Towards Accurate Modeling of Moving Contact Lines

HANNA HOLMGREN

UPPSALA UNIVERSITY
Department of Information Technology
Towards Accurate Modeling of Moving Contact Lines

Hanna Holmgren
hanna.holmgren@it.uu.se

November 2015

Division of Scientific Computing
Department of Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

http://www.it.uu.se/

Dissertation for the degree of Licentiate of Technology in Scientific Computing with specialization in Numerical Analysis

© Hanna Holmgren 2015
ISSN 1404-5117
Printed by the Department of Information Technology, Uppsala University, Sweden
Abstract

The present thesis treats the numerical simulation of immiscible incompressible two-phase flows with moving contact lines. The conventional Navier–Stokes equations combined with a no-slip boundary condition leads to a non-integrable stress singularity at the contact line. The singularity in the model can be avoided by allowing the contact line to slip. Implementing slip conditions in an accurate way is not straightforward and different regularization techniques exist where ad-hoc procedures are common. This thesis presents the first steps in developing the macroscopic part of an accurate multiscale model for a moving contact line problem in two space dimensions. It is assumed that a micro model has been used to determine a relation between the contact angle and the contact line velocity. An intermediate region is introduced where an analytical expression for the velocity field exists, assuming the solid wall is perfectly flat. This expression is used to implement boundary conditions for the moving contact line, at the macroscopic scale, along a fictitious boundary located a small distance away from the physical boundary. Model problems where the shape of the interface is constant throughout the simulation are introduced. For these problems, experiments show that the errors in the resulting contact line velocities converge with the grid size $h$ at a rate of convergence $p \approx 2$. Further, an analytical expression for the velocity field in the intermediate region for the case with a curved solid wall is derived. The derivation is based on perturbation analysis.
Acknowledgments

First, I would like to thank my main advisor Gunilla Kreiss for your guidance, advice and encouragement. I also want to thank Martin Kronbichler for all the help, advices, discussions, and collaboration, it has been very valuable. I thank Ababacar Diagne for the collaboration regarding Paper III. Further, I gratefully acknowledge all fundings for my work and travels, particularly the Swedish Research Council. Furthermore, I am very grateful for all my wonderful colleagues and friends at TDB. I want to especially mention Josefin Ahlkrona, Lina Meinecke, Fredrik Hellman, Martin Almquist and Sloboden Milovanovic. Also, thank you Siyang Wang for being a very kind office mate who is always willing to help, and for a lot of interesting discussions about everything. Finally, the work with this thesis would not have been possible without the support from Kristoffer Wahlberg, my great friend Anna Hellbom and my family: Sverker, Karolina, Johan, David, Stina, Felix, and Viggo.
List of Papers

This thesis is based on the following papers

   Contributions: Implementation (by modification of the existing two-phase flow solver presented in Paper III), experiments and writing of the manuscript. Ideas were developed in close cooperation between both authors.

   Contributions: Calculations and experiments. Ideas and manuscript were developed in close cooperation between both authors.

   Contributions: Contributed to the writing of the manuscript and to performing simulations. By applying the solver presented in this paper to the problem presented in Paper I, the author of this thesis contributed by finding errors and consequently improving parts of the solver. The author of this thesis worked mostly with the level-set-part of this solver.
Chapter 1

Introduction

In scientific computing mathematical models are constructed to perform computer simulations with the purpose to answer scientific and engineering questions. These simulations complement experiments and theory and are considered the third pillar of science. The availability of computer simulations has broaden what can be studied without expensive, cumbersome and possibly unethical experiments. One of the largest fields in scientific computing is computational fluid dynamics (CFD), which is used to understand and predict the behavior of fluids.

An important area in CFD is the modeling of two-phase immiscible flows. These are flows including two different kind of fluids that do not mix while forming an interface separating the two fluids. Examples of two-phase flow phenomena are oil drops in water or ocean waves (where the air is a fluid). Two-phase flow play an important role in for example process, oil and energy industry. Numerical simulations are a fundamental tool and accurate predictions are of growing interest. The challenges when modeling two-phase flows include keeping track of the interface between the two fluids (and possible topological changes) and dealing with surface tension. In Section 2 an overview of different models for the simulation of two phase-flows is given. In Paper I we use the so-called level set method [1, 2] to track the fluid interface. This method is explained in more detail in Section 2.2.1 and Paper I.

This thesis treats the numerical simulation of immiscible incompressible two-phase flows where the fluid-fluid interface is in contact with a solid wall. When the interface is moving over the solid wall, we have a moving contact line problem. The contact line is formed at the intersection of the fluid-fluid interface and the wall. Figure 1.1 illustrates a contact line problem in two space dimensions, where the contact line is reduced to a contact point.
Moving contact line problems appear in many natural and industrial processes, including any form of droplet spreading on a solid surface, coating flows and displacement flows [3, 4]. Examples from the everyday life are raindrops falling on a window or water bugs resting on water surfaces. In the case when the dynamics of the contact line are driving the flow the physical effects that occur at the contact line affect the overall behavior of the system [5]. This is the case in for example wetting or capillary driven flows. Industrial applications where the contact line behavior is important include gas and oil recovery in porous media, lubrication, inkjet printing, biological flows and microfluidics such as micropumps and so called lab-on-a-chip devices [6, 7, 8, 9, 10, 11]. The inkjet device consists of a long ink channel with a tip that ejects liquid drops. With numerical simulations quantities such as the drop size and the eJECTING speeds can be predicted [10]. A lab-on-a-chip device can be used to directly diagnose a patient for different diseases by analyzing a single drop of blood, instead of sending a blood sample to a laboratory [12]. These devices involve the flow of different kinds of fluids in miniature channels, and simulations can be used to investigate effects of changing the channel geometry or physical parameters.

Despite their importance, the physics of moving contact lines are poorly understood [3, 6]. Atomistic processes at the contact line come into play, and their influence is difficult to model. The standard Navier–Stokes model is singular when moving contact lines are present. An option is to use models where atomistic scales can be accounted for. Such alternatives include molecular dynamics (MD) or phenomenological based phase field methods (described in Section 2.2). However, it is not realistic to resolve the atomistic processes in a macroscopic domain. Instead, the singularity in the standard Navier–Stokes model can be removed by modifying the model to
take phenomena on microscopic length scales into account.

Section 3 discusses contact line problems further and gives a short overview of existing moving contact line models to overcome the difficulty with the singularity. Paper I presents the first steps in deriving a new accurate macroscale contact line model. Paper II formulates an analytical expression for the velocity field close (on a macroscopic length scale) to a contact line that moves steadily over a curved wall in two space dimensions. The analytical velocity field can be used to further develop the contact line model presented in Paper I to account for curved solid walls.
Chapter 2

Two-Phase Flow Models

In this section we consider a domain $\Omega$ occupied by two immiscible incompressible fluids separated by an interface $\Gamma$, see Figure 2.1. The density and (dynamic) viscosity of the first fluid are denoted $\rho_1$ and $\mu_1$ respectively and the domain occupied by this fluid is denoted $\Omega_1$, and accordingly for the second fluid.

![Figure 2.1: Model domain $\Omega$. Fluid 1 with density $\rho_1$ and viscosity $\mu_1$ occupies $\Omega_1$ and fluid 2 with density $\rho_2$ and viscosity $\mu_2$ occupies $\Omega_2$. The separating interface is represented by $\Gamma$.](image)

2.1 Navier–Stokes Equations

The separate motion of each fluid is given by the incompressible Navier–Stokes equations for the fluid velocity $\mathbf{u}$ and pressure $p$

$$\rho \left( \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \nabla \cdot (2\mu \nabla^{s} \mathbf{u}) + \mathbf{f}, \quad (2.1)$$

$$\nabla \cdot \mathbf{u} = 0. \quad (2.2)$$
The rank-2 tensor $\nabla^s u = \frac{1}{2}(\nabla u + \nabla u^T)$ denotes the rate of deformation tensor and the term $f$ represents external forces.

At the interface between the two immiscible fluids surface tension forces act in order to drive the interface into a state of minimum surface energy. Therefore, coupling conditions at the interface $\Gamma$ are needed

\[
\begin{align*}
[u]_\Gamma &= 0 & \text{-- continuity of velocity,} \\
[2\mu \nabla^s u \cdot n]_\Gamma \cdot t &= 0 & \text{-- continuity of shear stress,} \\
[-p n + 2\mu \nabla^s u \cdot n]_\Gamma &= \sigma \kappa n & \text{-- balance of normal stress and surface tension,}
\end{align*}
\]

where $[u]_\Gamma$ represents the jump of the quantity $u$ from $\Omega_1$ to $\Omega_2$, $n$ and $t$ denote normal and tangential directions to the interface $\Gamma$, $\sigma$ controls the magnitude of surface tension and $\kappa$ is the interface curvature.

Instead of solving equations (2.1)–(2.2) and the coupling conditions (2.3)–(2.5) twice, once for each fluid, the Navier–Stokes equations can be reformulated to be valid in the whole domain $\Omega$. In this formulation the effect of surface tension is modeled by adding a forcing to the momentum equation (2.1)

\[
f_{st} = \sigma \kappa n \delta_\Gamma,
\]

where the delta function $\delta_\Gamma$ localizes the force to the interface $\Gamma$. For example, $\delta_\Gamma$ localizes a smooth function $v$ by

\[
\int_\Omega \delta_\Gamma v \, d\Omega = \int_\Gamma v \, d\Gamma.
\]

The Navier–Stokes equations are completed by a divergence-free initial condition for the fluid velocity $u(\cdot, 0)$. At a fixed solid wall a no-slip boundary condition that prescribes a zero fluid velocity is often used.

A so-called creeping flow occurs if the viscous effects dominate the convection. The convection term and the time derivative on the left hand side of the momentum equation (2.1) can then be neglected and the non-linear Navier–Stokes equations reduce to the linear Stokes equations (under the assumption of linear viscosity).

### 2.2 Interface Representation

Two-phase flow models require a method for tracking and evolving the fluid interface in time. Several interface representation techniques have been proposed during the last decades. A historical overview of models and methods is given for example in [13].
2.2. Interface Representation

The two main classes of two-phase flow models are interface tracking and interface capturing methods. Interface tracking methods use an explicit representation of the interface. Examples are front-tracking methods [14, 15, 16] where a set of marker points are used to explicitly represent the interface, which is evolved by advecting each marker point by the local fluid velocity. Often, the fluid velocity is obtained by solving the Navier–Stokes equations on an Eulerian grid and the velocities at the marker points are then obtained by interpolation. After deformation of the interface the marker points will not be evenly distributed, for example in some parts there may be clustering of markers. Therefore, redistribution of the markers is necessary to retain accurate interface representations. Further, tracking methods suffer from unphysical volume changes of the two fluid phases. However, since a large number of marker points can be used a good accuracy of the method is possible.

In interface capturing methods the interface is represented implicitly by a higher dimensional function (compared to the interface). Interface capturing methods therefore provide a natural mechanism for handling topological changes such as merging or breaking of interfaces. In the volume-of-fluid (VOF) method [17], the so-called fluid function keeps track of the volume fraction of each fluid in each grid cell. The cells will have a volume fraction between zero and one depending on the volume fraction of the first fluid on the total fluid. The interface is usually advected by increasing or decreasing the volume fraction in each cell according to the velocity field and the composition of neighboring cells. A reconstruction of the interface from the volume fractions is required for the evaluation of surface tension forces. These reconstruction techniques use approximations by for example piecewise constants [17], piecewise linear functions [18] or splines [19]. The volume-of-fluid methods conserve mass well. However, the reconstruction of the interface makes them more complicated to implement compared to the front tracking methods for example.

Other examples of interface capturing methods are level set [1, 2] and phase field methods [20]. These methods use continuous functions to represent the interface (as opposed to the volume-of-fluid method where discrete representation is used). The level set method is used in Paper I and this method is presented in some more details in the following section and in Paper I.

In phase field methods the interface is modeled to have a finite thickness, i.e. a diffuse interface. The smoothed region between the different fluids is described in a thermodynamically consistent way, the idea comes from the van der Waals hypothesis that immiscible fluids do mix at a molecular level. The modeled thickness of the smoothed region is however usually much larger than it is in real systems. To update the so-called phase function, which
identifies the different fluids, a continuum advection-diffusion equation is used. This equation originates from the Cahn-Hilliard equation [21], which models the seeking of minimal energy of the physical system. A drawback with the phase field method is that since the propagation and properties of the interface depend sensitively on the dynamics in the smoothed region, a high resolution is required [22].

2.2.1 Level Set Methods

Level set methods make use of a continuous level set function for representing the interface. The level set function \( \phi(x) \) is defined on the whole domain \( \Omega \) (Figure 2.1), and the fluid interface \( \Gamma \) is defined as the zero level set of \( \phi \). In the standard level set method [23] used in Paper I \( \phi \) is a signed distance function

\[
\phi(x) = \begin{cases} 
\text{dist}(x, \Gamma) & \text{in the first fluid}, \\
-\text{dist}(x, \Gamma) & \text{in the second fluid}.
\end{cases}
\]  

(2.8)

The expression \( \text{dist}(x, \Gamma) \) represents the shortest distance from the position \( x \) to the interface \( \Gamma \). Figure 2.2 shows the signed distance function \( \phi \) for the case of a two-dimensional circular fluid interface. The black contour illustrates the zero level set, i.e. the interface.

