7.2. Curve-fitting discretization method

The curve-fitting discretization method (CFDM) was first presented in [37] and is based on the method by Macklin and Lowengrub [43]. The main idea is to identify kink regions with the quality function (3.68), and to use a curve parametrisation of the closest interface to calculate the curvature in regions where the quality function is larger than the threshold parameter, η.

The CFDM applied to the curvature or the normal vector at the grid point $\boldsymbol{x}_{i,j}$ can be summarized as follows. See also Figure 3.4, which shows an example of the CFDM used at $\boldsymbol{x}_{i,j}$.

1. If the quality of the level-set function in the neighbourhood of $\boldsymbol{x}_{i,j}$ is good, that is

$$Q(\boldsymbol{x}_{n,m}) \leq \eta \ \forall (n,m) \in [i-1, i+1] \times [j-1, j+1],$$

then we use a standard discretization. Otherwise continue to the next step.

2. Locate the closest interface, Γ.

3. Find a set of points on the located interface, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \Gamma$.

4. Create a parametrisation $\gamma(s)$ of the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.

5. Use the parametrisation $\gamma(s)$ to calculate a local level-set function.

6. Use a standard discretization of the local level-set function to calculate the curvature or the normal vector.

3.7.3. Local level-set extraction method

The local level-set extraction (LOLEX) method is based on the method presented by Salac and Lu [56], here called the SLM. It was found that the latter method was insufficient, because it did not treat all the kink problems. The LOLEX method is therefore a further development of the SLM, in that it handles the kink regions in a more general manner.

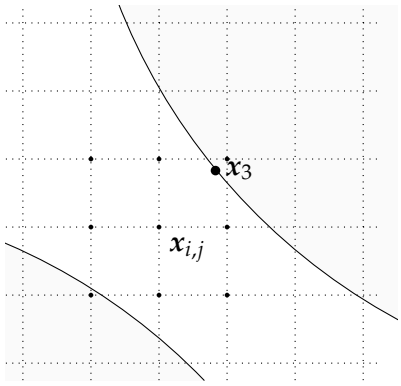
The LOLEX method applied to the curvature or the normal vector at a grid point $\boldsymbol{x}_{i,j}$ is summarized by the following algorithm. The algorithm is presented with 2D notation for clarity, and extends easily to 3D. See also Figure 3.5, which gives a simple example of the procedure.

1. If the quality in the neighbourhood of $\boldsymbol{x}_{i,j}$ is good, that is

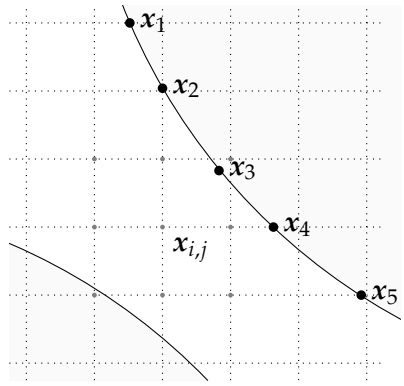
$$Q(\boldsymbol{x}_{n,m}) \leq \eta \ \forall (n,m) \in [i-1, i+1] \times [j-1, j+1],$$

then we use a standard discretization. Otherwise continue to the next step.

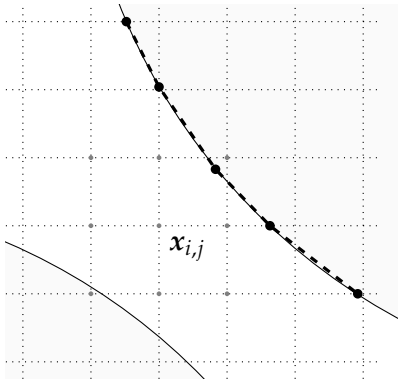
2. Copy a small, local square centred around $\boldsymbol{x}_{i,j}$ from the level-set function φ into a local array φ_{loc}.



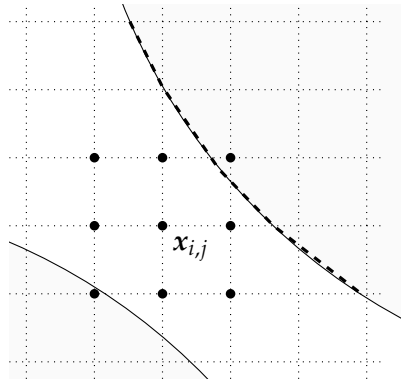
(a) First locate the closest interface, here represented with x_3.



(b) Then find a set of points along the closest interface.



(c) Construct a curve parametrisation from the points $x_1, \ldots, x_5$.



(d) Calculate a new local level-set function at the grid points around and including $x_{i,j}$.

Figure 3.4.: Example of the CFDM at a grid point $x_{i,j}$. First five points are found on the interface closest to $x_{i,j}$. Then a curve parametrisation (dashed line) is calculated, and the parametrisation is used to calculate a new local level-set function at the grid points surrounding and including $x_{i,j}$.

3. Identify and enumerate all the *bodies* in the local array φ_{loc}. A *body* is here defined as a set of neighbouring points where $\varphi_{\text{loc}} < 0$, see Figure 3.5. There will be $n \geq 0$ bodies in any given φ_{loc} array.

4. If no body is identified, that is, if $n = 0$, then use a standard discretization with the global level-set function φ. Otherwise continue to the next step.

5. For each body n, extract the relevant parts of φ_{loc} into an array φ^n_{loc}. If necessary, extrapolate values to ghost cells.

6. For each body, n, reinitialize φ^n_{loc}.

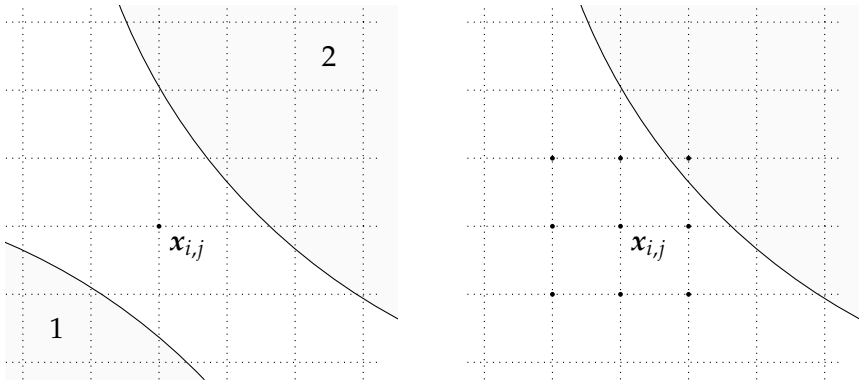
7. At this step, all the bodies in the local grid have their own local level-set functions that have been reinitialized to proper signed-distance functions. Due to the separation of the bodies, there are no longer any kinks.

8. Use the standard discretization of the curvature and the normal vector at the local level-set function that represents the body that is closest to $\boldsymbol{x}_{i,j}$.

3.8. Summary

In this chapter we have described the numerical methods that have been used to solve the Navier-Stokes equations for two-phase flow (2.9) and (2.10).

We first gave a brief introduction to the level-set method, which is used to capture the interface. We then presented the spatial and temporal discretization methods, including a brief overview of the projection methods that were used to decouple the pressure from the Navier-Stokes equations. In the final section, we presented the new methods for calculating the curvature and normal vector. These are the curve-fitting discretization scheme (CFDM) and the local level-set extraction (LOLEX) method.



(a) First enumerate the bodies. In this case there are $n = 2$ bodies. Then the bodies are extracted into separate level-set functions.

(b) Reinitialize the separated level-set functions, then use the function that represents the closest interface.

Figure 3.5.: An example of the LOLEX method at a grid point $\boldsymbol{x}_{i,j}$. First the bodies are identified and enumerated. Then they are extracted into separate level-set functions, which are reinitialized. Finally, the curvature or normal vector is calculated based on the level-set function for the closest body.

"If we want to solve a problem that we have never solved before, we must leave the door to the unknown ajar."

— Richard P. Feynman (1918–1988)

4

The diffuse-domain approach

In the previous chapters, we have considered the modelling of two-phase flows, and in particular methods for calculating the curvature and normal vector with the level-set method in a reliable manner. In this chapter, we consider a different problem of a more general nature: How to solve partial differential equations (PDEs) in complex domains. In particular, we consider an extension of a diffuse-domain method (DDM) by a high-order correction term that gives increased accuracy with respect to interface-width refinements.

We begin with a short introduction to the diffuse-domain approach. We then outline how it can be used to derive a DDM for the steady reaction-diffusion equation with Neumann boundary conditions. Next, we continue with a brief introduction to the method of matched asymptotic expansions. Finally, we introduce the high-order correction term derived in Paper E and show that the resulting DDM converge with second order in the diffuse-interface width to the original problem. The analysis also shows that the correction term is not necessary for second-order convergence.

4.1. Introduction

There exist several methods for solving PDEs in complex domains. Most of them have in common that they require tools or methods that are not frequently available in standard finite-element or finite-difference software packages. Examples of such methods include the immersed-interface method [39], the matched interface and boundary method [79], the extended and composite finite-element method [15], embedded boundary methods [29], cut-cell methods [27], and ghost-fluid methods [17]. A different approach, known as the fictitious domain method [19, 20] or the domain imbedding method [9], either augments the original system with equations for Lagrange multipliers to enforce the boundary conditions, or use the penalty method to enforce the boundary conditions weakly. For a more complete list of references, see Paper E.

The DDM is an alternative method for solving PDEs in complex domains. The main idea is to use an implicit representation of the boundary, where the sharp boundary is replaced by a diffuse layer. The PDEs are then reformulated on a larger, regular domain, and the boundary conditions are incorporated via source terms in the diffuse layer. When the thickness of the diffuse layer is reduced, these source terms tend towards singular source terms. The resulting PDEs can then be solved with the use of standard tools and methods.

The diffuse-domain approach was first introduced by Kockelkoren et al. [32] to study diffusion inside a cell with homogeneous Neumann boundary conditions at the cell boundary. It was later used by Li et al. [40] to develop a DDM for solving PDEs in complex evolving domains with Dirichlet, Neumann and Robin boundary conditions, which is hereafter called the DDM1. The DDM1 has been used by Teigen et al. [66], who modelled bulk-surface coupling of material quantities on a deformable interface. It was also used by Aland et al. [3] to simulate incompressible two-phase flows in complex domains in 2D and 3D, and by Teigen et al. [67] to study two-phase flows with soluble surfactants.

An analysis of the error behaviour of the diffuse-domain approach was done by Franz et al. [18] for a diffuse-domain approximation of an elliptic problem with Dirichlet boundary conditions. They considered the infinity norm of the difference of the approximated solution and the exact solution, and their analysis shows that the approximation quality

is of order one in the interface width.

In Paper E, we present the DDM2, which is an extension of the DDM1 by a high-order correction term. The DDM2 is derived for elliptic problems with Neumann and Robin boundary conditions, and it is shown to be asymptotically second-order accurate in the interface width. However, the analysis in Paper E is somewhat lacking in that it assumes that the DDM1 is only first-order accurate. In the following sections, we extend the analysis of Paper E and show that the DDM1 is also second-order accurate. The analysis is shown for the steady reaction-diffusion equation with Neumann boundary conditions, but the same technique also applies for the corresponding Robin problem.

4.2. The DDM for a Neumann problem

Consider the steady reaction-diffusion equation with Neumann boundary conditions,

$$\begin{aligned} \boldsymbol{\Delta} u - u &= f \qquad \text{in } D, \\ \boldsymbol{n} \cdot \boldsymbol{\nabla} u &= g \qquad \text{on } \partial D, \end{aligned} \tag{4.1}$$

where f and g are given. Let χ_D be the characteristic function of D,

$$\chi_D = \begin{cases} 1 & \text{if } x \in D, \\ 0 & \text{if } x \notin D. \end{cases} \tag{4.2}$$

The main idea with the diffuse-domain approach is to extend the original equation (4.1) into a larger and regular domain $\Omega \supset D$, as depicted in Figure 4.1. The extension can be written as

$$\boldsymbol{\nabla} \cdot (\chi_D \boldsymbol{\nabla} u) - \chi_D u + \text{BC} = \chi_D f, \tag{4.3}$$

where BC is a singular source term that represents the physical boundary condition on $\partial\Omega$.

The characteristic function is typically approximated by the phase-field function,

$$\chi_D \simeq \phi(\boldsymbol{x}, t) = \frac{1}{2}\left(1 - \tanh\left(\frac{3r(\boldsymbol{x}, t)}{\epsilon}\right)\right), \tag{4.4}$$

where ϵ is the interface width and $r(\boldsymbol{x}, t)$ is the signed-distance function with respect to the boundary ∂D, which is taken to be negative inside D.

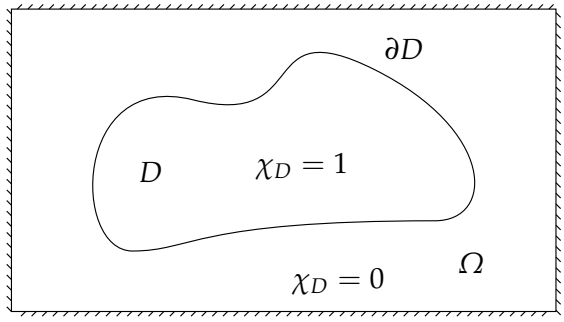


Figure 4.1.: A regular domain Ω that contains a complex domain D.

The main difficulty with the diffuse-domain approach is the derivation of approximations for the boundary condition term BC. Li et al. [40] give four approximations that are shown to converge asymptotically with first order in ϵ to the original equation when ϵ is decreased. In the following we consider the approximation

$$\text{BC} \simeq |\boldsymbol{\nabla}\phi| g. \tag{4.5}$$

If we combine the above approximations (4.4) and (4.5), we get a DDM1 equation for (4.1),

$$\boldsymbol{\nabla} \cdot (\phi \boldsymbol{\nabla} u) - \phi u + |\boldsymbol{\nabla}\phi| g = \phi f. \tag{4.6}$$

4.3. The method of matched asymptotic expansions

The following is a brief introduction to the method of matched asymptotic expansions, which is used to show that a given diffuse-domain approximation converges to the original problem when the interface width is decreased. More details can be found in Paper E and in [54].

Let u be some diffuse-domain variable. The asymptotic convergence of a given diffuse-domain approximation can be shown through expansions of the diffuse-domain variables in powers of the interface thickness ϵ in regions close to and far from the interface. For example, the expansions

of u are

$$u(\boldsymbol{x}) = \sum_{k=0}^{\infty} \epsilon^k u^{(k)}(\boldsymbol{x}), \tag{4.7}$$

$$\hat{u}(z, \boldsymbol{s}) = \sum_{k=0}^{\infty} \epsilon^k \hat{u}^{(k)}(z, \boldsymbol{s}), \tag{4.8}$$

where $u(\boldsymbol{x})$ and $\hat{u}(\boldsymbol{s}, z)$ denote the outer and inner expansions, respectively. Here z is a stretched variable,

$$z = \frac{r(\boldsymbol{x})}{\epsilon}, \tag{4.9}$$

where r is the signed distance from the point $\boldsymbol{x}$ to ∂D and is taken to be negative inside D. Further, z and $\boldsymbol{s}$ form a local coordinate system such that

$$\boldsymbol{x}(\boldsymbol{s}, z) = \boldsymbol{X}(\boldsymbol{s}) + \epsilon z \boldsymbol{n}(\boldsymbol{s}), \tag{4.10}$$

where $\boldsymbol{X}(\boldsymbol{s})$ is a parametrisation of the interface, $\boldsymbol{n}(\boldsymbol{s})$ is the interface normal vector, and z is a stretched variable.

When the inner and outer expansions are found, they are matched in a region where both solutions are valid and where $\epsilon z = \mathcal{O}(1)$, see Figure 4.2. The outer solution is then evaluated in the inner coordinates, which leads to a set of matching conditions that must hold when $\epsilon \to 0$. If we consider ϵ to be fixed and let $z \to \pm\infty$, we get the following asymptotic matching conditions:

$$\lim_{z \to \pm\infty} \hat{u}^{(0)}(z, \boldsymbol{s}) = u^{(0)}(\boldsymbol{s}), \tag{4.11}$$

and as $z \to \pm\infty$,

$$\hat{u}^{(1)}(z, \boldsymbol{s}) = u^{(1)}(\boldsymbol{s}) + z\boldsymbol{n} \cdot \boldsymbol{\nabla} u^{(0)}(\boldsymbol{s}) + o(1), \tag{4.12}$$

$$\begin{aligned} \hat{u}^{(2)}(z, \boldsymbol{s}) = {} & u^{(2)}(\boldsymbol{s}) + z\boldsymbol{n} \cdot \boldsymbol{\nabla} u^{(1)}(\boldsymbol{s}) \\ & + \frac{z^2}{2} (\boldsymbol{n} \cdot \boldsymbol{\nabla}) \boldsymbol{\nabla} u^{(0)}(\boldsymbol{s}) \cdot \boldsymbol{n} + o(1). \end{aligned} \tag{4.13}$$

We remark that the inner expansion is used to obtain the boundary condition on ∂D, and that the outer solution is used to obtain the sharp-interface equation inside the physical domain D.

To show that a given DDM approximation converges with second order, one must show that the order-one term of the outer solution of the DDM equation is zero in D. As an example, we consider the outer solution of the DDM1 equation (4.6), which is

$$\begin{aligned} \boldsymbol{\Delta} u^{(0)} - u^{(0)} &= f, \\ \boldsymbol{\Delta} u^{(1)} - u^{(1)} &= 0, \\ \boldsymbol{\Delta} u^{(k)} - u^{(k)} &= 0, \qquad k = 2, 3, \ldots. \end{aligned} \tag{4.14}$$

For the solution to be asymptotically second order, that is $u = u^{(0)} + \mathcal{O}\left(\epsilon^2\right)$, we must have that $u^{(0)}$ satisfies the original problem (4.1) and that $u^{(1)} = 0$. Thus the inner expansion must yield a boundary condition for $u^{(1)}$ to enforce $u^{(1)} = 0$.

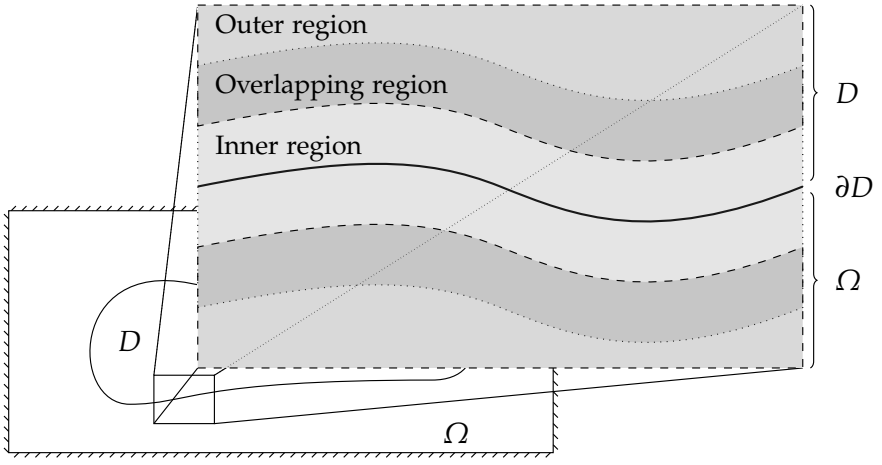


Figure 4.2.: A sketch of the regions used for the matched asymptotic expansions. The inner region is marked with a light gray color and the outer region with a slightly darker gray color. The overlapping region is marked with the darkest gray color.

4.4. Asymptotic analysis of the DDM1 and the DDM2

In Paper E we present the DDM2, which extends the DDM1 (4.6) with a high-order correction term,

$$\boldsymbol{\nabla} \cdot (\phi \boldsymbol{\nabla} u) - \phi u + |\boldsymbol{\nabla}\phi| g + r|\boldsymbol{\nabla}\phi| \left(f - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}} u + u\right) = \phi f. \quad (4.15)$$

Here $r|\boldsymbol{\nabla}\phi| \left(f - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}} u + u\right)$ is the correction term, κ is the curvature of the boundary ∂D, and $\boldsymbol{\Delta}_{\mathrm{s}} u$ is the surface Laplacian of u, which can be defined as

$$\boldsymbol{\Delta}_{\mathrm{s}} u \equiv (I - \boldsymbol{n} \otimes \boldsymbol{n}) \boldsymbol{\nabla} \cdot (I - \boldsymbol{n} \otimes \boldsymbol{n}) \boldsymbol{\nabla} u, \quad (4.16)$$

where I is the identity matrix and $\boldsymbol{n}$ is the normal vector. The curvature can be calculated from the phase-field function (4.4) as

$$\kappa = -\boldsymbol{\nabla} \cdot \frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}. \quad (4.17)$$

In the following, we use the method of matched asymptotic expansions to show that the DDM1 (4.6) and the DDM2 (4.15) are both second-order approximations of (4.1) in ϵ. First, it is easy to see that the outer expansions of both approximations are given by (4.14).

Next, we consider the inner expansion of the DDM2 (4.15), which is

$$\frac{1}{\epsilon^2} \left(\phi \hat{u}_z\right)_z + \frac{\kappa}{\epsilon} \phi \hat{u}_z + \phi \boldsymbol{\Delta}_{\mathrm{s}} \hat{u} - \phi \hat{u} - \frac{1}{\epsilon} \phi_z g - z \phi_z \left(\hat{u} + \hat{f} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}} \hat{u}\right) = \phi \hat{f}. \quad (4.18)$$

We expand $\hat{u}(z, \boldsymbol{s})$ in powers of ϵ and collect the lowest order terms,

$$\left(\phi \hat{u}_z^{(0)}\right)_z = 0. \quad (4.19)$$

If we integrate over all z, we get that $\hat{u}_z^{(0)} = 0$. The next order terms of (4.18) then give

$$\left(\phi \hat{u}_z^{(1)}\right)_z = \phi_z g, \quad (4.20)$$

and again we integrate, which gives that

$$\phi \hat{u}_z^{(1)} = \phi g + C \quad (4.21)$$

where the constant C must be zero, since $\lim_{z\to\infty}\phi(z) = 0$. Now consider the limit $z \to -\infty$ and use the matching condition (4.12) to get

$$\boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(0)} = g. \tag{4.22}$$

Thus $u^{(0)}$ satisfies the original problem at least to first order in ϵ. This shows that both DDM1 and DDM2 are first-order approximations of the sharp-interface problem.

To obtain the result for the next order, we need to apply the derivative of the matching condition (4.13),

$$\hat{u}_z^{(2)} = \boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(1)} + z(\boldsymbol{n}\cdot\boldsymbol{\nabla})\boldsymbol{\nabla}u^{(0)}\cdot\boldsymbol{n}. \tag{4.23}$$

Further, we use that $u^{(0)}$ satisfies

$$\boldsymbol{\Delta}u^{(0)} - u^{(0)} = f^{(0)}, \tag{4.24}$$

and that the Laplacian may be decomposed as

$$\boldsymbol{\Delta}u = (\boldsymbol{n}\cdot\boldsymbol{\nabla})\boldsymbol{\nabla}u\cdot\boldsymbol{n} + \kappa\boldsymbol{n}\cdot\boldsymbol{\nabla}u + \boldsymbol{\Delta}_{\mathrm{s}}u, \tag{4.25}$$

to get

$$(\boldsymbol{n}\cdot\boldsymbol{\nabla})\boldsymbol{\nabla}u^{(0)}\cdot\boldsymbol{n} = u^{(0)} + f^{(0)} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}}u^{(0)}. \tag{4.26}$$

Now insert (4.26) into (4.23) and use the matching condition (4.11) to obtain a modified matching condition,

$$\hat{u}_z^{(2)} - z\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)}\right) = \boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(1)}. \tag{4.27}$$

We are now ready to consider the zeroth order terms,

$$\begin{aligned}\left(\phi\hat{u}_z^{(2)}\right)_z + \phi\kappa\hat{u}_z^{(1)} + \phi\boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} - \phi\hat{u}^{(0)} \\ - z\phi_z\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)}\right) = \phi\hat{f}^{(0)}.\end{aligned} \tag{4.28}$$

The modified matching condition (4.27) motivates that we subtract and add the term

$$\left(z\phi\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)}\right)\right)_z$$

to (4.28), which gives

$$\begin{aligned}\Big(\phi\hat{u}_z^{(2)} - z\phi\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_s\hat{u}^{(0)}\right)\Big)_z \\ + \Big(z\phi\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_s\hat{u}^{(0)}\right)\Big)_z \\ + \phi\kappa\hat{u}_z^{(1)} + \phi\boldsymbol{\Delta}_s\hat{u}^{(0)} - \phi\hat{u}^{(0)} \\ - z\phi_z\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_s\hat{u}^{(0)}\right) = \phi\hat{f}^{(0)}. \quad (4.29)\end{aligned}$$

We expand the terms and use (4.21),

$$\begin{aligned}\Big(\phi\hat{u}_z^{(2)} - z\phi\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_s\hat{u}^{(0)}\right)\Big)_z \\ + \cancel{\phi\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_s\hat{u}^{(0)}\right)} \\ + z\phi\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_s\hat{u}^{(0)}\right)_z \\ + \cancel{\phi\kappa g + \phi\boldsymbol{\Delta}_s\hat{u}^{(0)} - \phi\hat{u}^{(0)}} = \cancel{\phi\hat{f}^{(0)}}, \quad (4.30)\end{aligned}$$

or

$$\begin{aligned}\Big(\phi\hat{u}_z^{(2)} - z\phi\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_s\hat{u}^{(0)}\right)\Big)_z \\ + z\phi\left(\cancel{\hat{u}_z^{(0)}} + \hat{f}_z^{(0)} - \cancel{(\kappa g)_z} - \cancel{\boldsymbol{\Delta}_s\hat{u}_z^{(0)}}\right) = 0. \quad (4.31)\end{aligned}$$

If we assume that $\hat{f}_z^{(0)}$, we get

$$\Big(\phi\hat{u}_z^{(2)} - z\phi\left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_s\hat{u}^{(0)}\right)\Big)_z = 0. \quad (4.32)$$

Finally, we integrate the left-hand side and take the limit,

$$\begin{aligned}\int_{-\infty}^{\infty} \left(\phi \hat{u}_z^{(2)} - z\phi \left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}} \hat{u}^{(0)} \right) \right)_z \mathrm{d}z \\ = \left[\phi \hat{u}_z^{(2)} - z\phi \left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}} \hat{u}^{(0)} \right) \right]_{-\infty}^{\infty} \\ = - \lim_{z \to -\infty} \left(\hat{u}_z^{(2)} - z \left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}} \hat{u}^{(0)} \right) \right) \\ = -\boldsymbol{n} \cdot \boldsymbol{\nabla} u^{(1)}, \end{aligned} \tag{4.33}$$

thus

$$\boldsymbol{n} \cdot \boldsymbol{\nabla} u^{(1)} = 0. \tag{4.34}$$

Combined with (4.14), this shows that $u^{(1)} = 0$, and so DDM2 converges asymptotically with second order to the original problem.

The analysis above also holds for DDM1, except instead of (4.32) we get

$$\begin{aligned}\left(\phi \hat{u}_z^{(2)} - z\phi \left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}} \hat{u}^{(0)} \right) \right)_z \\ = -z\phi_z \left(\hat{u}^{(0)} + \hat{f}^{(0)} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}} \hat{u}^{(0)} \right) = -z\phi_z D, \end{aligned} \tag{4.35}$$

where D is independent of z. Now we use that

$$\phi_z = -(3 \operatorname{sech}^2 3z)/2, \tag{4.36}$$

which follows from the definition of the phase-field function (4.4). We integrate the right-hand side, which gives

$$D \int_{-\infty}^{\infty} z\phi_z \, \mathrm{d}z = -D \frac{3}{2} \int_{-\infty}^{\infty} z \operatorname{sech}^2 3z \, \mathrm{d}z = 0. \tag{4.37}$$

Thus DDM1 is also second order in ϵ.

The difference between the DDM1 and the DDM2 is therefore that the correction term with the DDM2 directly cancels the term on the right-hand side in (4.35). This should give an increase of accuracy, but the convergence order remains the same.

The analysis for the corresponding Robin problem is essentially the same as the above. In Paper E, the DDM1 and DDM2 are compared

for several elliptic problems with both Neumann and Robin boundary conditions. The results of Paper E show that the correction term in the DDM2 leads to an increase of accuracy and that both DDM1 and DDM2 converge with second-order accuracy.

4.5. Summary

In this chapter, we have given a brief introduction to the diffuse-domain method (DDM). We considered a steady reaction-diffusion equation with Neumann boundary conditions (4.1) and two DDM approximations: DDM1 (4.6) and DDM2 (4.15). The DDM2 is an extension of DDM1 by a high-order correction term, and was first derived in Paper E.

Next, we gave an outline of the method of matched asymptotic expansions, and we used it to show that both the DDM1 and the DDM2 converged asymptotically with second order in the diffuse-interface width to the original equation (4.1). The analysis shows that the correction term in the DDM2 leads to a cancellation in the asymptotic expansions. By doing the integration, we see that this cancellation is not necessary for obtaining the second order convergence.

"Count what is countable, measure what is measurable, and what is not measurable, make measurable."

— Galileo Galilei (1564–1642)

5 Summary of contributions

This chapter presents summaries of the papers that constitute parts of this thesis. Each summary gives a brief discussion of the results of each paper, and the contribution of the author is highlighted for each paper.

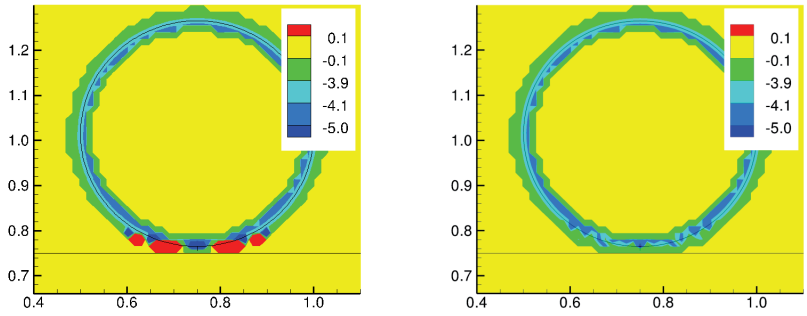
5.1. Paper A: Calculation of interface curvature with the level-set method

Karl Yngve Lervåg. Published in *MekIT'11 - 6th National Conference on Computational Mechanics, Trondheim*, 2011. ISBN: 978-82-519-2798-7.

In this paper I address a problem with the calculation of the interface curvature with the level-set method, cf. Section 3.7. The curvature can be calculated from the level-set function, ϕ, as

$$\kappa = \boldsymbol{\nabla} \cdot \boldsymbol{n} = \boldsymbol{\nabla} \cdot \frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}. \tag{5.1}$$

It is typically discretized by standard methods such as the second-order central-difference scheme (CD-2), and interpolated to the interface where needed [52, 30, 53]. However, the level-set function as a signed-distance function will tend to have kinks where its gradient is discontinuous. The



(a) CD-2 **(b)** CFDM

Figure 5.1.: A comparison of curvature calculations between standard discretization (CD-2) and the improved method (CFDM). The standard discretization leads to large errors in the curvatures in areas that are close to two interfaces.

standard methods may lead to large errors in the curvature close to these regions, which in turn may lead to errors in the surface tension force.

The main contribution of this paper is a new curve-fitting discretization method (CFDM) for the curvature (see Section 3.7.2). The method is based on the approach developed by Macklin and Lowengrub [43]. It differs in that it uses a cubic Hermite spline parametrisation of the interface, and that the curvature values are calculated on the grid and then interpolated to the interface as opposed to using a localised grid centred at the interface as in [43].

The CFDM is tested and compared with the CD-2 for two test cases, and it is shown to yield better results in both cases. Figure 5.1 shows one of these results, where the calculated curvature values are compared. In the example, a cylindrical drop impacts on a liquid film. The figure shows that the CD-2 leads to large errors in the curvature calculations in the kink regions, that is, the red and dark blue regions in Figure 5.1a near the liquid film. These errors are not present with the new method, cf. Figure 5.1b.

My contribution: I developed the method and implemented it into our in-house finite-difference code for two-phase flow based on the methods of Sections 3.2 to 3.5 and 3.7. I ran the numerical simulations. I wrote the paper and presented the work at the conference.

5.2. Paper B: Curvature calculations for the level-set method

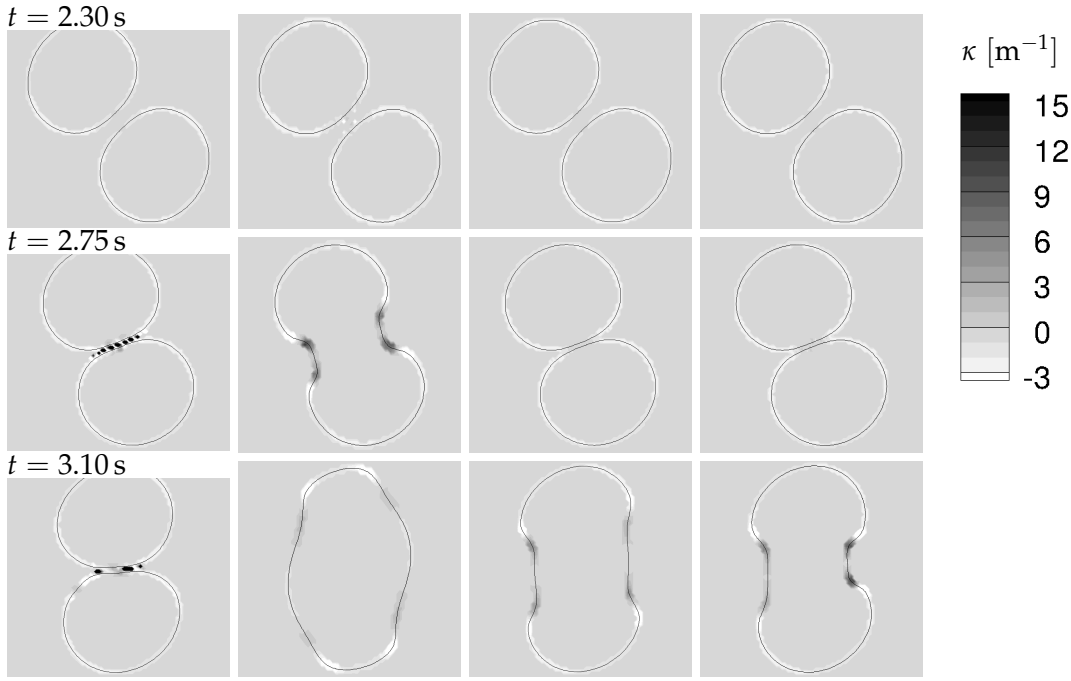
Karl Yngve Lervåg and Åsmund Ervik. Published in *ENUMATH 2011* proceedings volume, Springer, 2013. ISBN: 978-3642331336.

This paper is a continuation of Paper A. The main contribution in this paper is a comparison of different methods for calculating the curvature in a robust manner with the level-set method in the kink regions. In particular, the CFDM* that was presented in Paper A is compared with Macklin and Lowengrub's method (MLM) [43]. In addition, the method is compared with the second-order central-difference scheme (CD-2) and the more recent method presented by Salac and Lu [56], here called Salac and Lu's method (SLM).

The main result in the paper is shown in Figure 5.2, which shows a comparison of the methods for a case where two drops collide in a 2D shear flow. In particular, it shows snapshots of the evolution of the interfaces and the curvature at times $t = 2.30\,\mathrm{s}$, $t = 2.75\,\mathrm{s}$, and $t = 3.10\,\mathrm{s}$. The results show that all of the improved methods, that is MLM, CFDM, and SLM, handle the kink region in a more reliable manner than CD-2. The reason that the result with MLM differs from those with CFDM and SLM might be that it uses a localised grid centred at the interface to calculate the curvature, which means that it does not need to use interpolation of the curvature from the grid to the interface. Note that the difference is mainly that the MLM results in slightly earlier coalescence in the given case.

My contribution: I wrote the manuscript, implemented CFDM and MLM, and produced the results with CD-2, CFDM, and MLM. Åsmund Ervik implemented the SLM and ran the simulations that used the SLM.

*The method is called LM in the paper. Here CFDM is used, in order to be consistent with the rest of the thesis.



(a) CD-2 **(b)** MLM **(c)** CFDM **(d)** SLM

Figure 5.2.: A comparison between the different discretization schemes of the interface evolution and the curvature κ of drop collision in shear flow.

He also gave feedback on the manuscript. I presented the work at the conference.

5.3. Paper C: Calculation of the interface curvature and normal vector with the level-set method

Karl Yngve Lervåg, Bernhard Müller, and Svend Tollak Munkejord. Published in Computers and Fluids, volume 84 (2013), 218–230.

Paper A presented the curve-fitting discretization method (CFDM) for the calculation of the curvature with the level-set method. The method was designed to be robust in the calculation of the curvature in kink

regions, that is regions where the gradient of the level-set function is not smooth. This paper presents the details of the CFDM and applies it to the calculation of both the curvature and the normal vector.

In the paper we compare the CFDM with the second-order central-difference scheme (CD-2) for several test cases. In the first case, we consider the curvature calculations for a nontrivial geometry that includes some kink regions. This is a static test case with no flow, and the results show that the CD-2 leads to large errors for the curvature calculations in areas close to kink regions and that these errors are not present with the CFDM.

In the following two cases, we consider the collision of two drops in a 2D shear flow and in an axisymmetric flow. These cases show that the errors in the curvature calculations in the kink regions with the CD-2 lead to errors in the pressure that prevents coalescence. These errors are prevented with the CFDM. The curvature and the evolution of the interfaces for the axisymmetric case are shown in Figures 5.3 and 5.4. As in the earlier results of Papers A and B, the figures show that the errors in the curvature calculation with CD-2 prevent coalescence, in this case leading to a slower coalescence process.

In the final test case, we consider the calculation of the normal vector, and we compare the CD-2, the CFDM, and the direction-difference scheme (DDS) presented by Macklin and Lowengrub [44], cf. Section 3.7.1. The results show that both the CFDM and the DDS generally lead to good results, see Figure 5.5. Here the red and green vectors depict the DDS and CFDM results, respectively. The red vectors are plotted below the green vectors, and since the results agree well at most points, the red vector is often covered by its corresponding green vector. However, at the point in the middle between the drops, the DDS completely fails to calculate the normal vector. Here the CFDM still gives a reasonable result.

My contribution: I designed the new method, implemented it into our in-house finite-difference code, ran the simulations, and wrote the paper manuscript. The co-authors contributed with feedback on the manuscript and discussions of the results.

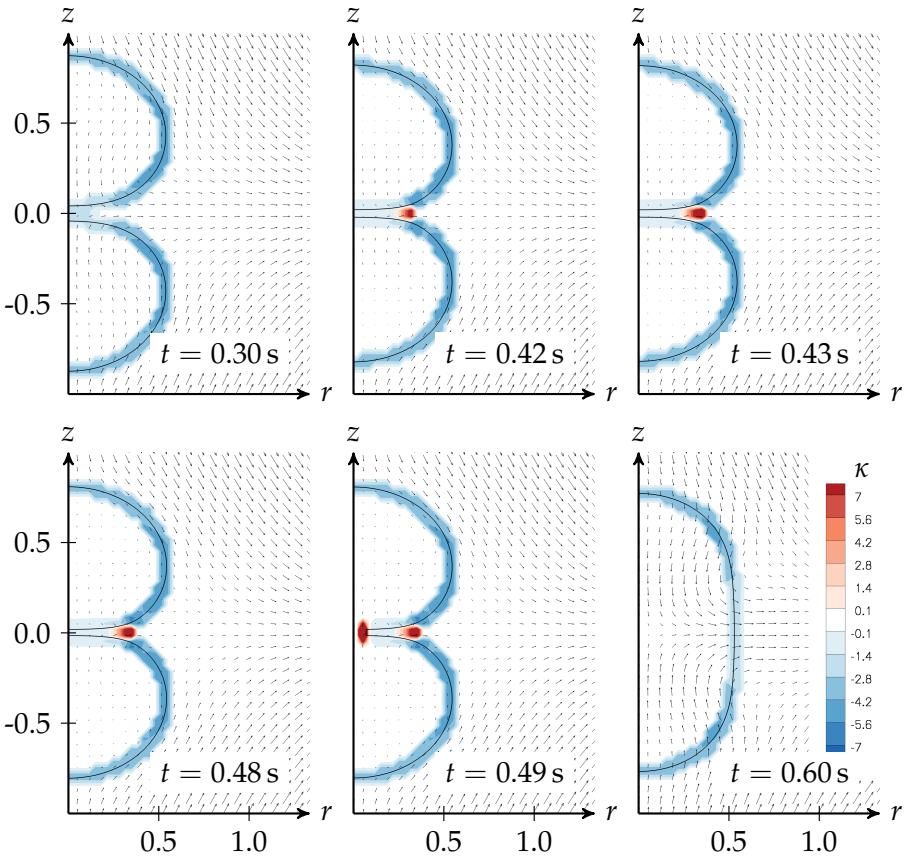


Figure 5.3.: Drop collision in axisymmetric flow calculated with the CD-2. The legend for the colour contours of the curvature κ is shown in the last image. The velocity vectors are displayed to show the evolution of the flow during the collision.

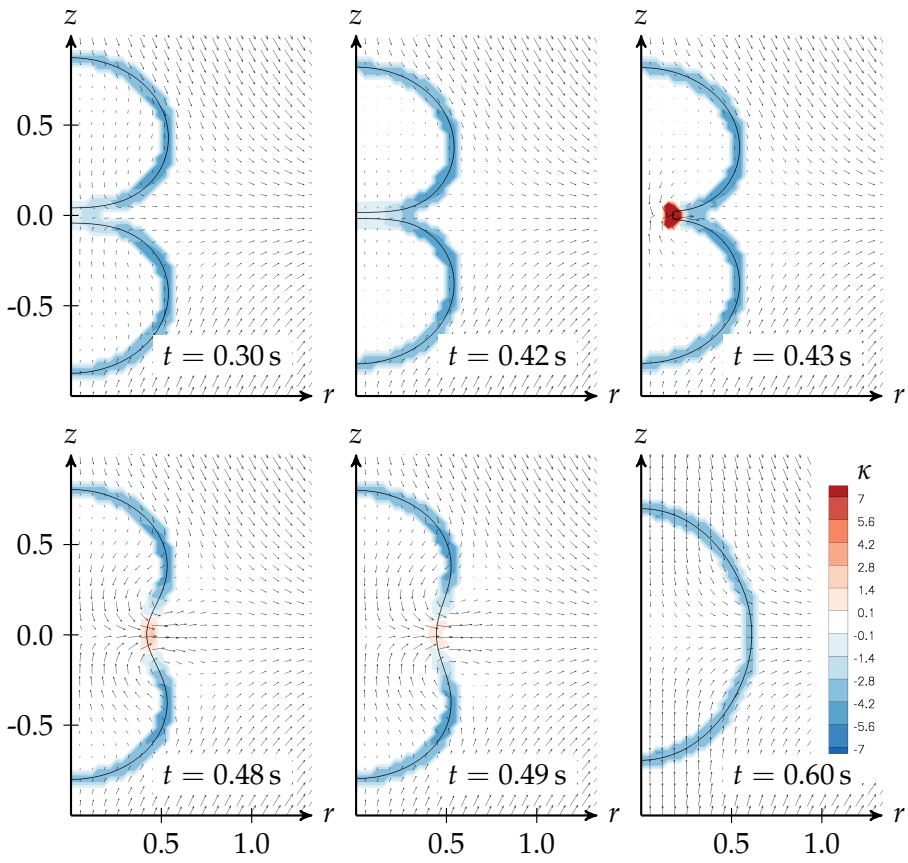


Figure 5.4.: Drop collision in axisymmetric flow calculated with the CFDM. The legend for the colour contours of the curvature κ is shown in the last image. The velocity vectors are displayed to show the evolution of the flow during the collision.

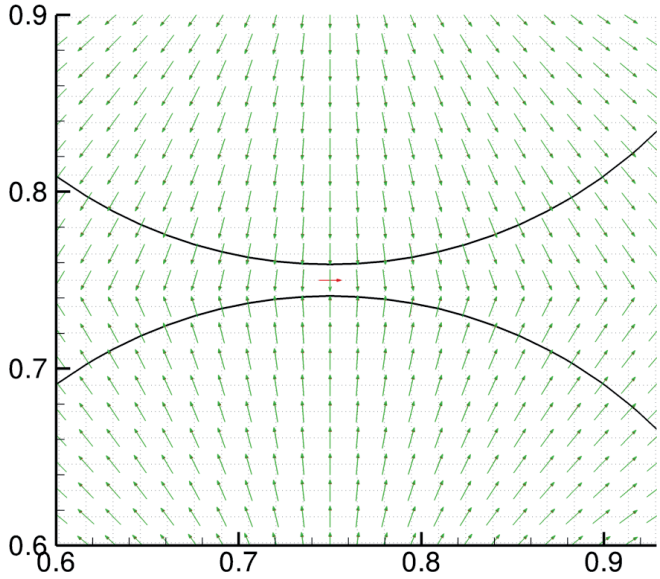


Figure 5.5.: A comparison of the DDS and the CFDM for calculating normal vectors. The thick black lines depict the interfaces, the green vectors are the results with the CFDM, and the red vectors are the results with the DDS. The red vectors are covered by the green vectors at most points, because the results agree well at those points.

5.4. Paper D: A robust method for calculating interface curvature and normal vectors using an extracted local level set

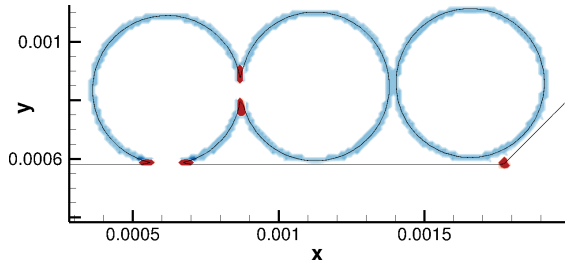
Åsmund Ervik, Karl Yngve Lervåg, and Svend Tollak Munkejord. Submitted to Journal of Computational Physics, 2013.

In this paper we present an alternative method for the calculation of the curvature and the normal vector of an interface with the level-set method in kink regions, hereafter called the local level-set extraction (LOLEX) method. The method is based on a method presented by Salac and Lu [56] (SLM), who handle the kink region by extracting different bodies of a domain into separate level-set functions. This procedure removes most of the kink regions, but it does not handle kink regions that are due to complex interfaces of single bodies. Our method extends the SLM by making it local. That is, we only consider the local area around the point for which we are calculating the curvature or the normal vector. This leads to a method that is more generally applicable, as shown in Figure 5.6. The figure shows a comparison between the LOLEX method, the SLM, and the standard central differences (CD-2). CD-2 leads to curvature spikes at the kink regions, as explained in Section 3.7. The SLM gives a better result for the kink regions around the rightmost disc. However, since the other two discs are connected to each other and to the film, they are considered to be the same body and are extracted into the same level-set function. Several kink regions are therefore not removed. Since the LOLEX method only considers the local area, as explained in Section 3.7.3, it is able to handle all the kink regions in a robust manner.

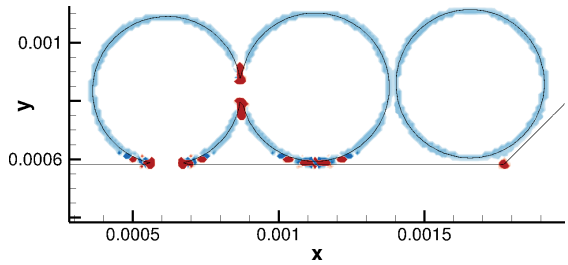
The LOLEX method has proven to be a good alternative to the CFDM presented in Paper A. Its main advantages are that it does not rely on complex algorithms as used in the CFDM or by Macklin and Lowengrub [43], and that the method easily extends to 3D as demonstrated in the paper in Section 4.4.

In the previous papers A–C, we used the DP1 projection method by Hansen [22]. In this paper we instead used the more standard Chorin projection method. These methods differ in that the DP1 assumes

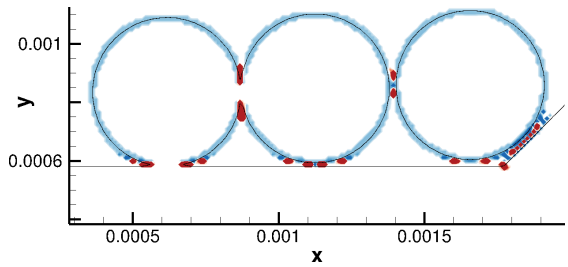
$$\boldsymbol{\nabla} \cdot \left(\frac{\partial \boldsymbol{u}}{\partial t} \right) = 0. \tag{5.2}$$



(a) LOLEX method



(b) SLM



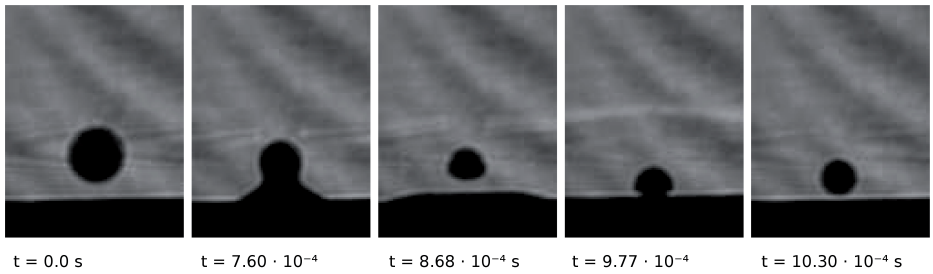
(c) CD-2

Figure 5.6.: Comparison of curvature calculation methods for three discs and a film with an angle at the right-hand side. The film is connected to the leftmost disc, which is connected to the middle disc. The rightmost disc is disjoint. The color indicates the curvature; white is zero, blue is negative and red is positive.

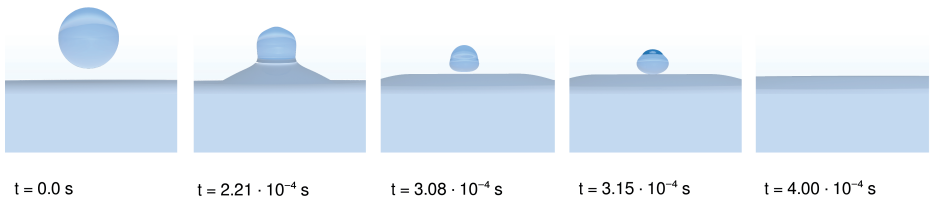
When compared with the Chorin method, this assumption becomes equivalent to assuming that $\nabla \cdot \boldsymbol{u}^n = 0$ in (3.20). That is, the DP1 assumes that the initial velocity field is divergence free. We have found that the DP1 works well in most cases, but that it is less robust than the Chorin method. In particular, the Chorin method is not equally affected by errors in the curvature calculations in kink regions. In other words, the difference between using a standard discretization and an improved discretization of the curvature is smaller with the Chorin method than with the DP1.

The LOLEX method is used for several test cases and compared with CD-2. The results indicate that even though we use the Chorin projection method, the LOLEX method outperforms CD-2 in all cases. Further, the results agree well with experiments, except for time instants, as shown in Figure 5.7. The exact reason why the time instants do not match is not known, but one reason may be that the initial condition of the numerical simulation did not match the corresponding state of the experiment.

When we study the drop-film collision processes, an important consequence of the error in the curvature calculation is a loss of kinetic energy. This can be seen in Figure 5.8, which compares the LOLEX method with CD-2 at two different stages of the collision process. The figure also compares two different frequencies of reinitialization of the level-set function: Every 7 time steps ((a) and (c)) and every single time step ((b) and (d)). The figure shows that the error in the curvature calculation with CD-2 leads to a shorter neck, as seen in Figure 5.8, (c) and (d). The error is larger for the higher frequency of reinitialization. The results indicate that CD-2 leads to a loss of kinetic energy during the collision process when compared with the LOLEX method. The LOLEX method is not significantly affected by the amount of reinitialization, and the kink region does not affect the curvature calculation, cf. Section 3.7.3. Thus the pressure field is more sensible, as seen in Figure 5.8 (a) and (b). Finally, we remark that some authors have noted [7] that the height of the neck and the dynamics of the capillary waves are important factors for the partial coalescence mechanism, which implies that the correct calculation of the curvature is important to capture the correct physical behavior.



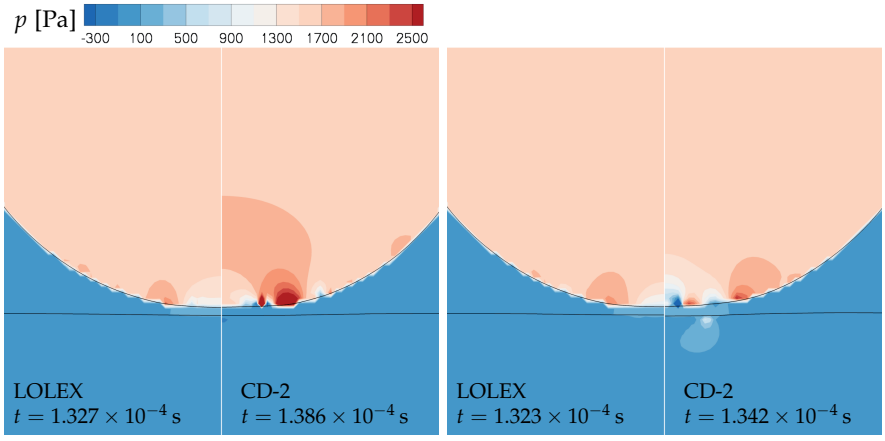
(a) Experimental result



(b) Simulation result

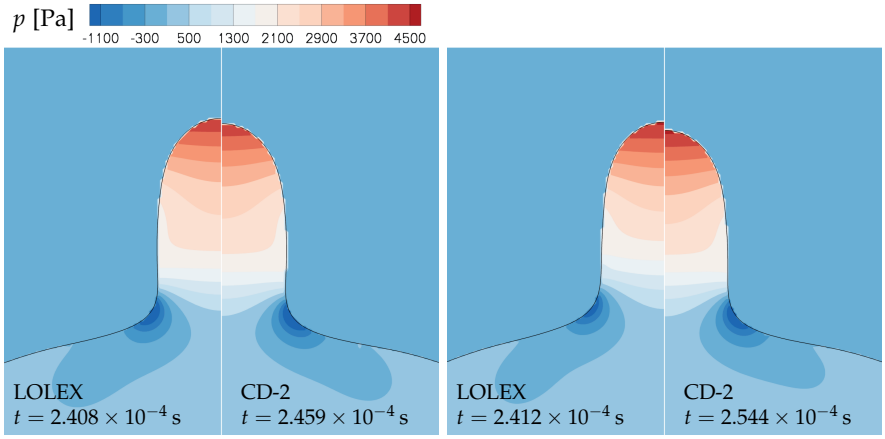
Figure 5.7.: Experimental results (top) and simulation results (bottom) for a 0.18 mm water drop falling through air and impacting a deep pool of water at 0.29 m/s. Figure (a) is reprinted from [77], Copyright (2011), with permission from Elsevier.

My contribution: The manuscript was written by Åsmund Ervik. The new method was developed and implemented into our in-house finite-difference code by Åsmund Ervik, and most of the numerical results are due to Åsmund Ervik. I contributed with discussions during the development of the new method, designed the test case in Section 4.1, ran simulations for Section 5.2, created some of the result figures, and gave feedback on the manuscript. I also assisted in some of the programming efforts for initializing the test cases in Chapter 5. Svend Tollak Munkejord contributed with discussions of the manuscript and some code testing.



(a) Reinitialization every 7 time steps

(b) Reinitialization every time step



(c) Reinitialization every 7 time steps

(d) Reinitialization every time step

Figure 5.8.: Water drop falling onto a pool, a comparison between the LOLEX method and CD-2. The interfaces are shown as solid black lines and the pressure field is shown as colored contours. (a) and (b): just before the interfaces merge. (c) and (d): when the neck reaches its highest position.

5.5. Paper E: Towards a second-order diffuse-domain approach for solving PDEs in complex geometries

Karl Yngve Lervåg and John Lowengrub. Submitted to Communications in Math. Sciences, 2013.