![Standard Level Set Method](image)

Figure 2.2: The signed distance function (in color) to the circular interface (the black contour).
2.2. Interface Representation

The interface is evolved by advection of the level set function with the local fluid velocity according to a Hamilton–Jacobi equation

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0.$$  \hfill (2.9)

For evaluation of the surface tension force (2.6) the calculation of normal vectors and curvature of the interface is required. The level set methods allow for straight-forward calculations by

$$\mathbf{n} = \frac{\nabla \phi}{|\nabla \phi|},$$

$$\kappa = -\nabla \cdot \mathbf{n}.$$

During the simulation the level set function will lose its signed distance property due to discretization errors and non-uniform velocity fields. To smooth the level set function and prevent the formation of large gradients, $\phi$ has to be restored regularly to take the form of a signed distance function. When $\phi$ is a signed distance function it has the property $|\nabla \phi| = 1$. The standard way to restore, or reinitialize, the level set function is to solve the following equation to steady state [2]

$$\frac{\partial \phi}{\partial \tau} + \text{sign}(\phi_0)(|\nabla \phi| - 1) = 0,$$  \hfill (2.10)

where $\phi_0$ is the level-set function before reinitialization and $\tau$ is a pseudo time. Often a regularized sign-function that smoothly changes sign from +1 in the first fluid to −1 in the second fluid is used.

The standard level set method often suffers from unphysical volume changes. Several methods exist to improve mass conservation. One example is the so-called conservative level set method [24, 25], which uses a smoothed color function that change sign from +1 to −1 over the interface instead of a signed distance function. Further, the reinitialization is posed as a conservation law where $\int_{\Omega} \phi \, d\mathbf{x}$ is preserved. There also exists hybrid methods, such as the combined level set–VOF method [26], to improve the mass conservation of the standard level set method.
Chapter 3

Contact Lines

When an interface separating two immiscible fluids is in contact with a solid wall, a contact line is formed at the intersection of the interface and the solid. In two space dimensions the contact line is reduced to a contact point (as illustrated in Figure 1.1). For the case when the fluids are at rest, the physics of the contact line has been understood for a long time [6]. The static contact angle \( \theta_s \), which is formed between the fluid interface and the solid, depends on the surface tension coefficient at the interface \( \sigma \) and the surface tension coefficients between the solid and the two fluids \( \sigma_1 \) and \( \sigma_2 \) respectively. Minimizing the total surface energy leads to the Young’s relation for the static contact angle \( \theta_s \)

\[
\sigma_2 - \sigma_1 = \sigma \cos \theta_s.
\]  

When the fluids are not at rest and the contact line moves, the situation is more complicated. The standard Navier–Stokes equations together with the conventional no-slip boundary condition predicts a singularity in the shear stresses at the contact line [27, 28]. In fact, molecular dynamics simulations show that there must be some fluid-wall slip in the region close (in an atomistic length scale) to the contact line [29, 30]. Therefore, microscopic length scales must be introduced into the problem [31]. Further, in the case of wetting or capillary driven flows, it has been observed that the microscopic contact line dynamics have a large impact on the overall macroscopic flow. However, the physics behind this interaction is not completely understood [8, 27, 5].

As mentioned in Section 2.2, the phase field method takes molecular processes at the interface (including the contact line) into account by diffusion [20]. Using this model together with the Navier–Stokes equations and a no-slip boundary condition avoids the stress singularity. The contact line moves by the diffusion included in the interface model. However, unrealistic
model parameters are often introduced leading to unphysical contact line dynamics [22]. Also, as mentioned in Section 2.2 a high resolution is required and using the phase field method for simulation of moving contact lines is regarded computationally demanding.

In simulations using the standard Navier–Stokes model together with a non-diffuse interface advection (e.g. the level set method) the stress singularity can be avoided if the contact line is allowed to slip. Several models for introducing slip have been proposed [8, 5, 6]. It is common to allow the contact line to move by using the Navier slip condition,

\[ \mathbf{u}_{\text{slip}} = \lambda \mathbf{n} \cdot (\nabla \mathbf{u} + (\nabla \mathbf{u})^T), \]

(3.2)

where the amount of slip is related to the shear stress. The parameter \( \lambda \) is the slip length and \( \mathbf{n} \) is the normal vector to the solid boundary.

For capillary and wetting flows, the slip boundary conditions for the velocity must be accompanied by models for contact line dynamics. It has been observed both in experiments and in simulations that the deviation of the contact angle from the static contact angle is important for the contact line motion [32, 33, 30, 34]. With this motivation there are models where the contact angle is explicitly prescribed. Several relations have been suggested for the implementation of the contact angle value [35, 36, 37, 38, 39]. The simplest approach is to prescribe the contact angle to be the static contact angle [39]. The angle can also be imposed according to experiments [37]. However, a singularity in the pressure remains at the contact line if a Navier condition for the velocity is used together with this approach. Further, it is still debated what is the correct way of prescribing the contact angle value [13].

### 3.1 Multiscale and Macroscale Models

To perform macroscale simulations of the overall flow dynamics of capillary or wetting flow, there are other alternatives than using a slip condition together with a prescribed contact angle. One approach is to construct velocity boundary conditions that not only allow for slip, but also approximate the effect of the microscopic region near the contact line. These methods are often based on models relating the contact line velocity to the contact angle by

\[ u_{CL} = f(\theta) \]

(3.3)

where \( u_{CL} \) is the contact line velocity. In [34] molecular dynamic simulations are used to derive such a relation. Further, microscopic models based on diffusion or semi-analytical models can be used [40, 41, 42]. In [34] the
3.1. Multiscale and Macroscale Models

relation (3.3) is imposed as a velocity boundary condition at the contact line, away from the contact line the Navier slip condition is used.

An alternative approach is the application of multiscale models where macroscopic models are coupled to detailed microscale models. One example is the multiscale model in [43] where two domains are used, one for the bulk flow and one close to the contact line. The two domains overlap and in each domain, the result from the other domain is imposed as boundary conditions. Results in each domain are computed using several iterations until the two models converge in the shared region. A more efficient way is provided by the heterogeneous multiscale method (HMM) [44].

Even though the contact line dynamics can be prescribed by a relation according to (3.3) it is not clear how this condition, which is only valid at the contact line, should be imposed as a velocity boundary condition along the whole solid wall. Often ad-hoc regularizations are used. In [45] for example, a slip profile that decays exponentially to no-slip far from the contact line is implemented. Common problems when imposing the condition (3.3) as boundary conditions in numerical simulations are grid effects and low accuracy. In [41] and [46], for example, only first-order accuracy of the resulting flow has been observed. Paper I presents the first steps in developing an accurate model to implement the relation (3.3) in the form of a velocity boundary condition for the Navier–Stokes model. The method is based on the so-called hydrodynamic model for steady movement of a contact line, introduced in [27]. The hydrodynamic model consists of an analytical expression for the fluid velocity field close to a moving contact line. This expression is derived from the creeping flow approximation of the Navier–Stokes equations together with appropriate boundary and interface conditions. In Paper I the velocity field from the hydrodynamic model is used to prescribe the velocity along a fictitious boundary located a small distance away from the physical solid boundary. There is no singularity, and the resulting velocity field advects the contact line at the correct rate. The approach in Paper I can be used both in macroscale models and in multiscale models where the Navier–Stokes equations are combined with a microscale model, from which the relation (3.3) is determined.
Chapter 4

Discretization and Implementation

Discrete approximations of the solution to the Navier–Stokes equations and the solution to the equation for interface advection are required for macroscopic numerical simulation of moving contact line problems. Both spatial and temporal discretizations are necessary. The most popular methods for discretization in space are the finite volume method, the finite difference method and the finite element method. In this thesis the finite element method is used.

4.1 Space Discretization by the Finite Element Method

The first step in the finite element approximation is to rewrite the equations on variational form. The variational form of the Navier–Stokes equations (2.1)–(2.2) reads: find \( u \in V_u \) and \( p \in V_p \) such that

\[
\left( v, \rho \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right) \right)_\Omega = \left( \nabla \cdot v, p \right)_\Omega - \left( \nabla^s v, 2\mu \nabla^s u \right)_\Omega + (v, f)_\Omega
\]

\[
(q, \nabla \cdot v)_\Omega = 0
\]

for all \( v \in V_u \) and \( q \in V_p \). Here \((\cdot, \cdot)_\Omega\) is the standard \(L_2\) inner product on \(\Omega\) and \(V_p = L_2(\Omega)\). \(V_u\) is the space of all square integrable vector-valued functions on the domain \(\Omega\) with integrable first derivatives that satisfy the boundary conditions, i.e. \(V_u = \{ u \in H^1(\Omega); u = 0 \text{ on } \partial\Omega \}\). For simplicity, the theory only for no-slip boundary conditions is presented, for additional forms for other types of conditions see [47].

In this thesis the level set method is used for interface advection. The variational formulation for the level set equation (2.9) reads: Find \( \phi \in V_\phi = \)
$H^1(\Omega)$ such that
\[
\left( \eta, \frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi \right)_\Omega = 0
\] (4.3)
for all $\eta \in V_\phi$.

In the next step, the solution function spaces $V_u, V_p$ and $V_\phi$ are replaced by finite-dimensional discrete subspaces $V_u^h, V_p^h$ and $V_\phi^h$. In the work presented here, the computational domain is decomposed into quadrilateral elements of size $h$. Further, the basis functions that span $V_u^h$ are piecewise quadratic polynomials and the functions that span $V_p^h$ and $V_\phi^h$ are piecewise linear polynomials. The basis functions are nonzero in exactly one node in the domain, and zero at all other nodes. The finite element method (without stabilization) for the Navier–Stokes equations does not allow for arbitrary pairings of basis functions for the velocity and pressure spaces due to the Babuška–Brezzi (inf–sup) condition [48]. This is the reason for the difference in the degrees of the polynomials that span the discrete velocity and pressure spaces. The discrete approximations $\mathbf{u}^h, p^h$ and $\phi^h$ are on the form
\[
\mathbf{u}^h = \sum_{j=1}^{N_u} U_j v_j^h, \quad p^h = \sum_{j=1}^{N_p} P_j q_j^h, \quad \phi^h = \sum_{j=1}^{N_\phi} \Phi_j \eta_j^h.
\] (4.4)
where the coefficients $U_j, P_j$ and $\Phi_j$ are to be determined and the functions $v^h, q^h$ and $\eta^h$ span the spaces $V_u^h, V_p^h$ and $V_\phi$ respectively. Now, the finite element approximation of the Navier–Stokes equations is to find the discrete velocity $\mathbf{u}^h \in V_u^h$ and pressure $p^h \in V_p^h$ such that a variational formulation of the form (4.1)–(4.2) is fulfilled for all test functions $v^h \in V_u^h$ and $q^h \in V_p^h$. Similarly, the level set equation is discretized by finding the discrete level set function $\phi^h \in V_\phi^h$ such that a variational formulation of the form (4.3) is fulfilled for all test functions $\eta^h \in V_\phi^h$.

### 4.2 Time Discretization

In this thesis the Navier–Stokes equation and the level set equation both are discretized in time using the implicit Backward Difference Formula of second order (BDF-2). However, to avoid an expensive coupling between the Navier–Stokes part and the level set part (via the variables $\mathbf{u}$ and $\phi$), an explicit splitting scheme between the two equations is introduced. At each time step $n$, an estimate of the level set function is extrapolated from the values at time steps $n-1$ and $n-2$. This estimate is used to evaluate the surface tension force. Then, the BDF-2 time step for the Navier–Stokes equations is performed. Finally, the level set function is propagated in time using the velocity $\mathbf{u}^n$ obtained from the Navier–Stokes step. The splitting
4.3 Implementation

The combination of the finite element method and implicit time discretization schemes for the solution of the Navier–Stokes equations result in a system of equations on saddle point form. Finding solutions to these problems are challenging. The implementation in this thesis make use of the effective solvers described in Paper III.

Instead of solving the Navier–Stokes equations using implicit time discretization, there are explicit methods where the saddle-point linear system is avoided. In these so-called fractional-step schemes the momentum equation (2.1) is first solved using a pressure that is extrapolated from old time levels. The resulting velocity is in general not divergence free, why a pressure Poisson equation with forcing according to the divergence of the intermediate step velocity is solved to correct the velocity. The two sub-problems result in systems that are considered to be of an easier structure than a saddle point problem. However, imposing boundary conditions is not straightforward and many naive implementations suffer from non-consistent approximations close to boundaries [50]. Further, in two-phase flow simulations the implementation of the case when the two fluids have different density is not straightforward and the poisson solvers are less efficient.

4.3 Implementation

In this thesis, the implementation is done using the existing two-phase flow solver described in Paper III with suitable modifications to account for moving contact lines. Additionally, the conservative level set method implemented in Paper III is changed to the standard level set method (described in Section 2.2.1) and the order in which the equations are solved in each time step is changed (so that the Navier–Stokes equations are solved before the interface is advected, as described in previous section). The solver is implemented in the C++ based finite element open source library deal.ii [51, 52].
The model presented in Paper I is restricted to model problems where the shape of the interface is constant thought the simulation. The reason to this is that the standard reinitialization of the level set function (see Section 2.2.1) does not preserve the contact line position. In order to focus on the accuracy of the velocity boundary conditions, we want to avoid errors due to reinitialization. Therefore, the model problems presented in Paper I are based on constant interface shapes. To be able to simulate problems with dynamic interface shapes, the model need to be extended with a reinitialization of the level set function that preserves the contact line position. An idea is to implement the reinitialization developed in [53] where a penalty term is introduced in the variational formulation of the reinitialization equation to preserve the interface shape. The penalty term is in the form of a line integral along the interface. A sharp implementation of this integral could be performed using so-called cut finite element methods.

Further, the model presented in Paper I should be extended to be able to handle curved solid walls. This extension is possible by using the result from Paper II. The intermediate model presented in Paper I can be changed to take the analytical expression for the velocity field derived in Paper II into account, instead of the velocity field presented in [27].
Bibliography


Towards Accurate Modeling of Moving Contact Lines

Hanna Holmgren        Gunilla Kreiss

Department of Information Technology, Uppsala University, Box 337, SE-751 05, Uppsala, Sweden

Abstract

A main challenge in numerical simulations of moving contact line problems is that the adherence, or no-slip boundary condition leads to a non-integrable stress singularity at the contact line. In this report we perform the first steps in developing the macroscopic part of an accurate multiscale model for a moving contact line problem in two space dimensions. We assume that a micro model has been used to determine a relation between the contact angle and the contact line velocity. An intermediate region is introduced where an analytical expression for the velocity exists. This expression is used to implement boundary conditions for the moving contact line at a macroscopic scale, along a fictitious boundary located a small distance away from the physical boundary.

Model problems where the shape of the interface is constant thought the simulation are introduced. For these problems, experiments show that the errors in the resulting contact line velocities converge with the grid size $h$ at a rate of convergence $p \approx 2$. 
Contents

1 Introduction ................................................. 3

2 Two Phase Flow Model ........................................ 5
   2.1 Navier–Stokes Equations .................................. 5
   2.2 The Level Set Method ...................................... 6

3 Contact Line Boundary Conditions .......................... 7
   3.1 Creeping Flow in the Intermediate Region ............ 7
   3.2 Velocity Boundary Conditions ............................ 10
      3.2.1 Choosing δ and Spatial Grid Size h ............... 12
   3.3 Interface Boundary Conditions ......................... 12

4 Discretization and Implementation .......................... 13

5 Numerical Experiments ...................................... 14
   5.1 Flat Fluid Interface with Creeping Flow Velocity Field .... 15
   5.2 Flat Fluid Interface Coupled to Navier–Stokes Equations ... 17
   5.3 Circular Fluid Interface Coupled to Navier–Stokes Equations 19

6 Summary and Conclusions .................................... 20
1 Introduction

Flow problems involving two immiscible incompressible fluids that are in contact with a solid are called moving contact line problems. The contact line is formed where the interface between the two fluids meets the solid wall. Figure 1 depicts a schematic illustration of a contact line problem in two space dimensions, where the contact line is reduced to a contact point. In capillary and wetting flows it is the dynamics of the moving contact line that drive the flow. In such cases the physical effects that occur at the contact line are important for the overall behavior of the system [1]. Examples of these phenomena are when a droplet is spreading on a solid surface or a liquid is rising in a narrow tube. Moving contact line problems form an important class of two-phase flows and appear both in nature and in many industrial applications [1]. Industrial applications where the contact line behavior is important include coating processes, lubrication, inkjet printing, biological flows and micro fluidics such as micropumps and so called lab-on-a-chip devices [2, 3, 4, 5, 6, 7, 8]. A lab-on-a-chip devices can be used to directly diagnose a patient for different diseases by analyzing one single drop of blood, instead of sending a blood sample to a laboratory [9].

![Figure 1: Schematic illustration of a moving contact line problem in two space dimensions.](image)

The moving contact line problem has been a subject of debate for many years. The physics governing the dynamics of moving contact lines is still not completely understood [4, 10, 1]. The conventional hydrodynamic theory combined with a no-slip boundary condition leads to a non-integrable stress singularity at the contact line [10, 11]. In fact, molecular dynamics simulations show that there is some sort of fluid-wall slip in the region close to the contact line [12, 13]. When performing numerical simulations of a moving contact line problem, the macroscale flow is usually the main interest. In these simulations it is too computationally demanding to resolve the problem to the microscopic scales. The singularity in the model can instead be avoided by allowing the contact line to slip [14]. A common approach
is to use a so-called Navier condition with a related slip length parameter [15, 16]. When the dynamics of the moving contact line is driving the flow only introducing slip is not enough. The effects of the microscopic dynamics of the contact line on the macroscopic flow need to be accounted for. These effects can be modeled by introducing additional conditions in the vicinity of the contact line [1]. Such conditions could for example relate the contact line velocity $U$ to the contact angle $\phi$ [1]

$$U = f(\phi).$$  

(1)

These relations are usually derived with the help of microscale simulations, such as molecular dynamics simulations [17, 18] or simulations where the phenomenologically based phase-field model is used [19].

Implementing slip conditions and relations on the form (1) in an accurate way is not straightforward. To impose slip conditions in the vicinity of the contact line different regularization techniques exist and ad-hoc procedures are common. One approach is to set the velocity in the proximity of the contact line to the given velocity from relation (1) and then let it smoothly approach the wall velocity as the distance to the contact line increases [20, 16]. Another approach is to use the Navier slip condition with a slip length parameter that is high at the contact line but reduced to zero over the next three grid points [19]. Further, to implement a contact model it is common to manipulate the representation of the interface, or the curvature of the interface, in order to follow a given contact line velocity [2, 21, 22, 23]. These procedures include using Dirichlet boundary conditions on equations describing the interface or fitting the interface so that is takes a given curvature [19]. Due to the ad-hoc regularization techniques to implement velocity boundary conditions, and/or due to the manipulation of the interface, common problems with contact line models are inaccuracies and grid effects [15, 24].

In this paper we perform the first steps in constructing a higher order accurate contact line model in two space dimensions based on multiscale modeling. Here, we focus on the macroscopic part of the model. It is assume that a micro model has been used to determine a relation between the contact angle and the contact point velocity as in (1). We focus on accurately imposing this given contact point velocity (from the micro model) at the macroscopic scale. The choice of micro model is general (as long as it relates the contact point velocity to the contact angle).

Our macroscopic model is based on introducing an intermediate scale to the problem, in a region in the vicinity of the contact point. In this region the continuum mechanics is assumed to be valid, however the following two assumptions are made:
1. The interface is essentially flat. (This is not the case if zooming in to the microscopic level where viscous bending effects are active [25]. If zooming in even further molecular phenomena also have an effect on the interface shape.)

2. The viscous effects dominate the convective and therefore the creeping flow approximation of the Navier–Stokes equations is valid.

With these assumptions the hydrodynamic model presented in [10] is valid. The model presented in [10] consists of an analytical expression for the velocity close to a contact point. This expression is here used to implement boundary conditions for the moving contact point along a fictitious boundary located a small distance away from the physical boundary.

To model the two-phase flow dynamics we use the Navier–Stokes equations coupled to the Level set method for keeping track of the fluid interface. As a part of the level set method a reinitialization step is usually required, see Section 2.2. However, we let the implementation of an accurate reinitialization be part of the future work. Here, in the first stage of developing an accurate model we restrict ourself to study the movement of an interface that has a constant shape, and where no reinitialization is needed.

Section 2 presents the equations used to model the two-phase flow dynamics. Section 3 describes the first steps in constructing the accurate macroscopic contact point model. Further, in Section 4 the discretization and implementation of the two-phase models are outlined and numerical experiments are presented in Section 5. Finally, a summary and conclusions, including future work, are given in Section 6.

2 Two Phase Flow Model

2.1 Navier–Stokes Equations

The motion of the two immiscible fluids is given by the incompressible Navier–Stokes equations for velocity $u$ and pressure $p$ in non-dimensional form,

$$\rho \frac{\partial u}{\partial t} + \rho u \cdot \nabla u = -\nabla p + \frac{1}{\text{Re}} \nabla \cdot (2\mu \nabla^s u) + F_{st},$$

$$\nabla \cdot u = 0.$$  (2)

Here, $F_{st}$ is the surface tension force at the fluid interface and Re denotes the Reynolds number, which controls the magnitude of viscous stresses. Further, $\nabla^s u = \frac{1}{2}(\nabla u + \nabla u^T)$ denotes the rate of deformation tensor and the parameters $\rho$ and $\mu$ denote the density and viscosity measured relative
to the parameters of fluid 1,

\[ \rho = \begin{cases} 1 & \text{in fluid 1,} \\ \frac{\rho_2}{\rho_1} & \text{in fluid 2,} \end{cases} \quad \mu = \begin{cases} 1 & \text{in fluid 1,} \\ \frac{\mu_2}{\mu_1} & \text{in fluid 2.} \end{cases} \]

### 2.2 The Level Set Method

The standard level set method [26] is used to keep track of the fluid interface and the moving contact line. The level set function \( \phi(x, t) \) is a signed distance function and the fluid interface \( \Gamma \) is given by the zero level set of \( \phi \). The subdomain \( \Omega_1 \) occupied by fluid 1 is given by \( \phi > 0 \) and the subdomain \( \Omega_2 \) occupied by fluid 2 is given by \( \phi < 0 \).

The level set function is advected in time by the fluid velocity according to the following Hamilton–Jacobi equation

\[
\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0.
\] (3)

After advecting the fluid interface, the surface tension force \( F_{st} \) is calculated,

\[ F_{st} = \frac{1}{\text{We}} \kappa \mathbf{n} \delta \Gamma, \] (4)

where \( \text{We} \) is the Weber number and \( \delta \Gamma \) is a Dirac delta function with support on \( \Gamma \). The normal and curvature of the interface can be computed using the level set function,

\[
\mathbf{n} = \frac{\nabla \phi}{|\nabla \phi|}, \quad \kappa = -\nabla \cdot \mathbf{n}.
\]

Over time the level set function will lose its signed distance property due to discretization errors and non-uniform velocity fields. To smooth the level set function and prevent the formation of large gradients, \( \phi \) has to be reinitialized with a regular interval. The standard way of doing this is to solve the following equations to steady state

\[
\frac{\partial \phi}{\partial \tau} = -\text{sign}(\phi_0)(|\nabla \phi| - 1),
\] (5)

where \( \phi_0 \) is the level-set function before reinitialization and \( \tau \) is a pseudo time step. In simulations, a regularized sign-function that smoothly changes sign from +1 in the first fluid to −1 in the second is often used.

Performing the reinitialization in (5) does not guarantee preservation of the interface position, and thus the contact point position. Further, unphysical volume changes in the fluid phases may occur. To prevent volume
changes the conservative level set method was developed in [27]. However, when deriving this method contact lines were not taken into account. When applying the conservative level set method to a problem where the interface intersects the wall, we found that the computation of the curvature of the fluid interface was not accurate close to the contact line. Therefore, as a first simple study case in developing an accurate contact line model, a steady interface shape is assumed throughout the whole simulation time. In each time step the contact line position is evaluated, and the level set function is then reinitialized by re-interpolation to take the form of a signed distance function depending on the contact line position, see Section 3 for details.

3 Contact Line Boundary Conditions

In this report we develop the macroscopic part of a multiscale model for a moving contact line problem in two space dimensions. To model the microscopic effects of the contact line on the macroscopic level, we assume there is a relation that gives the slip velocity at the contact line $U$ as a function of the wall contact angle $\phi$ (relation (1)). The wall contact angle $\phi$ is the angle between the interface and the wall in the macro model, see figure 2. This approach requires both a spatial and temporal scale separation between the local contact line behavior and global fluid flow [19]. When the dynamics of the moving contact line is driving the flow, the assumption that the flow on the microscopic scale is reacting much faster than the flow on the macroscale is justified. This temporal scale separation implies that the microscopic dynamics is in equilibrium for each wall contact angle at the macroscopic scale. Therefore no additional information from the macro model is required. These kind of relations between the contact line velocity and wall contact angle can be derived using micro models. Examples of such micro models are molecular dynamics simulation [17, 18] or phase field simulation [19]. The slip velocity at the contact line $U$ obtained from the microscopic relation is here used to determine velocity boundary conditions in the macroscopic model. To derive these boundary conditions an intermediate model in a region close to the contact line is introduced. In figure 2 a schematic illustration of the different scales is given.

3.1 Creeping Flow in the Intermediate Region

To derive the macroscopic boundary conditions an intermediate region of length scale $L$ is introduced; the red region in the schematic illustration in figure 2. This region is still assumed to be in the continuum region. However, at this length scale the viscous effects are assumed to dominate the convective, i.e. the Reynolds number $\text{Re} = \frac{\rho UL}{\mu} \ll 1$. Under these
conditions the creeping flow approximation of Navier–Stokes equations is valid. As illustrated in Figure 2, at this length scale the fluid interface could be assumed to have a flat shape. This assumption is not valid if you zoom in closer to the contact line, to a microscopic view. There viscous forces become very large and strongly bend the interface [25]. Additionally, even closer to the contact line molecular phenomena such as diffusive mass transport affect the interface shape.