Li et al. [40] developed a diffuse-domain method for solving partial-differential equations (PDEs) inside complex, dynamic geometries with Dirichlet, Neumann, and Robin boundary conditions. This method is in the following referred to as DDM1. They use the diffuse-domain approach [32], where the geometry is represented implicitly and the sharp boundary is replaced by a diffuse layer with a fixed interface width. The original governing equations are then reformulated on a larger, regular domain and the boundary conditions are incorporated via singular source terms. The method of matched asymptotic expansions is used to show that the reformulated problem converges asymptotically to the original problem.

In the present paper, we use the method of matched asymptotic expansions to extend the DDM1 with include a high-order correction term in the diffuse formulation, cf. Section 4.4. The extension is derived for elliptic problems with Neumann and Robin boundary conditions, where the correction term is shown to yield an asymptotically second-order accurate approximation of the original problem. The new method is referred to as the DDM2.

The DDM1 and DDM2 are compared for a selection of test problems. The resulting equations were discretized by standard second-order central-difference schemes on uniform grids, and solved by a multigrid method. A red-black Gauss-Seidel type iterative method was used as a smoother, see [74].

In addition to the comparison of DDM1 and DDM2, we compared two different approximations of the boundary conditions. These correspond to different diffuse-interface surface delta functions, and for the Neumann boundary conditions they are

$$\mathrm{BC1} = |\boldsymbol{\nabla}\phi| g, \tag{5.3}$$

and

$$\mathrm{BC2} = \epsilon |\boldsymbol{\nabla}\phi|^2 g. \tag{5.4}$$

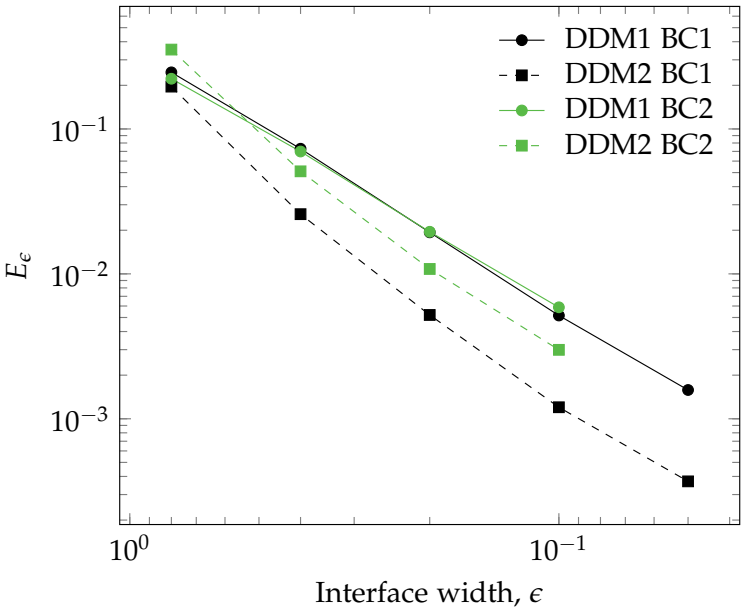


Figure 5.9.: Errors for the Neumann problem with respect to ϵ for Case 2, as labelled.

The approximations are similar for the Robin boundary condition, although here it was found that only BC1 resulted in a valid asymptotically second-order accurate DDM2.

Our results show that the global accuracy and convergence of DDM2 is better than DDM1, however both methods perform well and the global convergence rate is around two for each. The error was measured as the L_2 norm of the difference between an analytic solution and the solution u_ϵ for a given interface width, ϵ.

Figures 5.9 and 5.10 show two of the results, the first one with the Neumann boundary conditions and the second with the Robin boundary condition. The results indicate that DDM2 performs slightly better than DDM1. They also show that the approximation BC1 gives more accurate results than BC2. In particular, we found that with BC2 we needed much finer grids to obtain convergence for a given ϵ. For the smallest values of ϵ we were not able to refine the grids enough to obtain valid results.

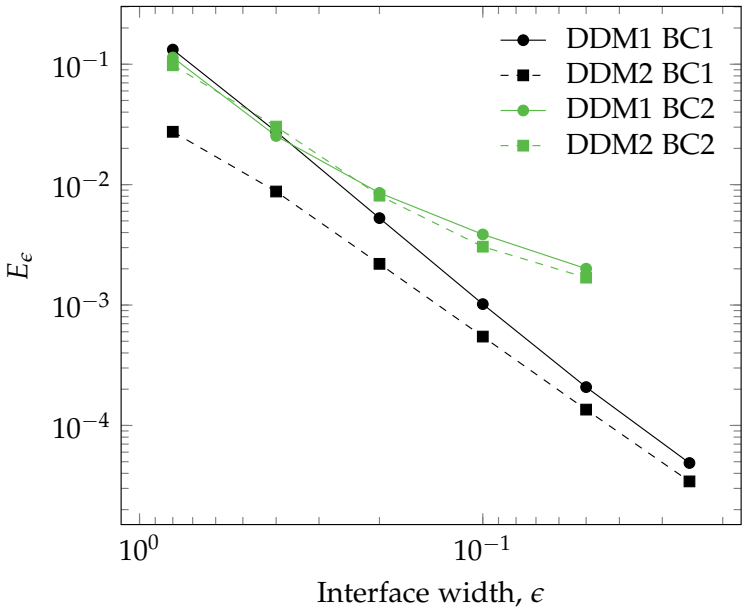


Figure 5.10.: Errors for the Robin problem with respect to ϵ for Case 2, as labelled.

My contribution: I took a leading role in analysing the equations, designing the numerical algorithms, selecting the test cases, and performing the numerical simulations. I wrote the manuscript. John Lowengrub contributed with important insights in the analysis, feedback on the manuscript, and discussions of the results.

"What is research but a blind date with knowledge?"

— Will Harvey (1967)

6 Conclusions and outlook

This thesis has considered two different problems: The discretization of interface curvature and normal vectors with the level-set method and a diffuse-domain approach for solving partial-differential equations (PDEs) in complex domains. In the following, some general concluding remarks and recommendations for further work are given.

Conclusions

The first part of the thesis considered the modelling of incompressible two-phase flow. The main motivation was to study two-phase flow phenomena that are relevant for compact heat exchangers, such as drop-drop and drop-film collisions. In particular, the thesis addressed a challenge with the calculation of interface curvature and normal vectors with the level-set method. Two methods were presented to handle the discretization in the kink regions: A curve-fitting discretization method (CFDM) and a local level-set extraction (LOLEX) method. Both methods were shown to be robust in the kink regions, where the standard central-difference scheme (CD-2) fails. Of these methods, the LOLEX method is the preferred method, because it relies on a less complicated algorithm that easily extends to 3D.

CD-2 and the LOLEX method were used for simulations of drop-film collisions that were compared with experiments. The results showed that CD-2 leads to errors in the curvature that cause unphysical pressure spikes during the coalescence. The errors are shown to lead to a dissipation of kinetic energy during the collision and to a slower coalescence process. The LOLEX method was shown to prevent these unphysical pressure spikes, and to produce to more accurate results.

The second part of the thesis considered the diffuse-domain approach for solving PDEs in complex domains. The main contribution, presented in Paper E, was the derivation of an asymptotically second-order diffuse-domain method (DDM2) for solving elliptic problems in complex geometries with Neumann and Robin boundary conditions. The new method is an extension by a high-order correction term of the method presented in [40], here called DDM1. The DDM1 and DDM2 were compared, and the results indicated that the DDM2 was slightly better than the DDM1.

The thesis has expanded on the results of Paper E with a new asymptotic analysis that shows that DDM1 is in fact also asymptotically second order. This analysis helps to explain why the performance of DDM2 presented in Paper E is only slightly better than that of DDM1. As such, the new analysis leads to a better understanding of the DDM1 and the DDM2.

Outlook

The following gives an outline of some possibilities for future work.

- A natural continuation of this work is to perform more in-depth comparisons of simulations with experiments for the drop-film collision phenomenon that was started in Paper D. In addition, it would be interesting to study other two-phase flow phenomena that are relevant for heat-exchanger processes, for instance drop-drop collisions or flow across tube bundles. The latter requires the treatment of more complex boundaries, which can be handled either with the diffuse-domain method or other methods from the literature.

- Paper C gave a short comparison of the standard discretization method (CD-2), the direction-difference scheme (DDS), and the curve-fitting discretization scheme (CFDM) applied to the calculation of the normal vectors. The results showed that the CD-2 leads to inaccurate results in the kink regions. The DDS gives robust and accurate results in most cases, but an example is given where only the CFDM yields an accurate result. However, the impact of inaccurate calculations of the normal vector should be investigated in more detail. The normal vector is used both for the solution of the level-set equations (3.4), (3.5), and (3.7), and for the calculation of the interface jumps (2.23) and (2.32), so one can expect that large errors in the calculation of the normal vector may lead to large errors in the numerical solution. This warrants a further study.

- Mass and heat transfer are obviously an important part of the heat-exchanger processes. The models that have been used in this thesis should therefore be expanded with additional models for heat transfer and mass transfer to enable the simulation of more relevant phenomena.

- The DDM2 was only derived for problems with Robin and Neumann boundary conditions. If possible, it should be extended to also work with Dirichlet boundary conditions.

- As explained in Paper E, we found that we were unable to solve the discrete system of equations when the surface Laplacian part of the correction term for the DDM2 was included. A further investigation of this problem should be done, and stable numerical methods to solve the full DDM2 equations should be developed.

- The diffuse-domain approach is a promising method for solving problems in complex and confined geometries with standard tools and methods. Aland et al. [3] provide a diffuse-domain formulation of the Navier-Stokes Cahn-Hilliard equations for incompressible two-phase flow and use it to compute two-phase flow in both complex and confined geometries. However, they do not provide an asymptotic analysis of the reformulated equations to show that the system converges. Such an analysis would be interesting, in

particular to verify that the equations do converge to the original problem when the interface width is decreased.

- Finally, it would be interesting to develop a second-order diffuse-domain formulation for the incompressible Navier-Stokes equations. A starting point would be to use the results and techniques from this thesis and Paper E.

Bibliography

[1] Adalsteinsson, D. and Sethian, J. A. "A Fast Level Set Method for Propagating Interfaces". In: *Journal of Computational Physics* 118 (1995), pp. 269–277.

[2] Adalsteinsson, D. and Sethian, J. A. "The fast construction of extension velocities in level-set methods." In: *Journal of Computational Physics* 148 (1999), pp. 2–22.

[3] Aland, S., Lowengrub, J., and Voigt, A. "Two-phase flow in complex geometries: A diffuse domain approach." In: *Computer Modeling in Engineering & Sciences* 57.1 (2010), pp. 77–106.

[4] Anderson, D. M., McFadden, G. B., and Wheeler, A. A. "Diffuse-interface methods in fluid mechanics". In: *Annual Review of Fluid Mechanics* 30.1 (1998), pp. 139–165.

[5] Aris, R. *Vectors, Tensors and the Basic Equations of Fluid Mechanics*. New York: Dover publications, 1989. ISBN: 0-486-66110-5.

[6] Balay, S., Brown, J., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H. *PETSc Web page*. Web page. 2013. URL: `http://www.mcs.anl.gov/petsc`.

[7] Blanchette, F. and Bigioni, T. P. "Partial coalescence of drops at liquid interfaces". In: *Nature Physics* 2 (2006), pp. 254–257.

[8] Brackbill, J. U., Kothe, D. B., and Zemach, C. "A continuum method for modeling surface tension". In: *Journal of Computational Physics* 100 (1992), pp. 335–354.

[9] Buzbee, B. L., Dorr, F. W., George, J. A., and Golub, G. H. "The direct solution of the discrete poisson equation on irregular regions". In: *SAIM J. Numer. Anal.* 8 (1971), pp. 722–736.

[10] Cahn, J. W. and Hilliard, J. E. "Free energy of a nonuniform system. I. Interfacial free energy." In: *Journal of Chemical Physics* 28 (1957), pp. 258–267.

[11] Chorin, A. J. and Marsden, J. E. *A Mathematical Introduction to Fluid Mechanics*. New York: Springer, 2000.

[12] Chorin, A. J. "Numerical solution of the Navier-Stokes equations". In: *Mathematics of computation* 22.104 (1968), pp. 745–762.

[13] Coyajee, E. and Boersma, B. J. "Numerical simulation of drop impact on a liquid-liquid interface with a multiple marker front-capturing method". In: *Journal of Computational Physics* 228 (20=9), pp. 4444–4467. DOI: 10.1016/j.jcp.2009.03.014.

[14] Cristini, V. and Tan, Y. "Theory and numerical simulation of droplet dynamics in complex flows — a review". In: *Lab on a Chip* 4 (2004), pp. 257–264.

[15] Dolbow, J. and Harari, I. "An efficient finite element method for embedded interface problems". In: *Int. J. Numer. Meth. Eng.* 78 (2009), pp. 229–252.

[16] Enright, D., Fedkiw, R., Ferziger, J., and Mitchell, I. "A hybrid particle level set method for improved interface capturing". In: *Journal of Computational Physics* 183 (2002), pp. 83–116.

[17] Fedkiw, R. P., Aslam, T., Merriman, B., and Osher, S. "A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the Ghost Fluid Method)". In: *Journal of Computational Physics* 152.2 (1999), pp. 457–492. DOI: 10.1006/jcph.1999.6236.

[18] Franz, S., Gärtner, R., Roos, H.-G., and Voigt, A. "A Note on the Convergence Analysis of a Diffuse-domain Approach". In: *Computational Methods in Applied Mathematics* 12.2 (2012), pp. 153–167.

[19] Glowinski, R., Pan, T. W., and Periaux, J. "A fictitious domain method for external incompressible viscous-flow modeled by Navier-Stokes equations". In: *Comput. Meth. Appl. Mech. Engin.* 112 (1994), pp. 133–148.

[20] Glowinski, R., Pan, T. W., Wells, R. O., and Zhou, X. D. "Wavelet and finite element solutions for the Neumann problem using fictitious domains". In: *J. Comput. Phys.* 126 (1996), pp. 40–51.

[21] Gottlieb, S., Shu, C. W., and Tadmor, E. "Strong stability-preserving high-order time discretization methods". In: *SIAM Review* 43 (2001), pp. 89–112.

[22] Hansen, E. B. "Numerical Simulation of Droplet Dynamics in the Presence of an Electric Field". ISBN 82-471-7318-2. Doctoral thesis. Trondheim: Norwegian University of Science, Technology, Department of Energy, and Process Engineering, Nov. 2005.

[23] Harlow, F. H. and Welch, J. E. "Numerical Calculation of Time Dependent Viscous Incompressible Flow of Fluid with Free Surface". In: *Physics of Fluids* 8 (1965), pp. 2182–2189. DOI: 10.1063/1.1761178.

[24] Hesselgreaves, J. E. *Compact Heat Exchangers: Selection, Design and Operation*. Gulf Professional Publishing, 2001. ISBN: 0-08-042839-8.

[25] Iserles, A. *A First Course in the Numerical Analysis of Differential Equations*. second. Cambridge: Cambridge University Press, 2009. ISBN: 978-0-521-73490-5.

[26] Jacqmin, D. "Calculation of two-phase Navier-Stokes flows using phase-field modeling". In: *Journal of Computational Physics* 155 (1999), pp. 96–127.

[27] Ji, H., Lien, F.-S., and Yee, E. "An efficient second-order accurate cut-cell method for solving the variable coefficient Poisson equation with jump conditions on irregular domains". In: *Int. J. Numer. Meth. Fluids* 52 (2006), pp. 723–748.

[28] Jiang, G.-S. and Shu, C.-W. "Efficient Implementation of Weighted ENO Schemes". In: *Journal of Computational Physics* 126 (1996), pp. 202–228.

[29] Johansen, H. and Colella, P. "A Cartesian grid embedded boundary method for Poisson's equation on irregular domains". In: *Journal of Computational Physics* 147 (1998), pp. 60–85.

[30] Kang, M., Fedkiw, R. P., and Liu, X.-D. "A boundary condition capturing method for multiphase incompressible flow". In: *Journal of Scientific Computing* 15.3 (2000), pp. 323–360.

[31] Ketcheson, D. I. and Robinson, A. C. "On the practical importance of the SSP property for Runge-Kutta time integrators for some common Godunov-type schemes". In: *International Journal for Numerical Methods in Fluids* 48 (2005), pp. 271–303.

[32] Kockelkoren, J., Levine, H., and Rappel, W. J. "Computational approach for modeling intra- and extracellular dynamics". In: *Phys. Rev.* E 68 (2003), p. 037702.

[33] Kraaijevanger, J. F. B. M. "Contractivity of Runge-Kutta methods". In: *BIT Numerical Mathematics* 31.3 (1991), pp. 482–528.

[34] Kwakkel, M., Breugem, W.-P., and Boersma, B. J. "An efficient multiple marker front-capturing method for two-phase flows". In: *Computers and Fluids* 63 (2012), pp. 47–56. DOI: 10.1016/j.compfluid.2012.04.004.

[35] Kwakkel, M., Breugem, W.-P., and Boersma, B. J. "Extension of a CLSVOF method for droplet-laden flows with a coalescence/breakup model". In: *Journal of Computational Physics* (2013). In press. DOI: 10.1016/j.jcp.2013.07.005.

[36] Lafaurie, B., Nardone, C., Scardovelli, R., Zaleski, S., and Zanetti, G. "Modeling merger and fragmentation in multiphase flows with SURFER". In: *Journal of Computational Physics* 113 (1994), pp. 34–47.

[37] Lervåg, K. Y. "Calculation of interface curvature with the level-set method". In: *Sixth National Conference on Computational Mechanics MekIT'11 (Trondheim, Norway)*. May 23-24 May 2011. ISBN: 978-82-519-2798-7.

[38] Lervåg, K. Y. "Simulation of two-phase flow with varying surface tension". MA thesis. Trondheim: Norwegian University of Science and Technology, Department of Mathematical Sciences, June 2008.

[39] LeVeque, R. J. and Li, Z. "The immersed interface method for elliptic equations with discontinuous coefficients and singular sources." In: *SIAM Journal of Numerical Analysis* 31 (1994), pp. 1019–1044.

[40] Li, X., Lowengrub, J., Rätz, A., and Voigt, A. "Solving PDEs in Complex Geometries: A Diffuse Domain Approach". In: *Communications in Mathematical Sciences* 7.1 (2009), pp. 81–107.

[41] Liu, X.-D., Fedkiw, R. P., and Kang, M. "A Boundary Condition Capturing Method for Poisson's Equation on Irregular Domains". In: *Journal of Computational Physics* 160 (2000), pp. 151–178.

[42] Lowengrub, J. and Truskinovsky, L. "Quasi-incompressible Cahn-Hilliard fluids and topological transitions". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454 (1998), pp. 2617–2654.

[43] Macklin, P. and Lowengrub, J. "An improved geometry-aware curvature discretziation for level set methods: Application to tumor growth". In: *Journal of Computational Physics* 215 (2006), pp. 392–401.

[44] Macklin, P. and Lowengrub, J. "Evolving interfaces via gradients of geometry-dependent interior Poisson problems: Application to tumor growth". In: *Journal of Computational Physics* 203 (2005), pp. 191–220.

[45] Macklin, P. and Lowengrub, J. S. "A New Ghost Cell/Level Set Method for Moving Boundary Problems: Application to Tumor Growth". In: *Journal of Scientific Computing* 35 (2008), pp. 266–299.

[46] Mallet, V., Keyes, D., and Fendell, F. "Modeling wildland fire propagation with level set methods". In: *Computers and Mathematics with Applications* 57.7 (2009), pp. 1089–1101. ISSN: 0898-1221.

[47] Melicher, V., Cimrak, I., and Keer, R. V. "Level set method for optimal shape design of MRAM core. Micromagnetic approach". In: *Physica B: Condensed Matter* 403 (2008), pp. 308–311. ISSN: 0921-4526.

[48] Nobari, M. R., Jan, Y., and Tryggvason, G. "Head-on collision of drops — A numerical investigation". In: *Physics of Fluids* 8 (1996), pp. 29–42.

[49] Olsen, R., Maråk, K. A., Zhao, H., and Munkejord, S. T. "An experimental and computational strategy for an increased understanding of two-phase flow of natural gas". In: *HEFAT 2007, Fifth International Conference on Heat Transfer, Fluid Mechanics and Thermodynamics*. Sun City, South Africa, July 2007.

[50] Olsson, E. and Kreiss, G. "A conservative level set method for two phase flow". In: *Journal of Computational Physics* 210 (2005), pp. 225–246.

[51] Olsson, E., Kreiss, G., and Zahedi, S. "A conservative level set method for two phase flow II". In: *Journal of Computational Physics* 225 (2007), pp. 785–807.

[52] Osher, S. and Sethian, J. A. "Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations". In: *Journal of Computational Physics* 79 (1988), pp. 12–49.

[53] Osher, S. and Fedkiw, R. P. *The Level-Set Method and Dynamic Implicit Surfaces*. New York: Springer, 2003. ISBN: 0387954821.

[54] Pego, R. L. "Front Migration in the nonlinear Cahn-Hilliard equation". In: *Proceedings of the Royal Society A* 422 (Apr. 1988), pp. 261–278. DOI: 10.1098/rspa.1989.0027.

[55] Saad, Y. *Iterative Methods for Sparse Linear Systems*. second. SIAM, 2003. ISBN: 0-89871-534-2.

[56] Salac, D. and Lu, W. "A Local Semi-Implicit Level-Set Method for Interface Motion". In: *Journal of Scientific Computing* 35 (2008), pp. 330–349.

[57] Scardovelli, R. and Zaleski, S. "Direct numerical simulation of free-surface and interfacial flow". In: *Annual Review of Fluid Mechanics* 31.1 (1999), pp. 567–603.

[58] Sethian, J. A. "Curvature and the evolution of fronts". In: *Communications in Mathematical Physics* 101.4 (1985), pp. 487–499.

[59] Sethian, J. A. and Smereka, P. "Level Set Methods for Fluid Interfaces". In: *Annual Review of Fluid Mechanics* 35 (2003), pp. 341–372.

[60] *Shell's Prelude FLNG Project, Browse Basin, Australia*. http://www.offshore-technology.com/projects/shell-project/. 2012.

[61] Shin, S. and Juric, D. "Modelling three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity". In: *Journal of Computational Physics* 180 (2002), pp. 427–470.

[62] Shu, C.-W. and Osher, S. "Efficient Implementation of Essentially Non-oscillatory Shock-Capturing Schemes". In: *Journal of Computational Physics* 77 (1988), pp. 439–471.

[63] Smereka, P. "Semi-Implicit Level Set Methods for Curvature and Surface Diffusion Motion". In: *Journal of Scientific Computing* 19 (2003), pp. 439–456. ISSN: 0885-7474.

[64] Sussman, M., Smereka, P., and Osher, S. "A level set approach for computing solutions to incompressible two-phase flow". In: *Journal of Computational Physics* 114 (1994), pp. 146–159.

[65] Sussman, M. and Puckett, E. G. "A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows". In: *Journal of Computational Physics* 162 (2000), pp. 301–337.

[66] Teigen, K. E., Li, X., Lowengrub, J., Wang, F., and Voigt, A. "A diffuse-interface approach for modeling transport, diffusion and adsorption/desorption of material quantities on a deformable interface". In: *Communications in Mathematical Sciences* 7.4 (Aug. 2009), pp. 1009–1037.

[67] Teigen, K. E., Song, P., Lowengrub, J., and Voigt, A. "A diffuse-interface method for two-phase flows with soluble surfactants". In: *Journal of Computational Physics* 230 (2011), pp. 375–393. DOI: 10.1016/j.jcp.2010.09.020.

[68] *The Prelude FLNG project*. http://www.shell.com.au/aboutshell/who-we-are/shell-au/operations/upstream/prelude.html. 2012.

[69] Tryggvason, G., Bunner, B., Esmaeeli, A., Juric, D., Al-Rawahi, N., Tauber, W., Han, J., Nas, S., and Jan, Y. J. "A front-tracking method for the computations of multiphase flow". In: *Journal of Computational Physics* 169.2 (2001), pp. 708–759.

[70] Tryggvason, G., Scrdovelli, R., and Zaleski, S. *Direct Numerical Simulations of Gas-Liquid Multiphase Flows*. Cambridge: Cambridge University Press, 2011.

[71] Tryggvason, G., Thomas, S., Lu, J., and Aboulhasanzadeh, B. "Multiscale issues in DNS of multiphase flows". In: *Acta Mathematica Scientia* 30 (2010), pp. 551–562.

[72] Walker, C. "Numerical Methods for Two–Phase Flow with Contact Lines". ISBN 978-82-471-3617-1. Doctoral thesis. Trondheim: Norwegian University of Science, Technology, Department of Energy, and Process Engineering, July 2012. ISBN: 978-82-471-3617-1.

[73] White, F. M. *Fluid Mechanics*. seventh. New York: McGraw-Hill, 2011.

[74] Wise, S., Kim, J., and Lowengrub, J. "Solving the regularized, strongly anisotropic Cahn-Hilliard equation by an adaptive nonlinear multigrid method". In: *Journal of Computational Physics* 226 (2007), pp. 414–446. DOI: 10.1016/j.jcp.2007.04.020.

[75] Xu, J.-J., Li, Z., Lowengrub, J., and Zhao, H.-K. "A Level Set Method for Interfacial Flows with Surfactants". In: *Journal of Computational Physics* 212.2 (Mar. 2006), pp. 590–616.

[76] Zhao, H. "An Experimental Investigation of Liquid Droplets Impinging Vertically on a Deep Liquid Pool". ISBN 978-82-471-1864-1. Doctoral thesis. Trondheim: Norwegian University of Science, Technology, Department of Energy, and Process Engineering, Oct. 2009.

[77] Zhao, H., Brunsvold, A., and Munkejord, S. T. "Transition between coalescence and bouncing of droplets on a deep liquid pool". In: *Journal of Multiphase Flow* 37 (2011), pp. 1109–1119.

[78] Zhao, H.-K., Chan, T., Merriman, B., and Osher, S. "A variational level set approach to multiphase motion". In: *Journal of Computational Physics* 127 (1996), pp. 179–195.

[79] Zhou, Y., Zhao, S., Feig, M., and Wei, G. "High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources". In: *J. Comput. Phys.* 213 (2006), pp. 1–30.

A

Calculation of interface curvature with the level-set method

K. Y. Lervåg
Published in *MekIT'11 - 6th National Conference on Computational Mechanics, Trondheim*, 2011. ISBN: 978-82-519-2798-7.

Calculation of interface curvature with the level-set method

Karl Yngve Lervåg
Norwegian University of Science and Technology
Department of Energy and Process Engineering
Kolbjørn Hejes veg 2
NO-7491 Trondheim, Norway
e–mail: karl.y.lervag@ntnu.no

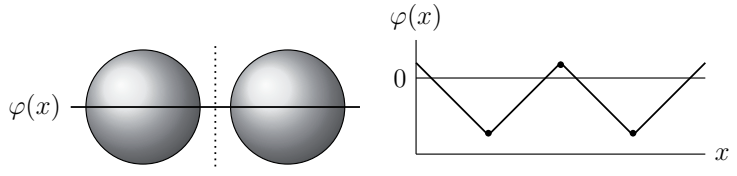
Summary The level-set method is a popular method for interface capturing. One of the advantages of the level-set method is that the curvature and the normal vector of the interface can be readily calculated from the level-set function. However, in cases where the level-set method is used to capture topological changes, the standard discretization techniques for the curvature and the normal vector do not work properly. This is because they are affected by the discontinuities of the signed-distance function half-way between two interfaces. This article addresses the calculation of normal vectors and curvatures with the level-set method for such cases. It presents a discretization scheme that is relatively easy to implement in to an existing code. The improved discretization scheme is compared with a standard discretization scheme, first for a case with no flow, then for a case where two drops collide in a shear flow. The results show that the improved discretization yields more robust calculations in areas where topological changes are imminent.

Introduction

The level-set method was introduced by Osher and Sethian [16]. It is designed to implicitly track moving interfaces through an isocontour of a function defined in the entire domain. In particular, it is designed for problems in multiple spatial dimensions in which the topology of the evolving interface changes during the course of events, c.f. [19].

This article addresses the calculation of interface geometries with the level-set method. This method allows us to calculate the normal vector and the curvature of an interface directly as the first and second derivatives of the level-set function. These calculations are typically done with standard finite-difference methods. Since the level-set function is chosen to be a signed-distance function, in most cases it will have areas where it is not smooth. Consider for instance two colliding droplets where the interfaces are captured with the level-set method, see Figure 1. The derivative of the level-set function will not be defined at the points outside the droplets that have an equal distance to both droplets. When the droplets are in near contact, this discontinuity in the derivative will lead to significant errors when calculating the interface geometries with standard finite-difference methods. For convenience the areas where the derivative of the level-set function is not defined will hereafter be refered to as kinks.

To the authors knowledge, this issue was first described in [11], where the level-set method was used to model tumor growth. Here Macklin and Lowengrub presented a direction difference to



(a) Droplets in near contact (b) A slice of the level-set function

Figure 1: (a) Two droplets in near contact. The dotted line marks a region where the derivative of the level-set function is not defined. (b) A one-dimensional slice of the level-set function $\varphi(x)$. The dots mark points where the derivative of $\varphi(x)$ is not defined.

treat the discretization across kinks for the normal vector and the curvature. They later presented an improved method where curve fitting was used to calculate the curvatures [12]. This was further expanded to include the normal vectors [13].

An alternative method to avoid the kinks is presented in [18], where a level-set extraction technique is presented. This technique uses an extraction algorithm to reconstruct separate level-set functions for each distinct body.

Accurate calculation of the curvature is important in many applications, in particular in curvature-driven flows. There are several examples in the literature of methods that improve the accuracy of the curvature calculations, but that do not consider the problem with the discretization across the kinks. The authors in [22] use a coupled level-set and volume-of-fluid method based on a fixed Eulerian grid, and they use a height function to calculate the curvatures. In [6] a refined level-set grid method is used to study two-phase flows on structured and unstructured grids for the flow solver. An interface-projected curvature-evaluation method is presented to achieve converging calculation of the curvature. In [14] they adopt a discontinuous Galerkin method and a pressure-stabilized finite-element method to solve the level-set equation and the Navier-Stokes equations, respectively. They develop a least-squares approach to calculate the normal vector and the curvature accurately, as opposed to using a direct derivation of the level-set function. This method is used in [2], where they show impressive results for simulation of turbulent atomization.

This article applies the level-set method to incompressible two-phase flow in two dimensions. The direction difference described in [11] is used to calculate the normal vectors, and a curvature discretization is presented which is based on the geometry-aware discretization given in [12]. The main advantage of the present scheme is that is is relatively easy to implement, since it requires very little change to a typical implementation of the level-set method.

The article starts by briefly describing the governing equations. It continues with a description of the numerical methods that are used. Then the discretization schemes for the normal vector and the curvature are presented, followed by a detailed description of the curvature discretization. Next a comparison of the improved discretization and the standard discretization is made, first on static interfaces in near contact, then on two drops colliding in a shear flow. Finally some concluding remarks are made.

Governing equations

Navier-Stokes equations for two-phase flow

Consider a two-phase domain $\Omega = \Omega^+ \cup \Omega^-$ where Ω^+ and Ω^- denote the regions occupied by the respective phases. The domain is divided by an interface $\Gamma = \delta\Omega^+ \cap \delta\Omega^-$ as illustrated in Figure 2. The governing equations for incompressible and immiscible two-phase flow in the domain Ω with an interface force on the interface Γ can be stated as

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0, \tag{1}$$

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} \right) = -\boldsymbol{\nabla} p + \boldsymbol{\nabla} \cdot (\mu \boldsymbol{\nabla} \boldsymbol{u}) + \rho \boldsymbol{f}_b + \int_\Gamma \sigma \kappa \boldsymbol{n}\, \delta(\boldsymbol{x} - \boldsymbol{x}_I(s))\, \mathrm{d}s, \tag{2}$$

where $\boldsymbol{u}$ is the velocity vector, p is the pressure, $\boldsymbol{f}_b$ is the specific body force, σ is the coefficient of surface tension, κ is the curvature, $\boldsymbol{n}$ is the normal vector which points to Ω^+, δ is the Dirac Delta function, $\boldsymbol{x}_I$ is a parametrization of the interface, ρ is the density and μ is the

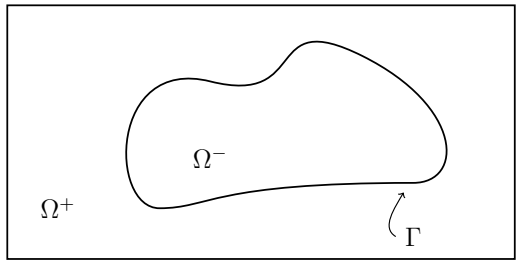


Figure 2: Illustration of a two-phase domain: The interface Γ separates the two phases, one in Ω^+ and the other in Ω^-.

viscosity. These equations are often called the Navier-Stokes equations for incompressible two-phase flow.

It is assumed that the density and the viscosity are constant in each phase, but they may be discontinuous across the interface. The interface force and the discontinuities in the density and the viscosity lead to a set of interface conditions,

$$[\boldsymbol{u}] = 0, \tag{3}$$
$$[p] = 2[\mu]\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{n} + \sigma\kappa, \tag{4}$$
$$[\mu\boldsymbol{\nabla}\boldsymbol{u}] = [\mu]\big((\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{n}\boldsymbol{n} + (\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{t})\boldsymbol{n}\boldsymbol{t} + (\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{t})\boldsymbol{t}\boldsymbol{n} + (\boldsymbol{t}\cdot\boldsymbol{\nabla}\boldsymbol{u}\cdot\boldsymbol{t})\boldsymbol{t}\boldsymbol{t}\big), \tag{5}$$
$$[\boldsymbol{\nabla}p] = 0, \tag{6}$$

where $\boldsymbol{t}$ is the tangent vector along the interface and $[\,\cdot\,]$ denotes the jump across an interface, that is

$$[\mu] \equiv \mu^+ - \mu^-. \tag{7}$$

See [7, 5] for more details and a derivation of the interface conditions.

Level-set method

The interface is captured with the zero level set of the level-set function $\varphi(\boldsymbol{x}, t)$, which is prescribed as a signed-distance function. That is, the interface is given by

$$\Gamma = \{\,(\boldsymbol{x}, t) \mid \varphi(\boldsymbol{x}, t) = 0\,\}, \quad \boldsymbol{x} \in \Omega, \quad t \in \mathbb{R}^+, \tag{8}$$

and for any $t \geq 0$,

$$\varphi(\boldsymbol{x}, t) \begin{cases} < 0 & \text{if } \boldsymbol{x} \in \Omega^- \\ = 0 & \text{if } \boldsymbol{x} \in \Gamma \\ > 0 & \text{if } \boldsymbol{x} \in \Omega^+ \end{cases}. \tag{9}$$

The interface is updated by solving an advection equation for φ,

$$\frac{\partial\varphi}{\partial t} + \hat{\boldsymbol{u}}\cdot\boldsymbol{\nabla}\varphi = 0, \tag{10}$$

where $\hat{\boldsymbol{u}}$ is the velocity at the interface extended to the entire domain. The interface velocity is extended from the interface to the domain by solving

$$\frac{\partial\hat{\boldsymbol{u}}}{\partial\tau} + S(\varphi)\boldsymbol{n}\cdot\boldsymbol{\nabla}\hat{\boldsymbol{u}} = 0, \quad \hat{\boldsymbol{u}}_{\tau=0} = \boldsymbol{u}, \tag{11}$$

to steady state, c.f. [24]. Here τ is pseudo-time and S is a smeared sign function which is equal to zero at the interface,

$$S(\varphi) = \frac{\varphi}{\sqrt{\varphi^2 + 2\Delta x^2}}. \tag{12}$$

When Equation (10) is solved numerically, the level-set function loses its signed-distance property due to numerical dissipation. The level-set function is therefore reinitialized regularly by solving

$$\begin{aligned} \frac{\partial \varphi}{\partial \tau} + S(\varphi_0)(|\boldsymbol{\nabla}\varphi| - 1) = 0, \\ \varphi(\boldsymbol{x}, 0) = \varphi_0(\boldsymbol{x}), \end{aligned} \tag{13}$$

to steady state as proposed in [20]. Here φ_0 is the level-set function that needs to be reinitialized.

One of the advantages of the level-set method is that normal vectors and curvatures can be readily calculated from the level-set function, i.e.

$$\boldsymbol{n} = \frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}, \tag{14}$$

$$\kappa = \boldsymbol{\nabla} \cdot \left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|} \right). \tag{15}$$

Numerical methods

The Navier-Stokes equations, (1) and (2), are solved by a projection method on a staggered grid as described in [5, Chapter 5.1.1]. The spatial terms are discretized by the second-order central difference scheme, except for the convective terms which are discretized by a fifth-order WENO scheme. The temporal discretization is done with explicit strong stability-preserving Runge-Kutta (SSP RK) schemes, see [4]. A three-stage third-order SSP-RK method is used for the Navier-Stokes equations (1) and (2), and a four-stage second-order SSP-RK method is used for the level-set equations (10), (11) and (13).

The method presented in [1] is used to improve the computational speed. The method is often called the narrow-band method, since the level-set function is only updated in a narrow band across the interface at each time step.

The interface conditions are treated in a sharp fashion with the Ghost-Fluid Method (GFM), which incorporates the discontinuities into the discretization stencils by altering the stencils close to the interfaces. For instance, the GFM requires that a term is added to the stencil on the right-hand side of the Poisson equation for the pressure. Consider a one-dimensional case where $[\rho] = [\mu] = 0$ and where the interface lies between x_i and x_{i+1}. In this case,

$$\frac{p_{i+1} - 2p_i + p_{i-1}}{\Delta x^2} = f_k \pm \frac{\sigma \kappa_\Gamma}{\Delta x^2}, \tag{16}$$

where f_k is the general right-hand side value and κ_Γ is the curvature at the interface. The sign of the added term depends on the sign of $\varphi(x_i)$. See [7] for more details on how the GFM is used for the Navier-Stokes equations and [9] for a description on how to use the GFM for a variable-coefficient Poisson equation.

The normal vector and the curvature defined by equations (14) and (15) are typically discretized by the second-order central difference scheme, c.f [7, 19, 23]. The curvatures are calculated on

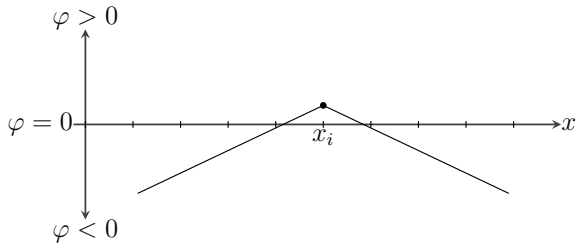


Figure 3: A level-set function that has one point where the derivative is discontinuous.

the grid nodes and then interpolated with simple linear interpolation to the interface, e.g. for κ_Γ in Equation (16),

$$\kappa_\Gamma = \frac{|\varphi_i|\kappa_{i+1} + |\varphi_{i+1}|\kappa_i}{|\varphi_i| + |\varphi_{i+1}|}. \tag{17}$$

If the level-set method is used to capture non-trivial geometries, the level-set function will in general contain areas where it is not smooth, i.e. kinks. This is depicted in Figure 3, which shows a level-set function in a one-dimensional domain that captures two interfaces, one on each side of the grid point x_i. The kink at x_i will lead to potentially large errors with the standard discretization both for the curvature and the normal vector. The errors in the curvature will lead to erroneous pressure jumps at the interfaces, and the errors in the normal vector affects both the discretized interface conditions and the advection of the level-set function. If the level-set method is used to study for example coalescence and breakup of drops, these errors may severly affect the simulations.

It should be noted that the kinks that appear far from any interfaces are handled by ensuring that the denominators do not become zero, as explained in [15, Sections 2.3 to 2.4]. This works fine, since only the values of the curvature at the grid nodes adjacent to any interface are used. Also, the normal vector only needs to be accurate close to the interface due to the narrow-band approach.

Improved discretization of geometrical quantities

The previous section explained why it is necessary to develop new discretization schemes for the normal vector and the curvature that can handle kinks in the level-set function. This section will give a brief presentation of a better discretization scheme for the normal vector and an overview of an algorithm to calculate the curvature.

The normal vector

A discretization scheme is presented in [11] which uses a quality function to ensure that the differences never cross kinks. The basic strategy is to use a combination of central differences and one-sided differences based on the values of a quality function,

$$Q(\boldsymbol{x}) = \left|1 - |\boldsymbol{\nabla}\varphi(\boldsymbol{x})|\right|, \tag{18}$$

which is approximated with central differences. The quality function effectively detects the areas where the level-set function differs from the signed-distance function. Let $Q_{i,j} = Q(\boldsymbol{x}_{i,j})$ and $\eta > 0$, then $Q_{i,j} > \eta$ can be used to detect kinks. The parameter η is tuned such that the quality function will detect all the kinks. The value $\eta = 0.1$ is used in the present work.

The quality function is used to define a direction function,

$$\boldsymbol{D}(\boldsymbol{x}_{i,j}) = (D_x(\boldsymbol{x}_{i,j}), D_y(\boldsymbol{x}_{i,j})), \tag{19}$$

where

$$D_x(\boldsymbol{x}_{i,j}) = \begin{cases} -1 & \text{if } Q_{i-1,j} < \eta \text{ and } Q_{i+1,j} \geq \eta, \\ 1 & \text{if } Q_{i-1,j} \geq \eta \text{ and } Q_{i+1,j} < \eta, \\ 0 & \text{if } Q_{i-1,j} < \eta \text{ and } Q_{i,j} < \eta \text{ and } Q_{i+1,j} < \eta, \\ 0 & \text{if } Q_{i-1,j} \geq \eta \text{ and } Q_{i,j} \geq \eta \text{ and } Q_{i+1,j} \geq \eta, \\ \text{undetermined} & \text{otherwise.} \end{cases} \tag{20}$$

$D_y(\boldsymbol{x}_{i,j})$ is defined in a similar manner. If D_x or D_y is undetermined, $\boldsymbol{D}(\boldsymbol{x}_{i,j})$ is chosen as the vector normal to $\boldsymbol{\nabla}\varphi(\boldsymbol{x}_{i,j})$. It is normalized, and the sign is chosen such that it points in the direction of best quality. See [11] for more details.

The direction difference is now defined as

$$\partial_x f_{i,j} = \begin{cases} \frac{f_{i,j} - f_{i-1,j}}{\Delta x} & \text{if } D_x(x_i, y_j) = -1, \\ \frac{f_{i+1,j} - f_{i,j}}{\Delta x} & \text{if } D_x(x_i, y_j) = 1, \\ \frac{f_{i+1,j} - f_{i-1,j}}{2\Delta x} & \text{if } D_x(x_i, y_j) = 0, \end{cases} \tag{21}$$

where $f_{i,j}$ is a piecewise smooth function. The normal vector is calculated using the direction difference on φ, which is equivalent to using central differences in smooth areas and one-sided differences in areas close to the kinks. This method makes sure that the differences do not cross any kinks, and the normal vector can be accurately calculated even close to a kink.

The curvature

The curvature is calculated with a discretization that is based on the improved geometry-aware curvature discretization presented by Macklin and Lowengrub [12]. This is a method where the curvature is calculated at the interfaces directly with the use of a least-squares curve parametrization of the interface. The curve parametrization is used to create a local level-set function from which the curvature is calculated using standard discretization techniques. The local level-set function only depends on one interface and is therefore free of kinks.

The main difference between the present method and that of Macklin and Lowengrub is that they calculate the curvature at the interface directly, whereas the present method instead calculates the curvature at the grid nodes. In other words, Macklin and Lowengrub calculate κ_Γ in Equation (16) directly, whereas the present method calculates κ_i and κ_{i+1} with the improved curvature discretization and then use linear interpolation as described in Equation (17) to find κ_Γ. The main motivation behind this difference is that the present method does not require a significant change to any existing code. Thus it is relatively straightforward to implement the present method even when the curvature is needed for more than the Capillary force term in the Navier-Stokes equations. An example of such a case is when the curvature is used to model interfacial flows with surfactant [23].

An important consequence of the previously explained difference is that it becomes more important to have an accurate representation of the interface. The curvature discretization presented here uses monotone cubic Hermite splines to parametrize the curve. The least-square parametrization used in [12] is only accurate very close to the point where the curvature needs to be calculated. The Hermite spline is more accurate along the entire interface representation.

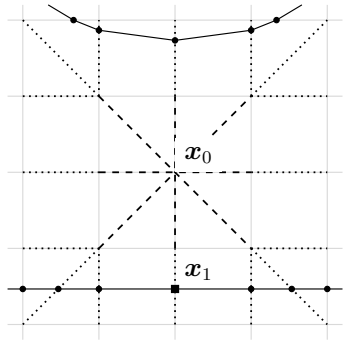


Figure 4: Sketch of a breadth-first search. The dashed lines depict the edges that are searched first, the dotted lines depict the edges that are searched next and the solid lines depict two interfaces. The circular dots mark where the algorithm finds interface points, and the rectangular dot marks the point which is returned for the depicted case.

The algorithm to calculate the curvature at $\boldsymbol{x}_{i,j}$ can be summarized as follows. The details are explained in the next section.

1. If $Q_{i+n,j+m} \leq \eta$, where $n = -1, 0, 1$ and $m = -1, 0, 1$, then it is safe to use the standard discretization. Otherwise continue to the next step.
2. Locate the closest interface, Γ.
3. Find a set of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \Gamma$.
4. Create a parametrization $\boldsymbol{\gamma}(s)$ of the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.
5. Calculate a local level-set function based on the parametrization $\boldsymbol{\gamma}(s)$.
6. Use the standard discretization on the local level-set function to calculate the curvature.

Details of the curvature algorithm

Locating the closest interface

A breadth-first search is used to to identify the closest interface, see Figure 4. Let $\boldsymbol{x}_0$ denote the starting point and $\boldsymbol{x}_1$ denote the desired point on the closest interface. The search iterates over all the eight edges from $\boldsymbol{x}_0$ to its neighbours and tries to locate an interface which is identified by a change of sign of $\varphi(\boldsymbol{x})$. If more than one interface is found, $\boldsymbol{x}_1$ is chosen to be the point that is closest to $\boldsymbol{x}_0$. If no interfaces are located the search continues at the next depth. The search continues in this manner until an interface is found. Note that this algorithm does not in general return the point on the interface which is closest to $\boldsymbol{x}_0$.

The crossing points between the grid edges and the interfaces are found with linear and bilinear interpolation. E.g. if an interface crosses the edge between (i, j) and $(i, j+1)$ at $\boldsymbol{x}_I$, the interface point is found by linear interpolation,

$$\boldsymbol{x}_I = \boldsymbol{x}_{i,j} + \theta(0, \Delta x), \tag{22}$$

where

$$\theta = \frac{\varphi(\boldsymbol{x}_{i,j})}{\varphi(\boldsymbol{x}_{i,j}) - \varphi(\boldsymbol{x}_{i,j+1})}. \tag{23}$$

In the diagonal cases the interface point is found with bilinear interpolation along the diagonal. This leads to

$$\boldsymbol{x}_I = \boldsymbol{x}_{i,j} + \theta(\Delta x, \Delta x), \tag{24}$$

where θ is the solution of

$$\alpha_1\theta^2 + \alpha_2\theta + \alpha_3 = 0. \tag{25}$$

The α values depend on the grid cell. For instance, when searching along the diagonal between (i, j) and $(i + 1, j + 1)$ the α values will be

$$\alpha_1 = \varphi_{i,j} - \varphi_{i+1,j} - \varphi_{i,j+1} + \varphi_{i+1,j+1}, \tag{26}$$

$$\alpha_2 = \varphi_{i+1,j} + \varphi_{i,j+1} - 2\varphi_{i,j}, \tag{27}$$

$$\alpha_3 = \varphi_{i,j}. \tag{28}$$

Searching for points on an interface

When an interface and a corresponding point $\boldsymbol{x}_1$ on the interface are found, the next step is to find a set of points $\boldsymbol{x}_2, \ldots, \boldsymbol{x}_k, \ldots, \boldsymbol{x}_n$ on the same interface. The points should be ordered such that when traversing the points from $k = 1$ to $k = n$, the phase with $\varphi(\boldsymbol{x}) < 0$ is on the left-hand side. Note that the ordering of the points may be done after all the points are found. Three criteria are used when searching for new points:

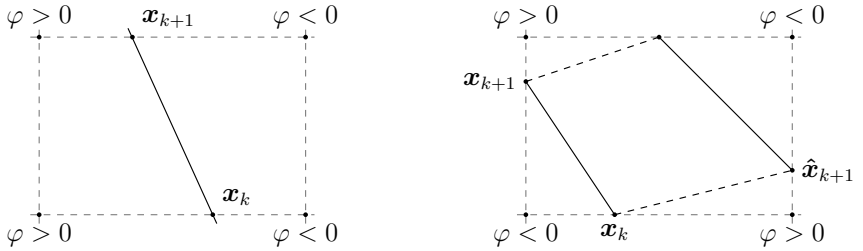
1. The points are located on the grid edges.
2. The distance between $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ for all k is greater than a given threshold μ.
3. The n points that are closest to $\boldsymbol{x}_0$ are selected, where $\boldsymbol{x}_0 = \boldsymbol{x}_{i,j}$ is the initial point where the curvature is to be calculated.

Let $\boldsymbol{x}_k \in \Gamma \cap [x_i, x_{i+1}) \times [y_j, y_{j+1})$ be given. To find a new point $\boldsymbol{x}_{k+1}$ on Γ, a variant of the marching-squares algorithm[1] is used. Given $\boldsymbol{x}_k$ and a search direction which is either clockwise or counter clockwise, the algorithm searches for all the points where an interface crosses the edges of the mesh rectangle $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$. In most cases there will be two such points and $\boldsymbol{x}_k$ is one of them. $\boldsymbol{x}_{k+1}$ is then selected based on the search direction. If $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, the search is continued in the adjacent mesh rectangle. The search process is depicted in Figure 5(a).

In some rare cases the algorithm must handle the ambiguous case depicted in Figure 5(b). In these cases there are four interface crossing-points and two solutions. Either solution is valid, and it is not possible to say which solution is better. The current implementation selects the first solution that it finds, which will be in all practical sense a random choice. Note that the ambiguous cases only occur when two interfaces cross a single grid cell. The ambiguity comes from the fact that the level-set method is not able to resolve the interfaces on a sub-cell resolution.

It was found that $n = 7$ points where necessary in order to ensure that the closest points on the interface with respect to the different grid points are captured with the spline parametrization.

[1]The marching-squares algorithm is an equivalent two-dimensional formulation of the well known marching-cubes algorithm presented in [10]. The algorithm was mainly developed for use in computer graphics.



(a) Marching squares (b) Ambiguous case

Figure 5: (a) The search starts by locating the two points where the interface crosses the mesh rectangle. $\boldsymbol{x}_k$ is the starting point, and if the search is counter clockwise it will select $\boldsymbol{x}_{k+1}$ as depicted. If the search is clockwise, it will select $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, and the search continues in the adjacent mesh rectangle $[x_i, x_{i+1}] \times [y_{j-1}, y_j]$. (b) An example of an ambiguous case. The solid black lines and the dashed black lines are two equally valid solutions for how the interfaces cross the mesh rectangle. If the search starts at $\boldsymbol{x}_k$ and searches counter clockwise, then both $\hat{\boldsymbol{x}}_{k+1}$ and $\boldsymbol{x}_{k+1}$ are valid solutions.

Curve fitting

Cubic Hermite splines are used to fit a curve to the set of points

$$X_{0,m} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}. \tag{29}$$

Let the curve parametrization be denoted $\boldsymbol{\gamma}(s)$ for $0 < s < 1$. A cubic spline is a parametrization where

$$\boldsymbol{\gamma}(s) = \begin{cases} \boldsymbol{\gamma}_1(s) & s_0 \leq s < s_1, \\ \boldsymbol{\gamma}_2(s) & s_1 \leq s < s_2, \\ \vdots & \\ \boldsymbol{\gamma}_m(s) & s_{m-1} \leq s \leq s_m, \end{cases} \tag{30}$$

where $0 = s_0 < s_1 < \cdots < s_m = 1$

$$\boldsymbol{\gamma}(s_i) = \boldsymbol{x}_i, \quad 0 \leq i \leq m, \tag{31}$$

and each interpolant $\boldsymbol{\gamma}_i(s) = (x_i(s), y_i(s))$ is a third-order polynomial. A Hermite spline is a spline where each interpolant is in Hermite form, see [17, Chapter 4.5]. The interpolants are created by solving the equations

$$\boldsymbol{\gamma}_i(s) = h_{00}(s)\boldsymbol{x}_{i-1} + h_{01}(s)\boldsymbol{x}_i + h_{10}(s)\boldsymbol{m}_{i-1} + h_{11}(s)\boldsymbol{m}_i, \tag{32}$$

for $1 \leq i \leq m$, where $\boldsymbol{m}_i$ is the curve tangents and h_{00}, h_{01}, h_{10} and h_{11} are Hermite basis polynomials,

$$\begin{aligned} h_{00}(s) &= 2s^3 - 3s^2 + 1, \\ h_{01}(s) &= s^3 - 2s^2 + s, \\ h_{10}(s) &= -2s^3 + 3s^2, \\ h_{11}(s) &= s^3 - s^2. \end{aligned} \tag{33}$$

The choice of the tangents is non-unique, and there are several possible options for a cubic Hermite spline.

It is essential that the spline is properly oriented. This is because we require to find both the distance and the position of a point on the grid relative to the spline in order to define a local level-set function. The orientation of the spline $\boldsymbol{\gamma}(s)$ is defined such that when s increases, Ω^- is to the left.

To ensure that our curve is properly oriented, the tangents are chosen as described in [3]. This will ensure monotonicity for each component as long as the input data is monotone. The tangents are modified as follows. First the slopes of the secant lines between successive points are computed,

$$\boldsymbol{d}_i = \frac{\boldsymbol{x}_i - \boldsymbol{x}_{i-1}}{s_i - s_{i-1}} \tag{34}$$

for $1 \leq i \leq m$. Next the tangents are initialized as the average of the secants at every point,

$$\boldsymbol{m}_i = \frac{\boldsymbol{d}_i + \boldsymbol{d}_{i+1}}{2} \tag{35}$$

for $1 \leq i \leq m-1$. The curve tangents at the endpoints are set to $\boldsymbol{m}_0 = \boldsymbol{d}_1$ and $\boldsymbol{m}_m = \boldsymbol{d}_m$. Finally let k pass from 1 through $m-1$ and set $\boldsymbol{m}_k = \boldsymbol{m}_{k+1} = 0$ where $\boldsymbol{d}_k = 0$, and $\boldsymbol{m}_k = 0$ where $\text{sign}(\boldsymbol{d}_k) \neq \text{sign}(\boldsymbol{d}_{k+1})$.

Local level-set function

The local level-set function, here denoted as $\phi(\boldsymbol{x}_{i,j}) \equiv \phi_{i,j}$, is calculated at the grid points surrounding and including $\boldsymbol{x}_0 = \boldsymbol{x}_{i,j}$. The curvature is then calculated with the standard discretization stencil where ϕ is used instead of the global level-set function, φ.

A precise definition of ϕ is

$$\phi(\boldsymbol{x}_{i,j}) = \min_s \left(\hat{d}(\boldsymbol{x}_{i,j}, \boldsymbol{\gamma}(s)) \right) \tag{36}$$

where $\hat{d}(\boldsymbol{x}, \boldsymbol{\gamma}(s))$ is the signed-distance function, which is negative in phase one and positive in phase two. This function is calculated by first finding the minimum distance between $\boldsymbol{x}$ and $\boldsymbol{\gamma}(s)$ and then deciding the correct sign. The minimum distance is found by minimizing the norm

$$d(\boldsymbol{x}, \boldsymbol{\gamma}(s)) = \|\boldsymbol{x} - \boldsymbol{\gamma}(s)\|_2. \tag{37}$$

When $\boldsymbol{\gamma}$ is composed of cubic polynomials as is the case for cubic Hermite splines, the computation of the distance requires the solution of several fifth-order polynomial equations. Sturm's method (see [21, Section 11.3] or [8, Chapter XI,§2]) is employed to locate and bracket the solutions and a combined Newton-Raphson and bisection method is used to refine them. The correct sign is found by solving

$$\text{sign}(\phi(\boldsymbol{x}_{i,j})) = \text{sign}\left((\boldsymbol{x}_{i,j} - \boldsymbol{\gamma}(s)) \times \boldsymbol{t}_{\boldsymbol{\gamma}}(s) \right)_z, \tag{38}$$

where $\boldsymbol{t}_{\boldsymbol{\gamma}}(s)$ is the tangent vector of $\boldsymbol{\gamma}(s)$.

Verification and testing

This section presents results of calculating normal vectors and curvatures with the improved discretization schemes. The results are compared with the standard discretization. Note that in both the following cases the standard second-order central differences are used as the standard discretization.

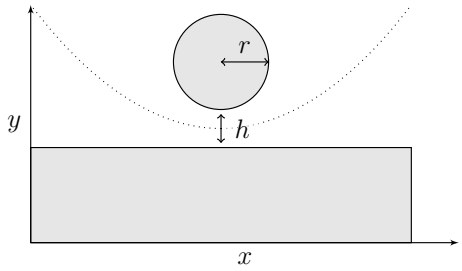


Figure 6: Initial setup for the circle and line test. The dotted line depicts the kink location.

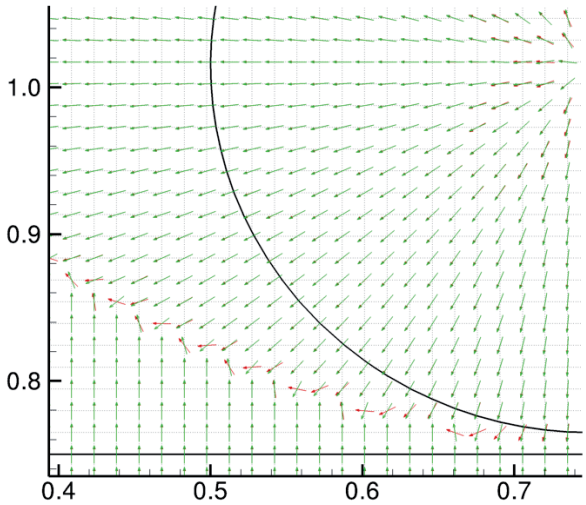


Figure 7: A comparison of the calculated normal vectors between the standard discretization in red and the improved method in green. The thick black lines depict the interface.

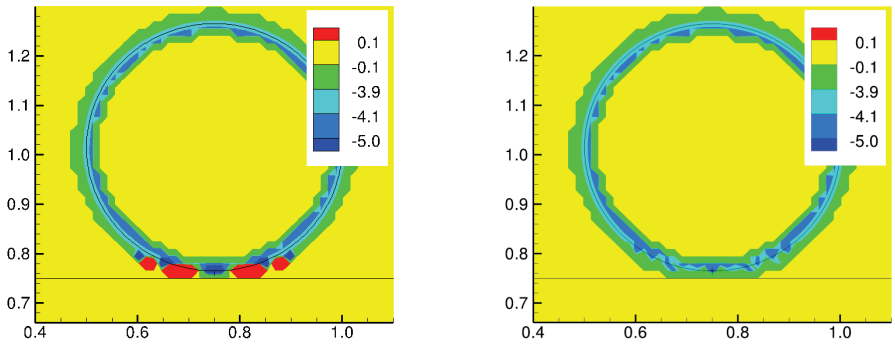
A static disc above a rectangle

The first case is a simple and static test-case where a disc of radius r is positioned at a distance h above a rectangle, see Figure 6. Only the level-set function and the geometrical quantities are considered. This means that none of the governing equations are solved (equations (1), (2), (10), (11) and (13)). When h is small, the kinks along the dotted line will affect the discretization stencils as has been explained.

The parameters for this case is $r = 0.25$ m and $h = \Delta x$. The domain is 1.5 m $\times$ 1.5m, and the straight line is positioned at $y = 0.75$ m. The grid size is 101×101.

Figure 7 shows a comparison of the calculated normal vectors. The standard discretization is depicted with red vectors and the direction difference is depicted with green vectors. The figure shows that the standard discretization yields much less accurate results along the kink region than the direction difference.

Figure 8 shows a comparison of the calculated curvatures. Note that the curvature is only cal-



(a) Standard discretization

(b) Improved method with curve fitting

Figure 8: A comparison of curvature calculations between standard discretization and the improved method. The standard discretization leads to large errors in the curvatures in areas that are close to two interfaces.

culated at grid points adjacent to the interfaces. At the grid points where it is not calculated, it is set to zero. The figure shows that the standard discretization leads to large errors in the calculated curvatures in the areas that are close to two interfaces. In particular note that the sign of the curvature becomes wrong. The analytic curvature for this case is $\kappa = -1/r = -4$, and the curvature spikes seen for the standard discretization is in the order of $|\kappa| \sim \frac{1}{\Delta x} \simeq 67.3$. These spikes will lead to large errors in the pressure jumps through Equation (17). The effect of these errors will become more clear in the next case.

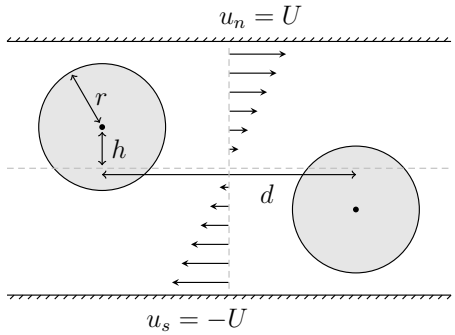


Figure 9: The drop in shear setup.

Drop collision in shear flow

The second case considers drop collision in shear flow, see Figure 9. The drops both have a radius r, and they are initially placed at a distance $d = 5r$ apart in a shear flow where the flow velocity changes linearly from $u_s = -U < 0$ at the bottom wall to $u_n = U$ at the top wall. The computational domain is $12r \times 8r$, and the grid size is 241×161. The size of the grid is chosen to be relatively coarse, such that the difference between the standard discretization of the curvature and the curve-fitting based discretization is properly revealed.

The purpose of this case is to study the behaviour of the level-set method, in particular the calculation of the curvatures, when the drops are in close proximity. It is therefore a natural simplification to only consider the case where the density difference and the viscosity difference of the phases are zero, i.e. there is no jump in density or viscosity across the interface.

The flow is governed by the Reynolds number and the Capillary number, which in the current case can be defined by

$$Re = \frac{\rho U r}{4\mu}, \tag{39}$$

$$Ca = \frac{\mu U}{4\sigma}. \tag{40}$$

In the following results the Reynolds and the Capillary numbers were set to

$$Re = 10, \quad Ca = 0.025. \tag{41}$$

The choice was made such that the drops would not be severely deformed in the shear flow. The radius of the drops was $r = 0.5$ m, and the distance from the center line to the drop centers was $h = 0.84r = 0.42$ m.

Figure 10 shows a comparison of the interface evolution and the curvature between the standard discretization and the improved discretization. The top and bottom rows show the evolution for the standard discretization and the improved discretization, respectively. The kinks between the drops lead to curvature spikes with the standard discretization, whereas the improved discretization calculates the curvature along the kink in a much more reliable manner. The curvature spikes are seen to prevent coalescence. This is due to the effect they have on the pressure field.

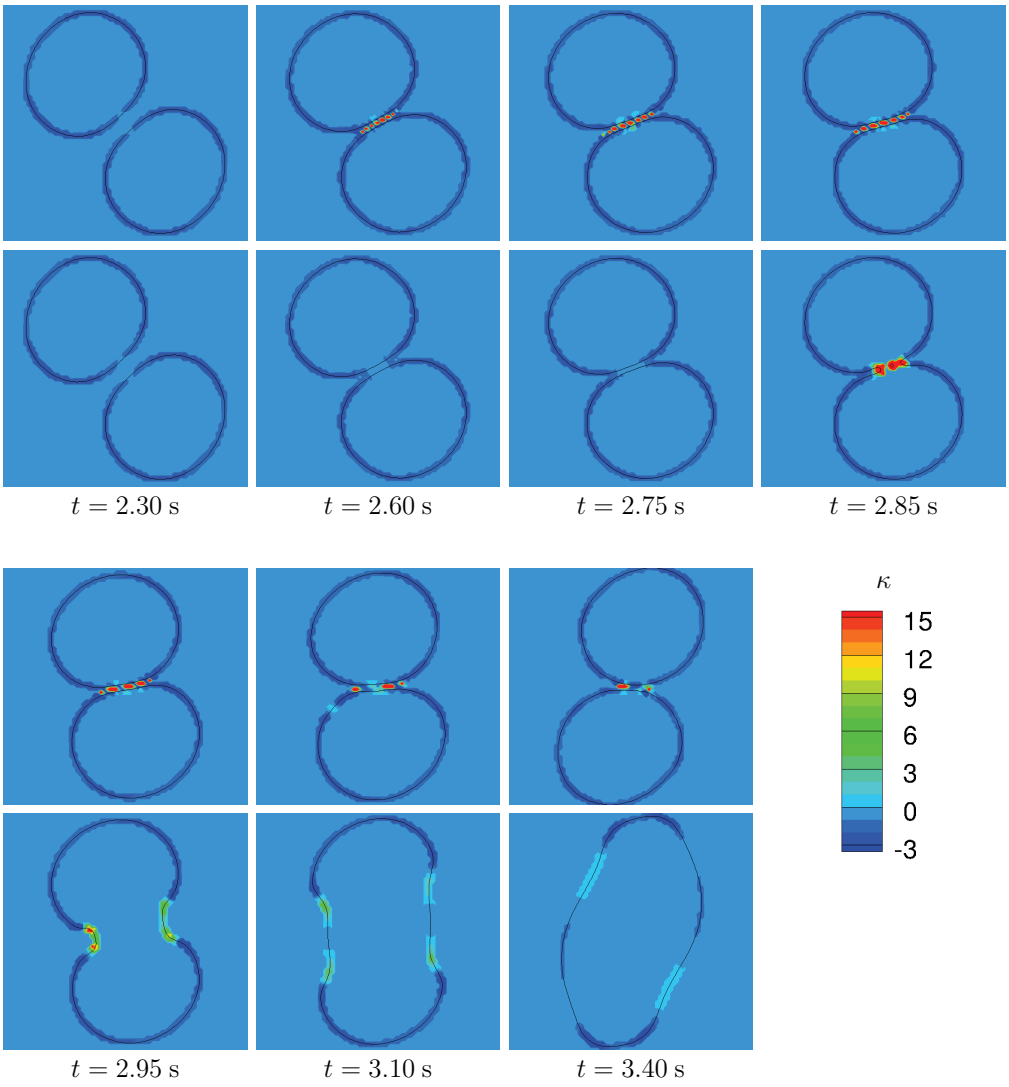
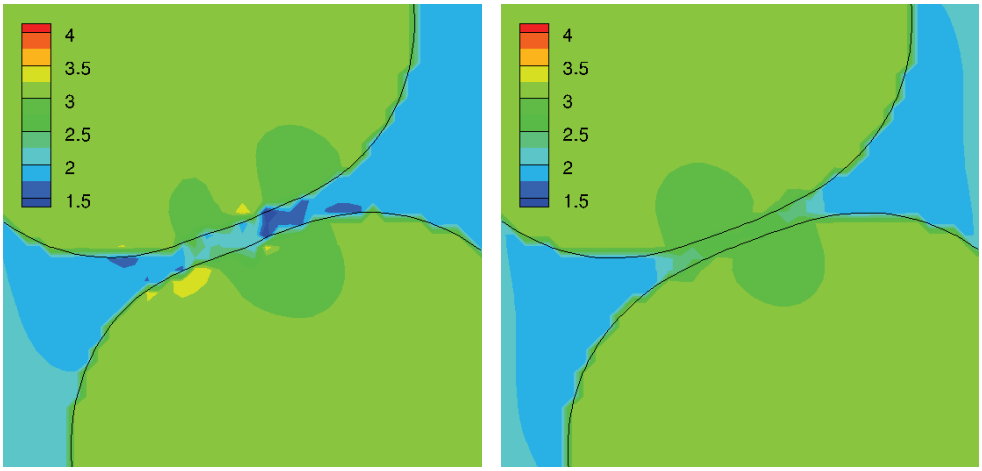


Figure 10: A comparison between the standard discretization (top row) and the improved discretization (bottom row) of the interface evolution and the curvature κ of drop collision in shear.



(a) Standard discretization (b) Improved method

Figure 11: Comparison of the pressure field in the thin film between the droplets at $t = 2.75$ s. The contour legends indicate the pressure in Pa.

The errors in the curvature with the standard discretization lead to an erroneous pressure field between the drops that prevents coalescence, c.f. Equation (16). Figure 11 shows the pressure field at $t = 2.75$ s. It can be seen that the pressure field for the standard discretization is distorted in the thin-film region. This distortion in the pressure leads to a flow in the film region which suppresses coalescence. The corresponding result for the improved method shows that the pressure is not distorted. It is high in the center of the thin-film region and lower at the edges. The pressure change induces a flow out of the region which is more as expected.

Conclusions

This article has implemented improved discretization schemes for the normal vector and the curvature of the interface between two phases. The normal vector was discretized by the direction difference which is presented in [11]. The curvature was discretized with a scheme that is based on the geometry-aware discretization presented in [12]. The main advantage of the present discretization method for the curvature is that it is relatively straightforward to implement in to an existing code since it does not require a change of the existing framework.

The implementation of the curvature discretization have been described in detail. The improved schemes are compared with the standard discretization in two different cases. The first case is a direct comparison of the schemes for a case with no flow. The second case compares the evolution of two drops colliding in shear flow. Both tests demonstrate that the standard discretization of the normal vector and the curvature leads to erroneous behaviour at the kink locations. The second case shows that this behaviour prevents coalescence from occurring due to an erroneous pressure field. The curvature spikes at the kink regions are not observed with the improved discretization schemes, and coalescence is achieved for the second case.

Acknowledgements

This work was financed through the Enabling Low-Emission LNG Systems project, and the author acknowledge the contributions of GDF SUEZ, Statoil and the Petromaks programme of the Research Council of Norway (193062/S60).

The author acknowledges Bernhard Müller (NTNU) and Svend Tollak Munkejord (SINTEF Energy Research) for valuable feedback on the manuscript. The author also acknowledges Leif Amund Lie and Eirik Svanes for several good discussions.

References

[1] D.Adalsteinsson and J. A.Sethian A fast level set method for propagating interfaces *Journal of Computational Physics*, **vol.118**, 269–277, 1995.

[2] O.Desjardins, V.Moureau and H.Pitsch An accurate conservative level set/ghost fluid method for simulating turbulent atomization *Journal of Computational Physics*, **vol.227**, 8395–8416, 2008.

[3] F. N.Fritsch and R. E.Carlson Monotone piecewise cubic interpolation *SIAM Journal of Numerical Analysis*, **vol.17**(2), 238–246, 1980.

[4] S.Gottlieb, C. W.Shu and E.Tadmor Strong stability-preserving high-order time discretization methods *SIAM Review*, **vol.43**, 89–112, 2001.

[5] E. B.Hansen *Numerical Simulation of Droplet Dynamics in the Presence of an Electric Field* Doctoral thesis, Norwegian University of Science and Technology, Department of Energy and Process Engineering, Trondheim, Nov. 2005 ISBN 82-471-7318-2.

[6] M.Herrmann A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids *Journal of Computational Physics*, **vol.227**, 2674–2706, 2008.

[7] M.Kang, R. P.Fedkiw and X.-D.Liu A boundary condition capturing method for multiphase incompressible flow *Journal of Scientific Computing*, **vol.15**(3), 323–360, 2000.

[8] S.Lang *Algebra* Graduate texts in mathematics. Springer, 2002.

[9] X.-D.Liu, R. P.Fedkiw and M.Kang A boundary condition capturing method for poisson's equation on irregular domains *Journal of Computational Physics*, **vol.160**, 151–178, 2000.

[10] W. E.Lorensen and H. E.Cline Marching cubes: A high resolution 3D surface construction algorithm *Computer Graphics*, **vol.21**(4), 163–169, July 1987.

[11] P.Macklin and J.Lowengrub Evolving interfaces via gradients of geometry-dependent interior poisson problems: Application to tumor growth *Journal of Computational Physics*, **vol.203**, 191–220, 2005.

[12] P.Macklin and J.Lowengrub An improved geometry-aware curvature discretziation for level set methods: Application to tumor growth *Journal of Computational Physics*, **vol.215**, 392–401, 2006.

[13] P.Macklin and J. S.Lowengrub A new ghost cell/level set method for moving boundary problems: Application to tumor growth *Journal of Scientific Computing*, **vol.35**, 266–299, 2008.

[14] E.Marchandise, P.Geuzaine, N.Chevaugeon and J.-F.Remacle A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics *Journal of Computational Physics*, **vol.225**, 949–974, 2007.

[15] S.Osher and R. P.Fedkiw *The Level-Set Method and Dynamic Implicit Surfaces* Springer, 2003.

[16] S.Osher and J. A.Sethian Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations *Journal of Computational Physics*, **vol.79**, 12–49, 1988.

[17] H.Prautzsh, W.Boehm and M.Paluszny *Bézier and B-spline Techniques* Springer, 2002.

[18] D.Salac and W.Lu A local semi-implicit level-set method for interface motion *Journal of Scientific Computing*, **vol.35**, 330–349, 2008.

[19] J. A.Sethian and P.Smereka Level set methods for fluid interfaces *Annual Review of Fluid Mechanics*, **vol.35**, 341–372, 2003.

[20] M.Sussman, P.Smereka and S.Osher A level set approach for computing solutions to incompressible two-phase flow *Journal of Computational Physics*, **vol.114**, 146–159, 1994.

[21] B.Waerden, E.Artin and E.Noether *Algebra* Number v. 1 in Algebra. Springer-Verlag, 2003.

[22] Z.Wang and A. Y.Tong A sharp surface tension modeling method for two-phase incompressible interfacial flows *International Journal for Numerical Methods in Fluids*, **vol.64**, 709–732, 2010.

[23] J.-J.Xu, Z.Li, J.Lowengrub and H.Zhau A level set method for interfacial flows with surfactants *Journal of Computational Physics*, **vol.212**(2), 590–616, March 2006.

[24] H. K.Zhao, T.Chan, B.Merriman and S.Osher A variational level set approach to multiphase motion *Journal of Computational Physics*, **vol.127**, 179–195, 1996.

B

Curvature calculations for the level-set method

K. Y. Lervåg and Å. Ervik
Published in *ENUMATH 2011* proceedings volume, Springer, 2013. ISBN: 978-3642331336.

Curvature calculations for the level-set method

Karl Yngve Lervåg and Åsmund Ervik

Abstract The present work illustrates a difficulty with the level-set method to accurately capture the curvature of interfaces in regions that are of equal distance to two or more interfaces. Such regions are characterized by kinks in the level-set function where the derivative is discontinuous. Thus the standard discretization scheme is not suitable. Three discretization schemes are outlined that are shown to perform better than the standard discretization on two selected test cases.

1 Introduction

This article addresses the calculation of interface curvature with the level-set method. In the level-set method, the normal vector and the curvature of an interface can be calculated directly from the level-set function. These calculations are usually done with standard finite-difference methods, typically the second-order central difference scheme (CD-2) [10, 12, 4].

A problem with these calculations may arise when the level-set function is defined to be a signed-distance function. The signed-distance function is in general not smooth, as can be seen in Figure 1. Here the derivative of the level-set function will be discontinuous at the regions that are of equal distance to more than one interface. When two droplets as in Figure 1 are in near contact, such discontinuities, or kinks, may lead to significant errors when calculating the interface geometries with standard finite difference methods.

Karl Yngve Lervåg, e-mail: karl.yngve@lervag.net
Norwegian University of Science and Technology, Department of Energy and Process Engineering, Kolbjørn Hejes veg 2, NO-7491 Trondheim, Norway.

Åsmund Ervik, e-mail: aaervik@gmail.com
SINTEF Energy Research, Sem Sælands veg 11, NO-7465 Trondheim, Norway.
Norwegian University of Science and Technology, Department of Physics, Høgskoleringen 5, NO-7491 Trondheim, Norway.

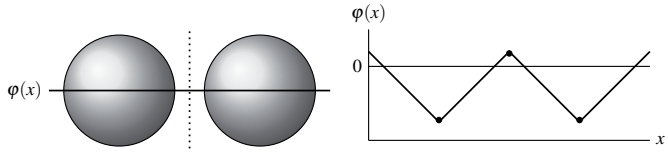


Fig. 1 (a) Two droplets in near contact. The dotted line marks a region where the derivative of the level-set function is not defined. (b) A one-dimensional slice of the level-set function $\varphi(x)$. The dots mark points where the derivative of $\varphi(x)$ is not defined.

2 Governing equations

2.1 Navier-Stokes equations for two-phase flow

Consider a domain $\Omega = \Omega^+ \cup \Omega^-$, where Ω^+ and Ω^- denote regions occupied by two respective phases, divided by an interface $\Gamma = \delta\Omega^+ \cap \delta\Omega^-$. The governing equations for incompressible and immiscible two-phase flow in the domain Ω with an interface force on the interface Γ are

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \nabla\cdot(\mu\nabla\mathbf{u}) + \rho\mathbf{f}_b + \int_\Gamma \sigma\kappa\mathbf{n}\,\delta(\mathbf{x}-\mathbf{x}_I(s))\,\mathrm{d}s. \tag{2}$$

Here $\mathbf{u}$ is the velocity vector, p is the pressure, $\mathbf{f}_b$ is the specific body force, σ is the coefficient of surface tension, κ is the curvature, $\mathbf{n}$ is the normal unit vector which points into Ω^+, δ is the Dirac delta function, $\mathbf{x}_I(s)$ is a parametrization of the interface, ρ is the density and μ is the viscosity.

It is assumed that the density and viscosity are constant in each phase, but may be discontinuous across the interface. The jump conditions across the interface are

$$[\![\mathbf{u}]\!] = 0, \tag{3}$$

$$[\![p]\!] = 2[\![\mu]\!]\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{n} + \sigma\kappa, \tag{4}$$

$$[\![\mu\nabla\mathbf{u}]\!] = [\![\mu]\!]\big((\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{n})\mathbf{nn} + (\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{t})\mathbf{nt} + (\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{t})\mathbf{tn} + (\mathbf{t}\cdot\nabla\mathbf{u}\cdot\mathbf{t})\mathbf{tt}\big), \tag{5}$$

where $\mathbf{t}$ is the tangent vector along the interface and $[\![\cdot]\!]$ denotes the jump across an interface, that is $[\![\mu]\!] \equiv \mu^+ - \mu^-$. Note that $\nabla\mathbf{u}$ and (e.g.) $\mathbf{nt}$ are rank-2 tensors. See [4, 3] for more details and a derivation of the interface conditions.

2.2 Level-set method

The interface is captured with the zero level set of the level-set function $\varphi(\mathbf{x},t)$, which is prescribed as a signed-distance function. It is updated by solving an advection equation for φ,

$$\frac{\partial \varphi}{\partial t} + \hat{\mathbf{u}} \cdot \nabla \varphi = 0, \tag{6}$$

where $\hat{\mathbf{u}}$ is the velocity at the interface, extended to the entire domain by solving

$$\frac{\partial \hat{\mathbf{u}}}{\partial \tau} + S(\varphi)\mathbf{n} \cdot \nabla \hat{\mathbf{u}} = 0, \quad \hat{\mathbf{u}}_{\tau=0} = \mathbf{u}, \tag{7}$$

to steady state, cf. [15]. Here τ is a pseudo-time and $S(\varphi) = \varphi/(\varphi^2 + 2\Delta x^2)^{1/2}$ is a smeared sign function which is equal to zero at the interface.

When (6) is solved numerically, the level-set function loses its signed-distance property due to numerical dissipation. The level-set function is therefore reinitialized regularly by solving

$$\begin{aligned} \frac{\partial \varphi}{\partial \tau} + S(\varphi_0)(|\nabla \varphi| - 1) &= 0, \\ \varphi(\mathbf{x},0) &= \varphi_0(\mathbf{x}), \end{aligned} \tag{8}$$

to steady state as proposed in [13], where φ_0 is the level-set function that needs to be reinitialized.

Normal vectors and curvatures can be readily calculated from the level-set function as

$$\mathbf{n} = \frac{\nabla \varphi}{|\nabla \varphi|} \quad \text{and} \quad \kappa = \nabla \cdot \left(\frac{\nabla \varphi}{|\nabla \varphi|} \right). \tag{9}$$

3 Numerical methods

The Navier-Stokes equations (1) and (2) are solved using a projection method on a staggered grid as described in [3, Chapter 5.1.1]. The spatial terms are discretized with CD-2, except for the convective terms which are discretized by a fifth-order WENO scheme. A third-order strong stability-preserving Runge-Kutta (SSP RK) method is used for the momentum equation (2), and a second-order SSP-RK method is used for the level-set equations (6) to (8) [2].

The interface conditions are treated in a sharp fashion with the Ghost-Fluid Method (GFM), which incorporates the discontinuities into the discretization stencils by altering the stencils close to the interfaces, cf. [1, 4, 6]. When using the GFM, the curvature is linearly interpolated from the grid points to the interface before it is used in the discretization stencils for the flow equations unless otherwise stated.

4 Curvature discretizations

The normal vector and the curvature (9) are typically discretized with the CD-2 at the grid points, cf. [4, 12, 14]. A problem with this is that CD-2 will not converge across kinks, and it may therefore introduce potentially large errors. The errors in the curvature will lead to erroneous pressure jumps at the interfaces, and the errors in the normal vector affect both the discretized interface conditions and the extrapolated velocity (7) which is used in the advection equation (6).

A direction difference scheme is presented in [7] which uses a combination of one-sided and central difference schemes to ensure that the differences never cross kinks. The same scheme is used in the present work to calculate the normal vector. The idea is choose which difference scheme to use based on the values of a quality function,

$$Q(\mathbf{x}) = |1 - |\nabla\varphi(\mathbf{x})||\,. \tag{10}$$

The quality function is itself calculated with central differences. It effectively detects the regions where the level-set function differs from the signed-distance function. Let $Q_{i,j} = Q(\mathbf{x}_{i,j})$ and $\eta > 0$, then $Q_{i,j} > \eta$ can be used to detect kinks. The parameter η is tuned such that the quality function will detect all the kinks. The value $\eta = 0.1$ is used in the present work.

In the following, three different improved discretization schemes for the curvature are outlined. Note that the first two schemes use the quality function to detect when the improved schemes should be used in favor of CD-2. Also note that the curvature is only calculated at grid points in a narrow band along the interface. At the points where it is not calculated, it is set to zero.

Macklin and Lowengrub's method (MLM) was presented in [8, 9]. With this method, the interface is parametrized with a second-order least-squares polynomial. The curvature is then calculated directly from the parametrization at the desired position on the interface.

To enable easy comparison with the other methods, the estimated curvature values are extrapolated from the interface to the adjacent grid points.

Lervåg's method (LM) was presented in [5] and is based on MLM, specifically [8]. The curve parametrization is used to create a local level-set function from which the curvature is calculated on the grid points using CD-2.

The main difference from MLM is that the curvature is calculated at the grid nodes and then interpolated to the interface afterwards. This is argued as a slight simplification of MLM, although an important consequence is that it becomes more important to have an accurate representation of the interface. Instead of using a least-squares parametrization, LM uses monotone cubic Hermite splines.

Salac and Lu's method (SLM) was presented in [11] and is a different approach than MLM and LM. Consider the 2D case of two circles in near contact, see Figure 2. SLM reconstructs two independent level-set functions ϕ_1 and ϕ_2 for the two circles. The reconstructed functions are then used to calculate the curvature. Since the two reconstructed cones have no kinks, the curvature can be calculated with CD-2. For points close to both circles, a weighted average of the curvature from ϕ_1 and

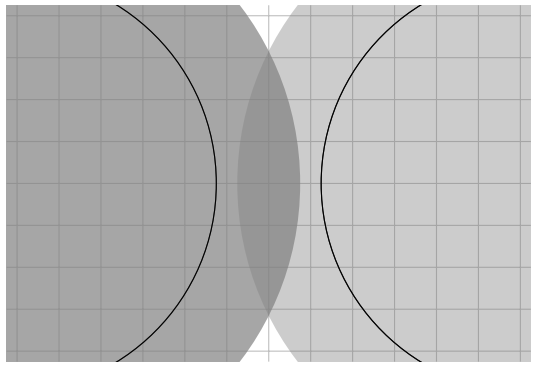


Fig. 2 Simple sketch of how SLM works. The two circles are represented by separate level-set functions.

from ϕ_2 is stored. For points close to only one circle, the appropriate curvature is stored. The weighted average is $\kappa = (\kappa_1\phi_2 + \kappa_2\phi_1)/(\phi_1 + \phi_2)$, where the subscripts refer to values calculated on the reconstructed level-set functions. This weighting will prefer κ_1 when closest to circle 1, and vice versa.

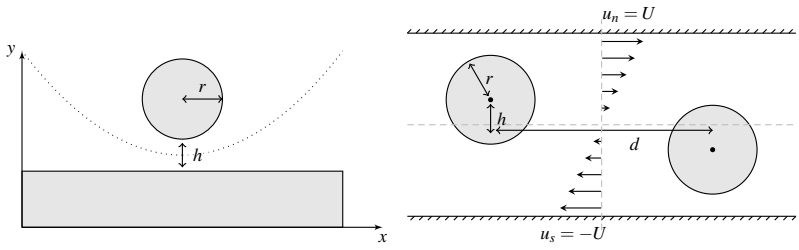
5 Comparison of the discretization schemes

5.1 A static disc above a rectangle

Consider a disc of radius r positioned at a distance h above a rectangle, see Figure 3(a). In this case, only the level-set function and the geometrical quantities are considered. None of the governing equations (1), (2) and (6) to (8) are solved.

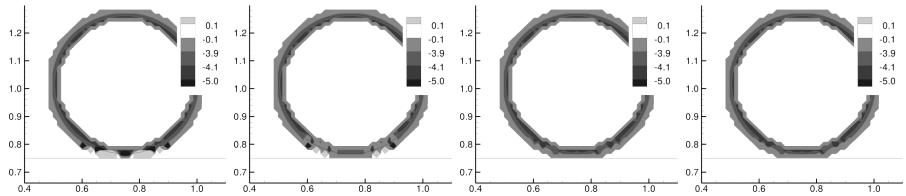
The parameters used for this case are $r = 0.25$ m and $h = \Delta x$. The domain is 1.5 m $\times$ 1.5m, and the rectangle height is 0.75 m. The grid size is 101×101.

Figure 4 shows a comparison of the calculated curvatures. The figure shows that CD-2 leads to large errors in the calculated curvatures in the areas that are close to two interfaces. In particular note that the sign of the curvature becomes wrong. The analytic curvature for this case is $\kappa = -1/r = -4$, and the curvature spikes seen for the standard discretization is of the order of $|\kappa| \sim \frac{1}{\Delta x} \simeq 67.3$. All of the improved methods give much better estimates of the curvature, as expected.



(a) Initial setup, first test case (b) Initial setup, second test case

Fig. 3 Initial setup for the circle and rectangle test, (a), and for the drop collision in shear flow test, (b). In (a), the dotted line depicts the kink location, and there is no flow. In (b) the flow is indicated by the velocity profile.



(a) CD-2 (b) MLM (c) LM (d) SLM

Fig. 4 A comparison of curvature calculations between standard discretization and the improved method. The standard discretization leads to large errors in the curvatures in areas that are close to two interfaces.

5.2 Drop collision in shear flow

Now consider two drops in a shear flow as depicted in Figure 3(b). Both drops have radius r and are initially placed a distance $d = 5r$ apart in the shear flow, where the flow velocity changes linearly from $u_s = -U < 0$ at the bottom wall to $u_n = U$ at the top wall. The computational domain is $12r \times 8r$, and the grid size is 241×161. The density and viscosity differences of the two phases are zero.

The shear flow is defined by the Reynolds number and the Capillary number,

$$Re = \frac{\rho U r}{\mu} \qquad \text{and} \qquad Ca = \frac{\mu U}{\sigma}. \tag{11}$$

The following results were obtained with $r = 0.5$ m, $h = 0.84r = 0.42$ m, $Re = 10$ and $Ca = 0.025$.

Figure 5 shows a comparison of the interface evolution and the curvature between the different discretization schemes. The first column shows the results with the CD-2. The next three columns show the results with the three improved schemes re-

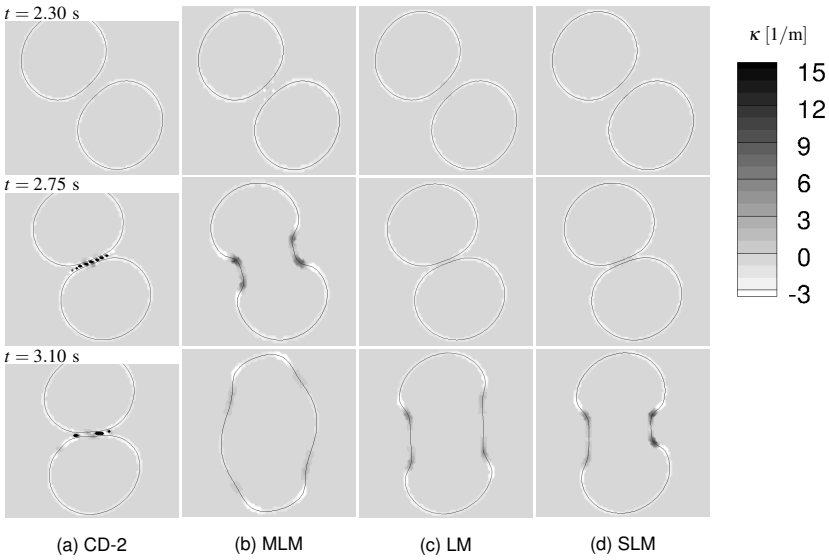
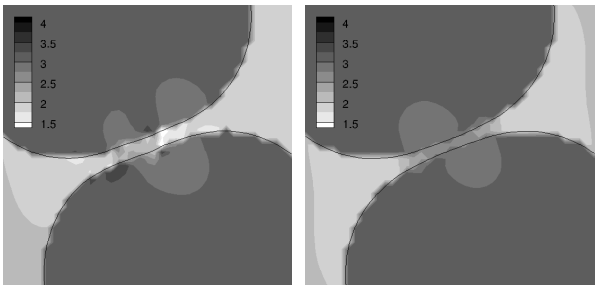


Fig. 5 A comparison between the different discretization schemes of the interface evolution and the curvature κ of drop collision in shear flow.

spetively. The kinks between the drops lead to curvature spikes with CD-2, whereas the improved discretizations calculate the curvature along the kink in a much more reliable manner. LM and SLM give very similar results. This is most likely due to the fact that both these methods calculate the curvature at the grid points and then interpolate, resulting in very similar algorithms as long as the curvature calculations are accurate. MLM on the other hand removes the interpolation step and calculates the curvature directly on the interface. Note that the difference is mainly that the MLM results in slightly earlier coalescence in the given case.

The curvature spikes in obtained with CD-2 are seen to prevent coalescence. This is due to the effect they have on the pressure field as displayed in Figure 6. Here it is shown that the errors in the curvature with CD-2 lead to an erroneous pressure field between the drops. The distortion of the pressure in the thin-film region leads to a flow into the film region that suppresses coalescence. The corresponding result with LM shows that when the pressure is not distorted, it leads to a flow directed out of the thin-film region.



(a) CD-2 (b) LM

Fig. 6 Comparison of the pressure field in the thin film between the droplets at $t = 2.75$ s. The contour legends indicate the pressure in Pa.

6 Conclusions

Three discretization schemes have been implemented to accurately calculate the curvature in regions close to kinks in the level-set function. It has been demonstrated in two test cases that the standard second-order central difference scheme (CD-2) leads to relatively severe errors across the kinks. Macklin and Lowengrub's method (MLM), Lervåg's method (LM), and Salac and Lu's method (SLM) all give better results. In the second test case where two droplets are put in a shear flow, CD-2 gives a qualitatively different result than all the three improved schemes due to an erroneous pressure field in the thin film region.

Acknowledgements

The authors acknowledge Bernhard Müller (Norwegian University of Science and Technology) and Svend Tollak Munkejord (SINTEF Energy Research) for valuable feedback on the manuscript.

This work was financed through the Enabling Low-Emission LNG Systems project, and the authors acknowledge the contributions of GDF SUEZ, Statoil and the Petromaks programme of the Research Council of Norway (193062/S60).

References

1. Fedkiw, R.P., Aslam, T., Merriman, B., Osher, S.: A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). Journal of Computational Physics **152**(2), 457–492 (1999). DOI 10.1006/jcph.1999.6236

2. Gottlieb, S., Shu, C.W., Tadmor, E.: Strong stability-preserving high-order time discretization methods. SIAM Review **43**, 89–112 (2001)
3. Hansen, E.B.: Numerical simulation of droplet dynamics in the presence of an electric field. Doctoral thesis, Norwegian University of Science and Technology, Department of Energy and Process Engineering, Trondheim (2005). ISBN 82-471-7318-2
4. Kang, M., Fedkiw, R.P., Liu, X.D.: A boundary condition capturing method for multiphase incompressible flow. Journal of Scientific Computing **15**(3), 323–360 (2000)
5. Lervåg, K.Y.: Calculation of interface curvature with the level-set method. In: Sixth National Conference on Computational Mechanics MekIT'11 (Trondheim, Norway) (23-24 May 2011)
6. Liu, X.D., Fedkiw, R.P., Kang, M.: A boundary condition capturing method for Poisson's equation on irregular domains. Journal of Computational Physics **160**, 151–178 (2000)
7. Macklin, P., Lowengrub, J.: Evolving interfaces via gradients of geometry-dependent interior Poisson problems: Application to tumor growth. Journal of Computational Physics **203**, 191–220 (2005)
8. Macklin, P., Lowengrub, J.: An improved geometry-aware curvature discretziation for level set methods: Application to tumor growth. Journal of Computational Physics **215**, 392–401 (2006)
9. Macklin, P., Lowengrub, J.S.: A new ghost cell/level set method for moving boundary problems: Application to tumor growth. Journal of Scientific Computing **35**, 266–299 (2008)
10. Osher, S., Sethian, J.A.: Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. Journal of Computational Physics **79**, 12–49 (1988)
11. Salac, D., Lu, W.: A local semi-implicit level-set method for interface motion. Journal of Scientific Computing **35**, 330–349 (2008)
12. Sethian, J.A., Smereka, P.: Level set methods for fluid interfaces. Annual Review of Fluid Mechanics **35**, 341–372 (2003)
13. Sussman, M., Smereka, P., Osher, S.: A level set approach for computing solutions to incompressible two-phase flow. Journal of Computational Physics **114**, 146–159 (1994)
14. Xu, J.J., Li, Z., Lowengrub, J., Zhao, H.K.: A level set method for interfacial flows with surfactants. Journal of Computational Physics **212**(2), 590–616 (2006)
15. Zhao, H.K., Chan, T., Merriman, B., Osher, S.: A variational level set approach to multiphase motion. Journal of Computational Physics **127**, 179–195 (1996)

C

Calculation of the interface curvature and normal vector with the level-set method

K. Y. Lervåg, B. Müller, and S. T. Munkejord
Published in Computers and Fluids, volume 84 (2013), 218–230.

Calculation of the interface curvature and normal vector with the level-set method

Karl Yngve Lervåg[a,*], Bernhard Müller[a], Svend Tollak Munkejord[b]

[a]*Department of Energy and Process Engineering, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway*
[b]*SINTEF Energy Research, P.O. Box 4761 Sluppen, NO-7465 Trondheim, Norway*

Abstract

This article addresses the use of the level-set method for capturing the interface between two fluids. One of the advantages of the level-set method is that the curvature and the normal vector of the interface can be readily calculated from the level-set function. However, in cases where the level-set method is used to capture topological changes, the standard discretization techniques for the curvature and the normal vector do not work properly. This is because they are affected by the discontinuities of the signed-distance function half-way between two interfaces. This article addresses the calculation of normal vectors and curvatures with the level-set method for such cases. It presents a discretization scheme based on the geometry-aware curvature discretization by Macklin and Lowengrub [1]. As the present scheme is independent of the ghost-fluid method, it becomes more generally applicable, and it can be implemented into an existing level-set code more easily than Macklin and Lowengrub's scheme [1]. The present scheme is compared with the second-order central-difference scheme and with Macklin and Lowengrub's scheme [1], first for a case with no flow, then for a case where two drops collide in a 2D shear flow, and finally for a case where two drops collide in an axisymmetric flow. In the latter two cases, the Navier-Stokes equations for incompressible two-phase flow are solved. The article also gives a comparison of the calculation of normal vectors with the direction difference scheme presented by Macklin and Lowengrub in [2] and with the present dis-

*Corresponding author
Email address: karl.y.lervag@ntnu.no (Karl Yngve Lervåg)
URL: http://folk.ntnu.no/lervag (Karl Yngve Lervåg)

cretization scheme. The results show that the present discretization scheme yields more robust calculations of the curvature than the second-order central difference scheme in areas where topological changes are imminent. The present scheme compares well to Macklin and Lowengrub's method [1]. The results also demonstrate that the direction difference scheme [2] is not always sufficient to accurately calculate the normal vectors.

Keywords: Level-set method, Curvature discretization, Normal-vector discretization, Curve-fitting discretization scheme, Finite differences, Ghost-fluid method.

1. Introduction

The level-set method was introduced by Osher and Sethian [3]. It is designed to implicitly track moving interfaces through an isocontour of a function defined in the entire domain. In particular, it is designed for problems in multiple spatial dimensions in which the topology of the evolving interface changes during the course of events, cf. [4].

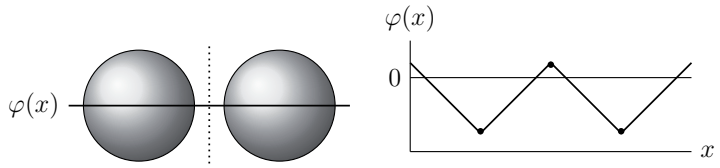
This article addresses the calculation of interface geometries with the level-set method. This method allows us to calculate the normal vector and the curvature of an interface directly as the first and second derivatives of the level-set function. These calculations are typically done with standard finite-difference methods. Since the level-set function is chosen to be a signed-distance function, in most cases it will have areas where it is not smooth. Consider for instance two colliding drops where the interfaces are captured with the level-set method, see Figure 1. The derivative of the level-set function will not be defined at the points outside the drops that have an equal distance to both drops. When the drops are in near contact, this discontinuity in the derivative will lead to significant errors when calculating the interface geometries with standard finite-difference methods. For convenience the areas where the derivative of the level-set function is not defined will hereafter be referred to as kinks.

To the authors knowledge, this issue was first described in [2], where the level-set method was used to model tumour growth. Here Macklin and Lowengrub presented a direction difference to treat the discretization across kinks for the normal vector and the curvature. They later presented an improved method where curve fitting was used to calculate the curvatures [1]. This was further expanded to include the normal vectors [5].

An alternative method to avoid the kinks is presented in [6], where a level-set extraction technique is presented. This technique uses an extraction algorithm to reconstruct separate level-set functions for each distinct body.

Accurate calculation of the curvature is important in many applications, in particular in curvature-driven flows. There are several examples in the literature of methods that improve the accuracy of the curvature calculations, but that do not consider the problem with the discretization across the kinks. The authors in [7] use a coupled level-set and volume-of-fluid method based on a fixed Eulerian grid, and they use a height function to calculate the curvatures. In [8] a refined level-set grid method is used to study two-phase flows on structured and unstructured grids for the flow solver. An interface-projected curvature-evaluation method is presented to achieve converging calculation of the curvature. Marchandise et. al [9] adopt a discontinuous Galerkin method and a pressure-stabilized finite-element method to solve the level-set equation and the Navier-Stokes equations, respectively. They develop a least-squares approach to calculate the normal vector and the curvature accurately, as opposed to using a direct derivation of the level-set function. This method is used by Desjardins et. al in [10], where they show impressive results for simulations of turbulent atomization.

This article is a continuation of the work presented in [11]. It applies the level-set method and the ghost-fluid method to incompressible two-phase flow in two dimensions. A curve-fitting discretization scheme is presented which is based on the geometry-aware discretization given in [1]. This scheme is mainly applied to the curvature discretization. The normal vectors are calculated both with the direction difference described in [2] and with a combination of the direction difference and the curve-fitting discretization



(a) Drops in near contact (b) A slice of the level-set function

Figure 1: (a) Two drops in near contact. The dotted line marks a region where the derivative of the level-set function is not defined. (b) A one-dimensional slice of the level-set function $\varphi(x)$. The dots mark points where the derivative of $\varphi(x)$ is not defined.

scheme.

The main advantage of the present scheme compared to the geometry-aware discretization [1] is that it is independent of the ghost-fluid method. That is, in [1] Macklin and Lowengrub calculate the curvature values directly on the interface when it is needed by the ghost-fluid method, whereas we compute the curvature values at the global grid points, indendent of the ghost-fluid method. Because of this, the scheme can be implemented more easily into existing Navier-Stokes codes employing the level-set method, since only small parts of the existing codes need modification. It is also more generally applicable, for instance it can be used with the continuum surface-force method [15]. Further, it allows for more accurate curvature values in models that require curvature values on the grid instead of on the interface, e.g. surfactant models [12–14].

The article starts by briefly describing the governing equations for two-phase flow and the level-set method in Section 2. It continues in Section 3 with a description of the numerical methods that are used for their solution. Then the discretization schemes for the normal vector and the curvature are presented in Section 4, followed by a detailed description of the method for curvature discretization in Section 5. Section 6 gives a convergence test and a comparison of the present discretization scheme with the second-order central difference scheme and Macklin and Lowengrub's scheme [1], first on static interfaces in near contact, then on two drops colliding in a 2D shear flow, and finally on a case where two drops collide in an axisymmetric flow. The section is concluded with a comparison of the direction difference scheme [2] with a combination of the direction difference and the curve-fitting discretization schemes for calculating normal vectors. Finally in Section 7 concluding remarks are made.

2. Governing equations

2.1. Navier-Stokes equations for two-phase flow

Consider a two-phase domain $\Omega = \Omega^+ \cup \Omega^-$ where Ω^+ and Ω^- denote the regions occupied by the respective phases. The domain is divided by an interface $\Gamma = \delta\Omega^+ \cap \delta\Omega^-$ as illustrated in Figure 2. The governing equations for incompressible and immiscible two-phase flow in the domain Ω with an

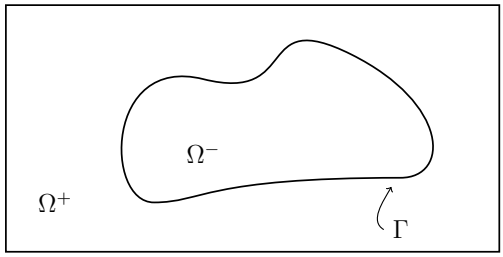


Figure 2: Illustration of a two-phase domain: The interface Γ separates the two phases, one in Ω^+ and the other in Ω^-.

interface force on the interface Γ can be stated as

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0, \tag{1}$$

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{u} \right) = -\boldsymbol{\nabla} p + \boldsymbol{\nabla} \cdot \left(\mu \left(\boldsymbol{\nabla} \boldsymbol{u} + (\boldsymbol{\nabla} \boldsymbol{u})^T \right) \right) + \rho \boldsymbol{f}_b + \int_\Gamma \sigma \kappa \boldsymbol{n} \, \delta(\boldsymbol{x} - \boldsymbol{x}_I(s)) \, \mathrm{d}s, \tag{2}$$

where $\boldsymbol{u}$ is the velocity vector, p is the pressure, $\boldsymbol{f}_b$ is the specific body force, σ is the coefficient of surface tension, κ is the curvature, $\boldsymbol{n}$ is the normal vector which points to Ω^+, δ is the Dirac Delta function, $\boldsymbol{x}_I$ is a parametrization of the interface, ρ is the density and μ is the viscosity. These equations are often called the Navier-Stokes equations for incompressible two-phase flow.

It is assumed that the density and the viscosity are constant in each phase, but they may be discontinuous across the interface. The interface force and the discontinuities in the density and the viscosity lead to a set of interface conditions,

$$[\boldsymbol{u}] = 0, \tag{3}$$

$$[p] = 2[\mu] \boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{n} + \sigma \kappa, \tag{4}$$

$$[\mu \boldsymbol{\nabla} \boldsymbol{u}] = [\mu] \big((\boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{n}) \boldsymbol{n} \otimes \boldsymbol{n} + (\boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t}) \boldsymbol{n} \otimes \boldsymbol{t} - (\boldsymbol{n} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t}) \boldsymbol{t} \otimes \boldsymbol{n} + (\boldsymbol{t} \cdot \boldsymbol{\nabla} \boldsymbol{u} \cdot \boldsymbol{t}) \boldsymbol{t} \otimes \boldsymbol{t} \big), \tag{5}$$

$$[\boldsymbol{\nabla} p] = 0, \tag{6}$$

where $\boldsymbol{t}$ is the tangent vector along the interface, $\otimes$ denotes the dyadic product, and $[\,\cdot\,]$ denotes the jump across an interface, that is

$$[\mu] \equiv \mu^+ - \mu^-. \tag{7}$$

See [16, 17] for more details and a derivation of the interface conditions.

2.2. Level-set method

The interface is captured with the zero level set of the level-set function $\varphi(\boldsymbol{x}, t)$, which is prescribed as a signed-distance function. That is, the interface is given by

$$\Gamma = \{\, (\boldsymbol{x}, t) \mid \varphi(\boldsymbol{x}, t) = 0 \,\}, \quad \boldsymbol{x} \in \Omega, \quad t \in \mathbb{R}^+, \tag{8}$$

and for any $t \geq 0$,

$$\varphi(\boldsymbol{x}, t) \begin{cases} < 0 & \text{if } \boldsymbol{x} \in \Omega^- \\ = 0 & \text{if } \boldsymbol{x} \in \Gamma \\ > 0 & \text{if } \boldsymbol{x} \in \Omega^+ \end{cases}. \tag{9}$$

The interface is updated by solving an advection equation for φ,

$$\frac{\partial \varphi}{\partial t} + \hat{\boldsymbol{u}} \cdot \boldsymbol{\nabla} \varphi = 0, \tag{10}$$

where $\hat{\boldsymbol{u}}$ is the velocity at the interface extended to the entire domain. The interface velocity is extended from the interface to the domain by solving

$$\frac{\partial \hat{\boldsymbol{u}}}{\partial \tau} + S(\varphi) \boldsymbol{n} \cdot \boldsymbol{\nabla} \hat{\boldsymbol{u}} = 0, \quad \hat{\boldsymbol{u}}_{\tau=0} = \boldsymbol{u}, \tag{11}$$

to steady state, cf. [18]. Here τ is pseudo-time and S is a smeared sign function which is equal to zero at the interface,

$$S(\varphi) = \frac{\varphi}{\sqrt{\varphi^2 + 2\Delta x^2}}. \tag{12}$$

When equation (10) is solved numerically, the level-set function loses its signed-distance property due to numerical dissipation. The level-set function is therefore reinitialized regularly by solving

$$\begin{aligned} \frac{\partial \varphi}{\partial \tau} + S(\varphi_0)(|\boldsymbol{\nabla} \varphi| - 1) &= 0, \\ \varphi(\boldsymbol{x}, 0) &= \varphi_0(\boldsymbol{x}), \end{aligned} \tag{13}$$

to steady state as proposed in [19]. Here φ_0 is the level-set function that needs to be reinitialized.

One of the advantages of the level-set method is that normal vectors and curvatures can be readily calculated from the level-set function, i.e.

$$\boldsymbol{n} = \frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|}, \tag{14}$$

$$\kappa = \boldsymbol{\nabla} \cdot \left(\frac{\boldsymbol{\nabla}\varphi}{|\boldsymbol{\nabla}\varphi|} \right). \tag{15}$$

3. Numerical methods

The Navier-Stokes equations, (1) and (2), are solved by a projection method on a staggered grid as described in [17, Chapter 5.1.1]. The spatial terms are discretized by the second-order central difference scheme, except for the convective terms which are discretized by a fifth-order WENO scheme. The temporal discretization is done with explicit strong stability-preserving Runge-Kutta (SSP RK) schemes, see [20]. A three-stage third-order SSP-RK method is used for the Navier-Stokes equations (1) and (2), and a four-stage second-order SSP-RK method is used for the level-set equations (10), (11) and (13).

The method presented in [21] is used to improve the computational speed. The method is often called the narrow-band method, since the level-set function is only updated in a narrow band across the interface at each time step.

The interface conditions are treated in a sharp fashion with the Ghost-Fluid Method (GFM), which incorporates the discontinuities into the discretization stencils by altering the stencils close to the interfaces. For instance, the GFM requires that a term is added to the stencil on the right-hand side of the Poisson equation for the pressure. Consider a one-dimensional case where $[\rho] = [\mu] = 0$ and where the interface lies between x_i and x_{i+1}. In this case,

$$\frac{p_{i+1} - 2p_i + p_{i-1}}{\Delta x^2} = f_k \pm \frac{\sigma \kappa_\Gamma}{\Delta x^2}, \tag{16}$$

where f_k is the general right-hand side value and κ_Γ is the curvature at the interface. The sign of the added term depends on the sign of $\varphi(x_i)$. See [16] for more details on how the GFM is used for the Navier-Stokes equations and [22] for a description on how to use the GFM for a variable-coefficient Poisson equation.

The normal vector and the curvature defined by equations (14) and (15) are typically discretized by the second-order central difference scheme, cf. [4,

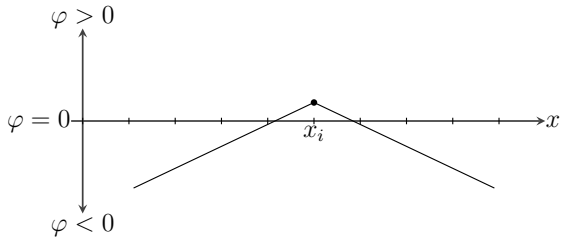


Figure 3: A level-set function that has one point where the derivative is discontinuous.

12, 16]. The curvatures are calculated on the grid nodes and then interpolated with simple linear interpolation to the interface, e.g. for κ_Γ in equation (16),

$$\kappa_\Gamma = \frac{|\varphi_i|\kappa_{i+1} + |\varphi_{i+1}|\kappa_i}{|\varphi_i| + |\varphi_{i+1}|}. \tag{17}$$

If the level-set method is used to capture non-trivial geometries, the level-set function will in general contain areas where it is not smooth, i.e. kinks. This is depicted in Figure 3, which shows a level-set function in a one-dimensional domain that captures two interfaces, one on each side of the grid point x_i. The kink at x_i will lead to potentially large errors with the standard discretization both for the curvature and the normal vector. The errors in the curvature will lead to erroneous pressure jumps at the interfaces, and the errors in the normal vector affects both the discretized interface conditions and the advection of the level-set function. If the level-set method is used to study for example coalescence and breakup of drops, these errors may severely affect the simulations.

It should be noted that the kinks that appear far from any interfaces are handled by ensuring that the denominators do not become zero, as explained in [23, Sections 2.3 to 2.4]. This works fine, since only the values of the curvature at the grid nodes adjacent to any interface are used. Also, the normal vector only needs to be accurate close to the interface due to the narrow-band approach.

4. Improved discretization of geometrical quantities

The previous section explained why it is necessary to develop new discretization schemes for the normal vector and the curvature that can handle

kinks in the level-set function. This section will give an overview of the curve-fitting discretization scheme and how it is applied to calculate the curvature. It will then give a brief presentation of the direction difference which is used to calculate the normal vector. A note is finally given on how to use the curve-fitting discretization scheme to calculate the normal vector.

4.1. The curvature

The curvature is calculated with a discretization that is based on the improved geometry-aware curvature discretization presented by Macklin and Lowengrub [1]. This is a method where the curvature is calculated at the interfaces directly with the use of a least-squares curve parametrization of the interface. The curve parametrization is used to create a local level-set function from which the curvature is calculated using standard discretization techniques. The local level-set function only depends on one interface and is therefore free of kinks.

The main motivation behind the present method is to improve the curvature calculations specifically at the grid points, as these values may be important for other models. Examples of such cases are the modelling of interfacial flows with surfactants [12–14].

The main difference between the present method and that of Macklin and Lowengrub [1] is that they modify the GFM to calculate the curvature at the interface directly, whereas the present method only changes the procedure to calculate the curvature at specific grid points. In other words, Macklin and Lowengrub calculate κ_Γ in equation (16) directly with a parametrized curve, whereas the present method uses parametrized curves to calculate κ_i and κ_{i+1}. The present method is therefore independent of the GFM, which makes it easier to adopt it into existing level-set codes.