Under the assumptions of a creeping flow approximation and a flat fluid interface shape, there exists an analytical expression for the fluid velocity field in the intermediate region. The analytical model was derived in [10] by rewriting the creeping flow approximation of Navier–Stokes equations in the form of a biharmonic equation for the stream function \( \psi(r, \theta) \) in plane polar coordinates \( r \) and \( \theta \). The origin of the polar coordinate system is fixed to the contact line position. In terms of the stream function the polar velocity components are \( v_r = -r^{-1} \frac{\partial \psi}{\partial \theta} \) and \( v_\theta = \frac{\partial \psi}{\partial r} \). By imposing appropriate boundary and interface conditions an analytical expression for the stream function in the region close to the contact line can be derived. The analytical expression depends on the wall contact angle \( 0 < \phi < 180 \), the contact line velocity \( U \) and the viscosity ratio \( Q \) and is given by

\[
\psi(r, \theta) = r(a \sin \theta + b \cos \theta + c \theta \sin \theta + d \theta \cos \theta),
\]

where the coefficients \( a, b, c \) and \( d \) for the two different fluids, denoted by 1

Figure 2: Schematic illustration of the different scales in the multiscale moving contact line model.
and 2 below, are given by

\begin{align}
    a_1 &= -U - \pi c_1 - d_1 \\
    b_1 &= -\pi d_1 \\
    c_1 &= US^2[S^2 - \gamma \phi + Q(\phi^2 - S^2)]/D \\
    d_1 &= USC[S^2 - \gamma \phi + Q(\phi^2 - S^2) - \pi \tan \phi]/D \\
    a_2 &= -U - d_2 \\
    b_2 &= 0 \\
    c_2 &= US^2[S^2 - \gamma^2 + Q(\delta \phi - S^2)]/D \\
    d_2 &= USC[S^2 - \gamma^2 + Q(\delta \phi - S^2) - Q\pi \tan \phi]/D,
\end{align}

where

\begin{align}
S &= \sin \theta \\
C &= \cos \theta \\
\gamma &= \phi - \pi \\
Q &= \mu_A/\mu_B \\
D &= (SC - \phi)(\gamma^2 - S^2) + Q(\delta - SC)(\phi^2 - S^2).
\end{align}

The resulting analytical velocity field constitutes a similarity solution since it is independent of the distance to the contact line \( r \). As mentioned above, the analytical model is not valid in the immediate neighborhood of the contact line, at the microscopic length scales. In fact the analytical velocity field is discontinuous at the contact line and shear stresses, pressure, and viscous dissipation rate increase without bound as the contact line is approached. Nevertheless, the model can be used do describe the dynamics in the intermediate region: “the model may approximate reality well in a slightly removed region where the fluid interface is substantially flat and the flow qualifies as creeping” [10].

In figure 3 the magnitude of the analytical velocity in the intermediate region (the red box in figure 2) is plotted for the case with a contact angle \( \phi = 45 \), contact velocity \( U = 1 \) and viscosity ratio \( Q = 1 \). In figure 4 the magnitude of the velocity along the three lines \( y = 0 \), \( y = 10^{-10} \) and \( y = 10^{-2} \) from the domain in figure 3 are plotted. It can be seen that the velocity is zero along the whole solid boundary, i.e. along the whole line \( y = 0 \), also at the contact line. However, the second plot in figure 4 depicts that just inside the boundary the velocity is non-zero in the vicinity of the contact point. This illustrates the velocity discontinuity at the contact point. The further away from the wall, the wider is the peak in the velocity at the contact point along a line \( y = \delta \), see the third plot in figure 4 for
example. More precisely, since the analytical velocity field constitutes a similarity solution, the width of the peak increase linearly with increasing δ.

Figure 3: The analytical velocity field in the intermediate region for φ = 45, U = 1 and Q = 1.

3.2 Velocity Boundary Conditions

The analytical model in the intermediate region described above, is here used to develop a velocity boundary condition for the macroscopic simulation. To avoid the singularity at the contact point in the analytical model, parts of the intermediate region will not be included in the macroscopic simulation. Consequently, a computational domain that is 2δ smaller in the direction perpendicular to the solid wall will be used, see figure 5. Along the new fictitious boundary, which is δ inside the physical boundary, we impose the analytical velocity from the intermediate model as a Dirichlet boundary condition for the velocity in the macroscopic model. If we use the domain in figure 5 with δ = 0.1 for example, the velocity function in figure 6 is used along the fictitious boundary. The information about the slip contact point velocity U from the microscopic model, i.e. the information about the movement of one single point, is transformed into a velocity boundary condition along the whole (fictitious) boundary.
Figure 4: The analytical velocity field in the intermediate region in figure 3 evaluated along three lines $y = 0$, $y = 10^{-10}$ and $y = 10^{-2}$.

Figure 5: Modified domain.

Figure 6: Velocity Dirichlet boundary condition derived from the analytical intermediate model.
3.2.1 Choosing $\delta$ and Spatial Grid Size $h$

When implementing the velocity boundary condition described above, care has to be taken when choosing the value of $\delta$. The creeping flow approximation is valid in a region with length scale $L$ (intermediate region) and $\delta < L$ is required. However, the smaller $\delta$ the sharper is the peak in the boundary velocity function at the contact point (figure 4). Therefore, when discretizing in space the grid must sufficiently resolve the features of the boundary function. To investigate what grid size $h$ is sufficient for a certain $\delta$, it is instructive to plot the resulting velocity function together with the corresponding discrete version of the boundary function. In figure 7 for example the resulting velocity boundary function for $\delta = 0.05$ and contact angle $\phi = 90$, contact point velocity $U = 1$ and viscosity ratio $Q = 1$ is plotted together with corresponding discrete versions where $h = 1/12$ and $h = 1/24$. There it can be seen that only the smaller grid size will be able to capture details of the peak in the boundary function. Since the width of the peak in the velocity function depends linearly on $\delta$ one knows how much the grid needs to be refined when decreasing $\delta$, assuming the limit of the grid size for a certain $\delta$ is known.

![Figure 7: Velocity boundary function and corresponding values at grid point for $h = 1/12$ and $h = 1/24$ ($\phi = 90$, $U = 1$, $Q = 1$).](image)

3.3 Interface Boundary Conditions

Forcing the interface to move according to the contact point velocity using Dirichlet boundary conditions might lead to incompatibility of the contact
line location between the velocity field and the interface representation via the level set method. Further, advecting the contact point explicitly using Dirichlet boundary conditions for the level set function might effect the mass conservation. Since the velocity boundary condition developed in previous subsection depends on the contact point velocity $U$ (given from the micro model), the interface close to the contact point should automatically be advected with that given velocity. Thus, there is no need for explicitly moving the interface using additional Dirichlet boundary conditions. Along the wall we simply use homogenous Neumann boundary conditions for the level set function. Possible oscillations in the level set function due to inflowing characteristics is smoothed during the reinitialization.

4 Discretization and Implementation

For the implementation we use the existing two-phase flow solver described in [28] with suitable modifications to account for moving contact lines. The conservative level set method implemented in MARTINBABA is also changed to the standard level set method (described in Section 2.2). Additionally, the order in which the equations are solved in each time step is changed so that the Navier–Stokes equations are solved before the interface is advected (and not the other way around as in [28]). The solver is implemented in the C++ based Finite Element open source library deal.ii [29, 30]. The equations in Section 2 are discretized in space using the Finite Element Method. For the level set function piecwise continuous linear shape functions on quadrilaterals, i.e. $Q_1$ elements, are used. For the incompressible Navier–Stokes equations we use the Taylor–Hood elements $Q_2Q_1$, i.e., shape functions of degree two for each component of the velocity and of degree one for the pressure. With these elements the Babuška–Brezzi (inf–sup) condition [31] is fulfilled in order to guarantee the existence of a solution. Finite element discretizations of equations of transport type, such as the level set equation, typically need to be stabilized. Here however, no stabilization is used since the reinitialization will take care of possible oscillations. Dirichlet boundary conditions are imposed strongly.

For time stepping, each of the level set equation and Navier–Stokes equations are discretized using the second order accurate, implicit BDF–2 scheme. In order to avoid an expensive coupling between the incompressible Navier–Stokes part and the level set part (via the variables $u$ and $\phi$) a temporal splitting scheme is introduced. In order to maintain second order accuracy in time, at each time step $n$ an estimate of the level set function is extrapolated from the values at time steps $n − 1$ and $n − 2$. This estimate is used to evaluate an approximation of the surface tension force. With this
surface tension force, the BDF-2 time step for the Navier–Stokes equations is then performed. Finally, the level set function is propagated in time, according to the velocity $u^n$ obtained from the Navier–Stokes step, again using the BDF-2 method. The splitting between the level set and Navier–Stokes parts corresponds to an explicit treatment of surface tension, which gives rise to a time step limit

$$\Delta t \leq c_1 \frac{We}{Re} h + \sqrt{\left(c_1 \frac{We}{Re} h \right)^2 + c_2 We h^3}, \quad (15)$$

where $c_1$ and $c_2$ are constants that do not depend on the mesh size $h$ or the material parameters. For more details about the time discretization see [28].

After time discretization and linearization of the Navier–Stokes equations using the implicit Newton method, linear systems need to be solved. For the level set equation, a BiCGStab solver [32] is used due to non-symmetry. The resulting system after discretization of the Navier–Stokes equations is of saddle point structure and solved by an iterative GMRES solver [32]. For preconditioning, a block-triangular operator constructed using the so called Schur complement of the block system is applied from the right [33]. Most of the iterative solvers spend the bulk of the computing time in matrix-vector products. Therefore, the fast matrix-free methods from [34, 35] based on cellwise integration are used for matrix-vector products. This enables matrix-free matrix-vector products that are up to an order of magnitude faster on $Q2$ elements. For more details about the linear solvers and matrix-free methods used, we again refer to [28].

5 Numerical Experiments

To investigate the moving contact line boundary condition developed in Section 3 we use three test problems. All test problems are performed on a two dimensional channel with a length of 10 and a height of 2, and the initial contact point position at $x = -0.5$. Further, in all simulations both fluids are assumed to have viscosity $\mu = 0.7$ and density $\rho = 1$. We start with a very simple set up in the first test problem, and proceed by adding more complexities into the two preceding test problems. At the stage of this report we do not use a specific relation between the contact angle and velocity from a specific micro model, instead we construct hypothetical examples for the relation between the contact point velocity $U$ and wall contact angle $\phi$.

For all problems the gradient of the level set function, $\nabla \phi$, gets distorted at the interface close to the contact point if no reinitialization is used. If very steep or very flat gradients are formed, the evaluation of the zero contour, the normal vectors, and the curvature accuracy becomes an issue. However,
as mentioned in Section 2.2 performing the standard reinitialization in (5) will modify the contact point position. Therefore, as a first simple study case to validate the contact line boundary conditions derived in last section, a steady interface shape is assumed throughout the whole simulation and “reinitialization” is performed in the end of each time step according to the following steps:

1. The contact point position is evaluated by cubic interpolation using the level set function values at the two degrees of freedoms closest to the contact point in each direction along the wall.

2. The level set function values in the whole domain are redefined to represent a distance function according to the constant interface shape (what shape depends on the problem set up) with contact point position according to the calculated position in the previous step.

We are interested in investigating the ability of the contact line boundary conditions to accurately advect the contact point according to the velocity $U$ obtained from the micro model. Therefore, it is important that the evaluation of the contact point position in the first step presented above is accurate, why a cubic interpolation is used. However, the procedure to modify the interface to take a fixed form introduces an error, in the end of each time step, that does not depend on the time step size $\Delta t$. To investigate spatial grid convergence the time step is therefore fixed to $\Delta t = 0.01$ for all simulations. The spatial grid sizes used are $h = 1/24, 1/32, 1/40, 1/48, 1/56$ and with the value of $\Delta t = 0.01$, the spatial discretization error is assumed to dominate the temporal error (for all spatial grid sizes).

5.1 Flat Fluid Interface with Creeping Flow Velocity Field

The first test problem consists of simple advection of a flat interface using a given velocity field (i.e. the Navier–Stokes part is not solved for). The given velocity field is the analytical velocity field from the creeping flow approximation in the intermediate model described in Section 3.1. The wall contact angle of the flat interface is taken to be $\phi = 135$ and we hypothetically relate this angle to a contact point velocity of $U = 0.02$. As explained in Section 3.2.1, when choosing $\delta$ we need to make sure $\delta < L$ where $L$ is the characteristic length scale of the intermediate region close to the contact line where the creeping flow approximation is valid. For the creeping flow to be valid we need $Re = \frac{\rho U L}{\mu} \ll 1$ or $L \ll \frac{\mu}{\rho U}$ which for this model problems implies the following condition on the distance to the physical boundary: $\delta \ll 35$ and we use $\delta = 0.05$. figure 8 shows the resulting velocity boundary function (see Section 3.2). Further, figure 9 illustrates the initial configuration of the first test problem.
Using the procedure described in Section 3.2.1, we find an upper limit of the spatial grid size $h$ of $1/24$. Thus, we perform simulations using the different grid sizes $h = 1/24, 1/32, 1/40, 1/48, 1/56$ for a non-dimensional total time $T = 6$. The resulting contact point velocity in each time step is plotted as a function of time in figure 10. The period of the oscillations corresponds to the time it takes for the contact line to pass one grid cell. At time $t = T$ we measure the relative error in the contact point position compared to the correct position $UT$ for the different grid sizes. In figure 11 it can be seen that grid convergence is obtained with a rate of convergence $p \approx 2$. In figure 11 it can be seen that the grid size $h = 1/24$ is probably not capable of resolving the velocity boundary function enough, since the rate of convergence is higher for the smaller grid sizes.
Figure 10: Resulting contact point velocity for the first model problem.

Figure 11: Error in contact point position at $T = 6$ for the first model problem.

5.2 Flat Fluid Interface Coupled to Navier–Stokes Equations

In next step, advection of an interface having a constant flat shape is still studied, but with the underlying velocity field coming from the Navier–Stokes solution, i.e. all equations in Section 2 are solved for. The wall contact angle of the flat interface is taken to be $\phi = 140$ and we hypothetically relate this angle to a contact point velocity of $U = 0.02$. Again, we choose $\delta = 0.05$ and we find an upper limit of the spatial grid size $h$ to be $1/24$. The resulting velocity boundary function for the second test problem is showed in figure 12.

Simulations with the different grid sizes are performed and the resulting velocity field at $T = 6$ for the grid size $h = 1/56$ is plotted in figure 13. Since the curvature of the interface is zero, the surface tension force in the Navier–Stokes equations is zero and the velocity field is therefore not affected by the interface. Further, the flat fluid interface is not realistic why we use a circular fluid interface shape in the third and final test problem.

However, the second model problem is used for studying the the grid
Figure 12: *Velocity boundary function at* \( \delta = 0.05 \) *for the first test problem:* \( \phi = 140, \ U = 0.02, \ Q = 1 \).