An important consequence of not calculating the curvature directly on the interface is that it becomes more important to have an accurate representation of the interface. This is due to the fact that the point $\boldsymbol{x}_i$ where the curvature is calculated is not on the interface, so the calculation becomes less local. Thus the parametrization needs to be more accurate at a larger distance from $\boldsymbol{x}_i$. The curve-fitting discretization scheme presented here uses monotone cubic Hermite splines to parametrize the curve. The least-square parametrization used in [1] is only accurate very close to the point where the curvature needs to be calculated. The Hermite spline is more accurate along the entire interface representation.

The algorithm to calculate the curvature at $\boldsymbol{x}_{i,j}$ can be summarized as follows. The details are explained in the next section.

1. If $Q_{i+n,j+m} \leq \eta$, where $n = -1, 0, 1$ and $m = -1, 0, 1$, then it is safe to use the central-difference discretization. Otherwise continue to the next step.
2. Locate the closest interface, Γ.
3. Find a set of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \Gamma$.
4. Create a parametrization $\boldsymbol{\gamma}(s)$ of the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.
5. Calculate a local level-set function based on the parametrization $\boldsymbol{\gamma}(s)$.
6. Use central-difference discretization on the local level-set function to calculate the curvature.

4.2. The normal vector

This section will describe two methods to calculate the normal vector close to kinks. The direction difference [2] will be described first. Then a method based on the curve-fitting scheme will be presented.

4.2.1. Direction difference

The direction-difference scheme uses a quality function to ensure that the difference stencils never cross kinks. The basic strategy is to use a combination of central differences and one-sided differences based on the values of a quality function,

$$Q(\boldsymbol{x}) = |1 - |\boldsymbol{\nabla}\varphi(\boldsymbol{x})||\,, \tag{18}$$

which is approximated with central differences. The quality function effectively detects the areas where the level-set function differs from the signed-distance function. Let $Q_{i,j} = Q(\boldsymbol{x}_{i,j})$ and $\eta > 0$, then $Q_{i,j} > \eta$ can be used to detect kinks. The parameter η is tuned such that the quality function will detect all the kinks. The value $\eta = 0.1$ is used in the present work.

The quality function is used to define a direction function,

$$\boldsymbol{D}(\boldsymbol{x}_{i,j}) = (D_x(\boldsymbol{x}_{i,j}), D_y(\boldsymbol{x}_{i,j})), \tag{19}$$

where

$$D_x(\boldsymbol{x}_{i,j}) = \begin{cases} -1 & \text{if } Q_{i-1,j} < \eta \text{ and } Q_{i+1,j} \geq \eta, \\ 1 & \text{if } Q_{i-1,j} \geq \eta \text{ and } Q_{i+1,j} < \eta, \\ 0 & \text{if } Q_{i-1,j} < \eta \text{ and } Q_{i,j} < \eta \text{ and } Q_{i+1,j} < \eta, \\ 0 & \text{if } Q_{i-1,j} \geq \eta \text{ and } Q_{i,j} \geq \eta \text{ and } Q_{i+1,j} \geq \eta, \\ \text{undetermined} & \text{otherwise.} \end{cases} \tag{20}$$

$D_y(\boldsymbol{x}_{i,j})$ is defined in a similar manner. If D_x or D_y is undetermined, $\boldsymbol{D}(\boldsymbol{x}_{i,j})$ is chosen as the vector normal to $\boldsymbol{\nabla}\varphi(\boldsymbol{x}_{i,j})$. It is normalized, and the sign is chosen such that it points in the direction of best quality. See [2] for more details.

The direction difference is now defined as

$$\partial_x f_{i,j} = \begin{cases} \frac{f_{i,j}-f_{i-1,j}}{\Delta x} & \text{if } D_x(x_i, y_j) = -1, \\ \frac{f_{i+1,j}-f_{i,j}}{\Delta x} & \text{if } D_x(x_i, y_j) = 1, \\ \frac{f_{i+1,j}-f_{i-1,j}}{2\Delta x} & \text{if } D_x(x_i, y_j) = 0, \end{cases} \tag{21}$$

where $f_{i,j}$ is a piecewise smooth function. The normal vector is calculated using the direction difference on φ, which is equivalent to using central differences in smooth areas and one-sided differences in areas close to the kinks. This method makes sure that the differences do not cross any kinks, and the normal vector can be accurately calculated even close to a kink.

4.2.2. Curve-fitting scheme

The direction difference is an elegant scheme which performs well in most cases. However, it will be shown later that in some rare cases where both direction functions are undetermined, this discretization scheme may become very inaccurate. An alternative is to use the curve-fitting discretization scheme on the normal vectors. But since this method starts by locating the closest interface with a breadth-first search (see next section), it will be slow when it is used far from any interfaces. It is therefore proposed to use a combination of the direction difference and the curve-fitting discretization scheme.

5. The curve-fitting discretization scheme

5.1. Locating the closest interface

A breadth-first search is used to to identify the closest interface, see Figure 4. Let $\boldsymbol{x}_0$ denote the starting point and $\boldsymbol{x}_1$ denote the desired point on the closest interface. The search iterates over all the eight edges from $\boldsymbol{x}_0$ to its neighbours and tries to locate an interface which is identified by a change of sign of $\varphi(\boldsymbol{x})$. If more than one interface is found, $\boldsymbol{x}_1$ is chosen to be the point that is closest to $\boldsymbol{x}_0$. If no interfaces are located the search continues at the next depth. The search continues in this manner until an interface is found. Note that this algorithm does not in general return the point on the interface which is closest to $\boldsymbol{x}_0$.

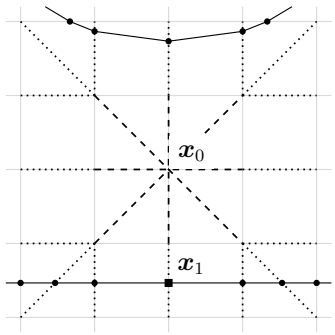


Figure 4: Sketch of a breadth-first search. The dashed lines depict the edges that are searched first, the dotted lines depict the edges that are searched next and the solid lines depict two interfaces. The circular dots mark where the algorithm finds interface points, and the rectangular dot marks the point which is returned for the depicted case.

The crossing points between the grid edges and the interfaces are found with linear and bilinear interpolation. E.g. if an interface crosses the edge between (i, j) and $(i, j + 1)$ at $\boldsymbol{x}_I$, the interface point is found by linear interpolation,

$$\boldsymbol{x}_I = \boldsymbol{x}_{i,j} + \theta(0, \Delta x), \tag{22}$$

where

$$\theta = \frac{\varphi(\boldsymbol{x}_{i,j})}{\varphi(\boldsymbol{x}_{i,j}) - \varphi(\boldsymbol{x}_{i,j+1})}. \tag{23}$$

In the diagonal cases the interface point is found with bilinear interpolation along the diagonal. This leads to

$$\boldsymbol{x}_I = \boldsymbol{x}_{i,j} + \theta(\Delta x, \Delta x), \tag{24}$$

where θ is the solution of

$$\alpha_1\theta^2 + \alpha_2\theta + \alpha_3 = 0. \tag{25}$$

The α values depend on the grid cell. For instance, when searching along the diagonal between (i, j) and $(i + 1, j + 1)$ the α values will be

$$\alpha_1 = \varphi_{i,j} - \varphi_{i+1,j} - \varphi_{i,j+1} + \varphi_{i+1,j+1}, \tag{26}$$
$$\alpha_2 = \varphi_{i+1,j} + \varphi_{i,j+1} - 2\varphi_{i,j}, \tag{27}$$
$$\alpha_3 = \varphi_{i,j}. \tag{28}$$

5.2. Searching for points on an interface

When an interface and a corresponding point $\boldsymbol{x}_1$ on the interface are found, the next step is to find a set of points $\boldsymbol{x}_2, \ldots, \boldsymbol{x}_k, \ldots, \boldsymbol{x}_n$ on the same interface. The points should be ordered such that when traversing the points from $k = 1$ to $k = n$, the phase with $\varphi(\boldsymbol{x}) < 0$ is on the left-hand side. Note that the ordering of the points may be done after all the points are found. Three criteria are used when searching for new points:

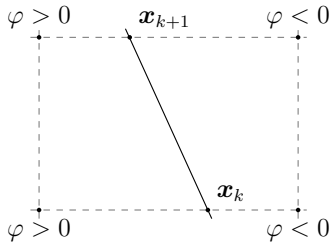
1. The points are located on the grid edges.
2. The distance between $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1}$ for all k is greater than a given threshold μ.
3. The n points that are closest to $\boldsymbol{x}_0$ are selected, where $\boldsymbol{x}_0 = \boldsymbol{x}_{i,j}$ is the initial point where the curvature is to be calculated.

Let $\boldsymbol{x}_k \in \Gamma \cap [x_i, x_{i+1}) \times [y_j, y_{j+1})$ be given. To find a new point $\boldsymbol{x}_{k+1}$ on Γ, a variant of the marching-squares algorithm[1] is used. Given $\boldsymbol{x}_k$ and a search direction which is either clockwise or counter clockwise, the algorithm searches for all the points where an interface crosses the edges of the mesh rectangle $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$. In most cases there will be two such points and $\boldsymbol{x}_k$ is one of them. $\boldsymbol{x}_{k+1}$ is then selected based on the search direction. If $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, the search is continued in the adjacent mesh rectangle. The search process is depicted in Figure 5(a).

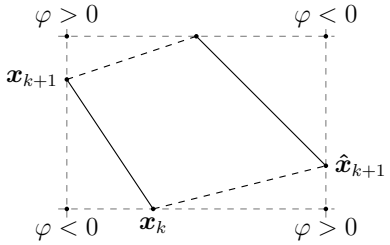
In some rare cases the algorithm must handle the ambiguous case depicted in Figure 5(b). In these cases there are four interface crossing-points and two solutions. Either solution is valid, and it is not possible to say which solution is better. The current implementation selects the first solution that it finds, which will be in all practical sense a random choice. Note that the ambiguous cases only occur when two interfaces cross a single grid cell. The ambiguity comes from the fact that the level-set method is not able to resolve the interfaces on a sub-cell resolution.

It was found that $n = 7$ points where necessary in order to ensure that the closest points on the interface with respect to the different grid points are captured with the spline parametrization.

[1]The marching-squares algorithm is an equivalent two-dimensional formulation of the well known marching-cubes algorithm presented in [24]. The algorithm was mainly developed for use in computer graphics.



(a) Marching squares



(b) Ambiguous case

Figure 5: (a) The search starts by locating the two points where the interface crosses the mesh rectangle. $\boldsymbol{x}_k$ is the starting point, and if the search is counter clockwise it will select $\boldsymbol{x}_{k+1}$ as depicted. If the search is clockwise, it will select $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$, and the search continues in the adjacent mesh rectangle $[x_i, x_{i+1}] \times [y_{j-1}, y_j]$. (b) An example of an ambiguous case. The solid black lines and the dashed black lines are two equally valid solutions for how the interfaces cross the mesh rectangle. If the search starts at $\boldsymbol{x}_k$ and searches counter clockwise, then both $\hat{\boldsymbol{x}}_{k+1}$ and $\boldsymbol{x}_{k+1}$ are valid solutions.

5.3. Curve fitting

Cubic Hermite splines are used to fit a curve to the set of points

$$X_{0,m} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}. \tag{29}$$

Let the curve parametrization be denoted $\boldsymbol{\gamma}(s)$ for $0 < s < 1$. A cubic spline is a parametrization where

$$\boldsymbol{\gamma}(s) = \begin{cases} \boldsymbol{\gamma}_1(s) & s_0 \leq s < s_1, \\ \boldsymbol{\gamma}_2(s) & s_1 \leq s < s_2, \\ \vdots & \\ \boldsymbol{\gamma}_m(s) & s_{m-1} \leq s \leq s_m, \end{cases} \tag{30}$$

where $0 = s_0 < s_1 < \cdots < s_m = 1$

$$\boldsymbol{\gamma}(s_i) = \boldsymbol{x}_i, \quad 0 \leq i \leq m, \tag{31}$$

and each interpolant $\boldsymbol{\gamma}_i(s) = (x_i(s), y_i(s))$ is a third-order polynomial. A Hermite spline is a spline where each interpolant is in Hermite form, see [25, Chapter 4.5]. The interpolants are created by solving the equations

$$\boldsymbol{\gamma}_i(s) = h_{00}(s)\boldsymbol{x}_{i-1} + h_{01}(s)\boldsymbol{x}_i + h_{10}(s)\boldsymbol{m}_{i-1} + h_{11}(s)\boldsymbol{m}_i, \tag{32}$$

for $1 \leq i \leq m$, where $\boldsymbol{m}_i$ is the curve tangents and h_{00}, h_{01}, h_{10} and h_{11} are Hermite basis polynomials,

$$\begin{aligned} h_{00}(s) &= 2s^3 - 3s^2 + 1, \\ h_{01}(s) &= s^3 - 2s^2 + s, \\ h_{10}(s) &= -2s^3 + 3s^2, \\ h_{11}(s) &= s^3 - s^2. \end{aligned} \tag{33}$$

The choice of the tangents is non-unique, and there are several possible options for a cubic Hermite spline.

It is essential that the spline is properly oriented. This is because we require to find both the distance and the position of a point on the grid relative to the spline in order to define a local level-set function. The orientation of the spline $\boldsymbol{\gamma}(s)$ is defined such that when s increases, Ω^- is to the left.

To ensure that our curve is properly oriented, the tangents are chosen as described in [26]. This will ensure monotonicity for each component as long

as the input data is monotone. The tangents are modified as follows. First the slopes of the secant lines between successive points are computed,

$$\boldsymbol{d}_i = \frac{\boldsymbol{x}_i - \boldsymbol{x}_{i-1}}{s_i - s_{i-1}} \tag{34}$$

for $1 \leq i \leq m$. Next the tangents are initialized as the average of the secants at every point,

$$\boldsymbol{m}_i = \frac{\boldsymbol{d}_i + \boldsymbol{d}_{i+1}}{2} \tag{35}$$

for $1 \leq i \leq m-1$. The curve tangents at the endpoints are set to $\boldsymbol{m}_0 = \boldsymbol{d}_1$ and $\boldsymbol{m}_m = \boldsymbol{d}_m$. Finally let k pass from 1 through $m-1$ and set $\boldsymbol{m}_k = \boldsymbol{m}_{k+1} = 0$ where $\boldsymbol{d}_k = 0$, and $\boldsymbol{m}_k = 0$ where $\operatorname{sign}(\boldsymbol{d}_k) \neq \operatorname{sign}(\boldsymbol{d}_{k+1})$.

5.4. Local level-set function

The local level-set function, here denoted as $\phi(\boldsymbol{x}_{i,j}) \equiv \phi_{i,j}$, is calculated at the grid points surrounding and including $\boldsymbol{x}_0 = \boldsymbol{x}_{i,j}$. The curvature is then calculated with the standard discretization stencil where ϕ is used instead of the global level-set function, φ.

A precise definition of ϕ is

$$\phi(\boldsymbol{x}_{i,j}) = \min_s \left(\hat{d}(\boldsymbol{x}_{i,j}, \boldsymbol{\gamma}(s)) \right) \tag{36}$$

where $\hat{d}(\boldsymbol{x}, \boldsymbol{\gamma}(s))$ is the signed-distance function, which is negative in phase one and positive in phase two. This function is calculated by first finding the minimum distance between $\boldsymbol{x}$ and $\boldsymbol{\gamma}(s)$ and then deciding the correct sign. The minimum distance is found by minimizing the norm

$$d(\boldsymbol{x}, \boldsymbol{\gamma}(s)) = \|\boldsymbol{x} - \boldsymbol{\gamma}(s)\|_2. \tag{37}$$

When $\boldsymbol{\gamma}$ is composed of cubic polynomials as is the case for cubic Hermite splines, the computation of the distance requires the solution of several fifth-order polynomial equations. Sturm's method (see [27, Section 11.3] or [28, Chapter XI,§2]) is employed to locate and bracket the solutions and a combined Newton-Raphson and bisection method is used to refine them. The correct sign is found by solving

$$\operatorname{sign}(\phi(\boldsymbol{x}_{i,j})) = \operatorname{sign}\left((\boldsymbol{x}_{i,j} - \boldsymbol{\gamma}(s)) \times \boldsymbol{t}_{\boldsymbol{\gamma}}(s) \right)_z, \tag{38}$$

where $\boldsymbol{t}_{\boldsymbol{\gamma}}(s)$ is the tangent vector of $\boldsymbol{\gamma}(s)$.

6. Verification and testing

This section presents results of calculating normal vectors and curvatures with the present discretization scheme. First a convergence test is considered. Then in the following three cases simulation results are compared with the second-order central difference scheme and Macklin and Lowengrub's scheme [1]. In the final case, the direction difference [2] is compared with the curve-fitting scheme outlined in Section 4.2.2.

6.1. Convergence test

The convergence of the present curve-fitting method is measured on a simple test case depicted in Figure 6. Here a disc of radius $r = 0.25$ and curvature $\kappa_0 = -4$ is placed a distance $h = 1.1\Delta x$ over a rectangle. The grid is aligned such that a grid cell fits between the disc and the rectangle, see Figure 7. The error for a grid of size $n \times n$ is defined as the 1-norm of the difference between κ_0 and the curvature values κ_i at the disc interface,

$$E_n = \frac{1}{m} \sum_{i=1}^{m} |\kappa_0 - \kappa_i|, \tag{39}$$

where m is the number of curvature values along the interface. The curvature values κ_i are calculated with linear interpolation (17) along the interface of the disc.

The convergence results for several different grid sizes n are shown in Table 1. The curvature calculated with central differences does not converge due to an error $\mathcal{O}(1/\Delta x)$ introduced by the kink region. It is seen that the present method converges, although the convergence order jumps between 0.6 and 3. Since h depends on the grid size, the case is slightly altered for each grid refinement. This might be one of the reasons that the convergence rate is slightly sporadic. Another reason is that the accuracy depends both on the second-order discretization stencil for the curvature, and on the locally generated level-set function. It is difficult to make a rigorous analysis of the accuracy of the latter, since it depends on several steps as described in the previous section. However, it is easy to see that the accuracy of the latter depends on the alignment of the interface with respect to the grid, and in particular the distance of the interface to the initial grid point $\boldsymbol{x}_{i,j}$. This could explain the varying convergence rate.

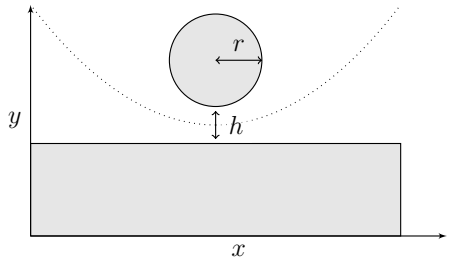


Figure 6: A disc that rests a distance h over a rectangle. The dotted line depicts the kink location.

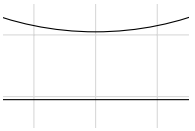


Figure 7: The grid and the rectangle are aligned on the grid such that there is one grid cell between the bodies.

Table 1: Convergence results for the curvature calculated for a disc that rests a distance $h = 1.1\Delta x$ above a rectangle. Results with the central difference on the left, and results with the present method on the right.

n	E_n	order
64	1.035	
128	1.213	-0.23
256	1.310	-0.11
512	1.351	-0.04
1024	1.401	-0.05
2048	1.394	0.01

n	E_n	order
64	$4.172 \cdot 10^{-2}$	
128	$1.123 \cdot 10^{-2}$	1.90
256	$3.950 \cdot 10^{-3}$	1.50
512	$2.583 \cdot 10^{-3}$	0.61
1024	$3.147 \cdot 10^{-4}$	3.00
2048	$1.164 \cdot 10^{-4}$	1.40

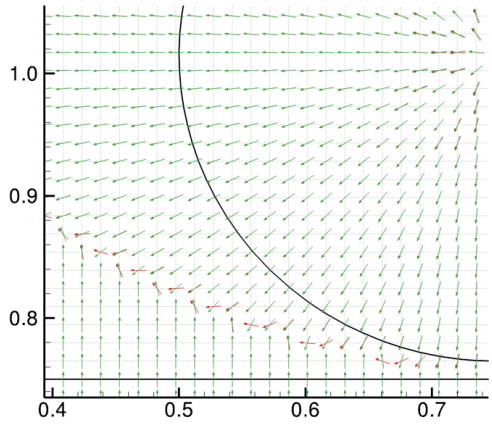


Figure 8: A comparison of the normal vectors as calculated with central differences [in red] and direction differences [in green]. The thick black lines depict the interfaces.

6.2. Disc and rectangle

Again consider the disc and rectangle, see Figure 6. It is of interest to compare the curvature and normal vector calculations, especially in the middle region close to the kink area. Only the level-set function and the geometrical quantities are considered, that is none of the governing equations is solved (equations (1), (2), (10), (11) and (13)). When h is small, the kinks along the dotted line will affect the discretization stencils as has been explained in Section 3.

The following results are obtained with $r = 0.25$ m and $h = \Delta x$. The domain is 1.5 m $\times$ 1.5m, and the straight line is positioned at $y = 0.75$ m. The grid size is 101×101.

Figure 8 shows a comparison of the calculated normal vectors. The results with central differences are depicted with red vectors and the results with direction differences are depicted with green vectors. The figure shows that the central-difference scheme yields much less accurate results for the normal vectors along the kink region than the direction-difference scheme.

Figure 9 shows a comparison of the calculated curvatures between the

central-difference scheme, Macklin and Lowengrub's method [1], and the present method. Note that for Macklin and Lowengrub's method [1] the values of the curvature at the grid points are first calculated with the central-difference scheme. Then for the interface locations that need special treatment the curvature values are copied from the interface to the adjacent grid points. This is done in order to compare the results. In all three cases, the curvature is set to zero at grid points that are far from any interface.

The figure shows that the present method yields similar results as Macklin and Lowengrub's method [1], which should be expected. Further, the central-difference scheme leads to large errors in the calculated curvatures in the areas that are close to two interfaces. In particular note that the sign of the curvature becomes wrong. The analytic curvature for this case is $\kappa = -1/r = -4$, and the curvature spikes seen with the central differences is in the order of $|\kappa| \sim \frac{1}{\Delta x} \simeq 67.3$. These spikes will lead to large errors in the pressure jumps through equation (4). The effect of these errors will become more clear in the next case.

6.3. Drop collision in shear flow

The second case considers drop collision in shear flow. The initial condition is sketched in Figure 10. Both drops have radius R and are initially placed at a distance $d = 5R$ apart in a shear flow. The initial flow velocity changes linearly from the bottom-wall velocity $u_s = -U < 0$ to the top-wall velocity $u_n = U$. The computational domain is $12R \times 8R$, and the grid size is 241×161.

The density and viscosity differences of the two phases are zero, and the shear flow is defined by the Reynolds number and the Capillary number,

$$Re = \frac{\rho U r}{\mu}, \qquad Ca = \frac{\mu U}{\sigma}. \tag{40}$$

The following results were obtained with $Re = 10$ and $Ca = 0.025$ for $R = 0.5$ m and $h = 0.42$ m. No-slip boundary conditions are used on all walls. The evolution of the flow field and the pressure is simulated by solving the Navier-Stokes equations (1) and (2) as described in Section 3.

Figure 11 shows the evolution of the interfaces and the velocity field for a simulation where the present method is used. We observe that the drops are deformed before they collide and that the evolution of the drops affects the velocity field. Figure 12 shows a comparison of the interface evolution and the curvature between the central-difference scheme and the present

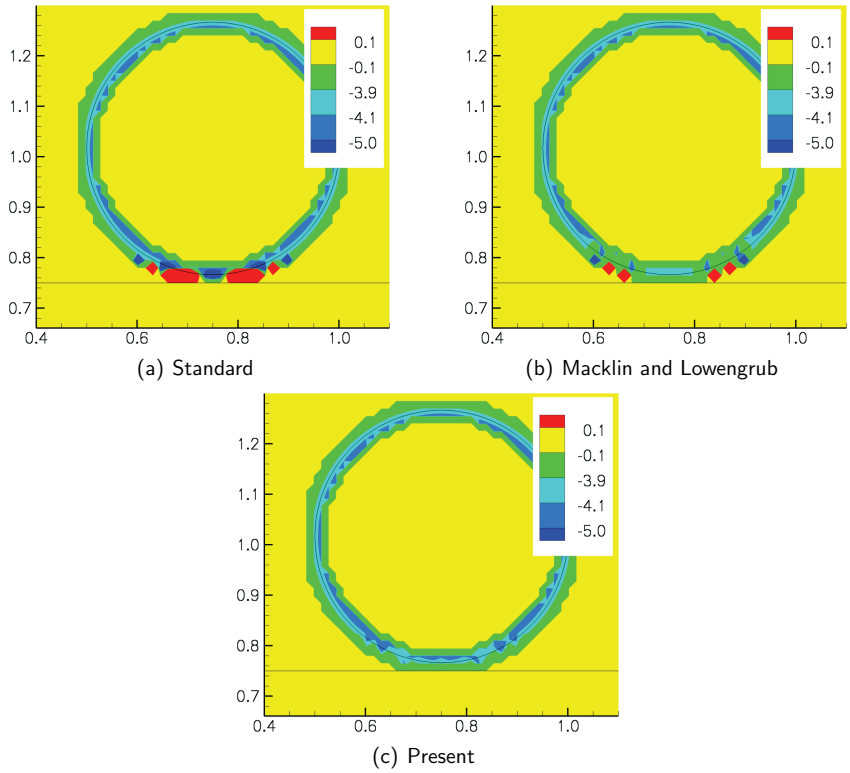


Figure 9: A comparison of curvature calculations between the central-difference scheme (Standard), Macklin and Lowengrub's method [1], and the present method. The central-difference scheme leads to large errors in the curvatures in areas that are close to two interfaces.

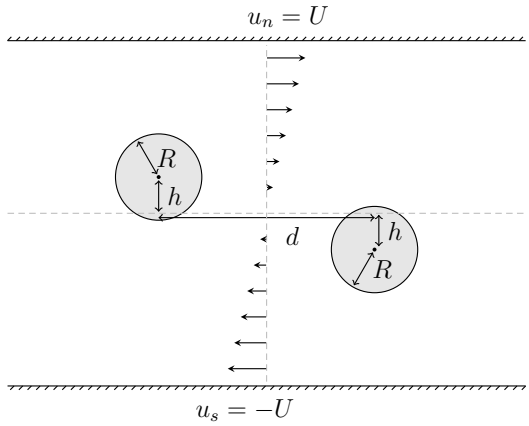


Figure 10: Sketch of the initial condition for the case with two drops in a shear flow.

method. The kinks between the drops again lead to curvature spikes for the central-difference scheme, whereas the improved discretization calculates the curvature along the kink in a much more reliable manner.

The curvature spikes in Figure 12 for the central-difference scheme are seen to prevent coalescence. This is due to the effect they have on the pressure field, cf. equation (16). Figure 13 shows the pressure field at $t = 2.75$ s. It can be seen that the pressure field for the central-difference scheme is distorted in the thin-film region. This distortion in the pressure leads to a flow in the film region which suppresses coalescence. The corresponding result for the present method shows that the pressure is not distorted. It is high in the centre of the thin-film region and lower at the edges. The pressure change induces a flow out of the region which is more as expected.

Finally, Figure 14 shows a comparison of Macklin and Lowengrub's method [1] and the present method. As can be expected, coalescence is observed also with Macklin and Lowengrub's method [1]. However, the time at which coalescence occurs is slightly different, which might be due to small differences in the flow of the thin film region just before coalescence.

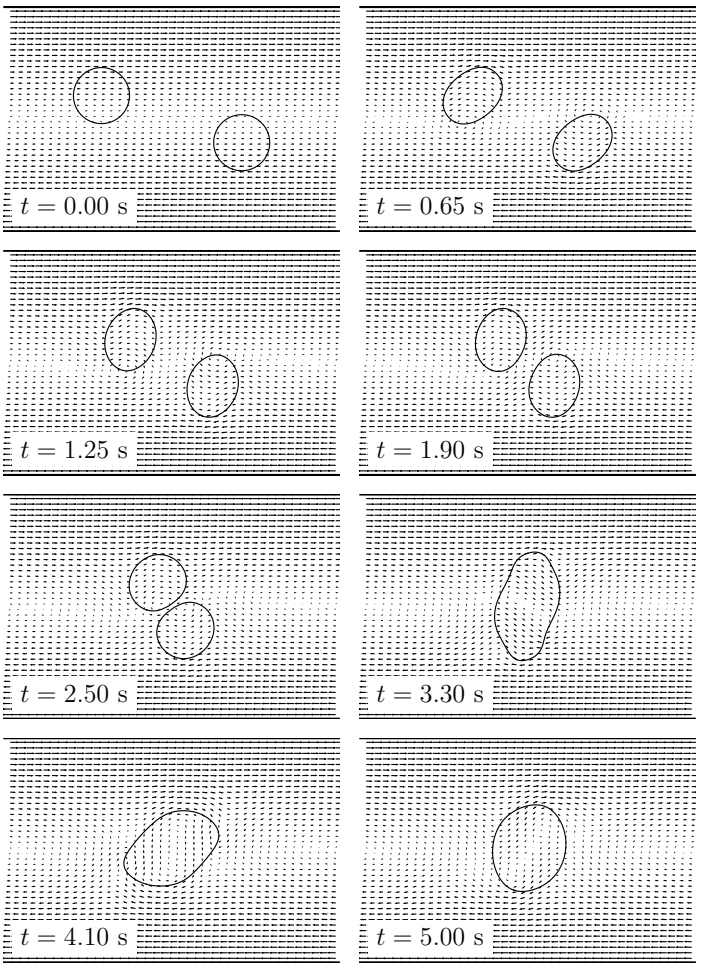


Figure 11: The evolution of the velocity field and the interfaces for drop collision in shear flow.

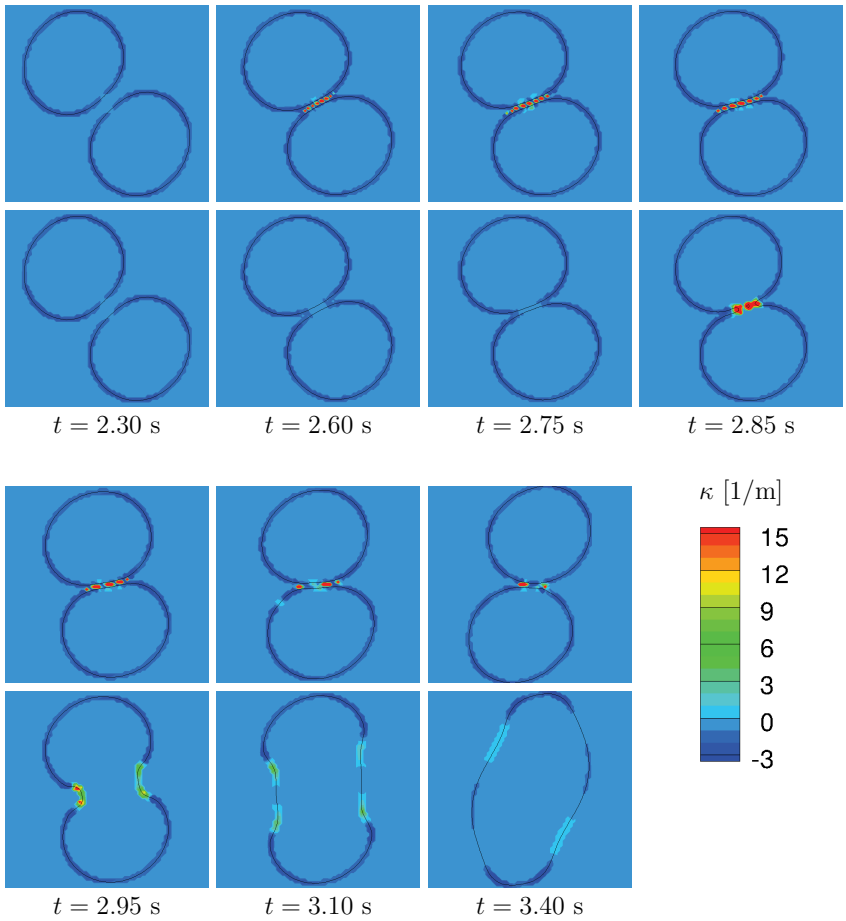
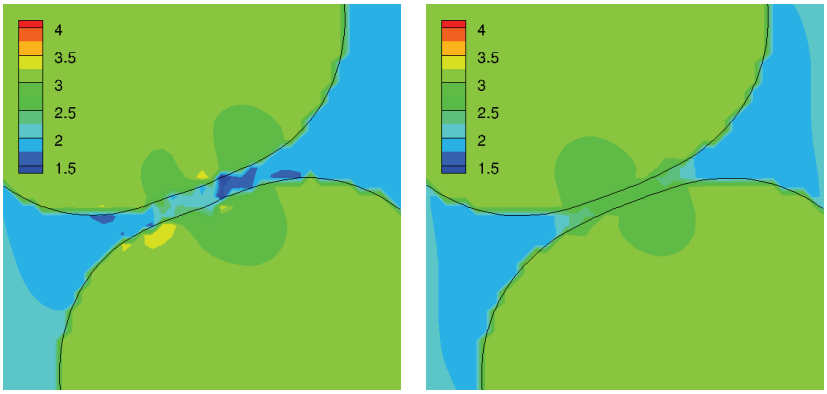


Figure 12: A comparison between the central-difference scheme (top row) and the present discretization scheme (bottom row) of the interface evolution and the curvature κ of drop collision in shear flow.



(a) Standard discretization (b) Improved method

Figure 13: Comparison of the pressure field in the thin film between the drops at $t = 2.75$ s. The contour legends indicate the pressure in Pa.

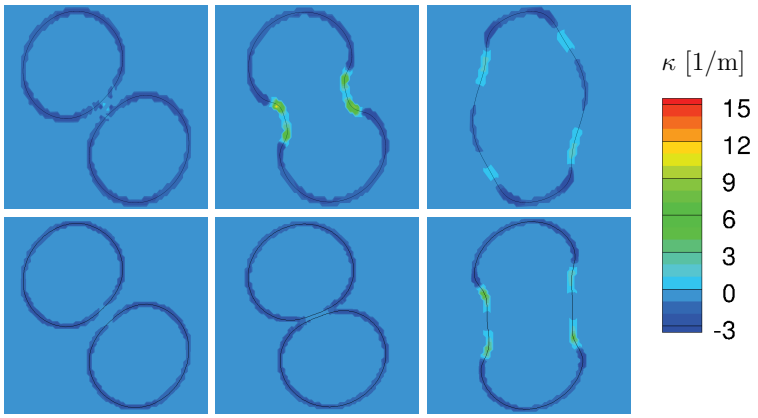


Figure 14: A comparison between Macklin and Lowengrub's method [1] (top row) and the present method (bottom row) of the interface evolution and the curvature κ of drop collision in shear flow.

6.4. Drop collision in axisymmetric flow

The third case considers the collision of two drops of radius R in an axisymmetric flow. The drops are initially placed at a distance $d = R$ apart in a linear flow field

$$\begin{aligned} u(r,z) &= \frac{r}{R}\,U_0, \\ v(r,z) &= -2\frac{z}{R}\,U_0. \end{aligned} \tag{41}$$

Here r is the radial coordinate, z is the axial coordinate, and U_0 is a scaling factor of the velocity. Figure 15 shows streamlines of the initial velocity field as well as the initial location of the drops with the centers at $z = \pm 0.75$ m.

The density and viscosity differences of the two phases are zero, and the case is defined by the Reynolds number and the Capillary number,

$$Re = \frac{\rho U_0 R}{\mu}, \qquad Ca = \frac{\mu U_0}{\sigma}. \tag{42}$$

The governing equations (1) and (2) are solved as explained in the previous sections, with some modifications: In axisymmetry the divergence and Laplacian operators become

$$\boldsymbol{\nabla} \cdot \boldsymbol{f} = \frac{1}{r}\frac{\partial}{\partial r}\left(r f_1\right) + \frac{\partial f_2}{\partial z}, \tag{43}$$

$$\boldsymbol{\Delta} g = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial g}{\partial r}\right) + \frac{\partial^2 g}{\partial z^2}, \tag{44}$$

where the subscripts indicate vector components, that is $\boldsymbol{f} = (f_1, f_2)$. In addition, one must add $-u/r^2$ to the viscous term in the radial component of the momentum equation, where u is the radial velocity component. Note that equation (43) applies to the calculation of the curvature through equation (15).

The following results were obtained with $Re = 0.5$ and $Ca = 0.025$ for $R = 0.5$ m and $U_0 = 0.5$ m/s. The computational domain was $8R \times 12R$, and the grid size was 120×160. The axis of symmetry coincides with the left boundary. At the other boundaries we specify (u, v) to match equation (41).

The case is run both with the standard discretization of curvature and with the present method. Figure 16 shows the interfaces, the curvature values, and the velocity vectors plotted at various stages of the collision process when the case is run with the standard discretization. The discretization stencil for the curvature starts to cross the kink at some time between

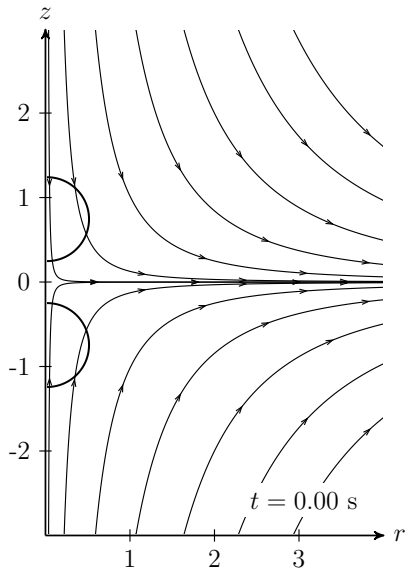


Figure 15: The initial drop interfaces and initial streamlines of the axisymmetric flow.

$t = 0.30$ s and $t = 0.042$ s, after which one can observe a spike in the calculated curvature where both the value and the sign are erroneous. This in turn affects the pressure calculation and leads to a slower coalescence process. Figure 17 shows that the spike in the curvature calculation is prevented with the present method. Note that right after the coalescence at $t = 0.43$ s, the value of the curvature in the thin filament area is and should be very high.

6.5. Normal vectors between two near discs

The final case is designed to show that the direction difference is not always sufficient to calculate the normal vectors. In this case two discs of radius r are placed at a distance h from each other as shown in Figure 18. As in the first case, only the level-set function and the geometrical quantities are considered.

The parameters for this case are $r = 0.25$ m and $h = 1.2\Delta x$. The domain is 1.5 m $\times$ 1.5m and the grid size is 101×101.

As was noted in Section 4.2.2, the curve-fitting discretization scheme may be used as an alternative. In this case the curve-fitting discretization scheme is used at the grid points that are within 1 grid cell from any interface. The direction difference is used at the other grid points.

Figure 19 shows a comparison of the calculated normal vectors. The normal vectors calculated with the direction difference are depicted with red vectors and the normal vectors calculated with a combination of the direction difference and the curve-fitting scheme are depicted with green vectors. The green vectors are on top of the red vectors, which shows that the results are almost identical. But at the centre grid point between the discs, the direction difference is not able to accurately calculate the normal vector. For more complex geometries this error may appear at more than one grid point. The error directly affects both the solutions of the level-set equations, (10), (11) and (13), and the jump conditions for the pressure and the gradient of the velocity, (4) and (5).

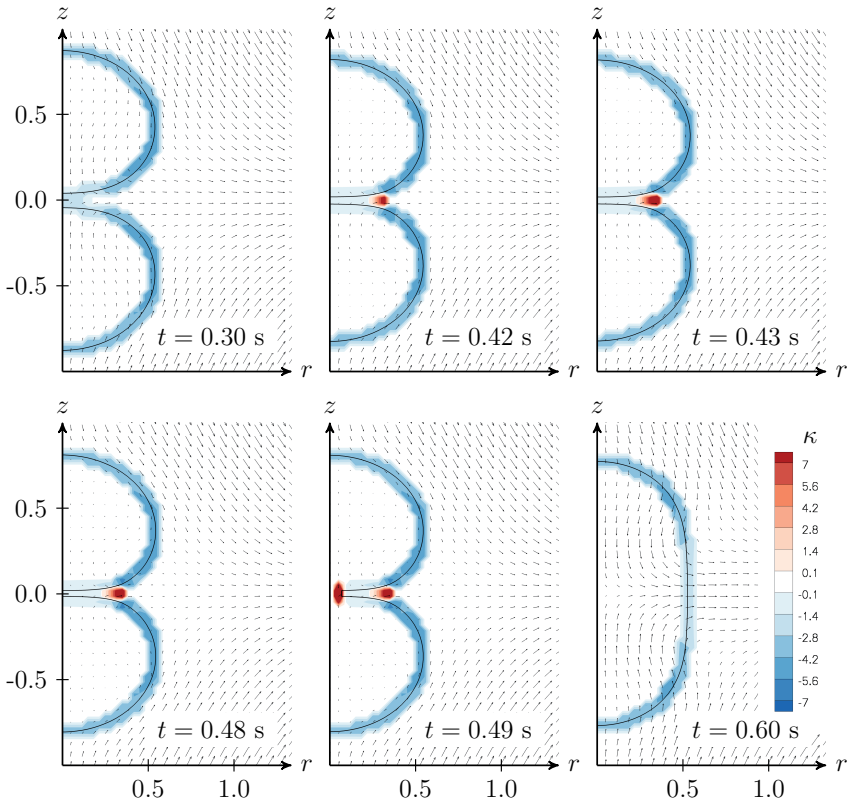


Figure 16: Drop collision in axisymmetric flow calculated with the standard method. The legend for the colour contours of the curvature κ is shown in the last image. The velocity vectors are displayed to show the evolution of the flow during the collision.

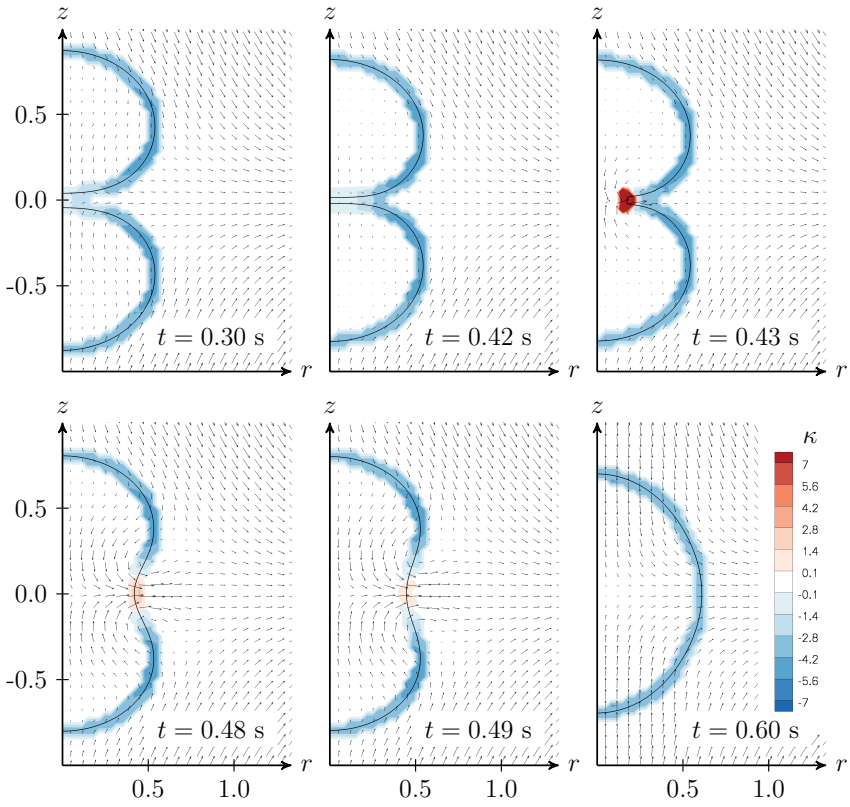


Figure 17: Drop collision in axisymmetric flow calculated with the present method. The legend for the colour contours of the curvature κ is shown in the last image. The velocity vectors are displayed to show the evolution of the flow during the collision.

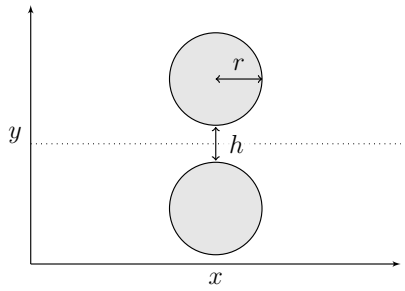


Figure 18: A sketch of the initial state for the two-disc test. The dotted line depicts the kink location.

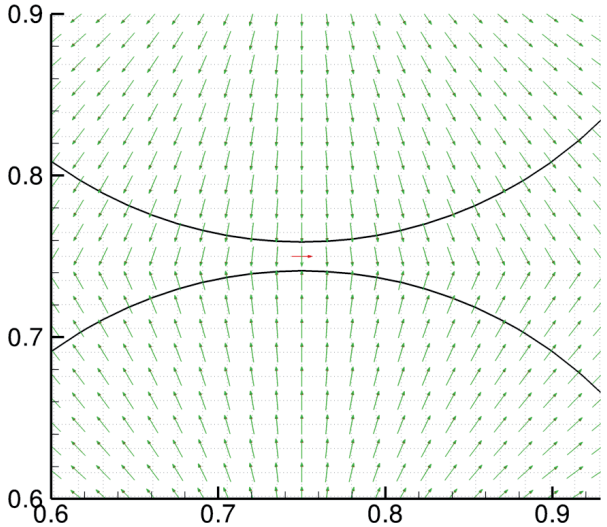


Figure 19: A comparison of the direction difference and the curve-fitting method for calculating normal vectors. The direction difference results are plotted in red below the curve-fitting method results which are depicted in green. The thick black lines depict the interfaces.

7. Conclusions

Improved discretization schemes for the normal vector and the curvature of the interface between two phases have been devised and tested. The curvature was discretized with a curve-fitting discretization scheme based on the geometry-aware discretization presented in [1]. The normal vector was discretized both by the direction difference presented in [2] and a combination of the direction difference and the curve-fitting discretization scheme. The main advantage of the present curvature discretization scheme is that it is independent of the ghost-fluid method. This makes it easier to be adopted into existing level-set codes, for instance codes that use the continuum surface-force method. In addition, it enables the use of models that require curvature values at the grid points, not just on the interface.

The present work has been restricted to two spatial dimensions. An extension to three dimensions would require bicubic parametrization of surfaces, and a local reconstruction of the level-set function based on calculating the minimum distances of parametrized curves to points on the grid. The complexity of this is much higher than for the two-dimensional problem. Note however, that the curve-fitting discretization scheme is directly applicable to axisymmetric cases, which is demonstrated in a test case.

The implementation of the curve-fitting discretization scheme has been described in detail. Our results show that the curvatures calculated with the present scheme converge when the grid size is reduced in a case where the standard scheme fails to converge.

The present discretization scheme is compared with the central-difference scheme in three different cases. The first case is a direct comparison of the schemes for a case with no flow. The second case compares the evolution of two drops colliding in shear flow. Both of these cases demonstrate that the central-difference scheme leads to erroneous behaviour at the kink locations. The second case shows that this behaviour prevents coalescence from occurring due to an erroneous pressure field. The curvature spikes at the kink regions are not observed with the present discretization scheme, and coalescence is achieved for the second case. The present scheme was also compared with Macklin and Lowengrub's method [1], and the results show that the present method gives similar results, as expected. The third case considers the collision of two drops in an axisymmetric flow. As in the previous cases, the central-difference scheme leads to erroneous curvatures at the kink, which is shown to lead to a slower coalescence.

Finally, a fourth test case demonstrates that the direction difference [2] does not always yield accurate results for calculating the normal vector. A combination of the curve-fitting discretization scheme and the direction difference is shown to remove the error in the given case. Accurate calculation of the normal vector is crucial, as it is used both to advect the level-set function (10), extrapolate the velocity vector (11), and to calculate the jumps across the interface (4) and (5). More work should therefore be done to investigate how much this error affects more complex cases.

Acknowledgements

This publication is based on results from the research project Enabling low emission LNG systems, performed under the Petromaks program. The author acknowledges the project partners; Statoil and GDF SUEZ, and the Research Council of Norway (193062/S60) for support.

The authors acknowledge Claudio Walker, Leif Amund Lie and Eirik Svanes for several good discussions.

[1] P. Macklin, J. Lowengrub, An improved geometry-aware curvature discretziation for level set methods: Application to tumor growth, Journal of Computational Physics 215 (2006) 392–401.

[2] P. Macklin, J. Lowengrub, Evolving interfaces via gradients of geometry-dependent interior Poisson problems: Application to tumor growth, Journal of Computational Physics 203 (2005) 191–220.

[3] S. Osher, J. A. Sethian, Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics 79 (1988) 12–49.

[4] J. A. Sethian, P. Smereka, Level set methods for fluid interfaces, Annual Review of Fluid Mechanics 35 (2003) 341–372.

[5] P. Macklin, J. S. Lowengrub, A new ghost cell/level set method for moving boundary problems: Application to tumor growth, Journal of Scientific Computing 35 (2008) 266–299.

[6] D. Salac, W. Lu, A local semi-implicit level-set method for interface motion, Journal of Scientific Computing 35 (2008) 330–349.

[7] Z. Wang, A. Y. Tong, A sharp surface tension modeling method for two-phase incompressible interfacial flows, International Journal for Numerical Methods in Fluids 64 (2010) 709–732.

[8] M. Herrmann, A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids, Journal of Computational Physics 227 (2008) 2674–2706.

[9] E. Marchandise, P. Geuzaine, N. Chevaugeon, J.-F. Remacle, A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics, Journal of Computational Physics 225 (2007) 949–974.

[10] O. Desjardins, V. Moureau, H. Pitsch, An accurate conservative level set/ghost fluid method for simulating turbulent atomization, Journal of Computational Physics 227 (2008) 8395–8416.

[11] K. Y. Lervåg, Calculation of interface curvature with the level-set method, in: Sixth National Conference on Computational Mechanics MekIT'11 (Trondheim, Norway), 23-24 May 2011.

[12] J.-J. Xu, Z. Li, J. Lowengrub, H.-K. Zhao, A level set method for interfacial flows with surfactants, Journal of Computational Physics 212 (2) (2006) 590–616.

[13] K. E. Teigen, K. Y. Lervåg, S. T. Munkejord, Sharp interface simulations of surfactant-covered drops in electric fields, in: Fifth European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2010, Lisbon, Portugal, 2010.

[14] K. E. Teigen, S. T. Munkejord, Influence of surfactant on drop deformation in an electric field, Physics of Fluids 22 (11) (2010) 112104. doi:10.1063/1.3504271.

[15] J. U. Brackbill, D. B. Kothe, C. Zemach, A continuum method for modeling surface tension, Journal of Computational Physics 100 (1992) 335–354.

[16] M. Kang, R. P. Fedkiw, X.-D. Liu, A boundary condition capturing method for multiphase incompressible flow, Journal of Scientific Computing 15 (3) (2000) 323–360.

[17] E. B. Hansen, Numerical simulation of droplet dynamics in the presence of an electric field, Doctoral thesis, Norwegian University of Science and Technology, Department of Energy and Process Engineering, Trondheim, iSBN 82-471-7318-2 (Nov. 2005).

[18] H.-K. Zhao, T. Chan, B. Merriman, S. Osher, A variational level set approach to multiphase motion, Journal of Computational Physics 127 (1996) 179–195.

[19] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, Journal of Computational Physics 114 (1994) 146–159.

[20] S. Gottlieb, C. W. Shu, E. Tadmor, Strong stability-preserving high-order time discretization methods, SIAM Review 43 (2001) 89–112.

[21] D. Adalsteinsson, J. A. Sethian, A fast level set method for propagating interfaces, Journal of Computational Physics 118 (1995) 269–277.

[22] X.-D. Liu, R. P. Fedkiw, M. Kang, A boundary condition capturing method for Poisson's equation on irregular domains, Journal of Computational Physics 160 (2000) 151–178.

[23] S. Osher, R. P. Fedkiw, The Level-Set Method and Dynamic Implicit Surfaces, Springer, 2003.

[24] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, Computer Graphics 21 (4) (1987) 163–169.

[25] H. Prautzsch, W. Boehm, M. Paluszny, Bézier and B-spline Techniques, Springer, 2002.

[26] F. N. Fritsch, R. E. Carlson, Monotone piecewise cubic interpolation, SIAM Journal of Numerical Analysis 17 (2) (1980) 238–246.

[27] B. Waerden, E. Artin, E. Noether, Algebra, no. v. 1 in Algebra, Springer-Verlag, 2003.
URL http://books.google.com/books?id=XDN8yR8R1OUC

[28] S. Lang, Algebra, Graduate texts in mathematics, Springer, 2002.
URL http://books.google.com/books?id=Fge-BwqhqIYC

D

A robust method for calculating interface curvature and normal vectors using an extracted local level set

Å. Ervik, K. Y. Lervåg, and S. T. Munkejord
Submitted to Journal of Computational Physics, 2013

A robust method for calculating interface curvature and normal vectors using an extracted local level set

Åsmund Ervik[a,b,*], Karl Yngve Lervåg[b], Svend Tollak Munkejord[a]

[a] *SINTEF Energy Research, P.O. Box 4761 Sluppen, NO-7465 Trondheim, Norway*
[b] *NTNU, Department of Energy and Process Engineering, Kolbjørn Hejes v 1B, NO-7491 Trondheim, Norway*

Abstract

The level-set method is a popular interface tracking method in two-phase flow simulations. An often-cited reason for using it is that the method naturally handles topological changes in the interface, e.g. merging drops, due to the implicit formulation. It is also said that the interface curvature and normal vectors are easily calculated. This last point is not, however, the case in the moments during a topological change, as several authors have already pointed out. Various methods have been employed to circumvent the problem. In this paper, we present a new such method which retains the implicit level-set representation of the surface and handles general interface configurations. It is demonstrated that the method extends easily to 3D. The method is validated on static interface configurations, and then applied to two-phase flow simulations where the method outperforms the standard method and the results agree well with experiments.

Keywords: Level-Set Method, Curvature, Normal vector, Droplet-film interaction

1. Introduction

Investigations of droplet collision phenomena have a long tradition in the study of multiphase flow, dating back to Lord Rayleigh [1] who in 1879 noted that a raindrop can bounce off a pool, and to Worthington [2] who in 1876 studied among other things the central jet that now bears his name. The early work predates the rise of computational studies, and consists of experimental studies that enabled a separation of the flow patterns into various regimes characterized by e.g. the Weber number and Ohnesorge number. A case which has long been the focus of study is that of a single droplet of one liquid, immersed in some other gas or liquid, and which collides with a deep pool of the first liquid. This could be e.g. a raindrop falling onto a pond, or a droplet of Liquefied Natural Gas (LNG) merging with a pool of LNG in a liquefaction heat exchanger, so the case is interesting also from an industry point of view. Such a system may seem simple at first, but experimental and numerical studies have shown that varied phenomena such as coalescence, bouncing, jetting and partial merging occur. The system

*Corresponding author
Email addresses: asmund.ervik@sintef.no (Åsmund Ervik), karl.y.lervag@ntnu.no (Karl Yngve Lervåg), svend.t.munkejord@sintef.no (Svend Tollak Munkejord)

is also not fully understood yet; as an example, Thoroddsen et al. [3] have recently shown that for high impact velocities a turbulent boundary layer forms between the droplet and the pool after they merge.

In order to study such a case using computer simulations, it is necessary to use a precise interface-tracking method to capture the physics before, during and after the collision. The Level-Set Method (LSM) is a popular choice for interface tracking in studies of collisions, since its implicit formulation means that the method can handle the topological change which occurs when two interfaces merge. The LSM is very general, and apart from fluid dynamics it has been used for modeling such diverse phenomena as tumor growth [4], wildland fire propagation [5] and computer RAM production [6]. For a good introduction to the LSM, see e.g. [7]. The LSM originated from the seminal article by Osher and Sethian [8].

In two-phase flow simulations using the LSM, accurate interface curvature and normal vector information is vital in order to get good results. Standard methods exist for calculating these geometric quantities, but they fail when the interface topology changes, e.g. when two drops collide and merge. Several approaches have been used to remedy this flaw. The first approach to this problem is described by Smereka in [9]. He describes the problem briefly, and increases the numerical smoothing in the curvature discretization to lessen the effect. This is not an optimal solution, and Smereka notes on one of the simulations with merging interfaces that "most of the area loss occurs at the topology change". Several non-smearing approaches have subsequently been developed, by Macklin and Lowengrub [4, 10], by Salac and Lu [11] and by Lervåg [12, 13]. The methods by Macklin and Lowengrub and by Lervåg use curve fitting to obtain an accurate representation of the interface, while the method by Salac and Lu extracts several level-set functions each representing only a single body, and uses these to calculate the curvature.

The present work proposes a new method, which is an extension of previous methods, for calculating the curvature and normal vectors. The proposed method is based on the method by Salac and Lu, but it handles more general interface configurations and topological changes, as it considers only the local area around a point. The quality function introduced by Macklin and Lowengrub is used to restrict the use of the proposed method to those areas where it is needed, thus reducing the computational cost. As the proposed method uses no curve fitting, it extends easily to three dimensions, as demonstrated here. The proposed method is compared to the standard method for demanding cases where the analytical curvature is known; for such a case the proposed method gives errors of 1–2% where the standard method gives errors of $\mathcal{O}(1/\Delta x) \gtrsim 100\%$.

The outline of this work is as follows: In Section 2, the theory of two-phase incompressible flow, the LSM and numerical methods are briefly reviewed. In Section 3, the proposed method is presented in detail. In Section 4, the method is validated on geometric test cases, and the results are compared to other methods. In Section 5, the results of two-phase flow simulations using the current method are reported and compared to experimental results. Finally, in Section 6, some concluding remarks are offered.

2. The Level-Set Method and two-phase flow

The LSM is one of the more successful interface-capturing methods used in computational physics. Since its introduction by Osher and Sethian in [8], it has been used for numerous physical applications, as well as in computer graphics. Perhaps the main virtue of the LSM is how intuitive it is; in 2D it can easily be explained to anyone with a basic knowledge of multivariate

calculus. This simplicity stems from the implicitness of the LSM, making the numerical implementation of the LSM relatively easy. The implicit formulation also means that changes in the interface topology are handled naturally. When comparing the LSM to other interface-tracking methods, such as the Front-Tracking Method [14] where the interface is represented by piecewise continuous functions, the simplicity becomes especially clear.

The main disadvantage of the LSM, on the other hand, is that it is not a conservative method. During the course of a simulation, a fraction of fluid 1 may be converted to fluid 2 in an unphysical fashion. Various methods have been invented to circumvent this, e.g. the HCR-2 reinitialization method [15], so it is only a small effect presently. Interface-tracking methods may be conservative; an example of this is the Volume-of-Fluid (VOF) Method, but then they typically have other disadvantages. In the VOF method, for instance, the advection equation cannot easily be solved, necessitating the use of interface-reconstruction methods [16]. Recent efforts have attempted to join the LSM and VOF in order to get the benefits of both methods; this approach seems to be fairly successful [17].

We give here the formal definition of the level-set function used in the LSM. Let Γ be the interface between two fluids, e.g. air and water, and S be the computational domain where the fluids are confined. To represent this interface, we define a *level-set function* $\phi : S \to \mathbb{R}$ with the property

$$\Gamma = \{\mathbf{x} \,|\, \phi(\mathbf{x}) = 0\}. \tag{1}$$

This only defines the value of ϕ at the interface Γ, and not elsewhere. The common choice here is a signed distance function. Thus ϕ is fully specified by

$$\phi(\mathbf{x}) = \begin{cases} -\text{dist}(\mathbf{x},\Gamma) & \text{if } \mathbf{x} \text{ is inside } \Gamma, \\ \text{dist}(\mathbf{x},\Gamma) & \text{if } \mathbf{x} \text{ is outside } \Gamma. \end{cases} \tag{2}$$

Here, the function $\text{dist}(\mathbf{x},\Gamma)$ is the shortest distance from the point $\mathbf{x} \in S$ to the interface Γ. With this definition of the level-set function, the normal vector to the interface is given by

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}. \tag{3}$$

From this, the curvature is calculated by the well-known formula

$$\kappa = \nabla \cdot \mathbf{n} = \nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|} \right). \tag{4}$$

With suitable discretizations of the derivatives involved, these quantities are easy to calculate numerically. This is often quoted as one of the nice features of the LSM, along with e.g. the very natural way the method handles topological changes [18]. In 2D, the standard discretization of the curvature is (see e.g. [19])

$$\kappa = \frac{\phi_{xx} + \phi_{yy}}{(\phi_x^2 + \phi_y^2 + \epsilon)^{1/2}} - \frac{\phi_x^2\phi_{xx} + \phi_y^2\phi_{yy} + 2\phi_x\phi_y\phi_{xy}}{(\phi_x^2 + \phi_y^2 + \epsilon)^{3/2}} \tag{5}$$

Here, e.g. ϕ_x denotes the first derivative of ϕ in x-direction, calculated using standard central differences. However, when curvature and normal vector calculations are done during a change

in the interface topology, this approach fails; the error in curvature is of the order $\mathcal{O}(1/\Delta x)$ [4]. In [9], Smereka notes that "One of the major advantages of level-set methods is their ability to easily handle topological changes. However for this problem we have found this not to be the case." It is this that the present method attempts to solve.

From the defining Equation (2), ϕ is initialized at the start of a simulation. For a given velocity field $\mathbf{u}$, ϕ should be transported so that the interface follows the flow. This is done by solving the advection equation,

$$\frac{\partial \phi}{\partial t} = v|\nabla \phi| = -\mathbf{u} \cdot \nabla \phi. \tag{6}$$

Here v is the velocity normal to the interface, and $\mathbf{u}$ is an *extrapolated* velocity field constructed using the method in [20]. This equation is not justified here, see e.g. [21].

Solving this equation will result in transportation of the interface, but it will also degrade the accuracy of the interface representation, as ϕ is deformed from a signed distance function. To avoid this, the level-set function is periodically *reinitialized*. We follow here the PDE-based approach introduced by Sussman, Smereka and Osher [21], which consists in solving

$$\frac{\partial \phi}{\partial \tau} + \text{sgn}(\phi)(|\nabla \phi| - 1) = 0. \tag{7}$$

Here τ is a pseudo-time which is not related to the physical time in simulations. This approach is both computationally fast and accurate when used as here with a narrow-band approach. The extrapolation of the velocity field as used in Equation (6) above is performed by solving a similar type of equation. These equations are solved using pseudo-CFL numbers of 1.0 for the velocity extrapolation and 0.5 for the reinitialization. It is noted that a numerical solution of the reinitialization equation needs accurate normal vectors at the interface.

A useful property of these equations is that the characteristics originate at the interface, meaning that solving the equations numerically for N pseudo-time steps using a CFL-number of C will yield a correct signed distance function $C \cdot N$ space steps away from the interface. This has led to the use of narrow-band methods, where the level-set function and other properties such as the curvature are only calculated and used in a narrow band around the interface. This reduces the computational time significantly.

In two-phase flow simulations, the LSM is coupled with the Navier-Stokes equations,

$$\nabla \cdot \mathbf{u} = 0, \tag{8}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} + \mathbf{f}. \tag{9}$$

Here $\nu = \mu/\rho$ is the kinematic viscosity, while μ is the dynamic viscosity, ρ is the density, $\mathbf{u}$ is the velocity field and p is the pressure. $\mathbf{f}$ is any external force, such as gravity, and may be zero.

These equations hold for single-phase fluid flow, but can be extended to two-phase flow using different methods. In the present work, the Ghost Fluid Method (GFM) [22] is used. This method prescribes jump conditions for e.g. the pressure across the interface based on the

interface properties. The jump conditions used here are

$$[\mathbf{u}] = 0, \tag{10}$$

$$[p] = 2[\mu]\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{n} + \sigma\kappa, \tag{11}$$

$$[\mu\nabla\mathbf{u}] = [\mu]\Big((\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{n})\mathbf{nn} + (\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{t})\mathbf{nt} \tag{12}$$

$$-(\mathbf{n}\cdot\nabla\mathbf{u}\cdot\mathbf{t})\mathbf{tn} + (\mathbf{t}\cdot\nabla\mathbf{u}\cdot\mathbf{t})\mathbf{tt}\Big), \tag{13}$$

$$[\nabla p] = 0. \tag{14}$$

based on [19]. Here, $\mathbf{t}$ is the tangent vector along the interface and $[\cdot]$ denotes the jump across an interface, that is $[\mu] \equiv \mu^+ - \mu^-$. Note that $\nabla\mathbf{u}$ and (e.g.) $\mathbf{nt}$ are rank-2 tensors. The pressure must also be decoupled from the velocity field in order to enable a numerical solution of the Navier-Stokes equations; we use here the projection method due to Chorin [23]. This gives a Poisson equation for the pressure which can be solved using freely available numerical libraries. The PETSc library is used here [24].

In the present numerical implementation, SSP-RK schemes [25, 26] are used for the time integration, while the WENO method [27] is used for the spatial discretizations. To determine the time step dynamically, we use the CFL criterion given by Kang et al. [19].

3. The Local Level-Set Extraction (LOLEX) Method

3.1. Introduction

Calculating the curvature κ of the interface between two phases is important, since it appears in the Young-Laplace formula for the capillary pressure, $\Delta p = \sigma\kappa$. Its value is used in e.g. the Ghost Fluid Method (GFM) (Equation (11)), or other methods of enforcing the jump conditions. The normal vectors to the interface are also important, e.g. when advecting the level-set function and when reinitializing it. Calculating these geometric quantities is straightforward in theory, using Equation (3) and Equation (4) to compute them from the level-set function.

However, as is often the case, in practice it is not so straightforward. The problems arise when the distance between two interfaces is of the order Δx. This is illustrated in Figure 1. The derivatives of ϕ are not defined at the kinks. As a result of this, the numerical stencils approximating the derivatives of ϕ will often produce large, erroneous values. When this happens, the curvatures and normal vectors will be erroneous. For the curvature, this error is of order $\mathcal{O}(1/\Delta x)$, which can be several orders of magnitude larger than the correct curvature value. It should be stressed that additional grid refinement does not solve this problem; e.g. for the simulation of colliding drops, one would have to continue refining the grid *ad infinitum*.

The earliest non-smearing approach to this problem, by Macklin and Lowengrub [4], uses a modification of the directional differences for points close to kinks, along with a mesh refinement for these points. The same authors introduced a curve-fitting method instead in [10], which is said to be an improvement on the directional differences and a simplification. The latter version will be referred to as the MLM (Macklin and Lowengrub Method). Further improvements to this method, and adaptations to an on-grid framework (i.e. calculating the curvature at the grid points, not at the interface), have been developed by Lervåg [12],[13].

These methods give good results in 2D, but are difficult to extend to 3D simulations due to the use of curve-fitting.

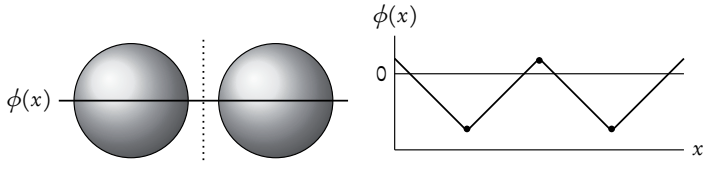
An alternative approach to the problem is due to Salac and Lu [11], and will be referred to as the Salac and Lu Method (SLM). In essence, this approach extracts the bodies represented by the level-set function, such that each body (e.g. drop) is momentarily given its own version of the computational domain. In this dedicated version, ϕ does not represent any other bodies that can induce kinks, and this temporary ϕ can be reinitialized and the geometric quantities can be calculated without problems. For a review and comparison of the SLM, MLM and the method by Lervåg, see Lervåg and Ervik [28]. It should also be noted that the recent article by Focke and Bothe [29] discusses a similar issue, in the context of thin lamellae which form when liquid drops collide off-center. The authors introduce a method which resembles the SLM, but which also has the ability to add small amounts of liquid to the lamella region, preventing a numerical rupture.

The method considered here is a further development of the SLM. It is referred to as the local level-set extraction method, or LOLEX method in short. The reason why the SLM is insufficient in some cases, as well as the details of the present method, is given below. Suffice it to say at this point that the present method is more general, so it applies both to the cases considered by Salac and Lu and those considered by Focke and Bothe (except the stabilization of thin lamellae which the latter introduce).

Another recently presented approach is due to Trontin et al. [30], who consider a hybrid particle/level-set method. Their approach is to use the information from the tracking particles to calculate the curvature and normal vectors, with good results. This can obviously not be applied to a pure level-set method as discussed here, or e.g. a hybrid LSM-VOF method as has recently become popular [17].

An approach which has not been considered here, or by other authors in the context of level-set methods as far as we are aware, is the use of filtering. Vliet and Verbeek [31] study the estimation of curvature from a discretely sampled greyscale image, using derivative-of-Gaussian filters, and note that this outperforms a traditional curvature estimate analogous to Equation (5).

The idea of Salac and Lu, on which the present method is based, is simple when compared to the curve-fitting scheme used by Macklin and Lowengrub [4] and later by Lervåg [12, 13]. This simplicity is more in keeping with the "spirit" of the level-set method: the LSM is an implicit alternative to front-tracking methods that employ curve fitting, and this implicitness makes extending to higher dimensions straightforward. In the same fashion, the SLM is easily



(a) Droplets in near contact (b) A slice of the level-set function

Figure 1: (a) Two droplets in near contact. The dotted line marks a region where the derivative of the level-set function is not defined. (b) A one-dimensional slice of the level-set function. The dots mark points where the derivative of ϕ is not defined.

extensible to 3D, while the methods employing curve fitting are not. There are, however, some drawbacks to the Salac and Lu method as well.

The primary issue stems from the fact that the Salac and Lu method is aware of the global topology of the interface. A problematic area, with a kink in the level-set function close to $\phi = 0$, can be caused either by two bodies in close proximity or by a single body folding back onto itself. In the latter case, as illustrated in Figure 2, the Salac and Lu method falls back to the standard discretization, and the calculated curvature will be erroneous. This may seem like an edge case not worth considering, but simulations have shown that this often happens, e.g. when a falling droplet merges into a pool. As pointed out by Smereka [9], errors like these can be the main contribution to unphysical area loss in a simulation. Another situation where this would often be the case is in tumor simulations like those performed by Macklin and Lowengrub, as can be seen in e.g. [4, Figure 6].

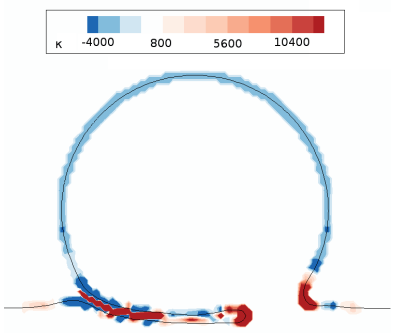


Figure 2: The curvature field plotted for the SLM. Note the red curvature field inside the air finger between the drop and the pool, which is incorrect. The color should be light blue in this area.(Figure best viewed in color.)

3.2. The idea of the LOLEX method

The method presented here tries to combine the best of the SLM with the best of the MLM. As illustrated in the previous section, the SLM is aware of the global topology of the interface, which is problematic in some cases. The MLM does not have this problem, as its curve fitting considers only the local area, but as previously stated it does not extend easily to 3D. A natural workaround to the "global awareness" is to make the Salac and Lu method consider only the local topology; say, a $10\times10\times10$ cube around the point where we calculate the curvature.

Since the SLM relies on reinitialization to remove kinks, a potential problem with this approach is computational efficiency, as reinitialization can be time-consuming. To avoid problems with this, we want to use the standard discretization as much as possible, only resorting to the LOLEX method when we have to, i.e. when kinks in ϕ are close to the interface. To easily identify kinks, we use the quality function $Q(\mathbf{x})$ which was introduced by Macklin and Lowengrub in [4]. It is defined as

$$Q(\mathbf{x}) = |1 - \nabla\phi(\mathbf{x})|, \tag{15}$$

i.e. the deviation of ϕ from a signed distance function. If $\max(Q(\mathbf{x}_{i,j,k})) > \eta$ for $\mathbf{x}_{i,j,k}$ in a 3×3×3 cube around the current grid point, we use the LOLEX method. A value of $\eta = 0.005$ is used here, and is seen to perform well. In addition to this, the current work uses the "narrow band" level-set method introduced in [32]. This means that quantities such as the curvature are only calculated in a narrow band around the zero level set, where they are needed. Together, these significantly restrict the use of the present method compared to the use of the normal method, keeping the computational cost low.

Having briefly presented the idea behind the present method and the scope in which it will be used, we give here a step-by-step outline of it. 2D notation is used here for clarity, but all steps are easily extensible to 3D. In this outline, a few arrays are introduced for storing data: `lookphi` is a copy of the global ϕ for the local area we are considering, `bodies` indicates the bodies present using increasing integers, and `locphi` holds the local ϕs that are extracted from the global ϕ and then refined into more accurate representations of the local bodies present. The quantities `ilmax`, `jlmax` and `klmax` represent the number of grid points, in the x, y and z directions respectively, of the *local* grid. The values of `ilmax`, `jlmax`, `klmax` are all set to 7 in the simulations performed here. Their values are independent of the global grid size.

- ↪ Loop over the computational domain using indices `i,j`
- ↪ If ($\mathbf{x}_{i,j}$ not close to interface) do nothing
- ↪ Else if ($Q(\mathbf{x}_{n,m}) \leq \eta \; \forall (n,m) \in [i-1, i+1] \times [j-1, j+1]$) use ordinary method
- ↪ Else use LOLEX method:
  - ↪ Copy ϕ in a `[-1,ilmax+2]*[-1,jlmax+2]` square around `i,j` into the `lookphi` array.
  - ↪ Identify the bodies present in the `[0,ilmax+1]*[0,jlmax+1]` square, store this in the `bodies` array.
  - ↪ For each body, extract the relevant part of the `lookphi` array into `locphi(:,:,bodyno)`. This array has 3 ghost cells on the boundary outside `ilmax*jlmax`; these are not used until the extrapolation further down. Extracting means
    - copying `lookphi` for the internal points of *this* body
    - copying `lookphi` for external points that are not next to more than one body
    - explicitly reconstructing the signed distance for external points that are next to more than one body
    - setting a value of `2*dx` for all other points
  - ↪ Once the `locphi` array has been filled for all bodies, the values are extrapolated into the ghost cells. The extrapolation is zeroth-order, as will be explained further down.
  - ↪ The `locphi` array is then reinitialized for all bodies. This erases the problematic kink, as well as the value of `2*dx` which was set previously. Thus this value is unimportant, as long as it is `> 0`.
  - ↪ Using these local ϕ's, the curvature and normal vectors can be calculated for each body. The curvature and normal vectors corresponding to the body which is closest to the current grid point are used.

The steps in this algorithm that warrant further comments are: identifying the bodies present, explicitly reconstructing the signed distance, extrapolating to the ghost cells, and reinitializing. These will be considered further in the next section and subsections.

3.3. Details of the method

Some steps of the algorithm outlined need further explanations. This is either because they are too technical to be fully described in the previous short outline, or because they have not been properly motivated yet. The steps that will be considered are identifying the bodies present (Section 3.3.1), explicitly reconstructing the signed distance (Section 3.3.2), extrapolating to the ghost cells (Section 3.3.3), and reinitializing (Section 3.3.4).

3.3.1. Identifying the bodies present

To identify the bodies present, a recursive routine is used, which starts at a seed point in a body and iterates through the entire body, marking it as a body in the `bodies` array. This routine is called `bodyscan` here. The `bodies` array starts with a value of `unchecked`, and bodies found are marked using increasing integers. The recursive subroutine will have marked the entire first body when its first call returns.

After the subroutine returns, we check if the present body is large enough to keep, or if it should be discarded. The reasoning behind discarding some bodies is twofold: a body which is large in the global domain but occupying only a few cells in the local area will not be accurately represented, and cannot be accurately reconstructed. If the body is small in the global domain and close to the central point where we want to calculate things, it will be insufficiently resolved anyway, and we fall back to the standard discretization. In the alternative case where all bodies present are large bodies far away from the central point, we would not be using the LOLEX method in the first place. For removed bodies, the points in the body are marked as `removed`. Points not inside a body are marked as `nobody`.

Several methods were tested for determining which bodies are to be discarded. The simplest and most effective method is to count the number of points in a body, and discard it if this number is less than some threshold. This method worked well, using a threshold of 25 cells for a 9×9 `bodies` array in 2D and 122 cells for a $9\times9\times9$ `bodies` array in 3D. These thresholds were chosen after tests with varying thresholds. They are large enough so that small bodies leading to erroneous values were discarded, but not so large as to remove bodies which are close to the central point (and thus important). Using a Gaussian weighting to give less importance to bodies with many points far from the centre of the local grid did not give any improvements.

A final point to note about the routine given here is that even though a recursive subroutine is used, memory usage will not be problematic. This is because the routine operates on a small array whose size is independent of the grid size. In 3 dimensions and with the presently used size of the local area, the array `bodyscan` would have $11*11*11 = 1331$ elements. This routine can maximally be called 1331 times, giving a worst-case memory consumption of 13.5 MB. This will not cause memory problems, although it is too large to fit in the CPU cache for some processors. The performance impact has not been tested here, as the 3D calculations are only considered as a proof-of-concept, and have not been optimized for speed. In 2D the memory use is naturally much smaller.

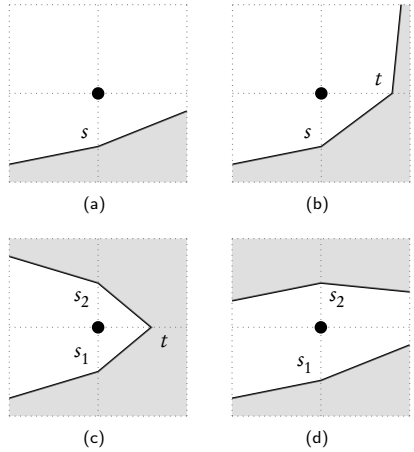


Figure 3: Cases for the neighborhood of a point.

3.3.2. Explicit reconstruction of the signed distance

For some points with $\phi > 0$, two or more bodies are within Δx of the point. This means that the value of ϕ is probably incorrect, since it has to be the distance to two separate bodies at the same time. We will call such points "dependent points". Because ϕ is likely incorrect for dependent points, we discard its value, and instead explicitly reconstruct the distance to the relevant interface. The procedure used is due to Adalsteinsson et al. [33].

When we consider such a dependent point, it lies right next to two interfaces. When reconstructing the distance, only one interface is of interest, so the other one is momentarily removed. Note that the signed distance is always positive for exterior points, so it is just the normal distance.

The procedure in [33] is as follows. The point (i, j) which we are considering is next to the interface of current interest. We ignore all other interfaces. Up to rotational symmetry, there are four possible cases. An illustration of these cases can be seen in Figure 3.

We examine the four cases (**a** to **d**) more closely:

a The interface crosses one of the lines from (i, j) to its four neighbors. In this case, we use the distance to the interface along this line as our distance. This distance is given by

$$s = \Delta y + \phi(i, j-1) \tag{16}$$

where we have assumed that $(i, j-1)$ is the neighbor on the other side of the interface. Since this neighbor is an internal point, it has $\phi < 0$. The distance to the interface is the distance to the neighboring grid point (Δy) minus the distance from that grid point to the interface, which gives this formula. It is best to use only the ϕ-value inside the body, since it is less likely to be distorted.

b The interface crosses two of the lines, and these two lines make out a corner of the 2×2 grid around (i, j). In this case we use the shortest distance to the straight line between the two

points of intersection. The distance d is given by the formula

$$\left(\frac{d}{s}\right)^2 + \left(\frac{d}{t}\right)^2 = 1. \tag{17}$$

As long as $s^2 + t^2 \neq 0$ this equation can be solved, and the positive solution is

$$d = \frac{st}{\sqrt{s^2+t^2}} \quad . \tag{18}$$

If we have $s^2 + t^2 = 0$, then $s = t = 0$, so it is obvious that the distance to the interface is $d = 0$.

c The interface crosses three lines. We construct the two straight lines between the points of intersection, and use the shortest distance to either of these two lines, given by

$$\left(\frac{d}{\min(s1, s2)}\right)^2 + \left(\frac{d}{t}\right)^2 = 1. \tag{19}$$

d The interface crosses two lines. These lines are on opposite sides of the point (i, j). In this case, we use the shortest of the two distances, so $d = \min(s1, s2)$.

These formulae can be extended to three dimensions, where the possible cases are more numerous. In 3D, the central point has two additional neighbors. This means there are more variations in addition to the cases considered above. This is not considered in detail here.

3.3.3. Extrapolation

After the interior of the `locphi` array has been filled, the ghost cells must be filled before we can reinitialize the local ϕ. Two ways of doing this are illustrated in Figure 4. A first approach is to use linear extrapolation, which should work well since ϕ is a linear function in 1D. However, it turns out that this does not work. A fundamental property of the reinitialization equation (7) is that its characteristics originate at the interface $\phi = 0$. This is why the present method (and the SLM) works – we only need a few cells directly next to the interface to have the correct value of ϕ, and reinitialization will fix the rest. It also means that reinitialization will never move the position of the interface, which is a desirable property in general.

The problem with linear extrapolation occurs when we extrapolate starting on the opposite side of the kink from the interface. In this case, the values of the local ϕ are tending towards 0 from above, which means that extrapolation can reintroduce the other body (which we removed in the first place). When this happens, reinitialization cannot fix the values beyond the kink, since it cannot move the interface reintroduced by extrapolation. A straightforward alternative is to use a zeroth-order extrapolation. This means simply copying the values along the edges into the ghost cells. It is obvious that this will never cross $\phi = 0$, so reinitialization works as intended.

The difference between these two is shown in Figure 4. In (a), a zoom in on the global level set of a droplet touching a pool is shown. In (b), the local level set of the lower body (the pool) is shown after extraction and explicit reconstruction. Here, the values on the edges are not

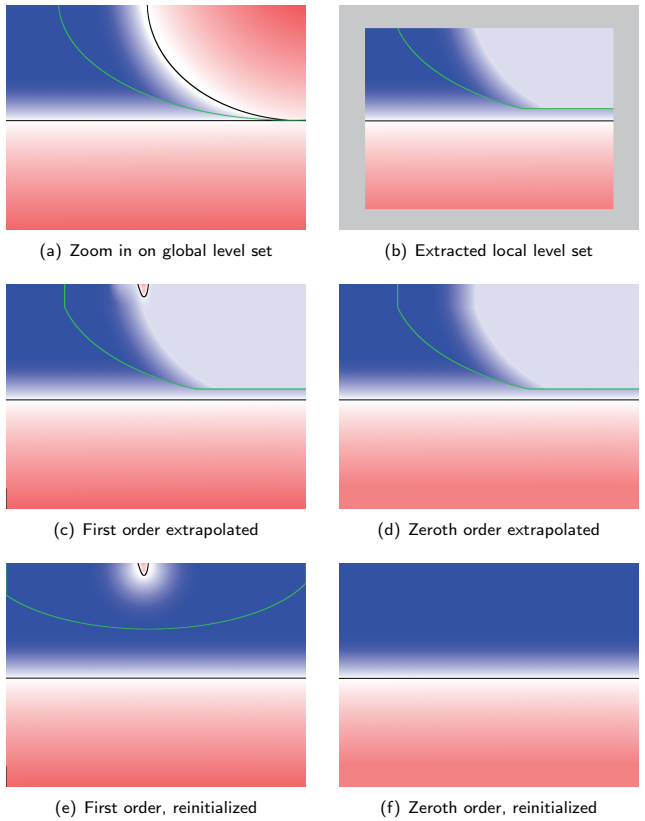


Figure 4: Extraction, extrapolation and reinitialization of the local level set is shown, for the lower body in Figure (a). Red indicates a negative value, blue a positive value, and white indicates zero. The green lines indicate kinks in the level set function, and the black lines are the zero level sets. A detailed explanation of the figures is given in Section 3.3.3. (Figure best viewed in color.)

set, indicated in grey. In (c), the same is shown after first-order extrapolation, and in (d) after zeroth-order extrapolation. In (e), the first-order extrapolated ϕ is shown reinitialized, and in (f) the zeroth-order extrapolated ϕ is shown reinitialized. Note in particular that in (e), a kink still exists after the entire procedure (green line), so the geometric quantities calculated would still be wrong if the derivatives cross the kink.

The corner cells on the boundaries must also be set. Here, these all get the value from the corresponding corner of the internal grid.

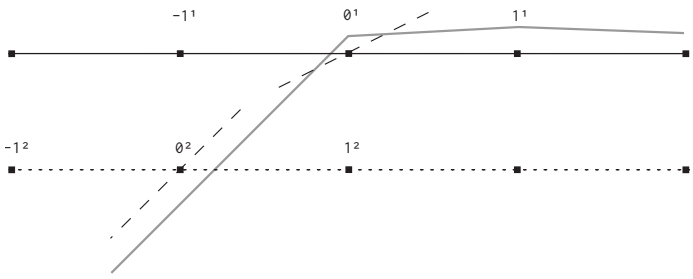


Figure 5: Why we reinitialize from a lower level set: At the lower level set, indicated by the dotted line, values of e.g. $\nabla\phi$ are more accurate at the grid point which is closest to the grey line than for the zero level set. The grey line indicates the local level-set function ϕ. The dashed lines indicate $\nabla\phi$ calculated using central differences.

3.3.4. Reinitialization

When the extracted local level-set has been extrapolated, it must then be reinitialized before the geometric quantities are calculated. This is essential in order to have good values of the level-set function outside the interface. The entire LOLEX method hinges on the fact that reinitialization restores the local level-set to a signed distance function, so that ordinary discretizations will not give errors. This is not entirely straightforward, however.

When reinitializing, we require at least some points on either side of the interface with decent ϕ-values, i.e. ϕ being the signed distance to the interface. In addition to this, we need to know the smeared sign function, and most crucially, the normal vectors at the interface. Thus we are faced with a bootstrapping problem: accurate normal vectors are required in order to accurately calculate the normal vectors. This is only a problem when the global interfaces are very close; when there is a moderate distance (i.e. more than one grid point between the interfaces), the normal vectors can be calculated at the interface using the local level-set.

The solution to this conundrum is to exploit the redundant information which is stored in the level-set function. To illustrate this redundancy, imagine that you are walking along a normal vector to the interface. At each grid point you pass, you are told the current distance to the interface. As long as you do not pass any kinks, this information is redundant: using the value at the first grid point you pass, you can calculate the value at the next grid point, and the one after that, given that you know the grid spacing.

What this means for the present case is that we have information inside the current body that we can use. Most importantly, we can calculate the normal vectors without problem for internal points. This means that we can reinitialize a level set different from $\phi = 0$, e.g. $\phi = -0.8\Delta x$, and get essentially the correct ϕ afterwards. We are not guaranteed to get exactly the correct ϕ, but as we cannot obtain the correct ϕ anyway, we will settle for a good approximation. An illustration of this in 1D is shown in Figure 5, where the extracted local level-set function ϕ is shown in grey. Note that e.g. the value of $\nabla\phi$ at the grid point 0^2, shown with a dashed line, is much closer to 1 than the value at the grid point 0^1. When the lower level set is used, we momentarily move the interface further to the left in this figure, so the grid point 0^2 is closest to the interface. It is obvious that we have a better chance of restoring a signed distance function with the correct location of the interface if we reinitialize from the lower level set.

The value of $-0.8\Delta x$ used here gives the most accurate results. If the value is too close to

zero, the benefit of reinitializing from a lower level set is reduced. However, if the value is too large, we risk having this lower level set too close to the edges of the local domain, and we increase the potential error caused by reinitializing from a different level than zero.

Another problem solved by this is the fact that the values directly outside the zero level set may be incorrect in some cases. In particular, this happens when an outside grid cell is not flagged as dependent, but its value of ϕ still deviates from that dictated by a signed distance function. Tests have shown that this sometimes occurs, and that it distorts the reconstructed local level set.

Reinitializing from a different level may sound somewhat complicated to do, but the implicit formulation springs to the rescue again. To reinitialize from a lower level set, we simply add a positive constant to ϕ at every local grid point, call the reinitialization routine on this ϕ, and then subtract the same constant from the reinitialized ϕ. The effect of this is illustrated in Figure 6, which is an extreme case. Here, reinitialization of two very close bodies (concentric circles) has distorted the global level-set function close to and outside the interfaces. The reinitialized local level-set function is also wrong, but the one which is reinitialized from a lower level set gives a much smoother representation of the interface, which agrees with the contour lines further into the body. This smoother representation will, in turn, give a significantly more accurate curvature. A plot of the curvature calculated with and without this improvement is shown in Figure 7 for the concentric circles case; this global interface configuration can also be seen in Figure 10 further down. This plot shows the curvature along the inner circle. It is seen that the improvement is large, particularly in this case when two interfaces are close. The curvature calculated using the standard method is not shown, as it is outside the y-axis range in this figure.

While the curvature calculated using the LOLEX method is close to the analytical value, there is still a more or less constant error of 1–2%. It turns out that this error is caused by the reinitialization of the local level set, as is indicated in this figure as well. The line captioned 'Forced LOLEX' shows the LOLEX method used on a single interface corresponding to the inner circle. Here, the level-set function is correct and the standard method gives an error for the single interface which is smaller than the line width in this figure. When we force the use of the LOLEX method, the only difference from the standard method is the extrapolation and reinitialization, meaning that these must be the culprits. To mitigate this, a more accurate reinitialization procedure could be used, e.g. the HCR method due to Hartmann et al. [15].

3.3.5. Parameters of the method

In the LOLEX method as presented here, there are a number of parameters that can be varied. An overview of these is given here, along with the values used presently, and sensible ranges, in Table 1.

After the local level sets have been extracted correctly, the standard discretizations can be used to calculate the normal vector and curvature. As the curvature and normal vector cannot be multiply defined at a single grid point, we must combine the information from different local level sets. To do this, we simply select the one corresponding to the interface which is closest to the central point.

As the present method uses reinitialization on a local grid for each grid point where it is used, the performance impact of the method could become large. To avoid this, the quality function $Q(\mathbf{x})$ is used to restrict the use of the method. In a typical falling drop simulation, the present method will only be used in a small percentage of the total number of time steps, and even then,

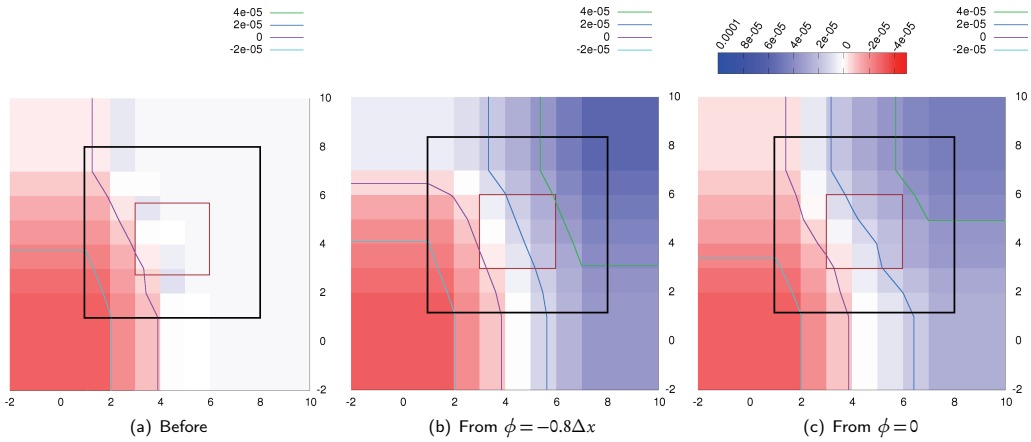


Figure 6: The LOLEX method on a global level set which is distorted due to reinitialization of very close bodies. The global bodies are two concentric circles. (a) Local ϕ before reinitialization. (b) Local ϕ reinitialized from $\phi = -0.8\Delta x$. (c) Local ϕ reinitialized from $\phi = 0.0$. The black square indicates the boundary to the ghost cells, and the red square indicates the 3×3 central points that are used in the final curvature calculation.

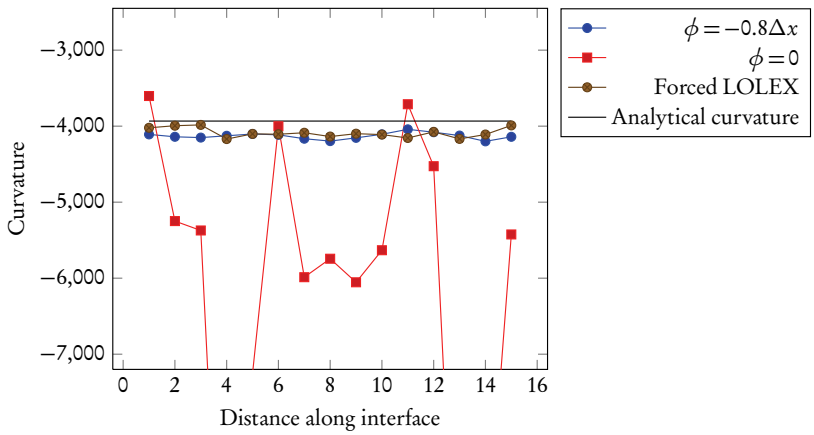


Figure 7: Lineplots of the curvature along the interface when reinitializing from both the zero level set and a lower level set. Also shown are the curvature calculated when forcing use of the LOLEX method on a single interface, and the analytical curvature.

Table 1: Parameters used in the LOLEX method, along with values used and sensible ranges.

Parameter	Value	Sensible range
Local grid size	7	5–11
Gradient threshold η	0.005	0.01–0.001
2D discard threshold	25	(given `ilmax=7` etc.)
3D discard threshold	122	(given `ilmax=7` etc.)
Reinit. level set	$-0.8\Delta x$	$-1.0\Delta x$ to $-0.5\Delta x$

it will typically not be used for all points along the interface. This means the computational cost of the present method has a low impact on the total runtime of a simulation.

3.4. Summary

In this section the presently used LOLEX method has been described in detail. The method is used for grid points where the level-set function deviates from being a signed distance function, where it extracts one or more local level sets, removes any kinks in these by use of reinitialization, and finally uses these local level sets to calculate the curvature and normal vectors. The values corresponding to the interface which is closest to the current grid point is used.

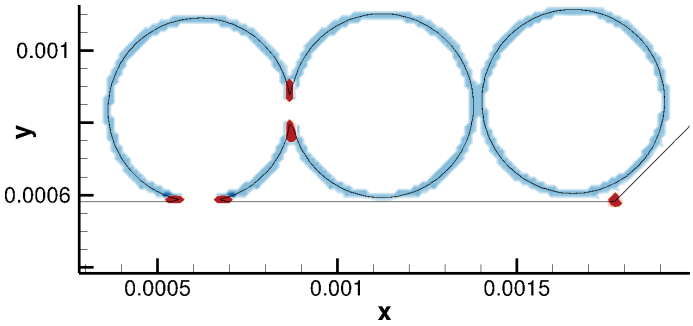
The method is motivated in that it is more general than the previous method by Salac and Lu [11], handling bodies which fold back onto themselves, and it extends more easily to 3D than the previous methods by Macklin and Lowengrub [4, 10] and by Lervåg [12, 13], which use curve-fitting schemes. The parameters of the method are given in Table 1. Results, both for static and dynamic simulations, are given in the next sections.

4. Geometric results

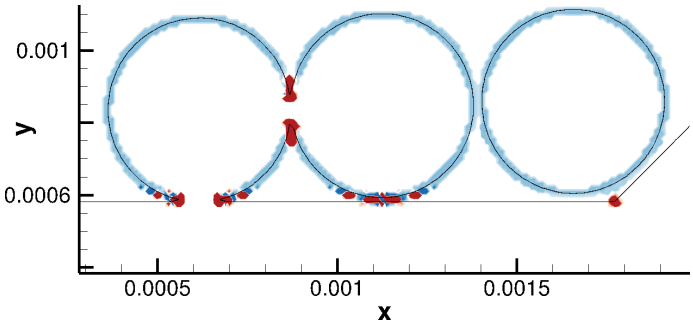
In order to test the LOLEX method, some static interface configurations were used that replicate typical situations occurring in simulations of droplet collisions.

4.1. Circles and straight interfaces

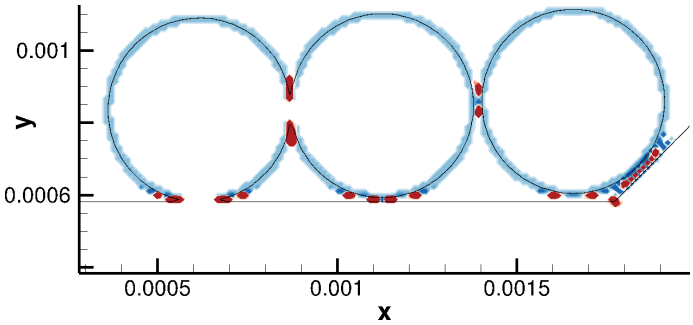
The first test case consists of three circles and a straight-lined interface, where two of the circles and the straight-lined interface are joined together. The results for this case are shown in Figure 8 for the LOLEX method, the SLM, and the standard method. In this figure, the interfaces are shown as black lines, and the color indicates the curvature. The background curvature of 0 is indicated in white, blue indicates a negative curvature and red indicates a positive curvature. The figure illustrates that the standard method produces positive unphysical curvatures several places, both between the circles and the straight interface and between circles. The Salac and Lu method remediates the situation somewhat, but still has problems where the circle folds back onto the straight interface, and at the bottom of the middle circle, which is particularly close to the straight interface. The LOLEX method produces positive curvatures only where they are expected and needed.



(a) LOLEX method



(b) Salac and Lu method



(c) Standard method

Figure 8: Comparison of curvature calculation methods for circles and straight interfaces. The color indicates the curvature; white is zero, blue is negative and red is positive.

4.2. Droplet falling onto a pool

In order to compare the behavior of the LOLEX and the standard method for different interface separations, a test case was considered which mimics a droplet falling onto a pool. In this case, a 0.2 m diameter circle and a horizontal line were initialized in a 1m ×1m domain. The separation between the circle and the line was varied from $3.6\Delta x$ down to $0\Delta x$ in increments of $0.1\Delta x$. For each separation, the curvature was calculated at all points within the narrow band close to the interfaces, and the supremum-norm $||\kappa||_\infty$ of the curvature values was calculated. This was done using the standard and the LOLEX method, for grid resolutions of 64×64, 256×256 and 1024×1024. The analytical curvature is 10 for the circle and 0 for the line, so the supremum norm should be close to 10. The results are shown in Figure 9.

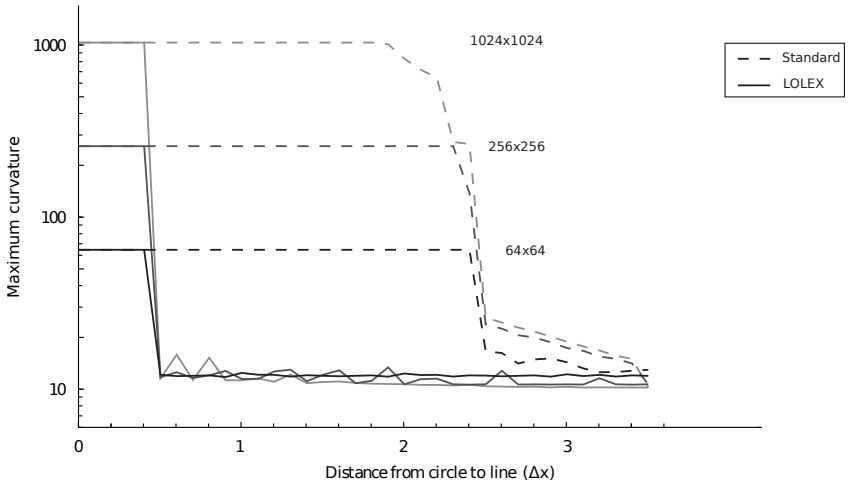


Figure 9: Supremum norm of the curvature for decreasing interface separation. Dashed lines: results using the standard method. Solid lines: results using the LOLEX method. The lines are shaded lighter with increasing grid resolution. The analytical curvature of the circle is 10.

As is seen in this figure, the standard method returns the value used in regularizing the curvature, $||\kappa||_\infty = \frac{1}{\Delta x}$, when the interface separation becomes smaller than about 2.4 Δx. Increasing the grid resolution does not improve the situation. Note that the y axis in this plot is logarithmic. Meanwhile, the LOLEX method gives decent values close to the analytical value of 10 all the way up to when the interfaces merge, which happens at a separation of 0.2 Δx. It is seen that the small deviations for the LOLEX method are reduced when the grid resolution is increased.

In addition to the curvature, accurate normal vectors close to the interface are desirable in level-set simulations. The importance in reinitialization has been suggested above, coming from the fact that normal vectors are used in finding the upwind direction. Normal vectors are equally important in calculating the extension velocity, where an error would lead to the interface not moving according to the flow.

4.3. Concentric circles

In order to compare the proposed method to the standard method, a geometric test case was considered which replicates the demands of simulating merging interfaces. The calculated normal vectors are compared both to the standard central-differences discretization, to a directional-differences discretization as described in [4], and to the curve-fitting method of Lervåg [13].

In this test case, two concentric circles were initialized, as if we had a thin ring of fluid 1 inside fluid 2. The width of this ring was $1.6\Delta x$. This test case is interesting, since it reveals grid effects or anisotropies. It also replicates the situation of a thin film that forms between a droplet and a pool for cases where the droplet deforms the pool surface before merging. This has been observed experimentally, see e.g. [34]. The results for all four methods are shown in Figure 10.

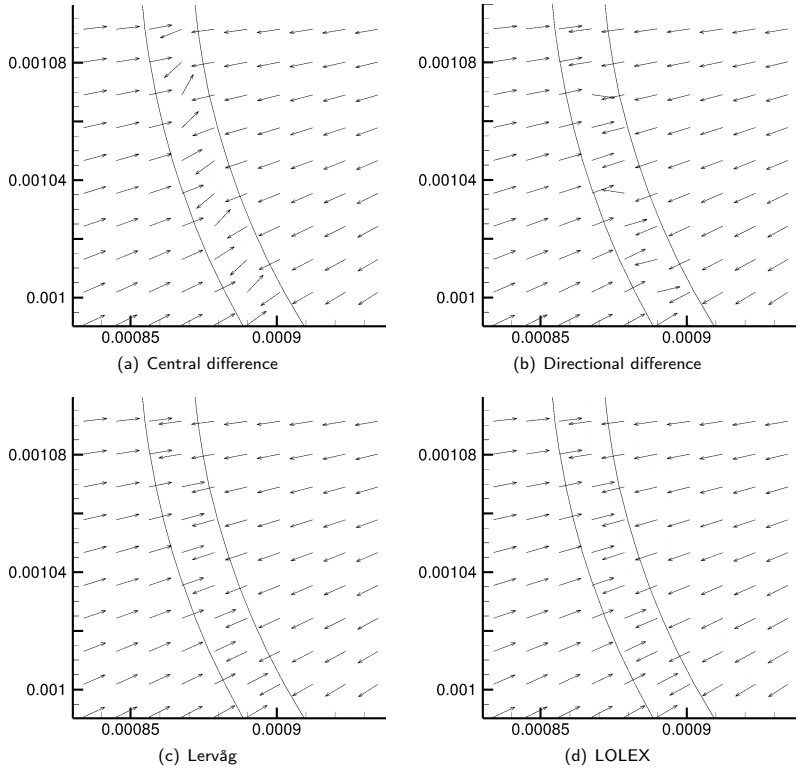


Figure 10: Comparison of normal vector calculations using different methods.

In this figure it is seen that the directional difference method is not much better than the central difference method. This is partly what prompted the use of curve fitting methods;

Macklin and Lowengrub initially used directional differences and additional grid refinement in [4], but switched to curve-fitting methods in [10]. As is seen in Figure 10 (c), curve fitting methods (the method by Lervåg is used here) give the correct result. In (d), we see that the LOLEX method also gives the correct result. It is impossible to distinguish the results in (c) and (d) without overlaying the figures and zooming in a lot. The difference is too small to have any impact on the simulation results.

As pointed out several times already, the main advantage of the present LOLEX method over methods employing curve fitting is that it scales easily to 3D. This is because the present method retains the implicitness of the level-set method. A 3D extension of the Macklin and Lowengrub method, on the other hand, would fit a local surface to the point of interest, as they indicate in [10]. Curvature estimation in 3D based on local surface fitting has long been a topic of research in computer vision, see [35] for a review of various methods including the use biquadratic surfaces and of splines. The conclusion of [35] is that these methods are very sensitive to numerical noise (in their context, sensor noise). In the current case, noise is to be expected, as can be seen in Figure 6 (b). Due to this fact, methods in computer vision that avoid local surface fitting and calculate only the sign of the curvature have been introduced, since this quantity can be calculated more reliably[36]. This is not a viable alternative in two-phase flow simulations as considered here.

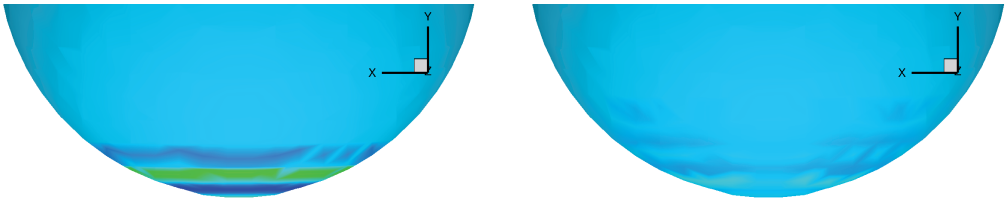
4.4. 3D bubble above a plane

A curvature calculation using the LOLEX method on a 3D case is shown in Figure 11. In this case, a bubble is placed above a plane, with distance 1.2 Δx at the closest. The grid is $50\times50\times50$, and the bubble radius is 12.5 Δx. The surfaces are colored according to the curvature (interpolated to the surface). In Figure 11 (a), the standard method is used. In 3D, this is the 27-point stencil given by Kang et al. [19]. In Figure 11 (b), the LOLEX method is used to extract the local level sets, and the curvature is then calculated using the same 27-point stencil on these local level sets. It is seen that the LOLEX method performs much better than the standard discretization in areas where the bubble and plane are in close proximity. Note that the plane is not shown here, only the bubble. The kink in the global ϕ is below the bubble.

Comparing to the analytical curvature, which in this case is -10 for the spherical bubble, it is seen that the standard discretization performs well away from kinks, where the variation in curvature is at most $\pm0.2\%$. Close to the kink, the standard discretization has errors of $\pm80\%$, seen as green and dark blue bands in Figure 11 (a). The LOLEX method has the same variation as the standard method away from kinks, while the variation is $\pm2\%$ close to the kink, seen as light blue spots in Figure 11 (b). Thus it is seen that the LOLEX method gives an error which is an order of magnitude lower than the standard method close to kinks in the level-set function. There is still a small error of the same size as reported above in 2D, which is again probably caused by reinitialization. A deviation of this magnitude is unlikely to have a large impact on simulations, in contrast to the errors from the standard discretization.

To the knowledge of the authors, improved calculation of geometric quantities for a pure level-set formulation in three spatial dimensions that handles general topologies have not been reported before in the literature. Salac and Lu report results of 3D simulations in [11], but it is not known how (or if) they handle problems like that illustrated in Figure 2, i.e. a body folding back onto itself. They also do not discuss the problem of needing good normal vectors at the interface in order to solve the reinitialization equation.

Given the current state of developments toward petascale supercomputers, and particularly the rapid evolution in GPU-accelerated solvers, dynamic 3D level-set simulations of colliding



(a) Standard discretization (b) LOLEX method

Figure 11: 3D bubble above a plane (not shown). Comparison of the standard curvature discretization (a) and the LOLEX method (b). The surfaces are colored according to the curvature, and the standard method is seen to give large errors close to the kink in the level-set function (which is below the sphere), seen as green and dark blue bands.

bodies are going to become more and more common. As this happens, a method such as the present one will be necessary in order to get trustworthy results for situations where accurate curvature is important.

5. Dynamic results

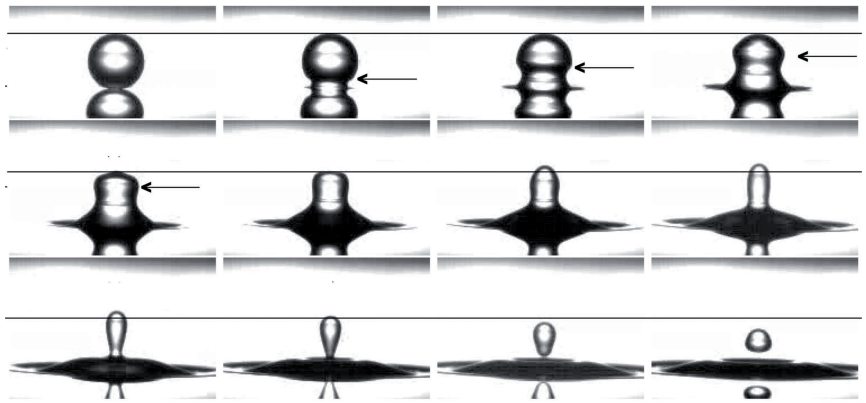
As discussed previously, the case of a single droplet of liquid falling onto a pool of the same liquid, either through gas or another liquid, has been widely studied. Thus it is a good benchmark on which to test the proposed method, since detailed experimental results are available.

When considering this case, the main dichotomy is between a droplet falling through gas and a droplet falling through liquid. We will consider both cases here, since both are interesting from an industry standpoint. These two cases present different challenges to numerical simulations. The liquid-in-gas case has a high density difference between the two fluids, which is known to be a difficult case. Sussman et al. have studied this problem, and have produced good results using a hybrid LSM-VOF method [17]. The liquid-in-liquid case, on the other hand, can be time-consuming to simulate due to the viscous term in the CFL-criterion used here [19], but is not challenging with respect to density differences.

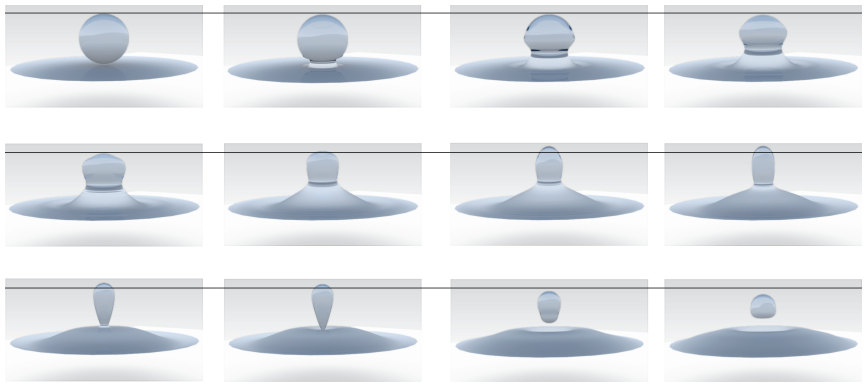
5.1. Decane droplet in water merging with decane pool

The simulation discussed here consider two immiscible liquids, where a droplet of the heaviest liquid is placed in the lightest liquid above a pool of the heaviest liquid. In the experimental work by Chen et al. [37], the droplet is made to rest on the pool, and then merging happens after some time. The heavy liquid is water, and the light liquid is a mix of 20 % polybutene in decane. The droplet diameter is 1.1 mm. As the droplet and interface are brought into proximity, a thin film is formed between them. This thin film drains, and after some time the film ruptures and the droplet merges with the interface. In the Chen et al. experiments, the merging happens at the central point, but off-center merging has also been reported for larger droplets [38].

A simulation was performed with the same fluid properties and droplet dimension as reported by Chen et al. The computational domain was 6×6 mm, the grid was 400×400, and



(a) Experimental result



(b) Simulation result

Figure 12: A 1.1 mm diameter water droplet merging with a water pool. The ambient fluid is 20% polybutene in decane. Snapshots are taken 542 μs apart, the arrow indicates the capillary wave, and the horizontal lines indicate the top of the bubble in the first frame. Figure (a) is the experimental result, reprinted with permission from [37], copyright 2006 American Institute of Physics. Figure (b) is the simulation result.

the CFL-number was 0.8. The results are shown in Figure 12. The agreement between the simulation and experimental results is very good.