Figure 13: *Resulting velocity field for the second test problem:* \( \phi = 140, \ U = 0.02, \ Q = 1 \).

convergence of the contact line boundary conditions in the Navier–Stokes equations. So, at time \( t = T \) we measure the relative error in the contact point position compared to the correct position \( UT \) for the different grid sizes and the result is shown in figure 14. Again, it can be seen that grid convergence is obtained with a rate of convergence \( p \approx 2 \).

Figure 14: *Error in contact point position at* \( T = 6 \) *for the second model problem.*
5.3 Circular Fluid Interface Coupled to Navier–Stokes Equations

In the third and final model problem, the interface shape is assumed to be in the shape of a circular arc throughout the simulation. The Navier–Stokes part is again included. A circular interface shape is realistic for example when a liquid is rising in a narrow tube (capillary rise). For the third model problem we perform two set of simulations. In the first set the wall contact angle is $\phi = 140$ and in the second the angle is $\phi = 170$. Again we hypothetically relate these angles to a contact point velocity of $U = 0.02$. Just as before, we choose $\delta = 0.05$. The resulting velocity boundary function for the third test problem with contact angle $\phi = 140$ is the same as for the second model problem. The velocity boundary function for $\phi = 170$ is given in figure 15.

![Figure 15](image)

Figure 15: Velocity boundary function at $\delta = 0.05$ for the third test problem where $\phi = 170$, $U = 0.02$, $Q = 1$.

All grid sizes, except $h = 1/24$ for the case with a contact angle $\phi = 170$, are used and the resulting velocity fields at $T = 6$ for the grid size $h = 1/56$ is plotted in figure 16. It can be seen that away from the interface the flow profile is a regular poiseuille profile with zero velocity at the boundary. Close to the interface and the contact line however, the velocity is non-zero at the wall due to the velocity boundary condition. For this test problem (as opposed to the second test problem), it is clear that the fluid interface also effects the velocity field (via the surface tension).
Figure 16: Resulting velocity fields for the third test problem: \( \phi = 140 \) (upper) and \( \phi = 170 \) (lower), \( U = 0.02, Q = 1 \).

The resulting contact point velocity in each time step is plotted as a function of time in figure 17 and in figure 18. It can be seen that grid convergence is again obtained with a rate of convergence \( p \approx 2 \) for the case where \( \phi = 140 \) and with rate of convergence \( p > 2 \) for the case where \( \phi = 170 \). The analytical model in Section 3.1 is valid for contact angles \( 0 < \phi < 180 \), but the case where \( \phi = 170 \) approaches the upper limit. This can be seen in that the errors are larger for that case than for \( \phi = 140 \), for all grid sizes except \( h = 1/56 \). The high rate of convergence for the case with the larger contact angle is probably due to that the courser grids are not capable of resolving the problem well enough.

Figure 17: Resulting contact point velocities for the third model problem, \( \phi = 140 \) (left) and \( \phi = 170 \) (right).
6 Summary and Conclusions

We present the first steps in constructing macroscopic velocity boundary conditions to impose a contact point velocity. The contact point velocity is assumed to be given as a result from microscopic models. The idea is based on introducing an intermediate scale where the creeping flow approximation is valid. This approximation is used to derive macroscopic velocity boundary conditions along a fictitious boundary, located a small distance inside the physical boundary.

Model problems where the shape of the interface is constant throughout the simulation are introduced. For these problems, experiments show that the errors in the resulting contact line velocities converge with the grid size $h$ at a rate of convergence $p \approx 2$. The result is sensitive to contact angles approaching the limit of $\phi = 0$ or $\phi = 180$, and for such cases a smaller grid size is needed to properly resolve the problem. Further, for an accurate implementation of the velocity boundary condition, it is important to accurately evaluate the contact line position in each time step. Here we use cubic interpolation.

To be able to simulate problems with dynamic interface shapes, the model needs to be extended with a reinitialization of the level set function that preserves the contact line position (see Section 2.2). One possible approach is to implement the reinitialization developed in [36] where a penalty term is introduced to preserve the interface shape.

Further, to improve the model an approach where the computational boundary is modified only close to the contact line should be considered. Also, curved solid boundaries should be taken into account.
References


A Hydrodynamic Model of Movement of a Contact Line Over a Curved Wall

Hanna Holmgren     Gunilla Kreiss

Department of Information Technology, Uppsala University, Box 337, SE-751 05, Uppsala, Sweden

Abstract

A main challenge in numerical simulations of moving contact line problems is that the adherence, or no-slip boundary condition leads to a non-integrable stress singularity at the contact line. The ultimate resolution of moving contact line dynamics must rest on molecular considerations on a microscopic length scale. However, it is still instructive to set up a hydrodynamic model to investigate the macroscopic behavior of the fluids close to a contact line. Such a hydrodynamical model exists for the case of steady, two-dimensional flow with a perfectly flat fluid interface moving over a flat solid wall. However, in many applications more complex geometries are common. In this paper we extend the existing hydrodynamical model to be valid for a curved solid wall. We are interested in the dynamics close to the contact line, where the quantity \( r/R \) is assumed to be small. Here \( r \) is the distance to the contact line and \( R \) is the radius of curvature of the wall. We use perturbation analysis over the small quantity \( r/R \) to determine the first two terms in the expansion of the velocity. We plan to use the model presented in this paper to provide accurate boundary conditions in numerical simulations of two-phase flow with moving contact lines.
1 Introduction

Consider two immiscible and incompressible fluids. In flow situations when both fluids are in contact with a solid there is a line along the solid where the interfaces separating the two fluids meet the solid. This line is called a contact line, and when it evolves with time we have a moving contact line problem. Such problems form an important class of two-phase flows and appear both in nature and in many industrial applications [1]. Examples of phenomena appearing in the nature are raindrops falling on a window or water bugs resting on water surfaces [2]. Industrial application areas include lubrication, inkjet printing, gas and oil recovery in porous media [3, 4, 5, 6, 7, 8] and the development of microfluidic devices such as micropumps and lab-on-a-chip devices [9].

The standard mathematical model is the incompressible Navier-Stokes equation, influenced by the interface through surface tension, while the interface is advected by the fluid velocity. A main challenge in numerical simulations of moving contact line problems is that the adherence, or no-slip boundary condition leads to a non-integrable stress singularity at the contact line [1, 10]. When one of the fluids displaces the other fluid over a resting solid the contact line is formed between the three materials and with a no-slip condition an infinite force would be required to move the contact line. However, the surface tension force which is believed to cause the contact line movement is finite [11]. To accurately model contact line dynamics additional phenomena must be taken into account. This can be done by considering a phenomenologically based phasefield model, or a molecular dynamics based model. In both cases smaller length scales are introduced into the problem.

In [12] they argue that even though the ultimate resolution of a moving contact line must rest on molecular considerations on a microscopic length scale, the problem can be approached through kinematics and dynamics of fluids. They mean that it is still instructive to set up a hydrodynamical model to investigate the macroscopic behavior of the fluids close to a contact line. In [12] such a hydrodynamical model for the case of steady, two-dimensional flow with a perfectly flat fluid interface, is constructed. The length scale at which this model is valid is here referred to as $L$. At these length scales the viscous effects dominates the convection and the creeping flow approximation of Navier–Stokes equation is valid. The analysis results in a family of solutions, which depend on three parameters: the velocity of the contact line, the angle between the interface and the wall, and the viscosity ratio of the two fluids. Far away from the contact line, the assumptions of a flat fluid interface, a low Reynolds number, and two dimensional flow are in general no longer valid and the approximation fails. Microscopic
behavior even closer to the contact line are not modeled, nor is the viscous bending of the interface, which becomes increasingly important close to the contact line.

The hydrodynamical model constructed in [12] can be used for developing macroscopic fluid mechanical boundary conditions for the purpose of performing accurate and predictive numerical simulations of moving contact line problems. However, the model developed in [12] is only valid for the movement of a contact line over a flat solid wall. A flat wall is also often considered in numerical simulations of moving contact line problems because of its simplicity [2]. However, in many applications more complex geometries are common [2]. Therefore, we will in this paper extend the hydrodynamical model presented in [12] to a curved solid boundary with radius of curvature $R > L$. The curvature is assumed to be essentially constant along the wall at length scales $< L$. We are interested in the dynamics close to the contact line, where the quantity $r/R$ is assumed to be small. Here $r$ is the distance to the contact line. We will use perturbation analysis over the small quantity $r/R$. In this paper we determine the first two terms in the expansion. However, if required it is possible to extend the theory and calculations presented here to account for more terms in the expansions.

In Section 2 we describe our two dimensional model problem, and discuss the conditions under which the creeping flow approximation, which is also called the Stokes model, can be used in the vicinity of the contact line. In the Section 3 we introduce a perturbation expansion, and solve for the first two terms. Section 4 discusses resulting velocity fields, while a brief summary is offered in Section 5.

2 Hydrodynamic Model

Consider a model problem in two space dimensions with constant curvature of the solid wall and constant contact line speed $U$, see figure 1. We will construct a hydrodynamical model for the velocity field close to the contact line. This velocity field will be useful also when the contact line moves along a curved wall with slowly varying curvature. A polar coordinate system $(r, \theta)$ is used, with the origin fixed to the contact line position, and with $\theta = 0, 180$ along the tangent of the wall at the contact line. This coordinate system is rotating with angular velocity $\Omega$ of magnitude $U/R$. In this rotating frame of reference the contact line appears to be in rest and the wall to move with constant speed $U$ (but with opposite direction to that of the contact line in a fixed frame). The fluid interface between phase A and B is assumed to be perfectly flat and inclined at an angle of $\phi$ from the line $\theta = 0$. 


2.1 Creeping flow approximation

In a rotating frame of reference centrifugal and coriolis forces appear in the Navier-Stokes equations:

\[
\frac{\partial \mathbf{v}'}{\partial t'} + (\mathbf{v}' \cdot \nabla) \mathbf{v}' + \frac{1}{\rho} \nabla p' = \frac{\mu}{\rho} \nabla^2 \mathbf{v}' - 2\Omega' \times \mathbf{v}' - \Omega' \times (\Omega' \times \mathbf{r}' \mathbf{e}_r).
\]

Here \(\mathbf{v}'\) is the fluid velocity, \(p'\) is the pressure, \(\mu\) is the viscosity, and \(\rho\) is the density. Introduce dimensionless quantities by

\[
v = U \mathbf{v}', \quad r = L \mathbf{r}', \quad t = \frac{L}{U} \mathbf{t}', \quad p = P \mathbf{p}', \quad \Omega = \frac{U}{R} \Omega'.
\]

(1)

Now the equations are

\[
\frac{U^2}{L} \frac{\partial \mathbf{v}}{\partial t} + \frac{U^2}{L} (\mathbf{u} \cdot \nabla) \mathbf{v} + \frac{P}{L\rho} \nabla p = \frac{\mu U}{\rho L^2} \nabla^2 \mathbf{v} - 2\frac{U^2}{R} \Omega \times \mathbf{v} - \frac{LU^2}{R^2} \Omega \times (\Omega \times \mathbf{r} \mathbf{e}_r).
\]

With Reynolds number \(\text{Re} = LU \rho / \mu\) we have after multiplying with \(\frac{\rho L^2}{\mu U}\)

\[
\text{Re} \frac{\partial \mathbf{v}}{\partial t} + \text{Re}(\mathbf{v} \cdot \nabla) \mathbf{v} + \frac{PL}{\mu U} \nabla p = \nabla^2 \mathbf{v} - 2\text{Re} \frac{L}{R} \Omega \times \mathbf{v} - \text{Re} \frac{L^2}{R^2} \Omega \times (\Omega \times \mathbf{r} \mathbf{e}_r).
\]

Close to the contact line the viscous effects dominate over convection, and \(\text{Re} \ll 1\). Thus we neglect inertia terms (first two terms above). In addition, if \(R \geq L\) we similarly neglect the centrifugal and Coriolis forces (last two terms). The result is that we can use the same creeping flow equation as in the flat wall case.

The creeping flow equations can be formulated to take the form of a two-dimensional biharmonic equation for the stream function \(\psi(r, \theta)\) [13]:

\[
\nabla^4 \psi = 0.
\]

(2)
In terms of the stream function the fluid velocity components in plane polar coordinates are:

\[ v_r = -\frac{1}{r} \frac{\partial \psi}{\partial \theta} \]  
\[ v_\theta = \frac{\partial \psi}{\partial r}. \]  

In our setting we will have one stream function in each fluid, \( \psi_A \) and \( \psi_B \), and corresponding velocities.

### 2.2 Boundary and Interface Conditions

To close the system we need boundary and interface conditions. Before applying the boundary conditions, we parametrize the circular wall as:

\[ \theta = \arcsin\left(\frac{r}{2R}\right) + \pi \quad \text{in A} \]
\[ \theta = -\arcsin\left(\frac{r}{2R}\right) \quad \text{in B} \]  

When applying the boundary conditions, the following reformulations of the parametrization in (5) will also be needed:

\[ \sin \theta = -\frac{r}{2R}, \quad \cos 2\theta = 1 - \frac{r^2}{4R^2} \quad \text{in both A and B} \]
\[ \cos \theta = -\sqrt{1 - \frac{r^2}{4R^2}} \quad \text{in A} \]
\[ \sin 2\theta = \frac{r}{R} \sqrt{1 - \frac{r^2}{4R^2}} \quad \text{in A} \]
\[ \cos \theta = \sqrt{1 - \frac{r^2}{4R^2}} \quad \text{in B} \]
\[ \sin 2\theta = -\frac{r}{R} \sqrt{1 - \frac{r^2}{4R^2}} \quad \text{in B} \]  

Further, the normals and tangents to the circle is described in plane polar coordinates by

\[ n = (-\sin \theta, \cos \theta) \quad \text{in both A and B} \]
\[ t_A = (-\cos \theta, -\sin \theta) \quad \text{in A} \]
\[ t_B = (\cos \theta, \sin \theta) \quad \text{in B} \]  

Note that the components of these vectors are in the radial and angular directions, respectively. Now, the following boundary and interface conditions can be applied:

(i) Kinematic conditions: vanishing normal component of velocity at the
solid wall and fluid interface.