In this simulation, the point of merging is decided mainly by grid effects when the droplet deforms the interface forming a thin film. With the present method, we must simply hope that precisely what happens at the time of merging does not significantly affect the following behaviour. Comparing Figure 12 (a) and (b) indicates that in this case the precise mechanisms of merging are not very important, as the numerical and experimental results agree very nicely. To accurately capture the thin film behaviour, the grid resolution would have to be extremely fine.

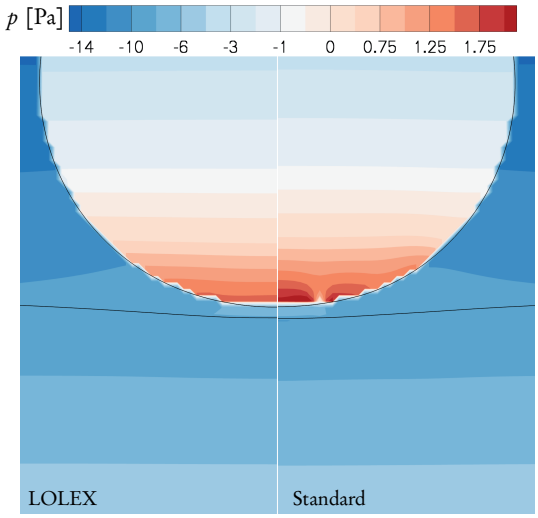


Figure 13: Water droplet in a mixture of polybutene and decane, about to merge with a water pool. Comparison of the pressure field using the LOLEX method and the standard method at $t = 0.007$ s.

Hodgson and Lee [39] report that the width of the thin film between a droplet and a pool for the water-toluene system they study is four orders of magnitude smaller than the droplet diameter, i.e. around 100 nanometers. It is possible that an adaptive grid could be able to resolve such a thin film, but since there is no analog to the Knudsen number for liquids, it is not immediately clear whether the continuum description of the Navier-Stokes equations still holds at this length scale.

Comparing the LOLEX method and the standard method on this case, the standard method gives a more oscillatory pressure field around the contact point, as seen in Figure 13. This increased pressure inside the thin film delays the rupture of the film, seen as a slightly increased width of the film in Figure 13 (b).

5.2. Water droplet falling through air onto a water pool

Considering the case of a liquid in gas, a simulation was performed of a 0.18 mm diameter water droplet falling through air at 0.29 m/s and impacting a deep pool of water. Experimental results for this case due to Zhao, Brunsvold and Munkejord are found in [40]. These results indicate that a partial coalescence occurs, but the high-speed camera used was not fast enough to capture all the details of the partial coalescence process.

The simulation was performed using axisymmetry. The computational domain was 0.7 mm × 0.7 mm, resolved using a 401×401 Cartesian grid. The CFL number was 0.25. The LOLEX method was used for curvature and normal vector calculation. A comparison of the experimental and simulation results are shown in Figure 14.

The time intervals between frames for the experimental and simulation results do not match in this figure. The intervals between the second and third frames are the ones that match best,

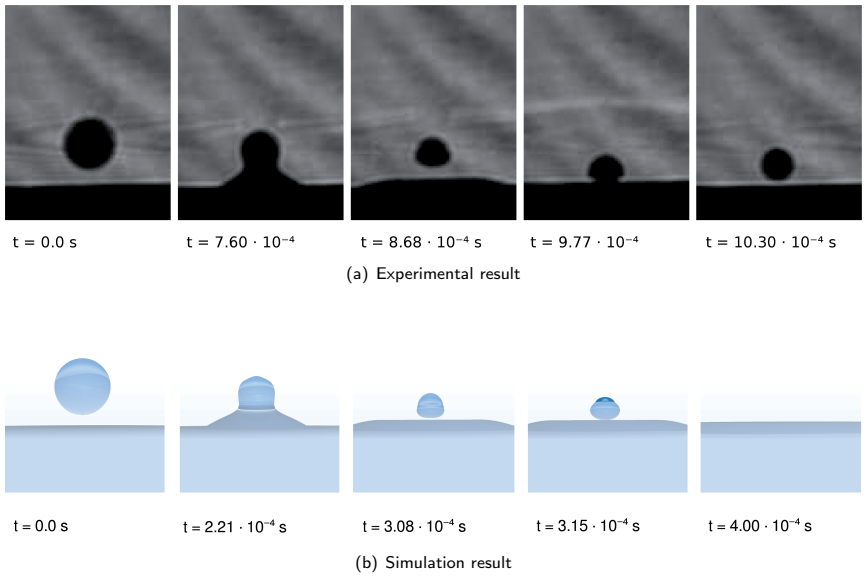


Figure 14: Experimental results (top) and simulation results (bottom) for a 0.18 mm water droplet falling through air and impacting a deep pool of water at 0.29 m/s. Figure (a) is reprinted from [40], Copyright (2011), with permission from Elsevier.

suggesting that the behaviour of the thin air film that forms between the droplet and the pool before coalescence is the major source of this discrepancy. The grid used in the simulation is unable to resolve the thin film. It is not clear that an increased grid resolution would amend this, as the continuum approximation may not be valid for the thin air film. The width of this film is not known from experiments.

As an order-of-magnitude estimate, we can use the results by Hodgson and Lee [39]. They report that the width of the thin film between a droplet and a pool before merging, for the water-toluene system they study, is around $L = 100$ nanometers. Since the mean free path in air at room temperature and atmospheric pressure is around $\lambda = 66$ nanometers [41], the Knudsen number is $Kn = \frac{\lambda}{L} \approx 0.7 \not\ll 1$, which would imply that the continuum description is no longer valid.

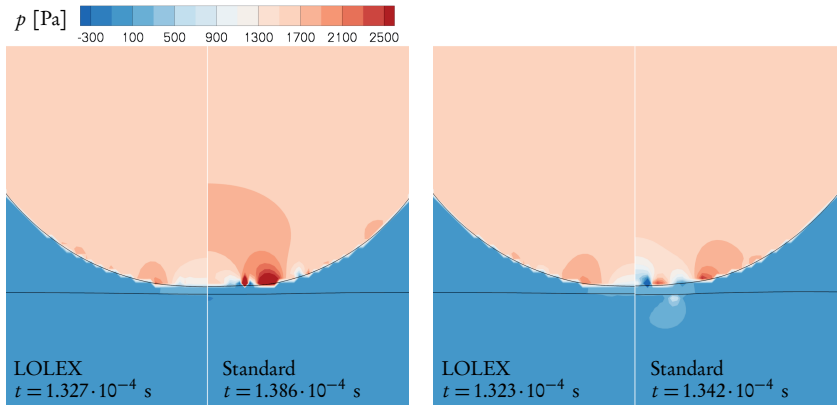
Nevertheless, the simulation is able to correctly predict the partial coalescence, and the simulation agrees well with experiments on the size of the daughter droplet produced. In the experiments, this daughter droplet subsequently bounces on the pool of water. The simulation is unable to predict this, again due to the thin air film formed, and shows the daughter droplet merging with the water pool instead.

A comparison between the LOLEX method and the standard method is shown in Figures 15 and 16. These figures show a section through the droplets just before collision and just when the neck is at its tallest, respectively. The pressure field is plotted as colored contours. The LOLEX method is plotted on the left side and the standard method is plotted on the right side.

It is seen from these figures that the curvature errors produced by the standard method give rise to significant oscillations in the pressure; note in particular the interleaved red and blue patches where the pressure changes sign. As the reinitialization is performed more frequently, the oscillations persist, and are even found inside the pool below the droplet.

An important effect of this erroneous pressure is a loss of kinetic energy, which can be seen in Figure 16, where the neck is clearly shorter with the standard method. It is also seen that more frequent reinitialization leads to a higher loss of kinetic energy. As some authors have noted [42], the height of the neck and the dynamics of the capillary waves are important factors for the partial coalescence mechanism.

The LOLEX method is not significantly affected by the amount of reinitialization, and gives a more sensible pressure field in both cases. It should be noted that the pressure difference across the droplet interface in Figure 15 is about 2500 Pa, which is very large, caused by the very small droplet diameter.



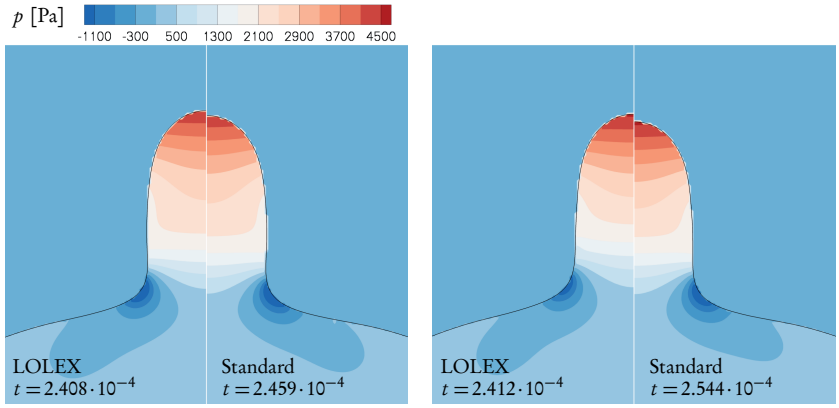
(a) Reinitialization every 7 time steps

(b) Reinitialization every time step

Figure 15: Water droplet falling onto a pool, just before the interfaces merge. Comparison between the LOLEX method and the standard method. The pressure field is shown as colored contours.

6. Concluding remarks

In the present work we have proposed a new method for calculating the curvature and normal vectors of an interface represented by a level-set function, and which gives accurate results before, during and after topological changes in the interface. The method is compared to the standard method for geometric test cases, where the analytical curvature is known, and it is seen that in areas where the standard method gives errors of around 100 %, the proposed method gives errors of around 1–2 %. The method is easily extended to 3D, as is demonstrated, where the same reduction in error is seen. The method is then employed in simulations of two-phase flow where a droplet merges with a pool. Here it is seen that the standard method gives rise to unphysical pressure oscillations before merging, which affect the subsequent capillary



(a) Reinitialization every 7 time steps

(b) Reinitialization every time step

Figure 16: Water droplet falling onto a pool, when the neck reaches its highest position. Comparison between the LOLEX method and the standard method. The pressure field is shown as colored contours.

waves, while the proposed method fares much better. The results of the simulations using the proposed method are compared to experimental results both for a liquid-in-liquid case, where the agreement is very good, and for a more demanding liquid-in-gas case where the agreement is qualitative, reproducing the partial coalescence behaviour.

Acknowledgements

This work was financed through the Enabling Low-Emission LNG Systems project at SINTEF Energy Research, and the authors acknowledge the contributions of GDF SUEZ, Statoil and the Petromaks programme of the Research Council of Norway (193062/S60).

References

[1] L. Rayleigh, in: Proc. R. Soc., 28, p. 406.
[2] A. M. Worthington, in: Proc. R. Soc., 25, pp. 261–272.
[3] S. T. Thoroddsen, K. Takehara, T. G. Etoh, S. Popinet, P. Ray, C. Josserand, S. Zaleski, M.-J. Thoraval, von Kármán vortex street within an impacting drop, Phys. Rev. Lett. 108 (2012) 264506.
[4] P. Macklin, J. Lowengrub, Evolving interfaces via gradients of geometry-dependent interior Poisson problems: application to tumor growth, Journal of Computational Physics 203 (2005) 191 – 220.
[5] V. Mallet, D. Keyes, F. Fendell, Modeling wildland fire propagation with level set methods, Computers and Mathematics with Applications 57 (2009) 1089 – 1101.
[6] V. Melicher, I. Cimrak, R. V. Keer, Level set method for optimal shape design of MRAM core. Micromagnetic approach, Physica B: Condensed Matter 403 (2008) 308 – 311.
[7] S. Osher, R. P. Fedkiw, Level set methods: An overview and some recent results, Journal of Computational Physics 169 (2001) 463 – 502.
[8] S. Osher, J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics 79 (1988) 12 – 49.

[9] P. Smereka, Semi-implicit level set methods for curvature and surface diffusion motion, Journal of Scientific Computing 19 (2003) 439–456.
[10] P. Macklin, J. Lowengrub, An improved geometry-aware curvature discretization for level set methods: Application to tumor growth, Journal of Computational Physics 215 (2006) 392 – 401.
[11] D. Salac, W. Lu, A local semi-implicit level-set method for interface motion, Journal of Scientific Computing 35 (2008) 330–349.
[12] K. Y. Lervåg, Calculation of interface curvature with the level-set method, in: Sixth National Conference on Computational Mechanics MekIT'11, Trondheim, Norway.
[13] K. Y. Lervåg, B. Müller, S. T. Munkejord, Calculation of the interface curvature and normal vector with the level-set method (2012). Submitted.
[14] S. O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible, multi-fluid flows, Journal of Computational Physics 100 (1992) 25 – 37.
[15] D. Hartmann, M. Meinke, W. Schröder, The constrained reinitialization equation for level set methods, Journal of Computational Physics 229 (2010) 1514 – 1535.
[16] J. E. Pilliod, Jr., E. G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, Journal of Computational Physics 199 (2004) 465 – 502.
[17] M. Sussman, K. Smith, M. Hussaini, M. Ohta, R. Zhi-Wei, A sharp interface method for incompressible two-phase flows, Journal of Computational Physics 221 (2007) 469 – 505.
[18] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, A PDE-based fast local level set method, Journal of Computational Physics 155 (1999) 410 – 438.
[19] M. Kang, R. P. Fedkiw, X.-D. Liu, A boundary condition capturing method for multiphase incompressible flow, Journal of Scientific Computing 15 (2000) 323–360.
[20] E. B. Hansen, Numerical simulation of droplet dynamics in the presence of an electric field, Ph.D. thesis, NTNU, 2005.
[21] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, Journal of Computational Physics 114 (1994) 146 – 159.
[22] R. P. Fedkiw, X. D. Liu, The Ghost Fluid Method for viscous flows, Presented at the "Solutions of PDE" Conference in honour of Prof. Phil Roe, 1998.
[23] A. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comp 22 (1968) 745–762.
[24] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, H. Zhang, PETSc Web page, 2012. http://www.mcs.anl.gov/petsc.
[25] J. Kraaijevanger, Contractivity of Runge–Kutta methods, BIT Numerical Mathematics 31 (1991) 482–528.
[26] D. I. Ketcheson, A. C. Robinson, On the practical importance of the SSP property for Runge–Kutta time integrators for some common Godunov-type schemes, International Journal for Numerical Methods in Fluids 48 (2005) 271–303.
[27] G.-S. Jiang, C.-W. Shu, I. L, Efficient implementation of weighted ENO schemes, J. Comput. Phys 126 (1996) 202–228.
[28] K. Y. Lervåg, Å. Ervik, Curvature calculations for the level-set method, in: ENUMATH 2011 Proceedings Volume, Leicester, England.
[29] C. Focke, D. Bothe, Direct numerical simulation of binary off-center collisions of shear thinning droplets at high Weber numbers, Physics of Fluids 24 (2012) 73105.
[30] P. Trontin, S. Vincent, J. Estivalezes, J. Caltagirone, A subgrid computation of the curvature by a particle/level-set method. application to a front-tracking/ghost-fluid method for incompressible flows, Journal of Computational Physics 231 (2012) 6990 – 7010.
[31] P. Verbeek, L. van Vliet, J. van de Weijer, Improved curvature and anisotropy estimation for curved line bundles, in: Jain, AK and Venkatesh, S and Lovell, BC (Ed.), Fourteenth International Conference on Pattern Recognition, Vols 1 and 2, International Conference on Pattern Recognition, pp. 528–533.
[32] D. Adalsteinsson, J. A. Sethian, A fast level set method for propagating interfaces, Journal of Computational Physics 118 (1995) 269 – 277.
[33] D. Adalsteinsson, J. A. Sethian, The fast construction of extension velocities in level set methods, Journal of Computational Physics 148 (1999) 2 – 22.
[34] Z. Mohamed-Kassim, E. K. Longmire, Drop impact on a liquid–liquid interface, Physics of Fluids 15 (2003) 3263–3273.
[35] P. Flynn, A. Jain, On reliable curvature estimation, in: Computer Vision and Pattern Recognition, 1989. Proceedings CVPR '89., IEEE Computer Society Conference on, pp. 110 –116.
[36] C.-K. Tang, G. Medioni, Curvature-augmented tensor voting for shape inference from noisy 3D data, Pattern Analysis and Machine Intelligence, IEEE Transactions on 24 (2002) 858 –864.
[37] X. Chen, S. Mandre, J. J. Feng, Partial coalescence between a drop and a liquid-liquid interface, Physics of Fluids 18 (2006) 051705.

[38] Z. Mohamed-Kassim, E. K. Longmire, Drop coalescence through a liquid/liquid interface, Physics of Fluids 16 (2004) 2170–2181.
[39] T. Hodgson, J. Lee, The effect of surfactants on the coalescence of a drop at an interface i, Journal of Colloid and Interface Science 30 (1969) 94 – 108.
[40] H. Zhao, A. Brunsvold, S. T. Munkejord, Transition between coalescence and bouncing of droplets on a deep liquid pool, Journal of Multiphase Flow 37 (2011) 1109–1119.
[41] S. G. Jennings, The mean free path in air, Journal of Aerosol Science 19 (1988) 159–166.
[42] F. Blanchette, T. P. Bigioni, Partial coalescence of drops at liquid interfaces, Nature Physics 2 (2006) 254–257.

E

Towards a second-order diffuse-domain approach for solving PDEs in complex geometries

K. Y. Lervåg and J. Lowengrub
Submitted to Communications in Mathematical Sciences, 2013

TOWARDS A SECOND-ORDER DIFFUSE-DOMAIN METHOD FOR SOLVING PDES IN COMPLEX GEOMETRIES*

KARL YNGVE LERVÅG† AND JOHN LOWENGRUB‡

Abstract. In recent work, Li et al. (Comm. Math. Sci. (2009) 7:81-107) developed a first-order accurate diffuse-domain method (DDM1) for solving partial differential equations in complex, dynamic geometries with Dirichlet, Neumann and Robin boundary conditions. Here, we extend this approach and develop higher order accurate diffuse-domain approximations focusing on Neumann and Robin boundary conditions. The diffuse-domain method uses an implicit representation of the geometry, where the sharp boundary is replaced by a diffuse layer, the equations are reformulated on a larger regular domain and the boundary conditions are incorporated via singular source terms. The resulting PDE is shown to converge asymptotically to the original problem. The present contribution is to include higher-order corrections to the diffuse formulation in order to obtain a second-order accurate approximation. Our analysis shows that the second-order DDM (DDM2) converges asymptotically with second-order to the original problem. The DDM2 system is then investigated numerically and the results are compared with those from the DDM1 system for selected cases with both Neumann and Robin boundary conditions. Two different approximations for the boundary conditions are also compared, which correspond to different diffuse-interface surface delta functions. The results indicate that the global accuracy and convergence of DDM2 is better than DDM1, although both DDM1 and DDM2 generally perform well and the global convergence rate is around two for each. The choice of boundary-condition approximation is also important for rapid global convergence and high accuracy. Approximating the surface delta function by the modulus of the gradient of the phase-field function yields more accurate and robust results than an alternative approximation of the surface delta function based on a scaled version of the squared modulus.

Key words. numerical solution of partial differential equations, phase-field approximation, implicit geometry representation, matched asymptotic analysis.

1. Introduction

There are many problems in computational physics that involve solving partial differential equations (PDEs) in complex geometries. Examples include fluid flows in complicated systems, vein networks in plant leaves, and tumors in human bodies. Standard solution methods for PDEs in complex domains typically involve triangulation and unstructured grids. This rules out coarse-scale discretizations and thus efficient geometric multi-level solutions. Also, mesh generation for three-dimensional complex geometries remains a challenge, in particular if we allow the geometry to evolve with time.

In the past several years, there has been much effort put into the development of numerical methods for solving partial differential equations in complex domains. However, most of these methods typically require non-standard tools not frequently available in standard finite element and finite difference software packages. Examples of such approaches include the extended and composite finite element methods (e.g., [29, 10, 22, 11, 30, 48, 6, 3]), immersed interface methods (e.g., [37, 39, 53, 40, 58]), matched interface and boundary methods (e.g., [65, 62, 61, 60, 64]), modified finite volume/embedded boundary/cut-cell methods/ghost-fluid methods (e.g., [25, 33, 19, 23, 24, 32, 42, 63, 43, 8, 41, 57, 44, 7]). In another approach, known as the fictitious domain method (e.g., [26, 27, 49, 31]), the original system is either augmented with

*Received: ...

†Department of Energy and Process Engineering, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway (karl.y.lervag@ntnu.no).

‡Department of Mathematics, University of California, Irvine, Irvine CA-92697, USA (lowengrb@math.uci.edu).

equations for Lagrange multipliers to enforce the boundary conditions, or the penalty method is used to enforce the boundary conditions weakly.

Here, we follow an alternate approach and use the diffuse domain method for simulating PDEs in complex, non-standard domains. In this approach, the domain is represented implicitly by a phase-field function, which is an approximation of the characteristic function of the domain. The domain boundary is replaced by a narrow diffuse interface layer such that the phase field function rapidly transitions from one inside the domain to zero in the exterior of the domain. The boundary of the domain can thus be represented as an isosurface of the phase field function. The PDE is then reformulated on a larger, regular domain with additional source terms that approximate the boundary conditions. This diffuse domain method does not require any modification of standard finite element or finite difference software. Although uniform grids can be used, local grid refinement near domain boundaries improves efficiency and enables the use of smaller interface thicknesses than are achievable using uniform grids. A related approach involves the level-set method [46, 52, 45] to describe the implicitly embedded surface and to obtain the appropriate surface operators (e.g., [28]).

The diffuse domain method (DDM) was introduced by Kockelkoren et al. [35] to study diffusion inside a cell with zero Neumann boundary conditions at the cell boundary (a similar approach was also used by [4, 5] using spectral methods). The DDM was later used to simulate electrical waves in the heart [20], and in Levine and Rappel [36] to simulate membrane-bound Turing patterns using bulk diffusion coupled to an ODE on the membrane surface. More recently, diffuse-interface methods have been developed for solving PDEs on stationary surfaces [50], evolving surfaces [9, 13, 14, 17, 16, 15], for solving PDEs in complex evolving domains with Dirichlet, Neumann and Robin boundary conditions by Li et al. [38] and by Teigen et al. [55] who modeled bulk-surface coupling. The DDM was also used by Aland et al. [1] to simulate incompressible two-phase flows in complex domains in 2D and 3D, and by Teigen et al. [56] to study two-phase flows with soluble surfactants.

The method of matched asymptotic expansions can be used to show that a given DDM formulation converges to the correct sharp-interface problem. As shown by Li et al. [38], there exist several approximations to the physical boundary conditions that converge asymptotically to the correct sharp-interface problem. Li et al. present some numerical convergence results for a few selected problems. However they do not perform any quantitative comparison between the different boundary-condition approximations. Recently, Reuter et al. [51] reformulated the DDM using an integral equation solver and demonstrated that their generalized DDM, with appropriate choices of approximate surface delta functions, converges with first order accuracy in the diffuse-interface width for solutions of the Poisson equation with Dirichlet boundary conditions.

The current versions of the DDM were developed such that they converge asymptotically to the sharp-interface problems with first order in the diffuse interface width. Inspired by the work of Karma and Rappel [34] and Almgren [2], who incorporated second-order corrections in their phase field models of crystal growth and by the work of Folch et al. [21] who added second-order corrections in phase field models of advection, the present article aims to further generalize the DDM by deriving corrections that give asymptotically second-order approximations to the sharp-interface problem in the diffuse interface width. The second-order DDM (DDM2) is then compared to the first-order DDM (DDM1) from Li et al. [38] on a set of test cases.

In addition, the present work considers a more quantitative comparison of two different boundary condition approximations derived in [38]. The first approximation uses the modulus of the gradient of the phase-field function, and the second approximation uses a scaled version of the squared modulus. The former is shown to be more natural when considering the second-order extension.

Although the DDM is applicable to transient problems and geometries that evolve with time, we will in this article only consider stationary problems. However, our approach applies to transient problems in the same way as shown in [38].

The outline of the paper is as follows. In Section 2 we introduce the diffuse-domain method and derive the second-order corrections. In Section 3 the numerical methods are described, and in Section 4 the test cases are introduced and numerical results are presented and discussed. We finally give some concluding remarks in Section 5.

2. The diffuse-domain method

The main idea of the diffuse-domain method (DDM) is to extend PDEs that are defined inside complex and possibly time-dependent domains into larger, regular domains. As a model problem, we will consider the Poisson equation in a domain D,

$$\boldsymbol{\Delta} u = f, \tag{2.1}$$

with Neumann or Robin boundary conditions. As shown in Li et al. [38], the results for the Poisson equation can be used directly to obtain diffuse-domain methods for more general second-order partial differential equations in evolving domains.

The DDM is defined in a larger domain $\Omega \supset D$ as

$$\boldsymbol{\nabla} \cdot (\phi \boldsymbol{\nabla} u) + BC = \phi f, \tag{2.2}$$

see Figure 2.1. Here ϕ approximates the characteristic function of D,

$$\chi_D = \begin{cases} 1 & \text{if } x \in D, \\ 0 & \text{if } x \notin D, \end{cases} \tag{2.3}$$

and BC is chosen to approximate the physical boundary condition, cf. [38], which typically involves diffuse-interface approximations of the surface delta function. A standard approximation of the characteristic function is the phase-field function,

$$\chi_D \simeq \phi(\boldsymbol{x},t) = \frac{1}{2}\left(1 - \tanh\left(\frac{3r(\boldsymbol{x},t)}{\epsilon}\right)\right). \tag{2.4}$$

Here ϵ is the interface thickness and $r(\boldsymbol{x},t)$ is the signed-distance function with respect to ∂D, which is taken to be negative inside D.

For a number of different choices of approximations for the boundary conditions, as detailed in [38], the standard DDM is shown to converge asymptotically to the sharp-interface equation when ϵ is decreased by use of the method of matched asymptotic expansions. In the following, the DDM will be extended to include corrections that give second-order asymptotic convergence in ϵ, for Robin and Neumann boundary conditions.

2.1. Asymptotic analysis

To show asymptotic convergence, we need to consider the expansions of the diffuse-domain variables in powers of the interface thickness ϵ in regions close to and far from the interface. These will be called inner and outer expansions, respectively.

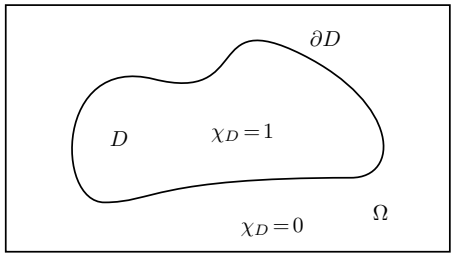


FIGURE 2.1. *A complex domain D covered by a larger, regular domain Ω.*

The two expansions are then matched in a region where both are valid. See Figure 2.2 and [47].

The outer expansion for the variable $u(\boldsymbol{x};\epsilon)$ is simply

$$u(\boldsymbol{x};\epsilon) = u^{(0)}(\boldsymbol{x}) + \epsilon u^{(1)}(\boldsymbol{x}) + \epsilon^2 u^{(2)}(\boldsymbol{x}) + \cdots. \tag{2.5}$$

The outer expansion of an equation is then found by inserting the expanded variables into the equation.

The inner expansion is found by introducing a local coordinate system near the interface ∂D,

$$\boldsymbol{x}(s,z;\epsilon) = \boldsymbol{X}(\boldsymbol{s};\epsilon) + \epsilon z \boldsymbol{n}(\boldsymbol{s};\epsilon), \tag{2.6}$$

where $\boldsymbol{X}(\boldsymbol{s};\epsilon)$ is a parametrization of the interface, $\boldsymbol{n}(\boldsymbol{s};\epsilon)$ is the interface normal vector that points out of D, z is the stretched variable

$$z = \frac{r(\boldsymbol{x})}{\epsilon}. \tag{2.7}$$

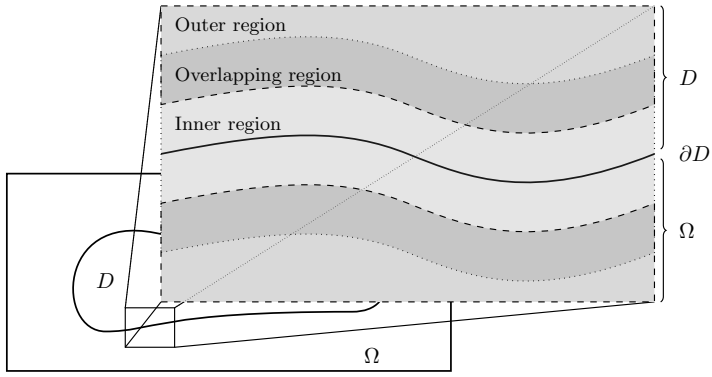


FIGURE 2.2. *A sketch of the regions used for the matched asymptotic expansions. The inner region is marked with a light gray color and the outer region with a slightly darker gray color. The overlapping region is marked with the darkest gray color.*

and r is the signed distance from the point $\boldsymbol{x}$ to ∂D, where r is taken to be negative inside D. In the local coordinate system, the derivatives become

$$\boldsymbol{\nabla} = \frac{1}{\epsilon}\boldsymbol{n}\partial_z + \boldsymbol{\nabla}_{\mathrm{s}}, \tag{2.8}$$

$$\boldsymbol{\Delta} = \frac{1}{\epsilon^2}\partial_{zz} + \frac{1}{\epsilon}\kappa\partial_z + \boldsymbol{\Delta}_{\mathrm{s}}, \tag{2.9}$$

where $\kappa \equiv \boldsymbol{\nabla}_{\mathrm{s}}\cdot\boldsymbol{n}$ is the curvature of the interface. The inner variable $\hat{u}(z,\boldsymbol{s};\epsilon)$ is now given by

$$\hat{u}(z,\boldsymbol{s};\epsilon) \equiv u(\boldsymbol{x};\epsilon) = u(\boldsymbol{X}(\boldsymbol{s};\epsilon) + \epsilon z\boldsymbol{n}(\boldsymbol{s};\epsilon);\epsilon), \tag{2.10}$$

and the inner expansion is

$$\hat{u}(z,\boldsymbol{s};\epsilon) = \hat{u}^{(0)}(z,\boldsymbol{s}) + \epsilon\hat{u}^{(1)}(z,\boldsymbol{s}) + \epsilon^2\hat{u}^{(2)}(z,\boldsymbol{s}) + \cdots. \tag{2.11}$$

To obtain the matching conditions, we assume that there is a region of overlap where both the expansions are valid. In this region, the solutions have to match. In particular, if we evaluate the outer expansion in the inner coordinates, this must match the limits of the inner solutions away from the interface, that is

$$u(\boldsymbol{X} + \epsilon z\boldsymbol{n};\epsilon) \simeq \hat{u}(z,\boldsymbol{s};\epsilon). \tag{2.12}$$

Insert the expansions into Eqs. (2.5) and (2.11) to get

$$\begin{aligned} u^{(0)}(\boldsymbol{X} + \epsilon z\boldsymbol{n}) + \epsilon u^{(1)}(\boldsymbol{X} + \epsilon z\boldsymbol{n}) + \epsilon^2 u^{(2)}(\boldsymbol{X} + \epsilon z\boldsymbol{n}) + \cdots \\ \simeq \hat{u}^{(0)}(z,\boldsymbol{s}) + \epsilon\hat{u}^{(1)}(z,\boldsymbol{s}) + \epsilon^2\hat{u}^{(2)}(z,\boldsymbol{s}) + \cdots. \end{aligned} \tag{2.13}$$

The terms on the left-hand side can be expanded as a Taylor series,

$$u^{(k)}(\boldsymbol{X} + \epsilon z\boldsymbol{n}) = u^{(k)}(\boldsymbol{s}) + \epsilon z\boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(k)}(\boldsymbol{s}) + \frac{\epsilon^2 z^2}{2}\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{\nabla}u^{(k)}(\boldsymbol{s})\cdot\boldsymbol{n} + \cdots, \tag{2.14}$$

where $k \in \mathbb{N}$ and $u^{(k)}(\boldsymbol{s}) \equiv u^{(k)}(\boldsymbol{X}(\boldsymbol{s};\epsilon))$. Now we end up with the matching equation

$$\begin{aligned} u^{(0)}(\boldsymbol{s}) &+ \epsilon\left(u^{(1)}(\boldsymbol{s}) + z\boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(0)}(\boldsymbol{s})\right) \\ &+ \epsilon^2\left(u^{(2)}(\boldsymbol{s}) + z\boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(1)}(\boldsymbol{s}) + \frac{z^2}{2}\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{\nabla}u^{(0)}(\boldsymbol{s})\cdot\boldsymbol{n}\right) \\ &+ \cdots \simeq \hat{u}^{(0)}(z,\boldsymbol{s}) + \epsilon\hat{u}^{(1)}(z,\boldsymbol{s}) + \epsilon^2\hat{u}^{(2)}(z,\boldsymbol{s}) + \cdots, \end{aligned} \tag{2.15}$$

which must hold when the interface width is decreased, that is $\epsilon \to 0$. In the matching region it is required that $\epsilon z = \mathcal{O}(1)$. If we consider ϵ to be fixed and let $z \to \pm\infty$, we get the following asymptotic matching conditions:

$$\lim_{z\to\pm\infty}\hat{u}^{(0)}(z,\boldsymbol{s}) = u^{(0)\pm}(\boldsymbol{s}), \tag{2.16}$$

and as $z \to \pm\infty$,

$$\hat{u}^{(1)}(z,\boldsymbol{s}) = u^{(1)\pm}(\boldsymbol{s}) + z\boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(0)\pm}(\boldsymbol{s}) + o(1), \tag{2.17}$$

$$\begin{aligned} \hat{u}^{(2)}(z,\boldsymbol{s}) = u^{(2)\pm}(\boldsymbol{s}) &+ z\boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(1)\pm}(\boldsymbol{s}) \\ &+ \frac{z^2}{2}\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{\nabla}u^{(0)\pm}(\boldsymbol{s})\cdot\boldsymbol{n} + o(1). \end{aligned} \tag{2.18}$$

Here $o(1)$ means that the expressions approach equality when $z \to \pm\infty$. That is, $o(1)$ is defined such that if some function $f(z) = o(1)$, then we have $\lim_{z\to\pm\infty} f(z) = 0$.

2.2. Poisson equation with Robin boundary conditions

Now we are ready to consider the Poisson equation with Robin boundary conditions,

$$\begin{aligned} \boldsymbol{\Delta} u &= f && \text{in } D, \\ \boldsymbol{n}\cdot\boldsymbol{\nabla} u &= k(u-g) && \text{on } \partial D. \end{aligned} \tag{2.19}$$

Consider a general DDM approximation,

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla} u)+\frac{1}{\epsilon^2}\psi=\phi f. \tag{2.20}$$

where ψ is a correction term we will *choose* and the scaling factor $1/\epsilon^2$ is taken for later convenience. If we assume that ψ is local to the interface (e.g., vanishes to all orders in ϵ away from ∂D) and that f is independent of ϵ (e.g., extended smoothly in the normal direction out of D), then the outer solution to this equation when $z\to-\infty$ satisfies

$$\boldsymbol{\Delta} u^{(0)} = f, \tag{2.21}$$

$$\boldsymbol{\Delta} u^{(1)} = 0, \tag{2.22}$$

$$\boldsymbol{\Delta} u^{(k)} = 0, \qquad k=2,3,\ldots. \tag{2.23}$$

The goal is to determine ψ such that $u^{(0)}$ satisfies (2.19) and $u^{(1)}=0$ so that the outer expansion $u\approx u^{(0)}+\epsilon^2 u^{(2)}+\ldots$ and therefore the DDM is asymptotically second-order accurate.

2.2.1. Matching conditions

Before we determine ψ, we will develop a higher-order matching condition based on Eqs. (2.17) and (2.18) that will match a Robin boundary condition for $u^{(1)}$. First we take the derivative of Eq. (2.18) with respect to z and subtract k times Eq. (2.17), which gives

$$\hat{u}_z^{(2)}-k\hat{u}^{(1)}=-ku^{(1)}-kz\boldsymbol{n}\cdot\boldsymbol{\nabla} u^{(0)}+\boldsymbol{n}\cdot\boldsymbol{\nabla} u^{(1)}+z\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{\nabla} u^{(0)}\cdot\boldsymbol{n}. \tag{2.24}$$

Keep the terms that make up a Robin condition for $u^{(1)}$ on the left-hand side, and move the rest to the right-hand side, that is

$$\boldsymbol{n}\cdot\boldsymbol{\nabla} u^{(1)}-ku^{(1)}=\hat{u}_z^{(2)}-k\hat{u}^{(1)}+kz\boldsymbol{n}\cdot\boldsymbol{\nabla} u^{(0)}-z\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{\nabla} u^{(0)}\cdot\boldsymbol{n}. \tag{2.25}$$

2.2.2. Inner expansions

Now consider the inner expansion of Eq. (2.20),

$$\frac{1}{\epsilon^2}(\phi\hat{u}_z)_z+\frac{1}{\epsilon}\kappa\phi\hat{u}_z+\phi\boldsymbol{\Delta}_{\mathrm{s}}\hat{u}+\frac{1}{\epsilon^2}\hat{\psi}=\phi\hat{f}. \tag{2.26}$$

Expand $\hat{u}$, $\hat{f}$ and $\hat{\psi}$ in powers of ϵ, to get

$$\begin{aligned} \frac{1}{\epsilon^2}\left(\phi\hat{u}_z^{(0)}\right)_z+\frac{1}{\epsilon}\left(\phi\hat{u}_z^{(1)}\right)_z+\left(\phi\hat{u}_z^{(2)}\right)_z+\frac{1}{\epsilon}\kappa\phi\hat{u}_z^{(0)}+\kappa\phi\hat{u}_z^{(1)} \\ +\phi\boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)}+\frac{1}{\epsilon^2}\hat{\psi}^{(0)}+\frac{1}{\epsilon}\hat{\psi}^{(1)}+\hat{\psi}^{(2)}=\phi\hat{f}^{(0)}+\mathcal{O}(\epsilon). \end{aligned} \tag{2.27}$$

and then collect the leading order terms. The lowest power of ϵ gives

$$\left(\phi\hat{u}_z^{(0)}\right)_z = -\hat{\psi}^{(0)}, \tag{2.28}$$

if we let $\hat{\psi}^{(0)} = 0$ we obtain $\hat{u}_z^{(0)} = 0$. The next order terms then give

$$\left(\phi\hat{u}_z^{(1)}\right)_z = -\hat{\psi}^{(1)}, \tag{2.29}$$

and by use of integration and the partial derivative with respect to z of the matching condition (2.17), we get

$$\boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(0)} = \int_{-\infty}^{\infty}\hat{\psi}^{(1)}\,\mathrm{d}z. \tag{2.30}$$

To get the desired boundary condition for $u^{(0)}$, we need that

$$\int_{-\infty}^{\infty}\hat{\psi}^{(1)}\,\mathrm{d}z = k(u^{(0)} - g), \tag{2.31}$$

which is satisfied if we assume that g is extended constant in the normal direction and if we choose

$$\hat{\psi}^{(1)} = -\phi_z k(\hat{u}^{(0)} - \hat{g}). \tag{2.32}$$

Thus we have that $u^{(0)}$ satisfies

$$\boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(0)} = k(u^{(0)} - g) \tag{2.33}$$

in the limit $z \to -\infty$. This shows that we have achieved first-order asymptotic convergence. We note that there exist other choices, for instance

$$\hat{\psi}^{(1)} = \phi_z^2 k(\hat{u}^{(0)} - \hat{g}). \tag{2.34}$$

It turns out, however, that the first choice (2.32) appears to be necessary in order to obtain the second-order correction. This is because the choice of ψ constrains both $\hat{\psi}^{(1)}$ and $\hat{\psi}^{(2)}$. This will become more apparent later.

Note that if we integrate Eq. (2.29), we get

$$\phi\hat{u}_z^{(1)} = \phi k(\hat{u}^{(0)} - \hat{g}) + C, \tag{2.35}$$

where $C = 0$ follows when we take the limit $z \to \infty$ and use the matching condition (2.17). Thus

$$\hat{u}_z^{(1)} = k(\hat{u}^{(0)} - \hat{g}), \tag{2.36}$$

which will be useful later.

Now consider the zeroth order terms,

$$\left(\phi\hat{u}_z^{(2)}\right)_z = \phi\hat{f} - \kappa\phi\hat{u}_z^{(1)} - \phi\boldsymbol{\Delta}_s\hat{u}^{(0)} - \hat{\psi}^{(2)}. \tag{2.37}$$

We will use that $u^{(0)}$ satisfies Eq. (2.19), that is

$$\begin{aligned} \boldsymbol{\Delta}u^{(0)} &= f && \text{in } D, \\ \boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(0)} &= k(u^{(0)} - g) && \text{on } \partial D. \end{aligned} \tag{2.38}$$

We note that the Laplacian can be decomposed into normal and tangential components as

$$\boldsymbol{\Delta} u = \boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{\nabla} u\cdot\boldsymbol{n} + \kappa\boldsymbol{n}\cdot\boldsymbol{\nabla} u + \boldsymbol{\Delta}_{\mathrm{s}} u, \tag{2.39}$$

which can be shown by writing the gradient vector as $\boldsymbol{\nabla} = \boldsymbol{n}\boldsymbol{n}\cdot\boldsymbol{\nabla} + \boldsymbol{\nabla}_{\mathrm{s}}$. We can therefore write

$$\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{\nabla} u^{(0)}\cdot\boldsymbol{n} = f - \kappa\boldsymbol{n}\cdot\boldsymbol{\nabla} u^{(0)} - \boldsymbol{\Delta}_{\mathrm{s}} u^{(0)} = \hat{f}^{(0)} - \kappa\boldsymbol{n}\cdot\boldsymbol{\nabla}\hat{u}^{(0)} - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)}, \tag{2.40}$$

where the last equality is valid in the matching region when we take the limit $z\to-\infty$. If we insert this into the matching condition (2.25), we get

$$\begin{aligned}\boldsymbol{n}\cdot\boldsymbol{\nabla} u^{(1)} - k u^{(1)} \\ = \hat{u}^{(2)}_z - k\hat{u}^{(1)} - z\left(\hat{f}^{(0)} - \kappa k(\hat{u}^{(0)} - \hat{g}) - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} - k^2(\hat{u}^{(0)} - \hat{g})\right).\end{aligned} \tag{2.41}$$

If we now subtract

$$\left(\phi k\hat{u}^{(1)} + z\phi\left(\hat{f} - \kappa k(\hat{u}^{(0)} - \hat{g}) - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} - k^2(\hat{u}^{(0)} - \hat{g})\right)\right)_z \tag{2.42}$$

from both sides of Eq. (2.37), we get

$$\begin{aligned}\left(\phi\hat{u}^{(2)}_z - \phi k\hat{u}^{(1)} - z\phi\left(\hat{f} - \kappa k\left(\hat{u}^{(0)} - \hat{g}\right) - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} - k^2\left(\hat{u}^{(0)} - \hat{g}\right)\right)\right)_z \\ = \phi\hat{f} - \kappa\phi\hat{u}^{(1)}_z - \phi\boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} - \hat{\psi}^{(2)} - \left(k\phi\hat{u}^{(1)}\right)_z \\ - \left(z\phi\left(\hat{f} - \kappa k\left(\hat{u}^{(0)} - \hat{g}\right) - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} - k^2\left(\hat{u}^{(0)} - \hat{g}\right)\right)\right)_z.\end{aligned} \tag{2.43}$$

Here we recognize that the matching condition (2.41) can be used on the left-hand side if we integrate. To get the desired boundary condition for $u^{(1)}$, we wish to find $\hat{\psi}^{(2)}$ such that the right-hand side of Eq. (2.43) is zero. Some of the terms cancel directly, so we are left with

$$\begin{aligned}\hat{\psi}^{(2)} = -k\phi_z\hat{u}^{(1)} - k\phi\hat{u}^{(1)}_z - \kappa\phi\hat{u}^{(1)}_z + \phi(k+\kappa)k\left(\hat{u}^{(0)} - \hat{g}\right) \\ - z\phi_z\left(\hat{f} - (k+\kappa)k\left(\hat{u}^{(0)} - \hat{g}\right) - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)}\right).\end{aligned} \tag{2.44}$$

Finally insert Eq. (2.36) to get

$$\hat{\psi}^{(2)} = -k\phi_z\hat{u}^{(1)} - z\phi_z\left(\hat{f} - (k+\kappa)k\left(\hat{u}^{(0)} - \hat{g}\right) - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)}\right). \tag{2.45}$$

Now we integrate Eq. (2.43) and use the matching condition (2.41), which yields the boundary condition,

$$\boldsymbol{n}\cdot\boldsymbol{\nabla} u^{(1)} = k u^{(1)}. \tag{2.46}$$

Thus since $u^{(1)}$ satisfies Eqs. (2.22) and (2.46), we conclude that $u^{(1)} = 0$, as desired.

It remains to find ψ in Eq. (2.20) that corresponds to $\hat{\psi}$. It is straightforward to see that

$$\begin{aligned}\frac{1}{\epsilon^2}\hat{\psi} = \frac{1}{\epsilon^2}\hat{\psi}^{(0)} + \frac{1}{\epsilon}\hat{\psi}^{(1)} + \hat{\psi}^{(2)} + \mathcal{O}(\epsilon) \simeq -\frac{1}{\epsilon}\phi_z k\left(\hat{u}^{(0)} - \hat{g}\right) \\ - \phi_z k\hat{u}^{(1)} - z\phi_z\left(\hat{f} - (k+\kappa)k\left(\hat{u}^{(0)} - \hat{g}\right) - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)}\right)\end{aligned} \tag{2.47}$$

corresponds to an inner expansion of

$$\frac{1}{\epsilon^2}\psi = -(\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi)k(u-g) - r\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi(f-(\kappa+k)k(u-g)-\boldsymbol{\Delta}_s u). \tag{2.48}$$

Here we notice the connection between $\hat{\psi}^{(2)}$ and $\hat{\psi}^{(1)}$ in that the first term of $\hat{\psi}^{(2)}$ is the second term of the expansion of the corresponding outer term of $\hat{\psi}^{(1)}$.

To summarize, we have shown that

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla}u) - (\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi)k(u-g) - r\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi(f-(\kappa+k)k(u-g)-\boldsymbol{\Delta}_s u) = \phi f \tag{2.49}$$

gives an asymptotically second-order approximation to the corresponding sharp-interface problem (2.19).

REMARK 2.1. *If we calculate the normal vector as*

$$\boldsymbol{n} = -\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}, \tag{2.50}$$

then

$$\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi = -|\boldsymbol{\nabla}\phi|$$

and Eq. (2.49) becomes

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla}u) + |\boldsymbol{\nabla}\phi|\big(k(u-g) + r(f-(\kappa+k)k(u-g)-\boldsymbol{\Delta}_s u)\big) = \phi f, \tag{2.51}$$

which is a second-order version of an approximation considered in [38].

REMARK 2.2. *If we chose to use the alternative approximation of the boundary condition, we would get the following DDM equation,*

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla}u) + \epsilon|\boldsymbol{\nabla}\phi|^2 k(u-g) - r\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi(f-(\kappa+k)k(u-g)-\boldsymbol{\Delta}_s u) = \phi f. \tag{2.52}$$

The zeroth order terms of the inner expansion would then give

$$\Big(\phi\hat{u}_z^{(2)} - \phi k\hat{u}^{(1)} - z\phi\Big(\hat{f} - \kappa k\left(\hat{u}^{(0)} - \hat{g}\right) - \boldsymbol{\Delta}_s\hat{u}^{(0)} - k^2\left(\hat{u}^{(0)} - \hat{g}\right)\Big)\Big)_z = k\hat{u}^{(1)}\left(\phi_z^2 - \phi_z\right), \tag{2.53}$$

where the integral of the right-hand side is no longer zero. Thus we no longer obtain the desired boundary condition for $u^{(1)}$.

2.3. Reaction-diffusion equation with Neumann boundary conditions

Since the Poisson equation with Neumann boundary conditions does not have a unique solution, we instead consider the steady reaction-diffusion equation with Neumann boundary conditions,

$$\begin{aligned} \boldsymbol{\Delta}u - u &= f \qquad \text{in } D, \\ \boldsymbol{n}\cdot\boldsymbol{\nabla}u &= g \qquad \text{on } \partial D. \end{aligned} \tag{2.54}$$

Again we consider a general DDM approximation,

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla}u) - \phi u + \frac{1}{\epsilon^2}\psi = \phi f. \tag{2.55}$$

Under the same conditions on ψ as in the previous section, the outer solution now satisfies

$$\boldsymbol{\Delta} u^{(0)} - u^{(0)} = f, \tag{2.56}$$

$$\boldsymbol{\Delta} u^{(k)} - u^{(k)} = 0, \qquad k = 1,2,3,\ldots, \tag{2.57}$$

and as before the goal is to find ψ such that $u^{(0)}$ satisfies Eq. (2.54) and $u^{(1)} = 0$.

2.3.1. Matching conditions

To construct the boundary condition for $u^{(1)}$, we consider the derivative of the matching condition (2.18),

$$\hat{u}_z^{(2)}(z,s) = \boldsymbol{n}\cdot\boldsymbol{\nabla} u^{(1)}(s) + z\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{\nabla} u^{(0)}(s)\cdot\boldsymbol{n}. \tag{2.58}$$

2.3.2. Inner expansions

The inner expansion of Eq. (2.55) is

$$\frac{1}{\epsilon^2}(\phi\hat{u}_z)_z + \frac{1}{\epsilon}\kappa\phi\hat{u}_z + \phi\boldsymbol{\Delta}_{\mathrm{s}}\hat{u} - \phi\hat{u} + \frac{1}{\epsilon^2}\hat{\psi} = \phi\hat{f}, \tag{2.59}$$

and based on the previous derivation we now choose directly $\hat{\psi}^{(0)} = 0$ to get $\hat{u}_z^{(0)} = 0$. To get the desired boundary condition for $u^{(0)}$, we need

$$\int_{-\infty}^{\infty} \hat{\psi}^{(1)}\,\mathrm{d}z = g. \tag{2.60}$$

Again there are several choices, and as before we choose

$$\hat{\psi}^{(1)} = -\phi_z\hat{g}, \tag{2.61}$$

so that

$$\phi\hat{u}_z^{(1)} = \phi\hat{g}. \tag{2.62}$$

Finally, the zeroth order terms are

$$\left(\phi\hat{u}_z^{(2)}\right)_z + \phi\kappa\hat{u}_z^{(1)} + \phi\boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} - \phi\hat{u}^{(0)} + \hat{\psi}^{(2)} = \phi\hat{f}^{(0)}. \tag{2.63}$$

In a similar manner as before, we use that $u^{(0)}$ satisfies Eq. (2.54), which gives

$$\boldsymbol{n}\cdot\boldsymbol{\nabla}\boldsymbol{\nabla} u^{(0)}\cdot\boldsymbol{n} = f^{(0)} - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}}u^{(0)} + u^{(0)}. \tag{2.64}$$

If we insert this into the matching condition (2.58), we get

$$\boldsymbol{n}\cdot\boldsymbol{\nabla} u^{(1)} = \hat{u}_z^{(2)} - z\left(\hat{f}^{(0)} - \kappa\hat{g} - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} + \hat{u}^{(0)}\right). \tag{2.65}$$

Now we subtract

$$\left(z\phi\left(\hat{f}^{(0)} - \kappa\hat{g} - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} + \hat{u}^{(0)}\right)\right)_z \tag{2.66}$$

on both sides of Eq. (2.63) to get

$$\begin{aligned}&\left(\phi\hat{u}_z^{(2)} - z\phi\left(\hat{f}^{(0)} - \kappa\hat{g} - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} + \hat{u}^{(0)}\right)\right)_z \\ &\quad = \phi\kappa\left(\hat{g} - \hat{u}_z^{(1)}\right) - z\phi_z\left(\hat{f}^{(0)} - \kappa\hat{g} - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} + \hat{u}^{(0)}\right) - \hat{\psi}^{(2)}.\end{aligned} \tag{2.67}$$

We let

$$\hat{\psi}^{(2)} = -z\phi_z\left(\hat{f}^{(0)} - \kappa\hat{g} - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} + \hat{u}^{(0)}\right), \tag{2.68}$$

so that

$$\left(\phi\hat{u}_z^{(2)} - z\phi\left(\hat{f}^{(0)} - \kappa\hat{g} - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} + \hat{u}^{(0)}\right)\right)_z = \phi\kappa\left(\hat{g} - \hat{u}_z^{(1)}\right) = 0, \tag{2.69}$$

where the last equality follows directly from Eq. (2.62). By integration and use of the matching condition (2.65), we then obtain the desired boundary condition,

$$\boldsymbol{n}\cdot\boldsymbol{\nabla}u^{(1)} = 0. \tag{2.70}$$

The correction is then

$$\frac{1}{\epsilon^2}\hat{\psi} \simeq -\frac{1}{\epsilon}\phi_z\hat{g} - z\phi_z\left(\hat{f}^{(0)} - \kappa\hat{g} - \boldsymbol{\Delta}_{\mathrm{s}}\hat{u}^{(0)} + \hat{u}^{(0)}\right), \tag{2.71}$$

which corresponds to

$$\frac{1}{\epsilon^2}\psi = -(\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi)\,g - r\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi(f - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}}u + u), \tag{2.72}$$

and the second-order DDM equation becomes

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla}u) - \phi u - (\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi)g - r\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi(f - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}}u + u) = \phi f. \tag{2.73}$$

Note that in this case, the choice of the first-order correction $\hat{\psi}^{(1)}$ is relatively independent of the second-order correction $\hat{\psi}^{(2)}$. The only requirement is that the first-order correction must be used to ensure that

$$\int_{-\infty}^{\infty}\phi\kappa\left(\hat{g} - \hat{u}_z^{(1)}\right)\mathrm{d}z = 0. \tag{2.74}$$

For instance, if we instead of Eq. (2.61) choose

$$\hat{\psi}^{(1)} = \phi_z^2\hat{g}, \tag{2.75}$$

then we get

$$\phi\hat{u}_z^{(1)} = \phi^2(3 - 2\phi)\hat{g}. \tag{2.76}$$

Now the integral must be evaluated, and the integrand is in this case

$$\phi\kappa\left(\hat{g} - \hat{u}_z^{(1)}\right) = \phi\kappa(\hat{g} - \phi(3 - 2\phi)\hat{g}) = \kappa\hat{g}\phi(\phi - 1)(2\phi - 1), \tag{2.77}$$

which becomes

$$\begin{aligned}\int_{-\infty}^{\infty}\phi(\phi-1)(2\phi-1)\,\mathrm{d}z &= \int_1^0 \frac{\phi(\phi-1)(2\phi-1)}{\phi_z}\,\mathrm{d}\phi \\ &= \int_1^0 \frac{\phi(\phi-1)(2\phi-1)}{6\phi(1-\phi)}\,\mathrm{d}\phi \\ &= \frac{1}{6}\int_1^0 (2\phi-1)\,\mathrm{d}\phi \\ &= \frac{1}{6}[\phi^2 - \phi]_0^1 \\ &= 0.\end{aligned} \tag{2.78}$$

This leads to an alternative second-order DDM,

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla}u) - \phi u + \epsilon|\boldsymbol{\nabla}\phi|^2 g - r\boldsymbol{n}\cdot\boldsymbol{\nabla}\phi(f - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}}u + u) = \phi f. \tag{2.79}$$

2.4. Summary

Using $\boldsymbol{n} = -\boldsymbol{\nabla}\phi/|\boldsymbol{\nabla}\phi|$, we have shown that the DDM equation

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla}u)+|\boldsymbol{\nabla}\phi|k(u-g)+r|\boldsymbol{\nabla}\phi|\,(f-(\kappa+k)k(u-g)-\boldsymbol{\Delta}_{\mathrm{s}}u)=\phi f \tag{2.80}$$

converges asymptotically with second order to the Poisson equation with Robin boundary conditions,

$$\begin{aligned}\boldsymbol{\Delta}u &= f && \text{in } D,\\ \boldsymbol{n}\cdot\boldsymbol{\nabla}u &= k(u-g) && \text{on } \partial D.\end{aligned} \tag{2.81}$$

We have also shown that the DDM equations

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla}u)-\phi u+|\boldsymbol{\nabla}\phi|g+r|\boldsymbol{\nabla}\phi|\,(f-\kappa g-\boldsymbol{\Delta}_{\mathrm{s}}u+u)=\phi f, \tag{2.82}$$

and

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla}u)-\phi u+\epsilon|\boldsymbol{\nabla}\phi|^2 g+r|\boldsymbol{\nabla}\phi|\,(f-\kappa g-\boldsymbol{\Delta}_{\mathrm{s}}u+u)=\phi f, \tag{2.83}$$

both converge asymptotically with second order to the steady reaction-diffusion equation with Neumann boundary conditions,

$$\begin{aligned}\boldsymbol{\Delta}u-u &= f && \text{in } D,\\ \boldsymbol{n}\cdot\boldsymbol{\nabla}u &= g && \text{on } \partial D.\end{aligned} \tag{2.84}$$

Note that in the higher-order correction formulas, the surface Laplacian has the opposite sign of the 2nd order elliptic operator $\nabla\cdot(\phi\nabla\cdot)$.

3. Discretizations and numerical methods

The equations are discretized with second-order central finite differences. The discrete system is solved using a multigrid method, where a red-black Gauss-Seidel type iterative method is used to relax the solutions (see [59]). The equations are solved in two-dimensions in a domain $\Omega=[-2,2]^2$ for all the test cases. Periodic boundary conditions are used on the domain boundaries $\partial\Omega$.

Since the phase-field function quickly tends to zero outside the physical domain D, it must be regularized in order to prevent the equations from becoming ill posed. We therefore use the modified phase-field function

$$\hat{\phi}=\tau+(1-\tau)\phi, \tag{3.1}$$

where the regularization parameter is set to $\tau=10^{-6}$ unless otherwise specified. In addition, one should note that the chosen boundary condition for the computational domain, Ω, should not interfere with the physical domain. Thus one has to make sure that the distance from the computational wall to the diffuse interface of D is large enough not to affect the results.

As discussed earlier, the normal vector (and the curvature) can be calculated from the phase-field function as

$$\boldsymbol{n}=-\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}, \tag{3.2}$$

and

$$\kappa=-\boldsymbol{\nabla}\cdot\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}. \tag{3.3}$$

The surface Laplacian can be found from the identity

$$\boldsymbol{\Delta}_{\mathrm{s}} \equiv (I-\boldsymbol{nn})\boldsymbol{\nabla}\cdot(I-\boldsymbol{nn})\boldsymbol{\nabla}, \tag{3.4}$$

where

$$(I-\boldsymbol{nn})\boldsymbol{\nabla} \equiv (\delta_{ij}-n_i n_j)\partial x_i. \tag{3.5}$$

In 2D we get

$$\begin{aligned}\boldsymbol{\Delta}_{\mathrm{s}}u = &\left(n_1n_2(n_1n_2)_x + n_1n_2(n_1^2)_y - (1-n_1^2)(n_1^2)_x - (1-n_2^2)(n_1n_2)_y\right)u_x\\ &+\left(n_1n_2(n_1n_2)_y + n_1n_2(n_2^2)_x - (1-n_2^2)(n_2^2)_y - (1-n_1^2)(n_1n_2)_x\right)u_y\\ &+\left(\left(1-n_1^2\right)^2 + n_1^2n_2^2\right)u_{xx}\\ &+\left(\left(1-n_2^2\right)^2 + n_1^2n_2^2\right)u_{yy}\\ &-2n_1n_2\left(\left(1-n_1^2\right)+\left(1-n_2^2\right)\right)u_{xy}.\end{aligned}$$

Below, we verify the accuracy of our numerical implementation on several test problems in which we manufacture a solution to the DDM approximations through particular choices of f. Interestingly, we find that when we include the surface Laplacian, we are unable to solve the discrete system. This is likely due to the fact that it has the opposite sign as the second-order elliptic operator $\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla}\cdot)$ in the bulk region. Thus, even though this term is confined to the interfacial region, it seems to prevent the convergence of our multigrid solver. We still consider the effect of this term, however, by using the surface Laplacian of the analytic solution in the DDM2 equations. The development of stable methods for the full DDM2 system are still under development.

4. Results

We next investigate the performance of the higher-order corrections, and compare them against the corresponding first-order approximations and the exact solution of the sharp-interface equations. We consider four different cases with Neumann boundary conditions and three different cases with Robin boundary conditions. For each case, we calculate and compare the error between the calculated solution u and an analytic solution u_{an} of the original PDE which is extended from D into Ω. The error is defined as

$$E_\epsilon = \frac{\|\phi(u_{\mathrm{an}}-u)\|_2}{\|\phi u_{\mathrm{an}}\|_2}, \tag{4.1}$$

where ϕ is used to restrict the error to the physical domain D. For a given ϵ, the error is calculated by refining the grids until the error has converged. The convergence rate in ϵ as $\epsilon\to 0$ is calculated as

$$k = \log\left(\frac{E_{\epsilon_i}}{E_{\epsilon_{i-1}}}\right)/\log\left(\frac{\epsilon_i}{\epsilon_{i-1}}\right), \tag{4.2}$$

for a decreasing sequence ϵ_i.

In the derivation of the second-order corrections, it was argued that a specific choice of the approximation of the boundary condition for $u^{(0)}$ was more natural. In [38] it was remarked that could be significant differences in accuracy between the different boundary-condition approximations, although no quantitative comparison was

made. It is therefore of interest to investigate the performance of different approximations of the boundary condition. For the cases with Neumann boundary conditions, these are

$$\mathrm{BC1} = |\boldsymbol{\nabla}\phi| g, \tag{4.3}$$

and

$$\mathrm{BC2} = \epsilon |\boldsymbol{\nabla}\phi|^2 g. \tag{4.4}$$

The same comparison will be performed for the cases with Robin boundary conditions. However it should be remarked that BC1 is required for the derivation of the second-order correction for the Robin boundary conditions to be fully valid while BC2 does not guarantee an asymptotically second-order method, as shown in Section 2.2.

4.1. Neumann boundary conditions

Consider the steady reaction-diffusion equation with Neumann boundary conditions,

$$\begin{aligned} \boldsymbol{\Delta} u - u &= f \quad \text{in } D, \\ \boldsymbol{n}\cdot\boldsymbol{\nabla} u &= g \quad \text{on } \partial D. \end{aligned} \tag{4.5}$$

In this section we will solve the first-order and second-order DDM approximations

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla} u) - \phi u + \mathrm{BC} = \phi f, \tag{4.6}$$

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla} u) - \phi u + \mathrm{BC} + r|\boldsymbol{\nabla}\phi|(f - \kappa g - \boldsymbol{\Delta}_{\mathrm{s}} u_{\mathrm{an}} + u) = \phi f, \tag{4.7}$$

denoted as DDM1 and DDM2, respectively. Here BC refers to either BC1 (4.3) or BC2 (4.4), and as remarked above the surface Laplacian term is not solved, rather the surface Laplacian of the analytic solution is used and is treated as a known source term.

4.1.1. Case 1

Consider the case where D is a circle of radius $R=1$ centered at $(0,0)$, and where the analytic solution to the reaction-diffusion equation in D is

$$u_{\mathrm{an}}(x,y) = \frac{1}{4}\left(x^2+y^2\right). \tag{4.8}$$

This corresponds to $f = 1-(x^2+y^2)/4$, $g = R/2$, and $\boldsymbol{\Delta}_{\mathrm{s}} u_{\mathrm{an}} = 0$. In this case, the curvature is $\kappa = 1/R$.

4.1.2. Case 2

Now consider the case where D is the square $D = [-1,1]^2$. Again let the analytic solution in D be

$$u_{\mathrm{an}}(x,y) = \frac{1}{4}\left(x^2+y^2\right), \tag{4.9}$$

so that $f = 1-(x^2+y^2)/4$, $g = 1/2$, and $\boldsymbol{\Delta}_{\mathrm{s}} u_{\mathrm{an}} = 1/2$. In this case the curvature is zero almost everywhere. To see the effect of the surface Laplacian term, this case is also run once where the surface Laplacian term has been set to zero in Eqs. (4.6) and (4.7).

To initialize the square domain D, the signed-distance function is defined as

$$r(x,y)=\begin{cases}|x|-1 & \text{if } |x|\geq|y|,\\ |y|-1 & \text{else.}\end{cases} \tag{4.10}$$

The phase-field function is then calculated directly from the signed-distance function in Eq. (2.4).

4.1.3. Case 3

Again let D be the circle centered at (0,0) with radius $R=1$, but now consider the case where the analytic solution is

$$u_{\text{an}}(x,y)=y\sqrt{x^2+y^2}, \tag{4.11}$$

which corresponds to

$$f=\frac{3y}{\sqrt{x^2+y^2}}-y\sqrt{x^2+y^2}, \tag{4.12}$$

$g=2y$, and

$$\boldsymbol{\Delta}_{\text{s}}u_{\text{an}}=-\frac{y}{\sqrt{x^2+y^2}}. \tag{4.13}$$

Note that in the DDM equations, g is extrapolated constantly in the normal direction off of the boundary ∂D. As in the previous case, we run this case also without the surface Laplacian term.

4.1.4. Case 4

For the final Neumann case we again let $D=[-1,1]^2$, and we consider the case where the analytic solution is

$$u_{\text{an}}(x,y)=e^r, \tag{4.14}$$

where $r=\frac{x^2+y^2}{4}$. This corresponds to

$$f=re^r. \tag{4.15}$$

The boundary function g and the surface Laplacian of the analytic function along the boundary are

$$g=\frac{1}{2}e^{\frac{1+\xi^2}{4}}, \tag{4.16}$$

$$\boldsymbol{\Delta}_{\text{s}}u_{\text{an}}=\frac{1}{4}\left(\xi^2+2\right)e^{\frac{1+\xi^2}{4}}, \tag{4.17}$$

where $\xi\equiv x$ along the bottom and top boundaries, and $\xi\equiv y$ along the left and right boundaries.

4.1.5. Results

Figures 4.1 to 4.4 and Table 4.1 show convergence results where ϵ is reduced for Cases 1 to 4 with DDM1 and DDM2, and with BC1 and BC2. The number of grid cells is $n=8192$ in each direction for all of the results. Although the DDM is most efficient when adaptive meshes are used, here we consider only uniform meshes to

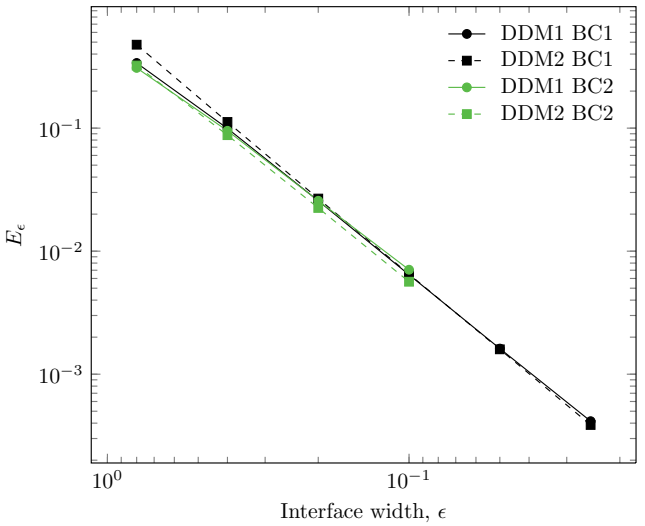


FIGURE 4.1. *Errors for the Neumann problem with respect to ϵ for Case 1, as labelled.*

more easily control the discretization errors in order to focus on the errors in the DDM. When ϵ is small, the required refinement to obtain a converged result becomes large, and in particular for the simulations that used BC2 we were not able to obtain fully converged results for the smallest values of ϵ.

The results indicate that the difference between DDM1 and DDM2 is small, however DDM2 consistently performs better than DDM1. Interestingly, both DDM1 and DDM2 seem to converge with roughly second-order accuracy in ϵ. Case 2 in particular shows a noticeable improvement of DDM2 over DDM1. Case 3 is the first case that has a nonconstant boundary condition, and the surface Laplacian of the analytic solution along the boundary is also nonconstant. An unexpected result for Case 3 is that DDM2 performs better if the surface Laplacian term is removed. One possible explanation to this is errors due to grid anisotropy. Therefore we also consider a fourth case, which again has a nonconstant boundary condition and nonconstant surface Laplacian of the analytic solution. Since the domain in this case is a square, the effect of grid anisotropy is lessened. Correspondingly DDM2 performs better when the surface Laplacian is included.

The difference between BC1 and BC2 is also noticable, especially with regard to the required amount of grid refinement that is needed to obtain a convergent result. Tables A.1 to A.11 in the Appendix show how each result for a given ϵ converges under grid refinement. These tables are used to generate the results in Table 4.1, and they show clearly that the convergence of the equations with respect to mesh size is much faster with BC1 than with BC2.

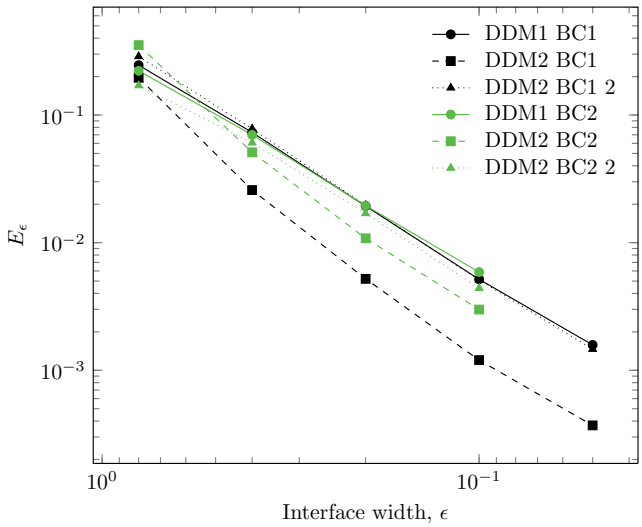


FIGURE 4.2. *Errors for the Neumann problem with respect to ϵ for Case 2, as labelled. Results where the surface-Laplacian term has been removed are also included as DDM2 BC1 2 and DDM2 BC2 2.*

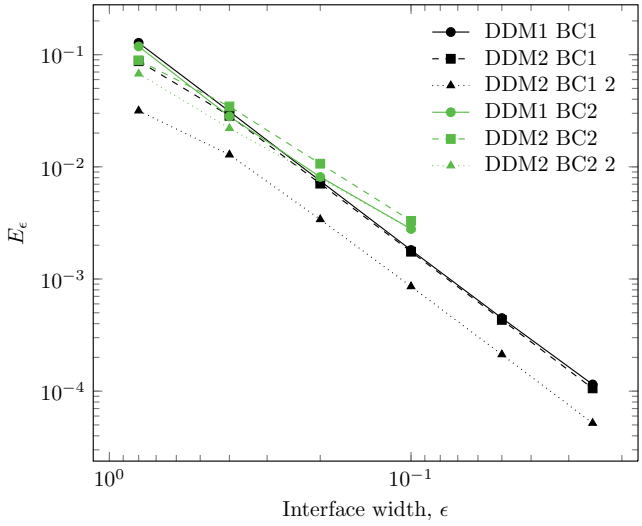


FIGURE 4.3. *Errors for the Neumann problem with respect to ϵ for Case 3, as labelled. Results where the surface-Laplacian term has been removed are also included as DDM2 BC1 2 and DDM2 BC2 2.*

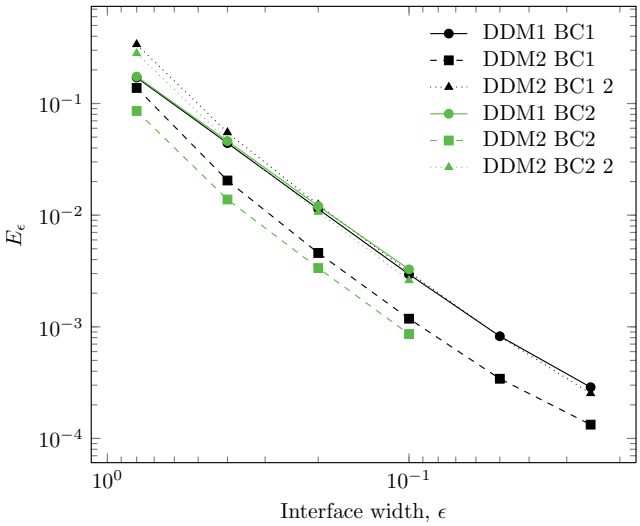


FIGURE 4.4. *Errors for the Neumann problem with respect to ϵ for Case 4, as labelled.*

ϵ	E DDM1 BC1	k	E DDM2 BC1	k	E DDM1 BC2	k	E DDM2 BC2	k
Case 1								
0.800	3.39×10^{-1}		4.77×10^{-1}		3.09×10^{-1}		3.23×10^{-1}	
0.400	9.94×10^{-2}	1.8	1.12×10^{-1}	2.1	9.52×10^{-2}	1.7	8.77×10^{-2}	1.9
0.200	2.57×10^{-2}	2.0	2.68×10^{-2}	2.1	2.57×10^{-2}	1.9	2.25×10^{-2}	2.0
0.100	6.43×10^{-3}	2.0	6.51×10^{-3}	2.0	7.07×10^{-3}	1.9	5.64×10^{-3}	2.0
0.050	1.61×10^{-3}	2.0	1.59×10^{-3}	2.0				
0.025	4.15×10^{-4}	2.0	3.87×10^{-4}	2.0				
Case 2								
0.800	2.46×10^{-1}		1.96×10^{-1}		2.22×10^{-1}		3.53×10^{-1}	
0.400	7.30×10^{-2}	1.8	2.58×10^{-2}	2.9	6.99×10^{-2}	1.7	5.10×10^{-2}	2.8
0.200	1.94×10^{-2}	1.9	5.21×10^{-3}	2.3	1.95×10^{-2}	1.8	1.08×10^{-2}	2.2
0.100	5.16×10^{-3}	1.9	1.20×10^{-3}	2.1	5.88×10^{-3}	1.7	2.99×10^{-3}	1.9
0.050	1.58×10^{-3}	1.7	3.70×10^{-4}	1.7				
Case 2 with no surface Laplacian term								
0.800			2.88×10^{-1}				1.70×10^{-1}	
0.400			7.81×10^{-2}	1.9			6.06×10^{-2}	1.5
0.200			1.96×10^{-2}	2.0			1.70×10^{-2}	1.8
0.100			5.10×10^{-3}	1.9			4.39×10^{-3}	1.9
0.050			1.46×10^{-3}	1.8				
Case 3								
0.800	1.27×10^{-1}		8.74×10^{-2}		1.18×10^{-1}		8.90×10^{-2}	
0.400	3.12×10^{-2}	2.0	2.85×10^{-2}	1.6	2.82×10^{-2}	2.1	3.45×10^{-2}	1.4
0.200	7.48×10^{-3}	2.1	7.08×10^{-3}	2.0	8.13×10^{-3}	1.8	1.07×10^{-3}	1.7
0.100	1.81×10^{-3}	2.0	1.75×10^{-3}	2.0	2.79×10^{-3}	1.5	3.30×10^{-3}	1.7
0.050	4.48×10^{-4}	2.0	4.32×10^{-4}	2.0				
0.025	1.15×10^{-4}	2.0	1.06×10^{-4}	2.0				
Case 3 with no surface Laplacian term								
0.800			3.16×10^{-2}				6.75×10^{-2}	
0.400			1.28×10^{-2}	1.3			2.21×10^{-2}	1.6
0.200			3.40×10^{-3}	1.9			7.92×10^{-3}	1.5
0.100			8.58×10^{-4}	2.0			2.78×10^{-3}	1.5
0.050			2.12×10^{-4}	2.0				
0.025			5.19×10^{-5}	2.0				
Case 4								
0.800	1.71×10^{-1}		1.38×10^{-1}		1.74×10^{-1}		8.58×10^{-2}	
0.400	4.42×10^{-2}	2.0	2.04×10^{-2}	2.8	4.61×10^{-2}	1.9	1.38×10^{-2}	2.6
0.200	1.14×10^{-2}	2.0	4.58×10^{-3}	2.2	1.20×10^{-2}	1.9	3.36×10^{-3}	2.0
0.100	2.95×10^{-3}	1.9	1.18×10^{-3}	2.0	3.27×10^{-3}	1.9	8.62×10^{-4}	2.0
0.050	8.23×10^{-4}	1.8	3.43×10^{-4}	1.8				
0.025	2.88×10^{-4}	1.5	1.33×10^{-4}	1.4				
Case 4 with no surface Laplacian term								
0.800			3.39×10^{-1}				2.81×10^{-1}	
0.400			5.51×10^{-2}	2.6			4.64×10^{-2}	2.6
0.200			1.24×10^{-2}	2.2			1.07×10^{-2}	2.1
0.100			3.09×10^{-3}	2.0			2.60×10^{-3}	2.0
0.050			8.20×10^{-4}	1.9				
0.025			2.53×10^{-4}	1.7				

TABLE 4.1. *The error for the Neumann problem as a function of ϵ for all cases. All results are calculated with $n=8192$ in each direction on uniform grids. Blank results indicate that the solutions require even finer grids to converge.*

4.2. Robin boundary conditions

Now consider the Poisson equation with Robin boundary conditions,

$$\begin{aligned} \boldsymbol{\Delta} u &= f && \text{in } D, \\ \boldsymbol{n}\cdot\boldsymbol{\nabla} u &= k(u-g) && \text{on } \partial D. \end{aligned} \tag{4.18}$$

As in the previous section, we solve the first-order and second-order DDM approximations

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla} u)+BC=\phi f, \tag{4.19}$$

$$\boldsymbol{\nabla}\cdot(\phi\boldsymbol{\nabla} u)+BC+r|\boldsymbol{\nabla}\phi|\,(f-(\kappa+k)k(u-g)-\boldsymbol{\Delta}_{\mathrm{s}}u_{\mathrm{an}})=\phi f, \tag{4.20}$$

respectively, where BC refers to either BC1 or BC2,

$$\mathrm{BC1}=|\boldsymbol{\nabla}\phi|k(u-g), \tag{4.21}$$

$$\mathrm{BC2}=\epsilon|\boldsymbol{\nabla}\phi|^2k(u-g). \tag{4.22}$$

4.2.1. Case 1

Consider the case where D is a circle of radius $R=1$ centered at $(0,0)$, and where the analytic solution to the Poisson equation in D is

$$u_{\mathrm{an}}(x,y)=\frac{1}{4}\left(x^2+y^2\right). \tag{4.23}$$

This corresponds to $f=1-(x^2+y^2)/4$,

$$g=\frac{1}{2}\left(\frac{1}{2}-\frac{1}{k}\right), \tag{4.24}$$

and $\boldsymbol{\Delta}_{\mathrm{s}}u_{\mathrm{an}}=0$. We will consider the case when $k=-1$, thus $g=3/4$.

4.2.2. Case 2

Again let D be the circle at $(0,0)$ with radius $R=1$, but now consider the case where the analytic solution is

$$u_{\mathrm{an}}(x,y)=y\sqrt{x^2+y^2}, \tag{4.25}$$

which corresponds to

$$f=\frac{3y}{\sqrt{x^2+y^2}}, \tag{4.26}$$

$$g=y\left(1-\frac{2}{k}\right), \tag{4.27}$$

and

$$\boldsymbol{\Delta}_{\mathrm{s}}u_{\mathrm{an}}=-\frac{y}{\sqrt{x^2+y^2}}. \tag{4.28}$$

Again let $k=-1$ so that $g=3y$. Similar to the Neumann case 3, g is extended constantly in the normal direction in the DDM equations.

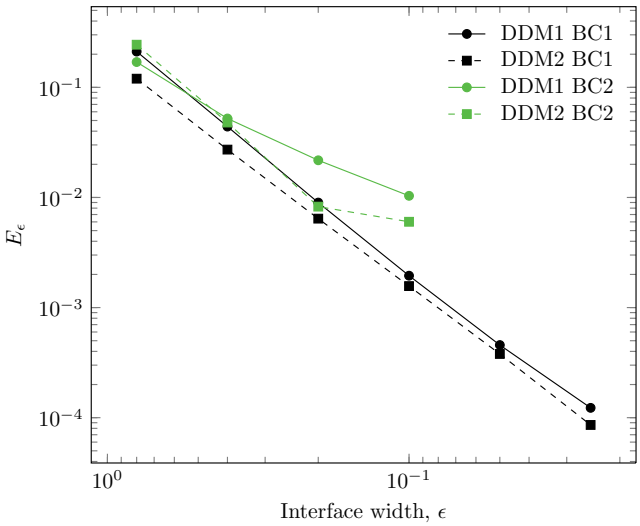


FIGURE 4.5. *Errors for the Robin problem with respect to ϵ for Case 1, as labelled.*

4.2.3. Case 3

For the final Robin case we let $D=[-1,1]^2$, and we consider a case that corresponds to the Neumann Case 4 where the analytic solution is

$$u_{\text{an}}(x,y)=e^r, \tag{4.29}$$

where $r=\frac{x^2+y^2}{4}$. This corresponds to

$$f=(r+1)e^r. \tag{4.30}$$

The boundary function g and the surface Laplacian of the analytic function along the boundary are

$$g=\frac{3}{2}e^{\frac{1+\xi^2}{4}}, \tag{4.31}$$

$$\boldsymbol{\Delta}_{\text{s}}u_{\text{an}}=\frac{1}{4}\left(\xi^2+2\right)e^{\frac{1+\xi^2}{4}}, \tag{4.32}$$

where $\xi\equiv x$ along the bottom and top boundaries, and $\xi\equiv y$ along the left and right boundaries.

4.2.4. Results

The convergence results are now presented in Figures 4.5 to 4.7 and Table 4.2. The grid convergence for each ϵ is presented in Tables A.12 to A.15 in the Appendix. Again the results indicate that DDM2 performs better than DDM1, although both methods are roughly second-order accurate, and that BC1 gives better results than BC2.

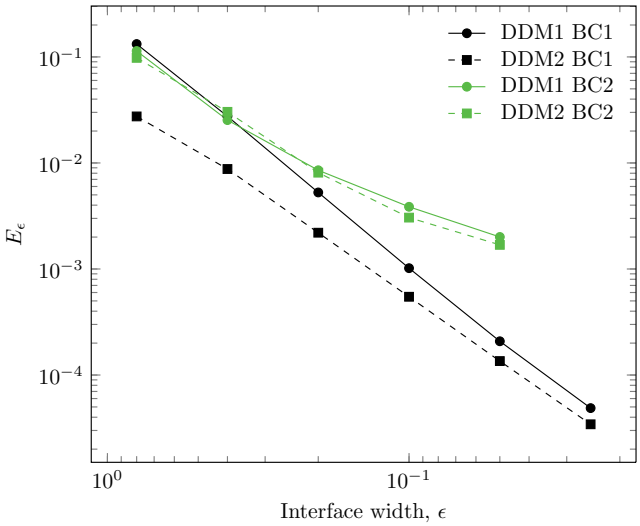


FIGURE 4.6. *Errors for the Robin problem with respect to ϵ for Case 2, as labelled.*

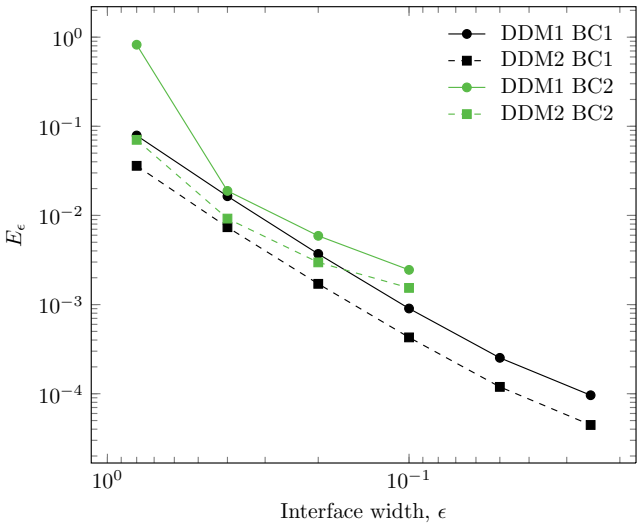


FIGURE 4.7. *Errors for the Robin problem with respect to ϵ for Case 3, as labelled.*

ϵ	E DDM1 BC1	k	E DDM2 BC1	k	E DDM1 BC2	k	E DDM2 BC2	k
Case 1								
0.800	2.11×10^{-1}		1.20×10^{-1}		1.70×10^{-1}		2.44×10^{-1}	
0.400	4.40×10^{-2}	2.3	2.72×10^{-2}	2.1	5.21×10^{-2}	1.7	4.79×10^{-2}	2.3
0.200	8.99×10^{-3}	2.3	6.42×10^{-3}	2.1	2.18×10^{-2}	1.3	8.26×10^{-3}	2.5
0.100	1.95×10^{-3}	2.2	1.57×10^{-3}	2.0	1.03×10^{-2}	1.1	6.01×10^{-3}	0.5
0.050	4.57×10^{-4}	2.1	3.79×10^{-4}	2.0				
0.025	1.23×10^{-4}	1.9	8.59×10^{-5}	2.1				
Case 2								
0.800	1.32×10^{-1}		2.75×10^{-2}		1.14×10^{-1}		9.80×10^{-2}	
0.400	2.75×10^{-2}	2.3	8.77×10^{-3}	1.6	2.54×10^{-2}	2.2	3.03×10^{-2}	1.7
0.200	5.27×10^{-3}	2.4	2.20×10^{-3}	2.0	8.54×10^{-3}	1.6	8.10×10^{-3}	1.9
0.100	1.02×10^{-3}	2.4	5.47×10^{-4}	2.0	3.86×10^{-3}	1.1	3.05×10^{-3}	1.4
0.050	2.08×10^{-4}	2.3	1.35×10^{-4}	2.0	2.01×10^{-3}	0.9	1.69×10^{-3}	0.9
0.025	4.88×10^{-5}	2.1	3.43×10^{-5}	2.0				
Case 3								
0.800	7.89×10^{-2}		3.61×10^{-2}		8.23×10^{-2}		7.05×10^{-2}	
0.400	1.64×10^{-2}	2.3	7.42×10^{-3}	2.3	1.89×10^{-2}	2.2	9.22×10^{-3}	2.5
0.200	3.70×10^{-3}	2.2	1.74×10^{-3}	2.1	5.90×10^{-3}	1.7	2.97×10^{-3}	1.6
0.100	9.04×10^{-4}	2.0	4.52×10^{-4}	2.0	2.45×10^{-3}	1.3	1.54×10^{-3}	1.0
0.050	2.53×10^{-4}	1.8	1.44×10^{-4}	1.8				
0.025	9.61×10^{-5}	1.4	7.16×10^{-5}	1.4				

TABLE 4.2. *The error for the Robin problem as a function of ϵ for all cases. All results are calculated with $n=8192$ in each direction on uniform grids, except for Case 3 with BC2, where the results are calculated with $n=4096$ in each direction. Blank results indicate that the solutions require even finer grids to converge.*

5. Conclusion

We have derived an asymptotically second-order diffuse domain method (DDM) for the Poisson equation with Robin boundary conditions and for the steady diffusion-equation with Neumann boundary conditions. The second-order DDM (DDM2) was tested for selected test cases and compared to the first-order DDM (DDM1) with two different approximations of the boundary condition, BC1, see Eqs. (4.3) and (4.21), and BC2, see Eqs. (4.4) and (4.22). Due to a problem with solving the full DDM2 equations, we instead solved the DDM2 equations using the surface Laplacian of the analytic solution in Eqs. (2.80), (2.82) and (2.83).

The results indicate that the global accuracy of DDM2 is better than that of DDM1. However, both methods generally perform well, and the global convergence rate is around 2. It was shown that the choice of boundary condition had a significant impact on the accuracy: BC1 generally performed much better than the alternative boundary condition BC2. Using DDM2 with BC1, we obtain a reliably second-order accurate method.

In future work, we plan to perform an explicit error analysis for the first and second order DDM-based schemes in order to better understand the convergence behaviour observed here. Further, this analysis should be also reveal why we have difficulties incorporating the surface Laplacian in the DDM equations and whether we can find an alternative numerical approach for handling this term.

Acknowledgement. KYL acknowledges support from the Fulbright foundation for a Visiting Researcher Grant to fund a stay at the University of California, Irvine. KYL also acknowledges support from Statoil and GDF SUEZ, and the Research Council of Norway (193062/S60) for the research project Enabling low emission LNG systems. JL acknowledges support from the National Science Foundation, Division of Mathematical Sciences, and the National Institute of Health through grant P50GM76516 for a Center of Excellence in Systems Biology at the University of California, Irvine. The authors gratefully thank Bernhard Müller (NTNU) and Svend Tollak Munkejord (SINTEF Energy Research) for helpful discussions and for feedback on the manuscript.

REFERENCES

[1] S. ALAND, J. LOWENGRUB, AND A. VOIGT, *Two-phase flow in complex geometries: A diffuse domain approach.*, Computer Modeling in Engineering & Sciences, 57 (2010), pp. 77–106.

[2] R. F. ALMGREN, *Second-order phase field asymptotics for unequal conductivities*, SIAM Journal on Applied Mathematics, 59 (1999), pp. 2086–2107.

[3] M.K. NERNAUER, R. HERZOG, *Implementation of an X-FEM solver for the classical two-phase Stefan problem*, J. Sci. Comput., 52 (2012), pp. 271-293.

[4] A. BUENO-OROVIO AND V. M. PEREZ-GARCIA, *Spectral smoothed boundary methods: the role of external boundary conditions*, Numer. Meth. Partial Diff. Eqns., 22 (2006), pp. 435–448.

[5] A. BUENO-OROVIO, V. M. PEREZ-GARCIA, AND F. H. FENTON, *Spectral methods for partial differential equations in irregular domains: the spectral smoothed boundary method*, SIAM Journal on Scientific Computing, 28 (2006), pp. 886–900.

[6] A. BYFUT, A. SCHROEDER, *hp-adaptive extended finite element method*, Int. J. Numer. Meth. Eng., 89 (2012), pp. 1293-1418.

[7] M. CISTERNINO, L WEYNANS, *A parallel second order Cartesian method for elliptic interface problems*, Comm. Comput. Phys., 12 (2012), pp. 1562-1587.

[8] H. JOHANSEN AND P. COLELLA, *Embedded boundary algorithms and software for partial differential equations*, J. Phys., 125 (2008), pp. 012084.

[9] A. DEMLOW AND G. DZIUK, *An adaptive finite element method for the Laplace-Beltrami operator on implicitly defined surfaces* , SIAM J. Numer. Anal., 45 (2007), pp. 421-442.

[10] J. Dolbow, I. Harari, *An efficient finite element method for embedded interface problems*, Int. J. Numer. Meth. Eng., 78 (2009), pp. 229-252.
[11] R. Duddu, D.L. Chopp, P. Voorhees, B. Moran, *Diffusional evolution of precipitates in elastic media using the extended finite element method and level set methods*, J. Comput. Phys., 230 (2011), pp. 1249-1264.
[12] G. Dziuk, and C.M. Elliott, *Finite elements on evolving surfaces*, IMA J. Numer. Anal., 27 (2007), pp. 262-292.
[13] G. Dziuk and C.M. Elliott, *Eulerian finite element method for parabolic PDEs on implicit surfaces*, Int. Free. Bound., 10 (2008), pp. 119-138.
[14] G. Dziuk and C.M. Elliott, *An Eulerian approach to transport and diffusion on evolving implicit surfaces*, Comput. Visual. Sci., 13, (2010), pp. 17-28.
[15] G. Dziuk and C.M. Elliott, *A fully discrete evolving surface finite element method*, SIAM J. Numer. Anal., 50, 5, (2012), pp. 2677-2694.
[16] C.M. Elliott, B. Stinner, V. Styles, R. Welford, *Numerical computation of advection and diffusion on evolving diffuse interfaces*, IMA J. Num. Anal., 31, (2011), pp. 245-269.
[17] C.M. Elliott and B. Stinner, *Analysis of a diffuse interface approach to an advection diffusion equation on a moving surface*, Math. Mod. Meth. Appl. Sci., (2009) in press.
[18] A.S. Fard, M.A. Hulsen, P.D. Anderson, *Extended finite element method for viscous flow inside complex three-dimensional geometries with moving boundaries*, Int. J. Numer. Meth. Fluids, 70 (2012), pp. 775-792.
[19] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, *A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method*, J. Comput. Phys., 152 (1999), pp. 457-492.
[20] F. H. Fenton, E. M. Cherry, A. Karma, and W. J. Rappel, *Modeling wave propagation in realistic heart geometries using the phase-field method*, CHAOS, 15 (2005).
[21] R. Folch, J. Casademunt, A. Hernandez-Machado, and L. Ramirez-Piscina, Phys. Rev. E, 60 (1999), pp. 1724.
[22] F.-P. Fries, T. Belytschko, *The extended/generalized finite element method: An overview of the method and its applications*, Int. J. Numer. Meth. Eng., 84 (2010), pp. 253-304.
[23] F. Gibou, R. Fedkiw, L.T. Cheng, and M. Kang, *A second order accurate symetric discretization of the Poisson equation on irregular domains*, J. Comput. Phys., 176 (2002), pp. 205-227.
[24] F. Gibou and R. Fedkiw, *A fourth order accurate discretization for the Laplace and heat equations on arbitrary domains with applications to the Stefan problem*, J. Comput. Phys., 202 (2005), pp. 577-601.
[25] J. Glimm and D. Marchesin and O. McBryan, *A numerical method for 2 phase flow with an unstable interface*, J. Comput. Phys., 39 (1981), pp. 179-200.
[26] R. Glowinski, T.W. Pan, and J. Periaux, *A fictitious domain method for external incompressible viscous-flow modeled by Navier-Stokes equations*, Comput. Meth. Appl. Mech. Engin., 112 (1994), pp. 133-148.
[27] R. Glowinski and T.W. Pan and R.O. Wells and X.D. Zhou, *Wavelet and finite element solutions for the Neumann problem using fictitious domains*, J. Comput. Phys., 126 (1996), pp. 40-51.
[28] J.B. Greer and A.L. Bertozzi and G. Sapiro, *Fourth order partial differential equations on general geometries*, J. Comput. Phys., 216 (2006), pp. 216-246.
[29] S. Gross, and A. Reusken, *An extended pressure finite element space for two-phase incompressible flows*, J. Comput. Phys., 224 (2007), pp. 40-48.
[30] X.M. He, T. Lin, Y.P. Lin, *Immersed finite element methods for elliptic interface problems with non-homogeneous jump conditions*, Int. J. Numer. Anal. Model., 8 (2011), pp. 284-301.
[31] R. Lohner and J.R. Cebral and F.F. Camelli and J.D. Baum and E.L. Mestreau and O.A. Soto, *Adaptive embedded/immersed unstructured grid techniques*, Arch. Comput. Meth. Eng., 14 (2007), pp. 279-301.
[32] H. Ji and F.-S. Lien and E. Yee, *An efficient second-order accurate cut-cell method for solving the variable coefficient Poisson equation with jump conditions on irregular domains*, Int. J. Numer. Meth. Fluids, 52 (2006), pp. 723-748.
[33] H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys., 147 (1998), pp. 60-85.
[34] A. Karma and W.-J. Rappel, *Quantitative phase-field modeling of dendritic growth in two and three dimensions*, Physical Review E, 57 (1998), pp. 4323–4349.
[35] J. Kockelkoren, H. Levine, and W. J. Rappel, *Computational approach for modeling intra- and extracellular dynamics*, Phys. Rev., E 68 (2003), p. 037702.
[36] H. Levine and W. J. Rappel, *Membrane-bound turing patterns*, Physical Review E, 72 (2005).

[37] R.J. LEVEQUE AND Z. LI, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal., 31 (1994), pp. 1019-1044.
[38] X. LI, J. LOWENGRUB, A. RÄTZ, AND A. VOIGT, *Solving pdes in complex geometries: A diffuse domain approach*, Communications in Mathematical Sciences, 7 (2009), pp. 81–107.
[39] Z. LI AND K. ITO, *The immersed interface method: Numerical solutions of PDEs involving interfaces and irregular domains*, SIAM Front. Appl. Math., 33 (2006).
[40] Z. LI, P. SONG, *An adaptive mesh refinement strategy for immersed boundary/interface methods*, Comm. Comput. Phys., 12 (2012), pp. 515-527.
[41] S.H. LUI, *Spectral domain embedding for elliptic PDEs in complex domains*, J. Comput. Appl. Math., 225 (2009), pp. 541-557.
[42] P. MACKLIN AND J. LOWENGRUB, *Evolving interfaces via gradients of geometry-dependent interior Poisson problems: Application to tumor growth*, J. Comput. Phys., 203 (2005), pp. 191-220.
[43] P. MACKLIN AND J. LOWENGRUB, *A new ghost cell/level set method for moving boundary problems: Application to tumor growth*, J. Sci. Comput., 35 (2008), pp. 266-299.
[44] M. OEVERMANN, C. SCHARFENBERG, AND R. KLEIN, *A sharp interface finite volume method for elliptic equations on Cartesian grids*, J. Comput. Phys., 228 (2009), pp. 5184-5206.
[45] S. OSHER AND R. FEDKIW, *Level set methods and dynamic implicit surfaces*, Springer (2003).
[46] S. OSHER, J.A. SETHIAN, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12-49.
[47] R. L. PEGO, *Front migration in the nonlinear Cahn-Hilliard equation*, Proceedings of the Royal Society A, 422 (1988), pp. 261–278.
[48] T. PREUSSER, M. RUMPF, S. SAUTER, AND L.O. SCHWEN, *3D composite finite elements for elliptic boundary value problems with discontinuous coefficients*, SIAM J. Sci. Comput., 35 (2011), pp. 2115-2143.
[49] I. RAMIERE AND P. ANGOT AND M. BELLIARD, *A general fictitious domain method with immersed jumps and multilevel nested structured meshes*, J. Comput. Phys., 225 (2007), pp. 1347-1387.
[50] A. RÄTZ AND A. VOIGT, *PDEs on surfaces—a diffuse interface approach*, Commun. Math. Sci., 4 (2006), pp. 575-590.
[51] M. G. REUTER, J. C. HILL, AND R. J. HARRISON, *Solving pdes in irregular geometries with multiresolution methods i: Embedded Dirichlet boundary conditions*, Computer Physics Communications, 183 (2012), pp. 1–7.
[52] J.A. SETHIAN, *Level set methods and fast marching methods*, Cambridge University Press (1999), ISBN 0-521-64557-3.
[53] J.A. SETHIAN AND Y. SHAN, *Solving partial differential equations on irregular domains with moving interfaces, with applications to superconformal electrodeposition in semiconductor manufacturing*, J. Comput. Phys., 227 (2008), pp. 6411-6447.
[54] I. SINGER-LOGINOVA AND H.M. SINGER, *The phase field technique for modeling multiphase materials*, Rep. Prog. Phys., 71 (2008), pp. 106501.
[55] K. E. TEIGEN, X. LI, J. LOWENGRUB, F. WANG, AND A. VOIGT, *A diffuse-interface approach for modeling transport, diffusion and adsorption/desorption of material quantities on a deformable interface*, Communications in Mathematical Sciences, 7 (2009), pp. 1009–1037.
[56] K.E. TEIGEN, P. SONG, A. VOIGT, AND J. LOWENGRUB, *A diffuse-interface method for two-phase flows with soluble surfactants*, J. Comput. Phys., 230 (2011), pp. 375-393.
[57] E. UZGOREN, J. SIM, AND W. SHYY, *Marker-based, 3-D adaptive Cartesian grid method for multiphase flows around irregular*, Comm. Comput. Phys., 5 (2009), pp. 1-41.
[58] X.H. WAN, Z. LI, *Some new finite difference methods for Helmholtz equations on irregular domains or with interfaces*, Disc. Cont. Dyn. Sys. B, 17 (2012), pp. 1155-1175.
[59] S. WISE, J. KIM, AND J. LOWENGRUB, *Solving the regularized, strongly anisotropic Cahn-Hilliard equation by an adaptive nonlinear multigrid method*, Journal of Computational Physics, 226 (2007), pp. 414–446.
[60] K. XIA, M. ZHAN, G. WEI, *MIB method for elliptic equations with multimaterial interfaces*, J. Comput. Phys., 230 (2011), pp. 4588-4615.
[61] S. ZHAO, *High order matched interface and boundary methods for the Helmholtz equation in media with arbitrarily curved interfaces*, J. Comput. Phys., 229 (2010), pp. 3155-3170.
[62] S. ZHAO, G. WEI, *Matched interface and boundary (MIB) for the implementation of boundary conditions in high-order central finite differences*, Int. J. Numer. Meth. Eng., 77 (2009), pp. 1690-1730.
[63] X.L. ZHONG, *A new high-order immersed interface method for solving elliptic equations with imbedded interface of discontinuity*, J. Comput. Phys., 225 (2007), pp. 1066-1099.
[64] Y.C. ZHOU, J. LIU, D.L. HARRY, *A matched interface and boundary method for solving mul-*

tiflow Navier-Stokes equations with applications to geodynamics, J. Comput. Phys., 231 (2012), pp. 223-242.

[65] Y.C. Zhou, S. Zhao, M. Feig, and G.W. Wei, *High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources*

Appendix A. Convergence tables with respect to grid refinement.

This appendix gives a list of tables that shows convergence with respect to grid refinement for each test case and for each ϵ. The tables are used to obtain the results that are presented in Section 4.

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	3.54×10^{-1}		5.16×10^{-1}		1.08×10^{-1}		2.00×10^{-1}	
64	3.43×10^{-1}	0.0	4.83×10^{-1}	0.1	1.03×10^{-1}	0.1	1.26×10^{-1}	0.7
128	3.40×10^{-1}	0.0	4.77×10^{-1}	0.0	1.00×10^{-1}	0.0	1.15×10^{-1}	0.1
256	3.39×10^{-1}	0.0	4.76×10^{-1}	0.0	9.96×10^{-2}	0.0	1.13×10^{-1}	0.0
512	3.39×10^{-1}	0.0	4.76×10^{-1}	0.0	9.95×10^{-2}	0.0	1.12×10^{-1}	0.0
1024	3.39×10^{-1}	0.0	4.76×10^{-1}	-0.0	9.94×10^{-2}	0.0	1.12×10^{-1}	0.0
2048	3.39×10^{-1}	0.0	4.76×10^{-1}	-0.0	9.94×10^{-2}	0.0	1.12×10^{-1}	0.0
4096	3.39×10^{-1}	0.0	4.77×10^{-1}	-0.0	9.94×10^{-2}	0.0	1.12×10^{-1}	0.0
8192	3.39×10^{-1}	0.0	4.77×10^{-1}	-0.0	9.94×10^{-2}	0.0	1.12×10^{-1}	0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	3.38×10^{-3}		3.57×10^{-1}		1.03×10^{-1}		8.54×10^{-1}	
64	2.37×10^{-2}	-2.8	8.09×10^{-2}	2.1	3.16×10^{-2}	1.7	2.92×10^{-1}	1.5
128	2.61×10^{-2}	-0.1	3.32×10^{-2}	1.3	1.88×10^{-3}	4.1	5.14×10^{-2}	2.5
256	2.58×10^{-2}	0.0	2.77×10^{-2}	0.3	6.21×10^{-3}	-1.7	1.10×10^{-2}	2.2
512	2.57×10^{-2}	0.0	2.69×10^{-2}	0.0	6.46×10^{-3}	-0.1	7.00×10^{-3}	0.6
1024	2.57×10^{-2}	0.0	2.68×10^{-2}	0.0	6.44×10^{-3}	0.0	6.59×10^{-3}	0.1
2048	2.57×10^{-2}	0.0	2.68×10^{-2}	0.0	6.43×10^{-3}	0.0	6.52×10^{-3}	0.0
4096	2.57×10^{-2}	0.0	2.68×10^{-2}	0.0	6.43×10^{-3}	0.0	6.51×10^{-3}	0.0
8192	2.57×10^{-2}	0.0	2.68×10^{-2}	0.0	6.43×10^{-3}	0.0	6.51×10^{-3}	0.0
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	1.62×10^{-1}		1.31		1.80×10^{-1}		1.44	
64	1.27×10^{-1}	0.4	8.10×10^{-1}	0.7	2.09×10^{-1}	-0.2	1.14	0.3
128	3.83×10^{-2}	1.7	2.77×10^{-1}	1.5	1.31×10^{-1}	0.7	7.53×10^{-1}	0.6
256	3.55×10^{-3}	3.4	4.34×10^{-2}	2.7	3.99×10^{-2}	1.7	2.70×10^{-1}	1.5
512	1.24×10^{-3}	1.5	5.39×10^{-3}	3.0	4.87×10^{-3}	3.0	4.11×10^{-2}	2.7
1024	1.60×10^{-3}	-0.4	1.93×10^{-3}	1.5	1.67×10^{-5}	8.2	3.94×10^{-3}	3.4
2048	1.62×10^{-3}	-0.0	1.63×10^{-3}	0.2	3.90×10^{-4}	-4.5	6.60×10^{-4}	2.6
4096	1.62×10^{-3}	0.0	1.60×10^{-3}	0.0	4.14×10^{-4}	-0.1	4.10×10^{-4}	0.7
8192	1.61×10^{-3}	0.0	1.59×10^{-3}	0.0	4.15×10^{-4}	-0.0	3.87×10^{-4}	0.1

TABLE A.1. *Grid convergence for Neumann Case 1 with BC1* (4.3).

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	5.94×10^{-1}		4.55×10^{-2}		1.11		8.99×10^{-1}	
64	3.83×10^{-1}	0.6	2.42×10^{-1}	-2.4	3.96×10^{-1}	1.5	2.17×10^{-1}	2.1
128	3.27×10^{-1}	0.2	3.02×10^{-1}	-0.3	1.74×10^{-1}	1.2	2.01×10^{-2}	3.4
256	3.13×10^{-1}	0.1	3.17×10^{-1}	-0.1	1.15×10^{-1}	0.6	6.80×10^{-2}	-1.8
512	3.10×10^{-1}	0.0	3.21×10^{-1}	-0.0	1.00×10^{-1}	0.2	8.28×10^{-2}	-0.3
1024	3.09×10^{-1}	0.0	3.22×10^{-1}	-0.0	9.64×10^{-2}	0.1	8.65×10^{-2}	-0.1
2048	3.09×10^{-1}	0.0	3.23×10^{-1}	-0.0	9.54×10^{-2}	0.0	8.74×10^{-2}	-0.0
4096	3.09×10^{-1}	0.0	3.23×10^{-1}	-0.0	9.52×10^{-2}	0.0	8.76×10^{-2}	-0.0
8192	3.09×10^{-1}	0.0	3.23×10^{-1}	-0.0	9.52×10^{-2}	0.0	8.77×10^{-2}	-0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	2.50		2.21		4.14		3.48	
64	1.03	1.3	9.43×10^{-1}	1.2	2.44	0.8	2.19	0.7
128	3.23×10^{-1}	1.7	2.74×10^{-1}	1.8	9.90×10^{-1}	1.3	9.42×10^{-1}	1.2
256	1.03×10^{-1}	1.6	5.74×10^{-2}	2.3	2.99×10^{-1}	1.7	2.84×10^{-1}	1.7
512	4.50×10^{-2}	1.2	7.67×10^{-3}	2.9	8.32×10^{-2}	1.8	7.15×10^{-2}	2.0
1024	3.04×10^{-2}	0.6	1.79×10^{-2}	-1.2	2.59×10^{-2}	1.7	1.45×10^{-2}	2.3
2048	2.68×10^{-2}	0.2	2.14×10^{-2}	-0.3	1.15×10^{-2}	1.2	2.77×10^{-3}	2.4
4096	2.59×10^{-2}	0.0	2.22×10^{-2}	-0.1	7.93×10^{-3}	0.5	4.87×10^{-3}	-0.8
8192	2.57×10^{-2}	0.0	2.25×10^{-2}	-0.0	7.07×10^{-3}	0.2	5.64×10^{-3}	-0.2
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	5.33		4.35		5.96		4.98	
64	4.08	0.4	3.42	0.3	5.33	0.2	4.45	0.2
128	2.42	0.8	2.19	0.6	4.12	0.4	3.51	0.3
256	9.74×10^{-1}	1.3	9.35×10^{-1}	1.2	2.41	0.8	2.18	0.7
512	2.91×10^{-1}	1.7	2.85×10^{-1}	1.7	9.67×10^{-1}	1.3	9.31×10^{-1}	1.2
1024	7.76×10^{-2}	1.9	7.45×10^{-2}	1.9	2.88×10^{-1}	1.7	2.84×10^{-1}	1.7
2048	2.09×10^{-2}	1.9	1.80×10^{-2}	2.1	7.59×10^{-2}	1.9	7.50×10^{-2}	1.9
4096	6.49×10^{-3}	1.7	3.68×10^{-3}	2.3	1.95×10^{-2}	2.0	1.88×10^{-2}	2.0
8192	2.96×10^{-3}	1.1	9.87×10^{-4}	1.9	5.22×10^{-3}	1.9	4.51×10^{-3}	2.1

TABLE A.2. *Grid convergence for Neumann Case 1 with BC2* (4.21).

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	3.37×10^{-1}		1.18×10^{-1}		1.75×10^{-1}		7.33×10^{-2}	
64	2.89×10^{-1}	0.2	1.50×10^{-1}	-0.4	1.22×10^{-1}	0.5	3.35×10^{-2}	1.1
128	2.67×10^{-1}	0.1	1.70×10^{-1}	-0.2	9.69×10^{-2}	0.3	2.18×10^{-2}	0.6
256	2.56×10^{-1}	0.1	1.83×10^{-1}	-0.1	8.46×10^{-2}	0.2	2.17×10^{-2}	0.0
512	2.51×10^{-1}	0.0	1.90×10^{-1}	-0.1	7.86×10^{-2}	0.1	2.34×10^{-2}	-0.1
1024	2.48×10^{-1}	0.0	1.93×10^{-1}	-0.0	7.56×10^{-2}	0.1	2.46×10^{-2}	-0.1
2048	2.47×10^{-1}	0.0	1.95×10^{-1}	-0.0	7.41×10^{-2}	0.0	2.53×10^{-2}	-0.0
4096	2.46×10^{-1}	0.0	1.96×10^{-1}	-0.0	7.34×10^{-2}	0.0	2.56×10^{-2}	-0.0
8192	2.46×10^{-1}	0.0	1.96×10^{-1}	-0.0	7.30×10^{-2}	0.0	2.58×10^{-2}	-0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	1.20×10^{-1}		7.59×10^{-2}		9.97×10^{-2}		7.78×10^{-2}	
64	6.93×10^{-2}	0.8	3.57×10^{-2}	1.1	5.41×10^{-2}	0.9	3.60×10^{-2}	1.1
128	4.37×10^{-2}	0.7	1.72×10^{-2}	1.1	2.94×10^{-2}	0.9	1.75×10^{-2}	1.0
256	3.12×10^{-2}	0.5	8.82×10^{-3}	1.0	1.70×10^{-2}	0.8	8.62×10^{-3}	1.0
512	2.51×10^{-2}	0.3	5.67×10^{-3}	0.6	1.09×10^{-2}	0.6	4.29×10^{-3}	1.0
1024	2.20×10^{-2}	0.2	4.99×10^{-3}	0.2	7.81×10^{-3}	0.5	2.27×10^{-3}	0.9
2048	2.05×10^{-2}	0.1	5.02×10^{-3}	-0.0	6.29×10^{-3}	0.3	1.47×10^{-3}	0.6
4096	1.97×10^{-2}	0.1	5.13×10^{-3}	-0.0	5.53×10^{-3}	0.2	1.24×10^{-3}	0.2
8192	1.94×10^{-2}	0.0	5.21×10^{-3}	-0.0	5.16×10^{-3}	0.1	1.20×10^{-3}	0.0
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	9.58×10^{-2}		1.16×10^{-1}		9.57×10^{-2}		1.16×10^{-1}	
64	4.87×10^{-2}	1.0	3.68×10^{-2}	1.7	4.77×10^{-2}	1.0	5.57×10^{-2}	1.1
128	2.55×10^{-2}	0.9	1.75×10^{-2}	1.1	2.41×10^{-2}	1.0	1.80×10^{-2}	1.6
256	1.33×10^{-2}	0.9	8.62×10^{-3}	1.0	1.23×10^{-2}	1.0	8.63×10^{-3}	1.1
512	7.24×10^{-3}	0.9	4.26×10^{-3}	1.0	6.32×10^{-3}	1.0	4.25×10^{-3}	1.0
1024	4.21×10^{-3}	0.8	2.12×10^{-3}	1.0	3.31×10^{-3}	0.9	2.11×10^{-3}	1.0
2048	2.71×10^{-3}	0.6	1.07×10^{-3}	1.0	1.81×10^{-3}	0.9	1.05×10^{-3}	1.0
4096	1.96×10^{-3}	0.5	5.73×10^{-4}	0.9	1.06×10^{-3}	0.8	5.20×10^{-4}	1.0
8192	1.58×10^{-3}	0.3	3.70×10^{-4}	0.6	6.83×10^{-4}	0.6	2.60×10^{-4}	1.0

TABLE A.3. *Grid convergence for Neumann Case 2 with BC1* (4.3).