At the circular wall given by (5):
\[ \mathbf{n} \cdot \mathbf{v}_A = \sin \theta \frac{1}{r} \frac{\partial \psi_A}{\partial \theta} + \cos \theta \frac{\partial \psi_A}{\partial r} = 0 \]  
(8)

At the circular wall given by (5):
\[ \mathbf{n} \cdot \mathbf{v}_B = \sin \theta \frac{1}{r} \frac{\partial \psi_B}{\partial \theta} + \cos \theta \frac{\partial \psi_B}{\partial r} = 0 \]  
(9)

At the interface, i.e. \( \theta = \phi \):
\[ \frac{\partial \psi_A}{\partial r} = 0 \]  
(10)

At the interface, i.e. \( \theta = \phi \):
\[ \frac{\partial \psi_B}{\partial r} = 0 \]  
(11)

(ii) Kinematic condition: no slip, i.e. continuity of velocity at the interface.

At the interface, i.e. \( \theta = \phi \) (\( r > 0 \)): \( v_{Ar} = v_{Br} \iff \frac{\partial \psi_A}{\partial \theta} = \frac{\partial \psi_B}{\partial \theta} \)  
(12)

(iii) Dynamic condition: continuity of tangential stress at the interface.

At the interface, i.e. \( \theta = \phi \) (\( r > 0 \)): \( \tau_{Ar\theta} = \tau_{Br\theta} \iff \mu_A \frac{\partial^2 \psi_A}{\partial \theta^2} = \mu_B \frac{\partial^2 \psi_B}{\partial \theta^2} \)  
(13)

Equations (10) and (11) have been used to simplify the second expression.

(iv) Kinematic condition: no slip, i.e., no tangential relative motion of fluid at the solid wall.

At the circular wall given by (5):
\[ \mathbf{t}_A \cdot \mathbf{v}_A = \cos \theta \frac{1}{r} \frac{\partial \psi_A}{\partial \theta} - \sin \theta \frac{\partial \psi_A}{\partial r} = U \]  
(14)

At the circular wall given by (5):
\[ \mathbf{t}_B \cdot \mathbf{v}_B = -\cos \theta \frac{1}{r} \frac{\partial \psi_B}{\partial \theta} - \cos \theta \frac{\partial \psi_B}{\partial r} = -U \]  
(15)
3 Approximation by Perturbation Analysis

In this section we use perturbation analysis to find an approximate expression for the velocity field close to a contact line at rest at a circular wall, which rotates with constant speed. We are looking for an expansion of the velocity field in terms of different powers of $r/R$,

$$
\mathbf{v} = v^0 + \frac{r}{R} v^1 + \ldots
$$

(16)

The starting point is the biharmonic equation (2) in plane polar coordinates for each fluid, together with boundary and interface conditions (8) - (15). The velocity is given by derivatives of the stream function according to (3,4). The general solution of the biharmonic equation in plane polar coordinates was derived in [14] by a separation of variables. The solution is in the form of an expansion containing terms with different powers of $r$. Here, just as in [12] we require that the velocity is bounded as $r \to 0$, and therefore all terms with negative powers of $r$ in the Michell solution are excluded. In [12] the expansion is to be used also far from the contact line, and therefore the velocity was required to be bounded also for large values of $r$. The consequence is that all terms with powers of $r$ higher than one in the expression for the stream function was also omitted. In our setting the expansion is only used close to the contact line. Further away it needs to be matched to an outer solution, and therefore there is no reason to omit terms with higher powers of $r$.

Based on [14] we use the following expansion of the stream function

$$
\psi(r, \theta) = r(a \sin \theta + b \cos \theta + c \sin \theta + d \theta \cos \theta) +
\frac{r^2}{R}(e + f \cos 2\theta + g \sin 2\theta + h\theta) + \ldots,
$$

(17)

for each of the fluids A and B. For the stream function expansion to correspond to (16), we have scaled the constants $e, f, g$ and $h$ in the second term above by $1/R$, compared to the solution given in [14] (i.e. the constants $e, f, g$ and $h$ here equals the constants from the Michell solution multiplied by $R$). Further, in the paper by Michell the term $r^2\theta$ is not included due to an assumption of theta-periodicity, which is not assumed here.

Using the expansion for the stream function (17), the zeroth and first order components (in $r/R$) of the expansion for the velocity (16) are given
by

\[ \begin{align*}
    v_r^0 &= (b + d\theta - c) \sin \theta - (a + c\theta + d) \cos \theta \\
    v_\theta^0 &= (a + c\theta) \sin \theta + (b + d\theta) \cos \theta \\
    v_r^1 &= 2f \sin 2\theta - 2g \cos 2\theta - h \\
    v_\theta^1 &= 2(e + f \cos 2\theta + g \sin 2\theta + h\theta).
\end{align*} \]

for each of the fluids A and B. We will in the following determine the coefficients in the two leading order terms of the expansions for each of the fluids by applying the boundary conditions (8) - (15). Note that it would be straightforward to continue the expansion.

To consider the boundary conditions along the circular wall to different powers in \( r/R \), we need to taylor expand the parametrization of the circular wall given in (5) and (6) around \( r/R = 0 \):

\[ \begin{align*}
    \theta &= \pi + \frac{r}{2R} + \mathcal{O}(r^2/R^2) \quad \text{in A} \\
    \theta &= 0 - \frac{r}{2R} + \mathcal{O}(r^2/R^2) \quad \text{in B} \bigg) \\
    \sin \theta &= -\frac{r}{2R}, \quad \cos 2\theta = 1 + \mathcal{O}(r^2/R^2) \quad \text{in both A and B} \\
    \cos \theta &= -1 + \mathcal{O}(r^2/R^2) \quad \text{in A} \bigg) \quad \sin 2\theta = 0 + \frac{r}{R} + \mathcal{O}(r^2/R^2) \quad \text{in A} \bigg) \\
    \cos \theta &= 1 + \mathcal{O}(r^2/R^2) \quad \text{in B} \bigg) \quad \sin 2\theta = 0 - \frac{r}{R} + \mathcal{O}(r^2/R^2) \quad \text{in B} \bigg). \end{align*} \]

We are now ready to apply the boundary and interface conditions (8) - (15) to the expression for the stream function (17), and consider separately different powers of \( r/R \). The zeroth order equations involve \([a_i, b_i, c_i, d_i], \ i = A; B\), and are exactly the same as in [12]. It follows that the velocity field derived in [12] for a contact line at rest at a flat wall, which translates with constant velocity, is the unperturbed solution \( v_0 \). For clarity we have stated the equations for the coefficients \([a_i, b_i, c_i, d_i], \ i = A; B\), and their solution, in the appendix. We note that the system is singular for \( \phi = 0, 180 \), and non-singular for other angles.

Next we collect terms which are linear in \( r/R \) and get the following system for the coefficients in the second term of (17):

\[ M^1 Z^1 = F^1. \] (23)

Here

\[ M^1 = \begin{pmatrix}
1 & 1 & 0 & \pi & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & \tilde{C} & \tilde{S} & \phi & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \tilde{C} & \tilde{S} & \phi \\
0 & 2\tilde{S} & -2\tilde{C} & -1 & 0 & -2\tilde{S} & 2\tilde{C} & 1 \\
0 & \mu_A\tilde{C} & \mu_A\tilde{S} & 0 & 0 & -\mu_B\tilde{C} & -\mu_B\tilde{S} & 0 \\
0 & 0 & -1 & -1/2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1/2 & 0
\end{pmatrix}. \]
\[ Z^1 = \begin{pmatrix} e_A \\ f_A \\ g_A \\ h_A \\ e_B \\ f_B \\ g_B \\ h_B \end{pmatrix} \quad \text{and} \quad F^1 = \begin{pmatrix} \frac{1}{2}(a_A + \pi c_A + d_A) \\ \frac{1}{2}(a_B + d_B) \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{2}(b_A - c_A + \pi d_A) \\ \frac{1}{2}(c_B - b_B) \end{pmatrix}. \]

and

\[ \tilde{S} = \sin 2\phi \\
\tilde{C} = \cos 2\phi. \]

The rows in system (23) originate from the boundary and interface conditions (8)-(15), respectively. This system is also singular for precisely \( \phi = 0, 180 \).

When all coefficients \([a_i, b_i, c_i, d_i, e_i, f_i, g_i, h_i], \ i = A; B\) have been calculated the resulting approximated velocity field is obtained using the expressions (18) - (21). The velocity field can be transformed to a frame of reference where the contact line moves over a stationary wall by subtracting a rigid body rotation, \( r_d \Omega \mathbf{e}_\theta \), centered at the center of curvature of the solid wall. Here \( \Omega = U/R \) is the magnitude of the angular velocity, \( r_d \) is the distance to the center of curvature of the wall, and \( \mathbf{e}_\theta \) is the unit vector in the rotational direction. The two latter quantities are

\[ r_d = \sqrt{r^2 \cos^2 \theta + (R + r \sin \theta)^2}, \quad \mathbf{e}_\theta = -\frac{R \cos \theta}{r_d} \mathbf{e}_r + \frac{R \sin \theta + r}{r_d} \mathbf{e}_\theta. \]

4 Results

We look at the resulting approximated velocity field for three different cases. In all cases the contact line is stationary while the circular solid is rotating with angular velocity \( U/R \) where \( U = 1 \). In the first case the angle between the fluid interface and the line \( \theta = 0 \) is \( \phi = 45 \) and the viscosity ratio is \( \mu_A/\mu_B = 1 \). In the second case the viscosity ratio is increased to \( \mu_A/\mu_B = 100 \) and in the third case the contact angle is increased to \( \phi = 170 \). The resulting velocity fields for the three cases are shown in figure 2, figure 3, and figure 4 respectively.

Increasing the viscosity ratio from \( \mu_A/\mu_B = 1 \) to \( \mu_A/\mu_B = 100 \) leads to a significant change in the flow pattern, compare case one in figure 2 and case two in figure 3. In the first case, when the two phases have the same
viscosity, the flow along the circular boundary in phase A and along the interface both are directed inwards towards the contact line. This leads to the formation of a jet out from the contact line in phase A (directed upwards in figure 2). In the second case, when the viscosity ratio is $\mu_A/\mu_B = 100$, the direction of the flow along the interface has changed sign compared to the equal viscosity case. Consequently the flow along the circular boundary in phase B and along the interface are now both directed outwards from the contact line, which leads to a jet inwards towards the contact line in phase B (the region with lower viscosity) instead. In case three, when the contact angle is increased to $\phi = 170$ but the viscosity ratio is kept at $\mu_A/\mu_B = 100$, the same flow pattern with an ingoing jet towards the contact line in the phase with lower viscosity, phase B, is observed. This behavior agrees with the observations in [12].

Figure 2: Approximated velocity field for the first case: $R = 1$, $\mu_A/\mu_B = 1$, $\phi = 45$, $U = 1$.

Figure 3: Approximated velocity field for the second case: $R = 1$, $\mu_A/\mu_B = 100$, $\phi = 45$, $U = 1$. 
To verify the perturbation expansion approximation we consider how well the first two terms satisfy the boundary conditions. The velocity component tangential to the circular wall should be unity, $U = 1$, due to the no slip boundary conditions (14) and (15). Further, the velocity component normal to the wall should be zero according to the boundary conditions (8) and (9). In figure 5 the error in the normal and tangential velocity components along the circular wall for case one are shown for both fluid phases A and B. The error in the velocity components decrease quadratically with decreasing $r$ ($R$ constant), which agrees with the perturbation expansion approximation (since we have neglected all terms containing powers of $r/R$ of two and higher). Similar second order behavior of the errors in case two and three are shown in figure 6 and figure 7 respectively. However, for case three, i.e. figure 7, a small deviation of the second order behavior of the error in the velocity component normal to the circular wall in phase B is observed for the larger values of $r$. The reason for the deviation in the second order behavior is that the error changes sign between $r = 0.2$ and $r = 0.3$. This can be seen in figure 8 where the behavior of the velocity component normal to the circular wall in phase B is plotted. For the smaller values of $r$, approximately $r < 0.25$, the normal velocity component is negative, i.e. too small, while for the larger values of $r$ it is too big. For $r > 0.25$ the error again follows a second order behavior.

The systems for determining the coefficients in the stream function expansion is singular for $\phi = 0$ and $\phi = 180$. In case three when $\phi$ is close to 180 an increase in the error is observed. Compare upper right figures in figure 6 and figure 7 for example. The error in the tangential velocity component along the circular wall for large values of $r$ in phase A is also clearly visible in figure 4.

For larger values of $r$ than included in the error plots presented here the
assumption of a small \( \frac{r}{R} \) is no longer valid, and the perturbation expansion approximation fails. Furthermore, far away from the contact line the viscous effects no longer dominate the convection, and the creeping flow approximation also fails. For small values of \( r \) on the other hand, the velocity field contains large gradients and is therefore very sensitive to small changes in \( r \) or \( \theta \).

Figure 5: Error in velocity components along circular wall in phase A (upper two figures) and phase B (lower two figures) for the first case.
Figure 6: Error in velocity components along circular wall in phase A (upper two figures) and phase B (lower two figures) for the second case.
Figure 7: Error in velocity components along circular wall in phase A (upper two figures) and phase B (lower two figures) for the third case.

Figure 8: Velocity component tangential to circular wall in phase A, for the third case.
5 Summary and outlook

We have presented a hydrodynamic model for the movement of a contact line over a curved wall. The model is valid close to the contact line, but only if the solid wall’s curvature is not too high or varies too rapidly. The model is based on Stokes equation, and is a perturbation expansion in terms of $r/R$, where $R$ is the radius of curvature of the wall and $r$ is the distance from the contact line. Furthermore, as for the original flat wall hydrodynamic model [12], the model is not valid in the intermediate vicinity of the contact line, at lengths scales where molecular phenomena come into play.

We plan to use the model to provide an accurate boundary condition in numerical simulations of two-phase flow with moving contact lines. Instead of applying the no-slip boundary condition along the wall we plan to use the velocity field given by the model as a boundary condition at a small distance from the wall, thus avoiding the singularity at the contact line. As with the flat wall model, a contact angle and a contact line velocity are required as input.

Appendix

The equations for the coefficients in the first term in the stream function expansion (17) are written in matrix form as

\[ M^0 Z^0 = F^0 \]

where

\[
M^0 = \begin{pmatrix}
0 & 1 & 0 & \pi & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
S & C & \phi S & \phi C & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\mu_A S & -\mu_A C & \mu_A (2C - \phi S) & -\mu_A (2S + \phi C) & \mu_B S & \mu_B C & -\mu_B (2C - \phi S) & \mu_B (2S + \phi C) \\
1 & 0 & 0 & \pi & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
\end{pmatrix},
\]

\[
Z^0 = \begin{pmatrix}
a_A \\
b_A \\
c_A \\
d_A \\
a_B \\
b_B \\
c_B \\
d_B \\
\end{pmatrix} \quad \text{and} \quad F^0 = \begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
-\mu \\
\end{pmatrix},
\]

and

\[
S = \sin \phi \\
C = \cos \phi.
\]
The rows in system (24) originates from the boundary conditions (8)-(15) respectively. The solution is

\[ a_A = -U - \pi c_A - d_A \]  
\[ b_A = -\pi d_A \]  
\[ c_A = US^2[S^2 - \delta \phi + Q(\phi^2 - S^2)]/D \]  
\[ d_A = USC[S^2 - \delta \phi + Q(\phi^2 - S^2) - \pi \tan \phi]/D \]  
\[ a_B = -U - d_B \]  
\[ b_B = 0 \]  
\[ c_B = US^2[S^2 - \delta^2 + Q(\delta \phi - S^2)]/D \]  
\[ d_B = USC[S^2 - \delta^2 + Q(\delta \phi - S^2) - Q\pi \tan \phi]/D, \]

where

\[ \delta = \phi - \pi \]
\[ Q = \mu_A/\mu_B \]
\[ D = (SC - \phi)(\delta^2 - S^2) + Q(\delta - SC)(\phi^2 - S^2). \]

Note that the system is non-singular for \(0 < \phi < 180\).

References


A fast massively parallel two-phase flow solver for the simulation of microfluidic chips

Martin Kronbichler*

Institute for Computational Mechanics, Technische Universität München, Boltzmannstr. 15, 85748 Garching b. München, Germany

Ababacar Diagne and Hanna Holmgren

Division of Scientific Computing, Department of Information Technology, Uppsala University, Box 337, 751 05 Uppsala, Sweden

October 22, 2015

Abstract

For the quantitative evaluation of three-dimensional dynamics in high-throughput microfluidic separation devices, detailed large-scale simulations are fundamental. For this application, a parallel finite element solver of incompressible two-phase flow is presented. The method relies on a conservative level set formulation for representing the interface and uses adaptive mesh refinement on forests of octrees. An implicit time stepping with efficient block solvers for the incompressible Navier–Stokes equations discretized with Taylor–Hood and augmented Taylor–Hood finite elements is presented. A matrix-free implementation is used that reduces the solution time for the Navier–Stokes system by a factor of three compared to the best matrix-based algorithms. Scalability of the chosen algorithms up to 32 768 cores and a billion of degrees of freedom is shown.

*Corresponding author; e-mail: kronbichler@lnm.mw.tum.de
1 Introduction

For the understanding of flow in microfluidic devices, numerical simulations are often the only available tool. Three-dimensional effects of various flow configurations must be captured by highly resolved computational studies that in turn require large-scale computational facilities. In this work, we present a solver framework for soft inertial microfluidics involving particles with deformable surfaces (Wu et al., 2009). Simulations detailing the flow patterns in these devices are still rare but promise to reveal novel physics and give better control over the device design. This includes the ability to find flow configurations to sort particles of various sizes and material parameters as well as to monitor surface stresses. As a means for representing deformable surfaces, we use a model for immiscible incompressible two-phase flow with surface tension.

Detailed multi-phase flow simulations require a very high numerical resolution to track the evolution of free surfaces with a sudden jump in stresses and material parameters between the different fluid phases. For indicating the interface location between fluids, several competing methods exist, which can either be interface tracking methods (Peskin, 1977) such as front tracking (Unverdi and Tryggvason, 1992), or interface capturing methods such as level set methods (Osher and Sethian, 1988), the volume–of–fluid method (Hirt and Nichols, 1981), or phase field methods (Jacqmin, 2000). For the representation of the interface forces in the incompressible Navier–Stokes equations, two strains of developments can be distinguished. In so-called extended finite element methods (XFEM), the interface is represented in a sharp way. Suitable jump and kink enrichments are added to the pressure and velocity, respectively (Groß and Reusken, 2007; Fries and Belytschko, 2010; Rasthofer et al., 2011), as a means to exactly include jump conditions in the finite element spaces. In order to obtain stable and robust schemes, suitable jump penalty terms are added that avoid the otherwise deteriorating effect of small cut regions on condition numbers. The second strain of methods is given by continuous surface tension models in the spirit of the original work by Brackbill et al. (1992), where the surface tension forces and changes in material parameters are smoothly applied in a narrow band of width proportional to the mesh size. XFEM-based methods are generally more accurate for a given element size, but at the cost of an almost continuous change in maps of degrees of freedom as the interface passes through the domain. Besides the increased cost of integration in cut elements, the difficulty of efficiently implementing changing maps has confined most XFEM methods to serial and relatively modest parallel computations.

The contribution of this work is a highly efficient and massively parallel realization of a continuous surface tension model using a conservative level
set representation of the interface (Olsson and Kreiss, 2005; Olsson et al., 2007). Parallel adaptive mesh refinement and coarsening is used in this work to dynamically apply high resolution close to the interface. The algorithm is based on unstructured coarse meshes that are refined in a structured way using a forest-of-trees concept with hanging nodes (Burstedde et al., 2011). In many complex three-dimensional flows, choosing two additional levels of refinement around the interface merely doubles the number of elements. For a range of flow configurations, continuous surface tension models on such a mesh provide solutions of similar quality as state-of-the-art XFEM techniques; thus, our solver is expected to be competitive with good XFEM implementations in the present context. For time discretization, second-order accurate time stepping based on BDF-2 is used. In space, we choose inf–sup stable Taylor–Hood elements $Q_2 Q_1$ for the representation of fluid velocity and pressure. These choices result in a considerably more accurate velocity representation as compared to the linear stabilized finite element case often used in the literature. Furthermore, we consider so-called augmented Taylor–Hood elements $Q_2 Q_1^+$ where an element-wise constant is added in order to guarantee element-wise mass conservation in the discretization of the incompressible Navier–Stokes equations (Boffi et al., 2012). These elements can provide additional accuracy, in particular with respect to the pressure representation, as compared to plain Taylor–Hood elements. Since these elements have not been studied in detail yet, we also present suitable iterative solvers for these elements.

On parallel architectures, many unstructured finite element solvers rely on (distributed) sparse matrix data structures, with sparse matrix-vector products dominating the run time. Unfortunately, these kernels are a poor fit for modern hardware due to the overwhelming memory bandwidth limit. Instead, our work uses fast matrix-free implementations based on cell-wise integration proposed in Kronbichler and Kormann (2012) for most kernels. The fast computation of integrals on the fly is realized by tensorial evaluation for hexahedra (sum factorization) that has its origin in the spectral element community (Karniadakis and Sherwin, 2005; Cantwell et al., 2011; Basini et al., 2012). For element degree 2, however, integration still increases the number of arithmetic operations by about a factor of three over sparse matrix-vector products on the scalar Laplacian (Kronbichler and Kormann, 2012). Nonetheless, performance can be gained if the increase in arithmetic does not outweigh the reduction of memory access. For systems of equations such as the incompressible Navier–Stokes equations with coupling between all velocity components (as they appear for variable material parameters and Newton linearization), fast integration has an additional advantage because coupling occurs only on the level of quadrature points. This enables matrix-free matrix-vector products that are up to an order
of magnitude faster already on $Q_2$ elements (Kronbichler and Kormann, 2012). These techniques are used in a fully implicit Navier–Stokes solver with block-triangular preconditioner and a selection of algebraic multigrid and incomplete factorizations for the individual blocks as appropriate. This work will demonstrate that these components give a solver which features

- massively parallel dynamic mesh adaptation,
- matrix-free solvers with good scalability and 2–4× faster solvers compared to matrix-based alternatives, and
- good memory efficiency, allowing to fit larger problems into a given memory configuration.

We want to point out that many algorithms for the incompressible Navier–Stokes equations presented in the context of two-phase flow in microfluidic devices are also applicable in other contexts. Moreover, the solvers extend straightforwardly to cubic and even higher-order polynomials, where the advantage over matrix-based algorithms is even more impressive. Furthermore, the higher arithmetic intensity and regular access structure makes these algorithms a promising development for future exascale hardware.

The remainder of the paper is as follows. Sec. 2 presents the numerical model and discretization and Sec. 3 discusses the selected linear solvers and implementational details of the fast matrix-free implementation. In Sec. 4, the microfluidic problem setting is introduced. Sec. 5 shows the performance results including strong and weak scalability tests, and Sec. 6 summarizes our findings.

## 2 Numerical model

We model the separation of species in a microfluidic device by the flow of two immiscible incompressible fluids as proposed in Wu et al. (2009). Surface tension at the fluid-fluid interface stabilizes the shape of the interface.

### 2.1 Incompressible Navier–Stokes equations

The motion of each fluid is given by the incompressible Navier–Stokes equations for velocity $\mathbf{u}$ and pressure $p$ in non-dimensional form,

$$
\rho^* \frac{\partial \mathbf{u}}{\partial t} + \rho^* \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{\text{Re}} \nabla \cdot (2\mu^* \nabla^s \mathbf{u}) + \frac{1}{\text{Fr}^2} \rho^* g e_g + \frac{1}{\text{We}} \kappa n \delta \Gamma,
$$

$$
\nabla \cdot \mathbf{u} = 0.
$$

(1)

Here, $\nabla^s \mathbf{u} = \frac{1}{2} (\nabla \mathbf{u} + \nabla \mathbf{u}^T)$ denotes the rate of deformation tensor. The parameter Re denotes the Reynolds number, Fr the Froude number, and
We the Weber number, which control the magnitude of viscous stresses, gravitational forces, and surface tension, respectively. The parameters $\rho^*$ and $\mu^*$ denote the density and viscosity measured relative to the parameters of fluid 1,

$$\rho^* = \begin{cases} 
1 & \text{in fluid 1,} \\
\frac{\rho_2}{\rho_1} & \text{in fluid 2,}
\end{cases} \quad \mu^* = \begin{cases} 
1 & \text{in fluid 1,} \\
\frac{\mu_2}{\mu_1} & \text{in fluid 2.}
\end{cases}$$

### 2.2 Level set description of two-phase flow

We denote by $\Omega_1$ the domain occupied by fluid 1 and by $\Omega_2$ the domain of fluid 2, and by $\Gamma$ the interface between $\Omega_1$ and $\Omega_2$ as sketched in Fig. 1. The computational domain $\Omega$ is the union of the two subdomains and the interface, $\Omega = \Omega_1 \cup \Gamma \cup \Omega_2$. The interface $\Gamma$ is captured by the conservative level set method from Olsson et al. (2007), i.e., by the 0 contour of a regularized characteristic function $\Phi$. Across the interface, $\Phi$ smoothly switches from $-1$ to $+1$ as depicted in Fig. 1. The evolution of $\Gamma$ in time is by transport of the level set function $\Phi$ with the local fluid velocity $u$,

$$\partial_t \Phi + u \cdot \nabla \Phi = 0, \quad \Phi(\cdot, 0) = \tanh \left( \frac{d(x, y)}{\varepsilon} \right).$$

(2)

At the initial time, the profile $\Phi$ is computed from a signed distance function $d(x, y)$ around the interface, where $\varepsilon$ is a parameter that controls the thickness of the transition region.

#### 2.2.1 Reinitialization procedure.

To preserve the profile thickness and shape of $\Phi$ during the simulation despite non-uniform velocity fields as well as discretization errors, a conservative reinitialization step is performed according to Olsson et al. (2007). The reinitialization seeks the steady state to the equation

$$\partial_\tau \Phi + \frac{1}{2} \nabla \cdot ((1 - \Phi^2)\mathbf{n}) - \nabla \cdot (\varepsilon (\nabla \Phi \cdot \mathbf{n})\mathbf{n}) = 0,$$

(3)

starting from the interface given by the advection equation (2), where $\tau$ is an artificial time. Using two to five pseudo time steps of equation (3) provides a good approximation of the steady state and ensures stable simulations.