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	5.98×10^{-1}		6.00×10^{-1}		1.10		9.56×10^{-1}	
64	3.54×10^{-1}	0.8	4.03×10^{-1}	0.6	4.25×10^{-1}	1.4	3.28×10^{-1}	1.5
128	2.72×10^{-1}	0.4	3.57×10^{-1}	0.2	1.83×10^{-1}	1.2	1.19×10^{-1}	1.5
256	2.43×10^{-1}	0.2	3.50×10^{-1}	0.0	1.08×10^{-1}	0.8	6.62×10^{-2}	0.8
512	2.31×10^{-1}	0.1	3.51×10^{-1}	-0.0	8.37×10^{-2}	0.4	5.44×10^{-2}	0.3
1024	2.26×10^{-1}	0.0	3.52×10^{-1}	-0.0	7.53×10^{-2}	0.2	5.17×10^{-2}	0.1
2048	2.24×10^{-1}	0.0	3.52×10^{-1}	-0.0	7.20×10^{-2}	0.1	5.12×10^{-2}	0.0
4096	2.23×10^{-1}	0.0	3.53×10^{-1}	-0.0	7.06×10^{-2}	0.0	5.10×10^{-2}	0.0
8192	2.22×10^{-1}	0.0	3.53×10^{-1}	-0.0	6.99×10^{-2}	0.0	5.10×10^{-2}	0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	2.11		1.97		3.19		3.04	
64	1.00	1.1	9.18×10^{-1}	1.1	2.05	0.6	1.98	0.6
128	3.41×10^{-1}	1.6	2.91×10^{-1}	1.7	9.47×10^{-1}	1.1	9.05×10^{-1}	1.1
256	1.14×10^{-1}	1.6	8.24×10^{-2}	1.8	3.05×10^{-1}	1.6	2.82×10^{-1}	1.7
512	4.77×10^{-2}	1.3	2.68×10^{-2}	1.6	8.94×10^{-2}	1.8	7.59×10^{-2}	1.9
1024	2.85×10^{-2}	0.7	1.40×10^{-2}	0.9	2.87×10^{-2}	1.6	2.03×10^{-2}	1.9
2048	2.25×10^{-2}	0.3	1.15×10^{-2}	0.3	1.21×10^{-2}	1.2	6.48×10^{-3}	1.6
4096	2.04×10^{-2}	0.1	1.09×10^{-2}	0.1	7.34×10^{-3}	0.7	3.52×10^{-3}	0.9
8192	1.95×10^{-2}	0.1	1.08×10^{-2}	0.0	5.88×10^{-3}	0.3	2.99×10^{-3}	0.2
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	4.10		3.92		4.61		4.43	
64	3.17	0.4	3.09	0.3	4.08	0.2	3.99	0.2
128	2.02	0.6	1.98	0.6	3.16	0.4	3.12	0.4
256	9.20×10^{-1}	1.1	8.99×10^{-1}	1.1	2.00	0.7	1.98	0.7
512	2.89×10^{-1}	1.7	2.79×10^{-1}	1.7	9.06×10^{-1}	1.1	8.96×10^{-1}	1.1
1024	8.02×10^{-2}	1.9	7.42×10^{-2}	1.9	2.82×10^{-1}	1.7	2.77×10^{-1}	1.7
2048	2.25×10^{-2}	1.8	1.91×10^{-2}	2.0	7.64×10^{-2}	1.9	7.36×10^{-2}	1.9
4096	7.19×10^{-3}	1.6	5.05×10^{-3}	1.9	2.02×10^{-2}	1.9	1.88×10^{-2}	2.0
8192	3.07×10^{-3}	1.2	1.66×10^{-3}	1.6	5.63×10^{-3}	1.8	4.77×10^{-3}	2.0

TABLE A.4. *Grid convergence for Neumann Case 2 with BC2* (4.21).

n	$E_{\text{DDM2 BC1}}$	k	$E_{\text{DDM2 BC2}}$	k	$E_{\text{DDM2 BC1}}$	k	$E_{\text{DDM2 BC2}}$	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	4.03×10^{-1}		1.51×10^{-1}		1.81×10^{-1}		8.46×10^{-1}	
64	3.37×10^{-1}	0.3	1.25×10^{-1}	0.3	1.22×10^{-1}	0.6	2.33×10^{-1}	1.9
128	3.12×10^{-1}	0.1	1.61×10^{-1}	-0.4	9.86×10^{-2}	0.3	3.71×10^{-2}	2.6
256	2.99×10^{-1}	0.1	1.70×10^{-1}	-0.1	8.78×10^{-2}	0.2	4.51×10^{-2}	-0.3
512	2.93×10^{-1}	0.0	1.71×10^{-1}	-0.0	8.27×10^{-2}	0.1	5.67×10^{-2}	-0.3
1024	2.90×10^{-1}	0.0	1.71×10^{-1}	0.0	8.02×10^{-2}	0.0	5.97×10^{-2}	-0.1
2048	2.89×10^{-1}	0.0	1.71×10^{-1}	0.0	7.90×10^{-2}	0.0	6.04×10^{-2}	-0.0
4096	2.88×10^{-1}	0.0	1.70×10^{-1}	0.0	7.84×10^{-2}	0.0	6.06×10^{-2}	-0.0
8192	2.88×10^{-1}	0.0	1.70×10^{-1}	0.0	7.81×10^{-2}	0.0	6.06×10^{-2}	-0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	1.10×10^{-1}		1.94		8.92×10^{-2}		3.04	
64	6.00×10^{-2}	0.9	8.94×10^{-1}	1.1	4.39×10^{-2}	1.0	1.97	0.6
128	3.83×10^{-2}	0.6	2.69×10^{-1}	1.7	2.33×10^{-2}	0.9	9.00×10^{-1}	1.1
256	2.85×10^{-2}	0.4	6.19×10^{-2}	2.1	1.37×10^{-2}	0.8	2.77×10^{-1}	1.7
512	2.38×10^{-2}	0.3	1.02×10^{-2}	2.6	9.18×10^{-3}	0.6	7.10×10^{-2}	2.0
1024	2.16×10^{-2}	0.1	1.32×10^{-2}	-0.4	6.98×10^{-3}	0.4	1.55×10^{-2}	2.2
2048	2.04×10^{-2}	0.1	1.60×10^{-2}	-0.3	5.90×10^{-3}	0.2	2.95×10^{-3}	2.4
4096	1.99×10^{-2}	0.0	1.68×10^{-2}	-0.1	5.36×10^{-3}	0.1	3.74×10^{-3}	-0.3
8192	1.96×10^{-2}	0.0	1.70×10^{-2}	-0.0	5.10×10^{-3}	0.1	4.39×10^{-3}	-0.2
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	1.24×10^{-1}		3.91		1.24×10^{-1}		4.43	
64	3.94×10^{-2}	1.6	3.09	0.3	5.74×10^{-2}	1.1	3.99	0.2
128	1.94×10^{-2}	1.0	1.98	0.6	1.86×10^{-2}	1.6	3.12	0.4
256	1.00×10^{-2}	1.0	8.98×10^{-1}	1.1	9.09×10^{-3}	1.0	1.98	0.7
512	5.52×10^{-3}	0.9	2.77×10^{-1}	1.7	4.59×10^{-3}	1.0	8.96×10^{-1}	1.1
1024	3.33×10^{-3}	0.7	7.30×10^{-2}	1.9	2.42×10^{-3}	0.9	2.77×10^{-1}	1.7
2048	2.26×10^{-3}	0.6	1.79×10^{-2}	2.0	1.35×10^{-3}	0.8	7.33×10^{-2}	1.9
4096	1.73×10^{-3}	0.4	3.88×10^{-3}	2.2	8.18×10^{-4}	0.7	1.85×10^{-2}	2.0
8192	1.46×10^{-3}	0.2	9.06×10^{-4}	2.1	5.54×10^{-4}	0.6	4.47×10^{-3}	2.0

TABLE A.5. *Grid convergence for Neumann Case 2 with BC1* (4.3) *and BC2* (4.21) *with no surface Laplacian term.*

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	1.33×10^{-1}		9.44×10^{-2}		3.50×10^{-2}		5.25×10^{-2}	
64	1.29×10^{-1}	0.0	8.89×10^{-2}	0.1	3.24×10^{-2}	0.1	3.24×10^{-2}	0.7
128	1.28×10^{-1}	0.0	8.77×10^{-2}	0.0	3.15×10^{-2}	0.0	2.93×10^{-2}	0.1
256	1.28×10^{-1}	0.0	8.75×10^{-2}	0.0	3.12×10^{-2}	0.0	2.87×10^{-2}	0.0
512	1.27×10^{-1}	0.0	8.75×10^{-2}	0.0	3.12×10^{-2}	0.0	2.86×10^{-2}	0.0
1024	1.27×10^{-1}	0.0	8.74×10^{-2}	0.0	3.12×10^{-2}	0.0	2.85×10^{-2}	0.0
2048	1.27×10^{-1}	0.0	8.74×10^{-2}	0.0	3.12×10^{-2}	0.0	2.85×10^{-2}	0.0
4096	1.27×10^{-1}	0.0	8.74×10^{-2}	-0.0	3.12×10^{-2}	0.0	2.85×10^{-2}	0.0
8192	1.27×10^{-1}	0.0	8.74×10^{-2}	0.0	3.12×10^{-2}	0.0	2.85×10^{-2}	0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	3.00×10^{-3}		9.62×10^{-2}		2.80×10^{-2}		2.31×10^{-1}	
64	7.38×10^{-3}	-1.3	2.19×10^{-2}	2.1	8.33×10^{-3}	1.8	7.93×10^{-2}	1.5
128	7.71×10^{-3}	-0.1	8.89×10^{-3}	1.3	7.43×10^{-4}	3.5	1.40×10^{-2}	2.5
256	7.56×10^{-3}	0.0	7.36×10^{-3}	0.3	1.78×10^{-3}	-1.3	2.99×10^{-3}	2.2
512	7.50×10^{-3}	0.0	7.14×10^{-3}	0.0	1.83×10^{-3}	-0.0	1.89×10^{-3}	0.7
1024	7.49×10^{-3}	0.0	7.10×10^{-3}	0.0	1.82×10^{-3}	0.0	1.77×10^{-3}	0.1
2048	7.49×10^{-3}	0.0	7.09×10^{-3}	0.0	1.81×10^{-3}	0.0	1.76×10^{-3}	0.0
4096	7.48×10^{-3}	0.0	7.08×10^{-3}	0.0	1.81×10^{-3}	0.0	1.75×10^{-3}	0.0
8192	7.48×10^{-3}	0.0	7.08×10^{-3}	0.0	1.81×10^{-3}	0.0	1.75×10^{-3}	0.0
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	4.72×10^{-2}		3.61×10^{-1}		5.43×10^{-2}		4.01×10^{-1}	
64	3.50×10^{-2}	0.4	2.21×10^{-1}	0.7	5.88×10^{-2}	-0.1	3.13×10^{-1}	0.4
128	1.04×10^{-2}	1.7	7.53×10^{-2}	1.6	3.61×10^{-2}	0.7	2.06×10^{-1}	0.6
256	9.47×10^{-4}	3.5	1.18×10^{-2}	2.7	1.09×10^{-2}	1.7	7.37×10^{-2}	1.5
512	3.54×10^{-4}	1.4	1.47×10^{-3}	3.0	1.33×10^{-3}	3.0	1.12×10^{-2}	2.7
1024	4.46×10^{-4}	-0.3	5.25×10^{-4}	1.5	1.79×10^{-5}	6.2	1.08×10^{-3}	3.4
2048	4.49×10^{-4}	-0.0	4.43×10^{-4}	0.2	1.08×10^{-4}	-2.6	1.81×10^{-4}	2.6
4096	4.48×10^{-4}	0.0	4.34×10^{-4}	0.0	1.15×10^{-4}	-0.1	1.12×10^{-4}	0.7
8192	4.48×10^{-4}	0.0	4.32×10^{-4}	0.0	1.15×10^{-4}	-0.0	1.06×10^{-4}	0.1

TABLE A.6. *Grid convergence for Neumann Case 3 with BC1* (4.3).

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	1.85×10^{-1}		4.56×10^{-2}		2.92×10^{-1}		2.25×10^{-1}	
64	1.34×10^{-1}	0.5	7.22×10^{-2}	-0.7	1.02×10^{-1}	1.5	5.18×10^{-2}	2.1
128	1.22×10^{-1}	0.1	8.46×10^{-2}	-0.2	4.55×10^{-2}	1.2	1.99×10^{-2}	1.4
256	1.19×10^{-1}	0.0	8.79×10^{-2}	-0.1	3.21×10^{-2}	0.5	3.02×10^{-2}	-0.6
512	1.19×10^{-1}	0.0	8.87×10^{-2}	-0.0	2.91×10^{-2}	0.1	3.34×10^{-2}	-0.1
1024	1.18×10^{-1}	0.0	8.89×10^{-2}	-0.0	2.84×10^{-2}	0.0	3.43×10^{-2}	-0.0
2048	1.18×10^{-1}	0.0	8.90×10^{-2}	-0.0	2.82×10^{-2}	0.0	3.45×10^{-2}	-0.0
4096	1.18×10^{-1}	0.0	8.90×10^{-2}	-0.0	2.82×10^{-2}	0.0	3.45×10^{-2}	-0.0
8192	1.18×10^{-1}	0.0	8.90×10^{-2}	-0.0	2.82×10^{-2}	0.0	3.45×10^{-2}	-0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	6.69×10^{-1}		5.83×10^{-1}		1.13×10^{-0}		9.36×10^{-1}	
64	2.73×10^{-1}	1.3	2.48×10^{-1}	1.2	6.59×10^{-1}	0.8	5.90×10^{-1}	0.7
128	8.48×10^{-2}	1.7	7.09×10^{-2}	1.8	2.67×10^{-1}	1.3	2.53×10^{-1}	1.2
256	2.65×10^{-2}	1.7	1.47×10^{-2}	2.3	8.03×10^{-2}	1.7	7.61×10^{-2}	1.7
512	1.20×10^{-2}	1.1	7.63×10^{-3}	0.9	2.21×10^{-2}	1.9	1.89×10^{-2}	2.0
1024	8.94×10^{-3}	0.4	9.73×10^{-3}	-0.4	6.87×10^{-3}	1.7	4.20×10^{-3}	2.2
2048	8.31×10^{-3}	0.1	1.04×10^{-2}	-0.1	3.51×10^{-3}	1.0	2.77×10^{-3}	0.6
4096	8.16×10^{-3}	0.0	1.06×10^{-2}	-0.0	2.91×10^{-3}	0.3	3.16×10^{-3}	-0.2
8192	8.13×10^{-3}	0.0	1.07×10^{-2}	-0.0	2.79×10^{-3}	0.1	3.30×10^{-3}	-0.1
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	1.47×10^{-0}		1.18×10^{-0}		1.67×10^{-0}		1.36×10^{-0}	
64	1.11×10^{-0}	0.4	9.28×10^{-1}	0.3	1.46×10^{-0}	0.2	1.21×10^{-0}	0.2
128	6.58×10^{-1}	0.8	5.94×10^{-1}	0.6	1.12×10^{-0}	0.4	9.56×10^{-1}	0.3
256	2.64×10^{-1}	1.3	2.54×10^{-1}	1.2	6.56×10^{-1}	0.8	5.93×10^{-1}	0.7
512	7.89×10^{-2}	1.7	7.72×10^{-2}	1.7	2.63×10^{-1}	1.3	2.54×10^{-1}	1.2
1024	2.10×10^{-2}	1.9	2.01×10^{-2}	1.9	7.85×10^{-2}	1.7	7.74×10^{-2}	1.7
2048	5.60×10^{-3}	1.9	4.82×10^{-3}	2.1	2.07×10^{-2}	1.9	2.04×10^{-2}	1.9
4096	1.85×10^{-3}	1.6	1.27×10^{-3}	1.9	5.30×10^{-3}	2.0	5.10×10^{-3}	2.0
8192	1.12×10^{-3}	0.7	9.85×10^{-4}	0.4	1.43×10^{-3}	1.9	1.24×10^{-3}	2.0

TABLE A.7. *Grid convergence for Neumann Case 3 with BC2* (4.21).

n	$E_{\text{DDM2 BC1}}$	k	$E_{\text{DDM2 BC2}}$	k	$E_{\text{DDM2 BC1}}$	k	$E_{\text{DDM2 BC2}}$	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	3.07×10^{-2}		1.06×10^{-1}		3.20×10^{-2}		2.46×10^{-1}	
64	3.13×10^{-2}	-0.0	7.08×10^{-2}	0.6	1.53×10^{-2}	1.1	6.88×10^{-2}	1.8
128	3.15×10^{-2}	-0.0	6.77×10^{-2}	0.1	1.32×10^{-2}	0.2	1.99×10^{-2}	1.8
256	3.16×10^{-2}	-0.0	6.75×10^{-2}	0.0	1.29×10^{-2}	0.0	1.95×10^{-2}	0.0
512	3.16×10^{-2}	-0.0	6.75×10^{-2}	0.0	1.28×10^{-2}	0.0	2.13×10^{-2}	-0.1
1024	3.16×10^{-2}	-0.0	6.75×10^{-2}	0.0	1.28×10^{-2}	0.0	2.19×10^{-2}	-0.0
2048	3.16×10^{-2}	-0.0	6.75×10^{-2}	-0.0	1.28×10^{-2}	0.0	2.20×10^{-2}	-0.0
4096	3.16×10^{-2}	-0.0	6.75×10^{-2}	0.0	1.28×10^{-2}	0.0	2.20×10^{-2}	-0.0
8192	3.16×10^{-2}	0.0	6.75×10^{-2}	-0.0	1.28×10^{-2}	0.0	2.21×10^{-2}	-0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	8.92×10^{-2}		5.90×10^{-1}		2.25×10^{-1}		9.43×10^{-1}	
64	1.74×10^{-2}	2.4	2.53×10^{-1}	1.2	7.77×10^{-2}	1.5	5.92×10^{-1}	0.7
128	4.93×10^{-3}	1.8	7.48×10^{-2}	1.8	1.29×10^{-2}	2.6	2.54×10^{-1}	1.2
256	3.60×10^{-3}	0.5	1.81×10^{-2}	2.1	2.04×10^{-3}	2.7	7.70×10^{-2}	1.7
512	3.43×10^{-3}	0.1	6.91×10^{-3}	1.4	9.80×10^{-4}	1.1	1.98×10^{-2}	2.0
1024	3.41×10^{-3}	0.0	7.33×10^{-3}	-0.1	8.76×10^{-4}	0.2	4.89×10^{-3}	2.0
2048	3.40×10^{-3}	0.0	7.76×10^{-3}	-0.1	8.61×10^{-4}	0.0	2.60×10^{-3}	0.9
4096	3.40×10^{-3}	0.0	7.88×10^{-3}	-0.0	8.58×10^{-4}	0.0	2.70×10^{-3}	-0.1
8192	3.40×10^{-3}	0.0	7.92×10^{-3}	-0.0	8.58×10^{-4}	0.0	2.78×10^{-3}	-0.0
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	3.49×10^{-1}		1.19		3.84×10^{-1}		1.37	
64	2.20×10^{-1}	0.7	9.29×10^{-1}	0.4	3.10×10^{-1}	0.3	1.21	0.2
128	7.49×10^{-2}	1.6	5.95×10^{-1}	0.6	2.05×10^{-1}	0.6	9.57×10^{-1}	0.3
256	1.16×10^{-2}	2.7	2.54×10^{-1}	1.2	7.36×10^{-2}	1.5	5.93×10^{-1}	0.7
512	1.24×10^{-3}	3.2	7.75×10^{-2}	1.7	1.12×10^{-2}	2.7	2.54×10^{-1}	1.2
1024	3.02×10^{-4}	2.0	2.03×10^{-2}	1.9	1.02×10^{-3}	3.5	7.75×10^{-2}	1.7
2048	2.23×10^{-4}	0.4	5.04×10^{-3}	2.0	1.26×10^{-4}	3.0	2.05×10^{-2}	1.9
4096	2.14×10^{-4}	0.1	1.41×10^{-3}	1.8	5.80×10^{-5}	1.1	5.16×10^{-3}	2.0
8192	2.12×10^{-4}	0.0	9.54×10^{-4}	0.6	5.19×10^{-5}	0.2	1.29×10^{-3}	2.0

TABLE A.8. *Grid convergence for Neumann Case 3 with BC1* (4.3) *and BC2* (4.21) *with no surface Laplacian term.*

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	1.97×10^{-1}		1.77×10^{-1}		7.09×10^{-2}		5.79×10^{-2}	
64	1.83×10^{-1}	0.1	1.48×10^{-1}	0.3	5.70×10^{-2}	0.3	3.42×10^{-2}	0.8
128	1.77×10^{-1}	0.1	1.42×10^{-1}	0.1	5.04×10^{-2}	0.2	2.62×10^{-2}	0.4
256	1.74×10^{-1}	0.0	1.39×10^{-1}	0.0	4.72×10^{-2}	0.1	2.30×10^{-2}	0.2
512	1.72×10^{-1}	0.0	1.38×10^{-1}	0.0	4.56×10^{-2}	0.0	2.16×10^{-2}	0.1
1024	1.72×10^{-1}	0.0	1.38×10^{-1}	0.0	4.49×10^{-2}	0.0	2.10×10^{-2}	0.0
2048	1.71×10^{-1}	0.0	1.38×10^{-1}	0.0	4.45×10^{-2}	0.0	2.07×10^{-2}	0.0
4096	1.71×10^{-1}	0.0	1.38×10^{-1}	0.0	4.43×10^{-2}	0.0	2.05×10^{-2}	0.0
8192	1.71×10^{-1}	0.0	1.38×10^{-1}	0.0	4.42×10^{-2}	0.0	2.04×10^{-2}	0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	3.74×10^{-2}		3.79×10^{-2}		2.60×10^{-2}		3.35×10^{-2}	
64	2.45×10^{-2}	0.6	1.74×10^{-2}	1.1	1.59×10^{-2}	0.7	1.39×10^{-2}	1.3
128	1.77×10^{-2}	0.5	9.93×10^{-3}	0.8	9.40×10^{-3}	0.8	6.58×10^{-3}	1.1
256	1.45×10^{-2}	0.3	6.96×10^{-3}	0.5	6.09×10^{-3}	0.6	3.60×10^{-3}	0.9
512	1.29×10^{-2}	0.2	5.67×10^{-3}	0.3	4.46×10^{-3}	0.4	2.28×10^{-3}	0.7
1024	1.21×10^{-2}	0.1	5.07×10^{-3}	0.2	3.66×10^{-3}	0.3	1.68×10^{-3}	0.4
2048	1.17×10^{-2}	0.0	4.79×10^{-3}	0.1	3.25×10^{-3}	0.2	1.39×10^{-3}	0.3
4096	1.15×10^{-2}	0.0	4.65×10^{-3}	0.0	3.05×10^{-3}	0.1	1.25×10^{-3}	0.2
8192	1.14×10^{-2}	0.0	4.58×10^{-3}	0.0	2.95×10^{-3}	0.0	1.18×10^{-3}	0.1
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	2.39×10^{-2}		4.33×10^{-2}		2.39×10^{-2}		4.33×10^{-2}	
64	1.30×10^{-2}	0.9	1.31×10^{-2}	1.7	1.25×10^{-2}	0.9	1.81×10^{-2}	1.3
128	7.22×10^{-3}	0.8	5.76×10^{-3}	1.2	6.49×10^{-3}	0.9	5.64×10^{-3}	1.7
256	3.98×10^{-3}	0.9	2.79×10^{-3}	1.0	3.43×10^{-3}	0.9	2.59×10^{-3}	1.1
512	2.34×10^{-3}	0.8	1.46×10^{-3}	0.9	1.81×10^{-3}	0.9	1.26×10^{-3}	1.0
1024	1.53×10^{-3}	0.6	8.48×10^{-4}	0.8	9.97×10^{-4}	0.9	6.46×10^{-4}	1.0
2048	1.13×10^{-3}	0.4	5.55×10^{-4}	0.6	5.91×10^{-4}	0.8	3.50×10^{-4}	0.9
4096	9.24×10^{-4}	0.3	4.12×10^{-4}	0.4	3.89×10^{-4}	0.6	2.05×10^{-4}	0.8
8192	8.23×10^{-4}	0.2	3.43×10^{-4}	0.3	2.88×10^{-4}	0.4	1.33×10^{-4}	0.6

TABLE A.9. *Grid convergence for Neumann Case 4 with BC1* (4.3).

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	2.69×10^{-1}		4.48×10^{-2}		2.79×10^{-1}		1.74×10^{-1}	
64	2.09×10^{-1}	0.4	7.16×10^{-2}	-0.7	1.28×10^{-1}	1.1	4.94×10^{-2}	1.8
128	1.88×10^{-1}	0.2	8.18×10^{-2}	-0.2	7.27×10^{-2}	0.8	1.02×10^{-2}	2.3
256	1.80×10^{-1}	0.1	8.44×10^{-2}	-0.0	5.53×10^{-2}	0.4	1.11×10^{-2}	-0.1
512	1.77×10^{-1}	0.0	8.51×10^{-2}	-0.0	4.96×10^{-2}	0.2	1.31×10^{-2}	-0.2
1024	1.75×10^{-1}	0.0	8.55×10^{-2}	-0.0	4.75×10^{-2}	0.1	1.37×10^{-2}	-0.1
2048	1.75×10^{-1}	0.0	8.57×10^{-2}	-0.0	4.66×10^{-2}	0.0	1.38×10^{-2}	-0.0
4096	1.74×10^{-1}	0.0	8.58×10^{-2}	-0.0	4.63×10^{-2}	0.0	1.38×10^{-2}	-0.0
8192	1.74×10^{-1}	0.0	8.58×10^{-2}	-0.0	4.61×10^{-2}	0.0	1.38×10^{-2}	-0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	4.82×10^{-1}		4.19×10^{-1}		7.30×10^{-1}		6.78×10^{-1}	
64	2.33×10^{-1}	1.1	1.96×10^{-1}	1.1	4.68×10^{-1}	0.6	4.41×10^{-1}	0.6
128	8.50×10^{-2}	1.5	5.98×10^{-2}	1.7	2.17×10^{-1}	1.1	2.02×10^{-1}	1.1
256	3.39×10^{-2}	1.3	1.43×10^{-2}	2.1	7.16×10^{-2}	1.6	6.24×10^{-2}	1.7
512	1.88×10^{-2}	0.9	2.88×10^{-3}	2.3	2.25×10^{-2}	1.7	1.62×10^{-2}	1.9
1024	1.43×10^{-2}	0.4	2.69×10^{-3}	0.1	8.65×10^{-3}	1.4	3.68×10^{-3}	2.1
2048	1.28×10^{-2}	0.2	3.18×10^{-3}	-0.2	4.78×10^{-3}	0.9	7.71×10^{-4}	2.3
4096	1.23×10^{-2}	0.1	3.32×10^{-3}	-0.1	3.64×10^{-3}	0.4	7.42×10^{-4}	0.1
8192	1.20×10^{-2}	0.0	3.36×10^{-3}	-0.0	3.27×10^{-3}	0.2	8.62×10^{-4}	-0.2
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	9.41×10^{-1}		8.81×10^{-1}		1.06		1.00	
64	7.28×10^{-1}	0.4	7.04×10^{-1}	0.3	9.42×10^{-1}	0.2	9.13×10^{-1}	0.1
128	4.63×10^{-1}	0.7	4.50×10^{-1}	0.6	7.28×10^{-1}	0.4	7.16×10^{-1}	0.4
256	2.11×10^{-1}	1.1	2.05×10^{-1}	1.1	4.60×10^{-1}	0.7	4.55×10^{-1}	0.7
512	6.69×10^{-2}	1.7	6.32×10^{-2}	1.7	2.08×10^{-1}	1.1	2.06×10^{-1}	1.1
1024	1.90×10^{-2}	1.8	1.67×10^{-2}	1.9	6.51×10^{-2}	1.7	6.35×10^{-2}	1.7
2048	5.73×10^{-3}	1.7	4.13×10^{-3}	2.0	1.78×10^{-2}	1.9	1.68×10^{-2}	1.9
4096	2.19×10^{-3}	1.4	9.36×10^{-4}	2.1	4.83×10^{-3}	1.9	4.25×10^{-3}	2.0
8192	1.21×10^{-3}	0.8	2.25×10^{-4}	2.1	1.45×10^{-3}	1.7	1.04×10^{-3}	2.0

Table A.10. *Grid convergence for Neumann Case 4 with BC2* (4.21).

n	$E_{\text{DDM2 BC1}}$	k	$E_{\text{DDM2 BC2}}$	k	$E_{\text{DDM2 BC1}}$	k	$E_{\text{DDM2 BC2}}$	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	3.76×10^{-1}		2.32×10^{-1}		9.97×10^{-2}		1.32×10^{-1}	
64	3.44×10^{-1}	0.1	2.60×10^{-1}	-0.2	7.05×10^{-2}	0.5	1.36×10^{-2}	3.3
128	3.39×10^{-1}	0.0	2.73×10^{-1}	-0.1	6.13×10^{-2}	0.2	3.11×10^{-2}	-1.2
256	3.38×10^{-1}	0.0	2.77×10^{-1}	-0.0	5.77×10^{-2}	0.1	4.25×10^{-2}	-0.5
512	3.38×10^{-1}	-0.0	2.79×10^{-1}	-0.0	5.63×10^{-2}	0.0	4.54×10^{-2}	-0.1
1024	3.38×10^{-1}	-0.0	2.80×10^{-1}	-0.0	5.56×10^{-2}	0.0	4.62×10^{-2}	-0.0
2048	3.38×10^{-1}	-0.0	2.81×10^{-1}	-0.0	5.53×10^{-2}	0.0	4.63×10^{-2}	-0.0
4096	3.39×10^{-1}	-0.0	2.81×10^{-1}	-0.0	5.51×10^{-2}	0.0	4.64×10^{-2}	-0.0
8192	3.39×10^{-1}	-0.0	2.81×10^{-1}	-0.0	5.51×10^{-2}	0.0	4.64×10^{-2}	-0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	5.07×10^{-2}		4.07×10^{-1}		3.77×10^{-2}		6.74×10^{-1}	
64	2.67×10^{-2}	0.9	1.87×10^{-1}	1.1	1.69×10^{-2}	1.2	4.38×10^{-1}	0.6
128	1.82×10^{-2}	0.6	5.15×10^{-2}	1.9	8.86×10^{-3}	0.9	2.00×10^{-1}	1.1
256	1.49×10^{-2}	0.3	6.46×10^{-3}	3.0	5.62×10^{-3}	0.7	6.04×10^{-2}	1.7
512	1.36×10^{-2}	0.1	6.60×10^{-3}	-0.0	4.24×10^{-3}	0.4	1.42×10^{-2}	2.1
1024	1.29×10^{-2}	0.1	9.64×10^{-3}	-0.5	3.61×10^{-3}	0.2	1.81×10^{-3}	3.0
2048	1.26×10^{-2}	0.0	1.04×10^{-2}	-0.1	3.31×10^{-3}	0.1	1.65×10^{-3}	0.1
4096	1.25×10^{-2}	0.0	1.06×10^{-2}	-0.0	3.16×10^{-3}	0.1	2.41×10^{-3}	-0.5
8192	1.24×10^{-2}	0.0	1.07×10^{-2}	-0.0	3.09×10^{-3}	0.0	2.60×10^{-3}	-0.0
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	4.59×10^{-2}		8.79×10^{-1}		4.59×10^{-2}		9.97×10^{-1}	
64	1.41×10^{-2}	1.7	7.03×10^{-1}	0.3	1.87×10^{-2}	1.3	9.13×10^{-1}	0.1
128	6.52×10^{-3}	1.1	4.50×10^{-1}	0.6	5.89×10^{-3}	1.7	7.16×10^{-1}	0.3
256	3.36×10^{-3}	1.0	2.04×10^{-1}	1.1	2.78×10^{-3}	1.1	4.54×10^{-1}	0.7
512	1.97×10^{-3}	0.8	6.27×10^{-2}	1.7	1.40×10^{-3}	1.0	2.05×10^{-1}	1.1
1024	1.34×10^{-3}	0.6	1.62×10^{-2}	2.0	7.73×10^{-4}	0.9	6.34×10^{-2}	1.7
2048	1.04×10^{-3}	0.4	3.65×10^{-3}	2.2	4.72×10^{-4}	0.7	1.67×10^{-2}	1.9
4096	8.92×10^{-4}	0.2	4.80×10^{-4}	2.9	3.26×10^{-4}	0.5	4.13×10^{-3}	2.0
8192	8.20×10^{-4}	0.1	4.27×10^{-4}	0.2	2.53×10^{-4}	0.4	9.24×10^{-4}	2.2

TABLE A.11. *Grid convergence for Neumann Case 4 with BC1* (4.3) *and BC2* (4.21) *with no surface Laplacian term.*

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	2.24×10^{-1}		1.21×10^{-1}		5.18×10^{-2}		4.35×10^{-2}	
64	2.15×10^{-1}	0.1	1.20×10^{-1}	0.0	4.63×10^{-2}	0.2	2.90×10^{-2}	0.6
128	2.12×10^{-1}	0.0	1.20×10^{-1}	0.0	4.46×10^{-2}	0.1	2.74×10^{-2}	0.1
256	2.12×10^{-1}	0.0	1.20×10^{-1}	-0.0	4.42×10^{-2}	0.0	2.72×10^{-2}	0.0
512	2.12×10^{-1}	0.0	1.20×10^{-1}	0.0	4.41×10^{-2}	0.0	2.72×10^{-2}	0.0
1024	2.11×10^{-1}	0.0	1.20×10^{-1}	-0.0	4.40×10^{-2}	0.0	2.72×10^{-2}	0.0
2048	2.11×10^{-1}	0.0	1.20×10^{-1}	-0.0	4.40×10^{-2}	0.0	2.72×10^{-2}	0.0
4096	2.11×10^{-1}	0.0	1.20×10^{-1}	-0.0	4.40×10^{-2}	0.0	2.72×10^{-2}	0.0
8192	2.11×10^{-1}	0.0	1.20×10^{-1}	0.0	4.40×10^{-2}	0.0	2.72×10^{-2}	0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	5.97×10^{-3}		1.11×10^{-1}					
64	8.67×10^{-3}	-0.5	2.57×10^{-2}	2.1	1.64×10^{-2}		1.19×10^{-1}	
128	9.31×10^{-3}	-0.1	8.32×10^{-3}	1.6	8.97×10^{-4}	4.2	2.14×10^{-2}	2.5
256	9.11×10^{-3}	0.0	6.61×10^{-3}	0.3	1.86×10^{-3}	-1.1	3.43×10^{-3}	2.6
512	9.02×10^{-3}	0.0	6.45×10^{-3}	0.0	1.96×10^{-3}	-0.1	1.74×10^{-3}	1.0
1024	9.00×10^{-3}	0.0	6.43×10^{-3}	0.0	1.95×10^{-3}	0.0	1.59×10^{-3}	0.1
2048	8.99×10^{-3}	0.0	6.42×10^{-3}	0.0	1.95×10^{-3}	0.0	1.57×10^{-3}	0.0
4096	8.99×10^{-3}	0.0	6.42×10^{-3}	0.0	1.95×10^{-3}	0.0	1.57×10^{-3}	0.0
8192	8.99×10^{-3}	0.0	6.42×10^{-3}	0.0	1.95×10^{-3}	0.0	1.57×10^{-3}	0.0
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	4.54×10^{-2}		4.76×10^{-1}		3.52×10^{-2}		5.12×10^{-1}	
64	5.93×10^{-2}	-0.4	3.43×10^{-1}	0.5	9.37×10^{-2}	-1.4	3.84×10^{-1}	0.4
128	1.92×10^{-2}	1.6	1.25×10^{-1}	1.5	6.30×10^{-2}	0.6	2.88×10^{-1}	0.4
256	2.13×10^{-3}	3.2	2.03×10^{-2}	2.6	1.98×10^{-2}	1.7	1.26×10^{-1}	1.2
512	2.84×10^{-4}	2.9	2.16×10^{-3}	3.2	2.52×10^{-3}	3.0	1.99×10^{-2}	2.7
1024	4.50×10^{-4}	-0.7	5.28×10^{-4}	2.0	9.31×10^{-5}	4.8	1.82×10^{-3}	3.5
2048	4.58×10^{-4}	-0.0	3.95×10^{-4}	0.4	1.11×10^{-4}	-0.2	2.16×10^{-4}	3.1
4096	4.57×10^{-4}	0.0	3.81×10^{-4}	0.1	1.22×10^{-4}	-0.1	9.63×10^{-5}	1.2
8192	4.57×10^{-4}	0.0	3.79×10^{-4}	0.0	1.23×10^{-4}	-0.0	8.59×10^{-5}	0.2

TABLE A.12. *Grid convergence for Robin Case 1 with BC1* (4.4).

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	3.26×10^{-1}		1.22×10^{-1}		6.66×10^{-1}		5.10×10^{-1}	
64	2.07×10^{-1}	0.7	2.12×10^{-1}	-0.8	2.13×10^{-1}	1.6	1.13×10^{-1}	2.2
128	1.79×10^{-1}	0.2	2.36×10^{-1}	-0.2	9.18×10^{-2}	1.2	2.25×10^{-2}	2.3
256	1.72×10^{-1}	0.1	2.42×10^{-1}	-0.0	6.18×10^{-2}	0.6	3.94×10^{-2}	-0.8
512	1.70×10^{-1}	0.0	2.43×10^{-1}	-0.0	5.45×10^{-2}	0.2	4.58×10^{-2}	-0.2
1024	1.70×10^{-1}	0.0	2.44×10^{-1}	-0.0	5.27×10^{-2}	0.0	4.74×10^{-2}	-0.1
2048	1.70×10^{-1}	0.0	2.44×10^{-1}	-0.0	5.23×10^{-2}	0.0	4.78×10^{-2}	-0.0
4096	1.70×10^{-1}	0.0	2.44×10^{-1}	-0.0	5.22×10^{-2}	0.0	4.79×10^{-2}	-0.0
8192	1.70×10^{-1}	0.0	2.44×10^{-1}	-0.0	5.21×10^{-2}	0.0	4.79×10^{-2}	-0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	1.94		1.69					
64	6.11×10^{-1}	1.7	5.62×10^{-1}	1.6	1.87		1.64	
128	1.78×10^{-1}	1.8	1.55×10^{-1}	1.9	5.84×10^{-1}	1.7	5.57×10^{-1}	1.6
256	6.12×10^{-2}	1.5	4.08×10^{-2}	1.9	1.63×10^{-1}	1.8	1.57×10^{-1}	1.8
512	3.15×10^{-2}	1.0	1.32×10^{-2}	1.6	4.90×10^{-2}	1.7	4.42×10^{-2}	1.8
1024	2.41×10^{-2}	0.4	8.70×10^{-3}	0.6	1.99×10^{-2}	1.3	1.52×10^{-2}	1.5
2048	2.23×10^{-2}	0.1	8.30×10^{-3}	0.1	1.26×10^{-2}	0.7	8.12×10^{-3}	0.9
4096	2.19×10^{-2}	0.0	8.26×10^{-3}	0.0	1.08×10^{-2}	0.2	6.42×10^{-3}	0.3
8192	2.18×10^{-2}	0.0	8.26×10^{-3}	0.0	1.03×10^{-2}	0.1	6.01×10^{-3}	0.1
$\epsilon = 0.050$					$\epsilon = 0.025$			
64	4.88		3.39					
128	1.86	1.4	1.62	1.1	5.04		3.81	
256	5.70×10^{-1}	1.7	5.47×10^{-1}	1.6	1.84	1.5	1.59	1.3
512	1.56×10^{-1}	1.9	1.53×10^{-1}	1.8	5.63×10^{-1}	1.7	5.41×10^{-1}	1.6
1024	4.34×10^{-2}	1.8	4.22×10^{-2}	1.9	1.52×10^{-1}	1.9	1.51×10^{-1}	1.8
2048	1.46×10^{-2}	1.6	1.34×10^{-2}	1.6	4.06×10^{-2}	1.9	4.03×10^{-2}	1.9
4096	7.36×10^{-3}	1.0	6.24×10^{-3}	1.1	1.20×10^{-2}	1.8	1.18×10^{-2}	1.8
8192	5.57×10^{-3}	0.4	4.46×10^{-3}	0.5	4.85×10^{-3}	1.3	4.57×10^{-3}	1.4

TABLE A.13. *Grid convergence for Robin Case 1 with BC2* (4.22).

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	1.37×10^{-1}		2.73×10^{-2}		3.16×10^{-2}		1.78×10^{-2}	
64	1.34×10^{-1}	0.0	2.73×10^{-2}	0.0	2.86×10^{-2}	0.1	9.98×10^{-3}	0.8
128	1.33×10^{-1}	0.0	2.74×10^{-2}	-0.0	2.78×10^{-2}	0.0	8.95×10^{-3}	0.2
256	1.32×10^{-1}	0.0	2.75×10^{-2}	-0.0	2.76×10^{-2}	0.0	8.80×10^{-3}	0.0
512	1.32×10^{-1}	0.0	2.75×10^{-2}	0.0	2.76×10^{-2}	0.0	8.78×10^{-3}	0.0
1024	1.32×10^{-1}	0.0	2.75×10^{-2}	-0.0	2.75×10^{-2}	0.0	8.77×10^{-3}	0.0
2048	1.32×10^{-1}	0.0	2.75×10^{-2}	-0.0	2.75×10^{-2}	0.0	8.77×10^{-3}	0.0
4096	1.32×10^{-1}	0.0	2.75×10^{-2}	-0.0	2.75×10^{-2}	0.0	8.77×10^{-3}	0.0
8192	1.32×10^{-1}	0.0	2.75×10^{-2}	0.0	2.75×10^{-2}	0.0	8.77×10^{-3}	0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	5.54×10^{-3}		4.18×10^{-2}					
64	5.62×10^{-3}	-0.0	9.72×10^{-3}	2.1	5.52×10^{-3}		4.29×10^{-2}	
128	5.46×10^{-3}	0.0	3.06×10^{-3}	1.7	7.81×10^{-4}	2.8	7.67×10^{-3}	2.5
256	5.33×10^{-3}	0.0	2.30×10^{-3}	0.4	1.02×10^{-3}	-0.4	1.25×10^{-3}	2.6
512	5.29×10^{-3}	0.0	2.22×10^{-3}	0.1	1.03×10^{-3}	-0.0	6.18×10^{-4}	1.0
1024	5.27×10^{-3}	0.0	2.20×10^{-3}	0.0	1.02×10^{-3}	0.0	5.57×10^{-4}	0.2
2048	5.27×10^{-3}	0.0	2.20×10^{-3}	0.0	1.02×10^{-3}	0.0	5.49×10^{-4}	0.0
4096	5.27×10^{-3}	0.0	2.20×10^{-3}	0.0	1.02×10^{-3}	0.0	5.48×10^{-4}	0.0
8192	5.27×10^{-3}	0.0	2.20×10^{-3}	0.0	1.02×10^{-3}	0.0	5.47×10^{-4}	0.0
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	1.79×10^{-2}		1.85×10^{-1}					
64	2.13×10^{-2}	-0.2	1.28×10^{-1}	0.5				
128	6.75×10^{-3}	1.7	4.49×10^{-2}	1.5	2.26×10^{-2}		1.07×10^{-1}	
256	7.41×10^{-4}	3.2	7.19×10^{-3}	2.6	7.04×10^{-3}	1.7	4.55×10^{-2}	1.2
512	1.67×10^{-4}	2.1	7.75×10^{-4}	3.2	8.89×10^{-4}	3.0	7.07×10^{-3}	2.7
1024	2.08×10^{-4}	-0.3	1.91×10^{-4}	2.0	4.12×10^{-5}	4.4	6.46×10^{-4}	3.5
2048	2.09×10^{-4}	-0.0	1.42×10^{-4}	0.4	4.54×10^{-5}	-0.1	7.99×10^{-5}	3.0
4096	2.08×10^{-4}	0.0	1.36×10^{-4}	0.1	4.87×10^{-5}	-0.1	3.79×10^{-5}	1.1
8192	2.08×10^{-4}	0.0	1.35×10^{-4}	0.0	4.88×10^{-5}	-0.0	3.43×10^{-5}	0.1

TABLE A.14. *Grid convergence for Robin Case 2 with BC1* (4.4).

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	1.51×10^{-1}		5.95×10^{-2}		2.07×10^{-1}		1.51×10^{-1}	
64	1.22×10^{-1}	0.3	8.75×10^{-2}	-0.6	7.03×10^{-2}	1.6	3.36×10^{-2}	2.2
128	1.16×10^{-1}	0.1	9.53×10^{-2}	-0.1	3.44×10^{-2}	1.0	2.20×10^{-2}	0.6
256	1.14×10^{-1}	0.0	9.73×10^{-2}	-0.0	2.72×10^{-2}	0.3	2.79×10^{-2}	-0.3
512	1.14×10^{-1}	0.0	9.78×10^{-2}	-0.0	2.58×10^{-2}	0.1	2.97×10^{-2}	-0.1
1024	1.14×10^{-1}	0.0	9.79×10^{-2}	-0.0	2.55×10^{-2}	0.0	3.02×10^{-2}	-0.0
2048	1.14×10^{-1}	0.0	9.80×10^{-2}	-0.0	2.54×10^{-2}	0.0	3.03×10^{-2}	-0.0
4096	1.14×10^{-1}	0.0	9.80×10^{-2}	-0.0	2.54×10^{-2}	0.0	3.03×10^{-2}	-0.0
8192	1.14×10^{-1}	0.0	9.80×10^{-2}	-0.0	2.54×10^{-2}	0.0	3.03×10^{-2}	-0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	5.28×10^{-1}		4.66×10^{-1}					
64	1.93×10^{-1}	1.5	1.77×10^{-1}	1.4	5.17×10^{-1}		4.64×10^{-1}	
128	5.87×10^{-2}	1.7	5.02×10^{-2}	1.8	1.88×10^{-1}	1.5	1.79×10^{-1}	1.4
256	2.02×10^{-2}	1.5	1.36×10^{-2}	1.9	5.52×10^{-2}	1.8	5.30×10^{-2}	1.8
512	1.10×10^{-2}	0.9	7.81×10^{-3}	0.8	1.66×10^{-2}	1.7	1.49×10^{-2}	1.8
1024	9.09×10^{-3}	0.3	7.89×10^{-3}	-0.0	6.78×10^{-3}	1.3	5.36×10^{-3}	1.5
2048	8.66×10^{-3}	0.1	8.04×10^{-3}	-0.0	4.50×10^{-3}	0.6	3.45×10^{-3}	0.6
4096	8.56×10^{-3}	0.0	8.09×10^{-3}	-0.0	3.98×10^{-3}	0.2	3.12×10^{-3}	0.1
8192	8.54×10^{-3}	0.0	8.10×10^{-3}	-0.0	3.86×10^{-3}	0.0	3.05×10^{-3}	0.0
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	1.56		1.11					
64	1.01	0.6	8.02×10^{-1}	0.5	1.53		1.27	
128	5.16×10^{-1}	1.0	4.61×10^{-1}	0.8	1.03	0.6	8.68×10^{-1}	0.5
256	1.85×10^{-1}	1.5	1.78×10^{-1}	1.4	5.13×10^{-1}	1.0	4.58×10^{-1}	0.9
512	5.35×10^{-2}	1.8	5.26×10^{-2}	1.8	1.84×10^{-1}	1.5	1.77×10^{-1}	1.4
1024	1.50×10^{-2}	1.8	1.46×10^{-2}	1.8	5.25×10^{-2}	1.8	5.20×10^{-2}	1.8
2048	5.04×10^{-3}	1.6	4.65×10^{-3}	1.7	1.42×10^{-2}	1.9	1.41×10^{-2}	1.9
4096	2.59×10^{-3}	1.0	2.24×10^{-3}	1.1	4.21×10^{-3}	1.8	4.11×10^{-3}	1.8
8192	2.01×10^{-3}	0.4	1.69×10^{-3}	0.4	1.70×10^{-3}	1.3	1.60×10^{-3}	1.4

TABLE A.15. *Grid convergence for Robin Case 2 with BC2* (4.22).

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	9.54×10^{-2}		4.57×10^{-2}		2.98×10^{-2}		1.87×10^{-2}	
64	8.63×10^{-2}	0.1	4.01×10^{-2}	0.2	2.24×10^{-2}	0.4	1.25×10^{-2}	0.6
128	8.24×10^{-2}	0.1	3.75×10^{-2}	0.1	1.93×10^{-2}	0.2	9.76×10^{-3}	0.4
256	8.06×10^{-2}	0.0	3.67×10^{-2}	0.0	1.78×10^{-2}	0.1	8.52×10^{-3}	0.2
512	7.97×10^{-2}	0.0	3.64×10^{-2}	0.0	1.71×10^{-2}	0.1	7.92×10^{-3}	0.1
1024	7.92×10^{-2}	0.0	3.62×10^{-2}	0.0	1.67×10^{-2}	0.0	7.63×10^{-3}	0.1
2048	7.90×10^{-2}	0.0	3.61×10^{-2}	0.0	1.66×10^{-2}	0.0	7.49×10^{-3}	0.0
4096	7.89×10^{-2}	0.0	3.61×10^{-2}	0.0	1.65×10^{-2}	0.0	7.42×10^{-3}	0.0
8192	7.89×10^{-2}	0.0	3.60×10^{-2}	0.0	1.64×10^{-2}	0.0	7.38×10^{-3}	0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	1.56×10^{-2}		4.95×10^{-3}		1.14×10^{-2}		8.47×10^{-3}	
64	9.30×10^{-3}	0.7	6.16×10^{-3}	-0.3	6.28×10^{-3}	0.9	4.14×10^{-3}	1.0
128	6.33×10^{-3}	0.6	3.78×10^{-3}	0.7	3.49×10^{-3}	0.8	2.39×10^{-3}	0.8
256	4.96×10^{-3}	0.4	2.67×10^{-3}	0.5	2.14×10^{-3}	0.7	1.35×10^{-3}	0.8
512	4.30×10^{-3}	0.2	2.15×10^{-3}	0.3	1.49×10^{-3}	0.5	8.50×10^{-4}	0.7
1024	3.98×10^{-3}	0.1	1.90×10^{-3}	0.2	1.18×10^{-3}	0.3	6.12×10^{-4}	0.5
2048	3.82×10^{-3}	0.1	1.79×10^{-3}	0.1	1.02×10^{-3}	0.2	5.02×10^{-4}	0.3
4096	3.74×10^{-3}	0.0	1.74×10^{-3}	0.0	9.42×10^{-4}	0.1	4.52×10^{-4}	0.2
8192	3.70×10^{-3}	0.0	1.71×10^{-3}	0.0	9.04×10^{-4}	0.1	4.28×10^{-4}	0.1
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	1.06×10^{-2}		1.31×10^{-2}		1.06×10^{-2}		1.31×10^{-2}	
64	5.35×10^{-3}	1.0	3.97×10^{-3}	1.7	5.19×10^{-3}	1.0	6.06×10^{-3}	1.1
128	2.81×10^{-3}	0.9	1.95×10^{-3}	1.0	2.60×10^{-3}	1.0	1.91×10^{-3}	1.7
256	1.49×10^{-3}	0.9	1.04×10^{-3}	0.9	1.33×10^{-3}	1.0	9.33×10^{-4}	1.0
512	8.45×10^{-4}	0.8	5.50×10^{-4}	0.9	6.90×10^{-4}	1.0	4.74×10^{-4}	1.0
1024	5.27×10^{-4}	0.7	3.13×10^{-4}	0.8	3.71×10^{-4}	0.9	2.43×10^{-4}	1.0
2048	3.69×10^{-4}	0.5	1.98×10^{-4}	0.7	2.14×10^{-4}	0.8	1.28×10^{-4}	0.9
4096	2.91×10^{-4}	0.3	1.44×10^{-4}	0.5	1.35×10^{-4}	0.7	7.16×10^{-5}	0.8
8192	2.53×10^{-4}	0.2	1.19×10^{-4}	0.3	9.61×10^{-5}	0.5	4.46×10^{-5}	0.7

TABLE A.16. *Grid convergence for Robin Case 3 with BC1* (4.4).

n	E_{DDM1}	k	E_{DDM2}	k	E_{DDM1}	k	E_{DDM2}	k
$\epsilon = 0.800$					$\epsilon = 0.400$			
32	1.43×10^{-1}		3.54×10^{-2}		1.70×10^{-1}		1.02×10^{-1}	
64	1.02×10^{-1}	0.5	4.94×10^{-2}	-0.5	6.33×10^{-2}	1.4	2.69×10^{-2}	1.9
128	8.88×10^{-2}	0.2	5.37×10^{-2}	-0.1	3.22×10^{-2}	1.0	9.21×10^{-3}	1.5
256	8.42×10^{-2}	0.1	5.74×10^{-2}	-0.1	2.32×10^{-2}	0.5	8.56×10^{-3}	0.1
512	8.23×10^{-2}	0.0	7.05×10^{-2}	-0.3	2.04×10^{-2}	0.2	8.88×10^{-3}	-0.1
1024					1.94×10^{-2}	0.1	9.03×10^{-3}	-0.0
2048					1.90×10^{-2}	0.0	9.15×10^{-3}	-0.0
4096					1.89×10^{-2}	0.0	9.22×10^{-3}	-0.0
$\epsilon = 0.200$					$\epsilon = 0.100$			
32	4.00×10^{-1}		8.56×10^{-1}		9.52×10^{-1}		8.39×10^{-1}	
64	1.42×10^{-1}	1.5	1.20×10^{-1}	2.8	3.84×10^{-1}	1.3	3.58×10^{-1}	1.2
128	4.41×10^{-2}	1.7	3.36×10^{-2}	1.8	1.31×10^{-1}	1.5	1.23×10^{-1}	1.5
256	1.65×10^{-2}	1.4	9.86×10^{-3}	1.8	3.79×10^{-2}	1.8	3.42×10^{-2}	1.8
512	8.96×10^{-3}	0.9	4.26×10^{-3}	1.2	1.18×10^{-2}	1.7	9.62×10^{-3}	1.8
1024	6.81×10^{-3}	0.4	3.19×10^{-3}	0.4	4.84×10^{-3}	1.3	3.39×10^{-3}	1.5
2048	6.14×10^{-3}	0.1	3.00×10^{-3}	0.1	2.98×10^{-3}	0.7	1.89×10^{-3}	0.8
4096	5.90×10^{-3}	0.1	2.97×10^{-3}	0.0	2.45×10^{-3}	0.3	1.54×10^{-3}	0.3
$\epsilon = 0.050$					$\epsilon = 0.025$			
32	2.34		1.86		5.26×10^{-0}		3.63×10^{-0}	
64	9.48×10^{-1}	1.3	8.89×10^{-1}	1.1	2.34×10^{-0}	1.2	2.09×10^{-0}	0.8
128	3.77×10^{-1}	1.3	3.65×10^{-1}	1.3	9.46×10^{-1}	1.3	9.16×10^{-1}	1.2
256	1.27×10^{-1}	1.6	1.24×10^{-1}	1.6	3.74×10^{-1}	1.3	3.68×10^{-1}	1.3
512	3.54×10^{-2}	1.8	3.39×10^{-2}	1.9	1.25×10^{-1}	1.6	1.24×10^{-1}	1.6
1024	9.96×10^{-3}	1.8	9.17×10^{-3}	1.9	3.43×10^{-2}	1.9	3.36×10^{-2}	1.9
2048	3.34×10^{-3}	1.6	2.85×10^{-3}	1.7	9.18×10^{-3}	1.9	8.85×10^{-3}	1.9
4096	1.61×10^{-3}	1.1	1.27×10^{-3}	1.2	2.70×10^{-3}	1.8	2.51×10^{-3}	1.8

TABLE A.17. *Grid convergence for Robin Case 3 with BC2* (4.22).