#### 2.2.2 Computation of surface tension.

The evaluation of surface tension in (1) requires the normal vector $\mathbf{n}$ onto the interface as well as the interface curvature $\kappa$. These quantities are computed...
in terms of the level set function $\Phi$,

$$
\mathbf{n} = \frac{\nabla \Phi}{|\nabla \Phi|} \quad \text{and} \quad \kappa = -\nabla \cdot \mathbf{n}.
$$

(4)

The surface tension force is defined by a slight modification of the continuous surface tension force according to Brackbill et al. (1992),

$$
\mathbf{F} = \frac{1}{\text{We}} \kappa \nabla \Phi,
$$

(5)

where $H_\Phi$ denotes a Heaviside function constructed from $\Phi$, with the gradient $\nabla H_\Phi$ replacing the term $\mathbf{n} \delta_\Phi$ in the original work (Brackbill et al., 1992; Olsson et al., 2007). The gradient form of surface tension has the advantage that the surface tension force for constant curvature, i.e., circular shapes, can be represented exactly by the pressure gradient without spurious velocities (Zahedi et al., 2012). The simple choice $H_\Phi = \Phi/2$ as proposed in Olsson et al. (2007)\(^1\) is undesirable, though, because $\nabla \Phi$ has support on the whole domain, albeit exponentially decaying for smooth $\Phi$. For the adaptive meshes with a higher level of refinement around the interface according to Sec. 2.4 below, small distortions in the level set profile at faces with different refinement give rise to large non-physical curvatures and thus, spurious force contributions. To localize surface tension, we instead use the definition

$$
H_\Phi \equiv H_{\epsilon}(d(\mathbf{x})), \quad d(\mathbf{x}) = \log \left( \frac{1 + \Phi(\mathbf{x})}{1 - \Phi(\mathbf{x})} \right),
$$

(6)

where $d(\mathbf{x})$ denotes a signed distance function reconstructed from $\Phi$, supplemented with suitable limit values for the regions where numerical errors in $\Phi$ yield values outside the open interval $(-1, 1)$. The function $\mathbb{H}_\epsilon$ denotes a one-dimensional smoothed Heaviside function that changes from 0 to 1 over a length scale proportional to $\epsilon$. We choose the primitive function of the discrete delta function with vanishing first order moments derived in (Peskin, 2002, Sec. 6) for $\mathbb{H}_\epsilon$, scaled such that the transition occurs in a band of width $2\epsilon$ around the interface. Note that this region approximately corresponds to a band between the $-0.76$ and $+0.76$ contours of $\Phi$.

To improve robustness, the equations for the normal vector field and the curvature (4) are each solved by a projection step of the level set gradient and normal vector divergence to the space of continuous finite elements, respectively, with mesh-dependent diffusion $4h^2$ added according to the discussion in Zahedi et al. (2012). Likewise, a projected normal vector $\mathbf{n}$ is computed that is used throughout the pseudo time stepping of (3). We emphasize

\(^1\)The factor $\frac{1}{2}$ accounts for the range $[-1, 1]$ of the smoothed indicator-like function $\Phi$ in this work as opposed to the range $[0, 1]$ in Olsson et al. (2007).
that these projection steps are essential for the robustness of the method on unstructured and 3D meshes. Slightly distorted normals, in particular the ones determining the curvature, can spoil the simulation.

2.3 Time discretization

For time stepping, an implicit/explicit variant of the BDF-2 scheme is used. In order to avoid an expensive monolithic coupling between the Navier–Stokes part (1) and the level set transport step (2) via the variables $u$ and $\Phi$, an explicit (time lag) scheme between the two equations is introduced. In each time step, we first propagate the level set function with the local fluid velocity, run the reinitialization algorithm, and then perform the time step of the incompressible Navier–Stokes equations with surface tension evaluated at the new time. Each of the two fields is propagated fully implicitly using BDF-2. To maintain second order of accuracy in time, the velocity field for the level set advection is extrapolated from time levels $n - 1$ and $n - 2$ to the new time level $n$,

$$u^{n,0} = 2u^{n-1} - u^{n-2},$$

and with suitable modifications when using variable time step sizes. Note that the splitting between the level set and Navier–Stokes parts corresponds to an explicit treatment of surface tension, which gives rise to a time step limit

$$\Delta t \leq c_1 \frac{We}{Re} h + \sqrt{(c_1 \frac{We}{Re} h)^2 + c_2 We h^3},$$

where $c_1$ and $c_2$ are constants that do not depend on the mesh size $h$ and the material parameters, cf. Galusinski and Vigneaux (2008). There exist methods to overcome this stability limit, see e.g. Hysing (2006); Sussman and Ohta (2009). For the examples considered in this work, however, this only imposes a mild restriction with the first term dominating.

2.4 Space discretization and mesh adaptivity

We discretize all solution variables in space using the finite element method. To this end, the computational domain is partitioned into a set of elements. On each element, polynomial solutions of the variables are assumed, and continuity is enforced over the element boundaries. In each time step, the approximations for $\Phi_h^n, u_h^n, p_h^n$ are of the form

$$\Phi_h^n(x) = \sum_{j=1}^{N_k} \Phi_j^n \varphi_j^h(x), \quad u_h^n(x) = \sum_{j=1}^{N_u} U_j^n \varphi_j^h(x), \quad p_h^n(x) = \sum_{j=1}^{N_p} P_j^n \varphi_j^p(x),$$

(9)
where the coefficient values $\Phi_j^n, U_j^n, P_j^n$ are to be determined. When choosing the finite element ansatz spaces, i.e., the spaces spanned by the shape functions $\varphi^\Phi, \varphi^u$ and $\varphi^p$, respectively, we consider the following factors:

- The function represented by $\Phi$ is a smoothed Heaviside function according to Sec. 2.2 whose width needs to be kept small for accurate pointwise mass conservation and interface positions. This sets high resolution requirements. Continuous linear functions on hexahedra, $Q_1$, defined by a tensor product of 1D functions, are used for $\varphi^\Phi$. With this choice sharp transitions are better represented than with higher order functions for the same number of degrees of freedom, because they avoid over- and undershoots.

- For the incompressible Navier–Stokes equations, finite element methods without stabilization do not allow for arbitrary pairings of velocity and pressure shape functions due to the Babuška–Brezzi (inf–sup) condition (Girault and Raviart, 1986). We consider the following two inf–sup stable options:
  
  - **Taylor–Hood (TH):** We use elements $Q_q Q_{q-1}$, i.e., shape functions of degree $q$ for each component of velocity and shape functions of degree $q-1$ for the pressure with $q \geq 2$. In this work, the lowest-order case $q = 2$ is selected.
  
  - **Augmented Taylor–Hood (ATH):** These elements use the same velocity space as TH elements but an extended space $Q_{q-1}^+$ for the pressure where an element-wise constant function is added for $\varphi^p$. The additional constant function forces the velocity field to be element-wise conservative (divergence-free) (Boffi et al., 2012) and is consistent because no derivatives on pressure variables appear in the weak form of the Navier–Stokes equations.

We use a mesh that can be dynamically adapted by local refinement and coarsening (adaptive mesh refinement). This allows us to increase the mesh resolution close to the fluid-fluid interface where rapid changes in pressure as well as material parameters need to be captured. Moreover, a fine mesh around the interface keeps the approximation error in normals and curvatures from the level set variable $\Phi$ small. In order to meet the different resolution requirements for the Navier–Stokes variables on the one hand and the indicator-like level set function on the other hand, a finer mesh is used for the latter. We choose the level set mesh to be a factor of three to four finer than the Navier–Stokes mesh, depending on the desired balance between costs in the level set part and the Navier–Stokes part. To avoid a mismatch in pressure space and the term $H_\Phi$ according to Zahedi et al.
(2012), we apply an interpolation of $H_\Phi$ onto the pressure space $Q_1$ on the Navier–Stokes mesh before evaluating surface tension.

As a mesh refinement criterion, we mark cell $K$ for refinement if

$$\log \left( \max_K |\nabla \Phi|_\varepsilon \right) > -4 \quad \text{or} \quad \log \left( \max_K |\nabla \Phi|_\varepsilon \right) + 4\Delta t \frac{\mathbf{u} \cdot \nabla \Phi}{\varepsilon |\nabla \Phi|} > -7, \quad (10)$$

where the last term is evaluated in the center of the cell. Recall that $\varepsilon$ controls the width of the transition region of the conservative level set function. In these formulas, the terms involving logarithms approximate the number of cells between cell $K$ and the interface, with 0 indicating a cell cut by the interface and negative numbers the respective distance. Thus, the first criterion specifies that cells up to four layers away from the interface should be refined. The second formula makes the refinement biased towards the direction of the local flow field: A distance-only approach would adjust the mesh optimally to the current interface position and soon be outdated again. To extend the time of viability of the refined mesh and reduce the frequency of re-meshing, the second term adds a layer of approximately three mesh cells in downstream direction.

In case the distance to the interface is larger than the above values, cells are marked for coarsening. The mesh smoothing algorithms from p4est (Burstedde et al., 2011) ensure that the levels of neighboring cells in the adapted mesh differ at most by a factor 2:1 both over faces, edges, and vertices. In each time step, we check the cheap (and less strict) criterion whether $\log \left( \max_K |\nabla \Phi|_\varepsilon \right) > -3.5$, and in case there is at least one such cell, criterion (10) is evaluated for each cell and the mesh adaptation algorithm is called. The frequency of mesh updates is typically between ten and a few hundreds of time steps, depending on the flow field and the time step size.

Based on the mesh and the finite element shape functions, weak forms of the equations (1) and (2) are derived. In the usual finite element fashion, the equations are multiplied by test functions, divergence terms are integrated by parts and boundary conditions are inserted. For the discrete version of the incompressible Navier–Stokes equations (1), we implement the skew-symmetric form of the convective term $\rho^* \mathbf{u} \cdot \nabla \mathbf{u} + \frac{\rho^*}{2} \nabla \cdot \mathbf{u}$ in order to guarantee discrete energy conservation (Tadmor, 1984). Finite element discretizations of equations of transport type, such as the level set equation, typically need to be stabilized. Here however, no stabilization is used since the reinitialization will take care of possible oscillations. Dirichlet boundary conditions such as no-slip conditions on velocities or the prescribed level set at inflow boundaries, are imposed strongly by setting the respective values of $U^n_j$ and $\Phi^n_j$ to the given values. For Neumann boundary conditions, e.g. for imposing non-zero pressure levels on outflow boundaries, boundary integrals are added to the equations.
2.5 Summary of solution algorithm

To summarize, one time step of our solver consists of computing the coefficient values $\Phi^j_n$, $U^j_n$ and $P^j_n$ in the finite element expansion (9) by the following steps:

1. Extrapolate all fields to the new time level using second order extrapolation similar to (7), resulting in $\Phi^{n,0}$, $u^{n,0}$, $p^{n,0}$, and apply boundary conditions at the new time step. The successive steps can then compute for increments in time to these fields with homogeneous boundary conditions.

2. Compute increment $\delta \Phi^n$ by solving the weak form of the advection equation (2) with velocity $u^{n,0}$ and BDF-2 discretization in $\Phi$. Then, we set $\tilde{\Phi}^n = \Phi^{n,0} + \delta \Phi^n$.

3. Project $\nabla \tilde{\Phi}^n$ onto the space of linear finite elements, including diffusion of size $4h^2$, and evaluate $\tilde{n}^n = \nabla \tilde{\Phi}^n / |\nabla \tilde{\Phi}^n|$ on each node of the level set mesh.

4. Perform $n_{\text{reinit,steps}}$ reinitialization steps according to (3), based on the normal vector approximation $\tilde{n}^n$. The nonlinear compression term $\frac{1}{2} \nabla \cdot ((1-\Phi^2)\tilde{n}^n)$ is treated in an explicit Euler fashion and the diffusion term $\nabla \cdot (\varepsilon (\nabla \Phi \cdot \tilde{n}^n)\tilde{n}^n)$ in an implicit Euler fashion. The result of this procedure is the final level set field $\Phi^n$.

5. Project $\nabla \Phi^n$ onto the space of linear elements, including diffusion of size $4h^2$, and evaluate $\n^n$ on each node of the level set mesh.

6. Compute curvature $\kappa$ by projecting $-\nabla \cdot \n^n$ onto the space of linear elements, including diffusion of size $4h^2$.

7. Compute the discrete Heaviside function $H^n_{\Phi}$ from $\Phi^n$ by evaluating (6) on each node of the finite element mesh. Interpolate $H^n_{\Phi}$ to the pressure finite element space.

8. Evaluate all forcing for the momentum equation, including surface tension according to (5). Evaluate the relative density $\rho^*$ and viscosity $\mu^*$ based on the Heaviside function $H^n_{\Phi}$ in each quadrature point and store them for use in the Navier–Stokes solver.

9. Newton iteration for velocity and pressure, iteration index $k \geq 1$:

   (a) Compute nonlinear residual of momentum and continuity equations.

   (b) Solve for increment $[\delta u^{n,k}, \delta p^{n,k}]$ and add to $u^{n,k-1}, p^{n,k-1}$. 


At convergence, we obtain the fields $u^n$ and $p^n$ at the new time level.

3 Solution of linear systems

After time discretization and linearization in the algorithm from Sec. 2.5, linear systems for the level set equations and for the Navier–Stokes equations need to be solved. The solution of linear systems represents the main computational effort in our solver and is thus discussed in more detail. For the level set advection equation in step 2, the system matrix is

$$
\left( \frac{3}{2\Delta t} M + C(u) \right) \delta \Phi = R\Phi,
$$

(11)

where $M$ denotes the level set mass matrix and $C(u)$ the convection matrix depending on the current velocity. The vector $R\Phi$ denotes the discrete residual of the level set equation, evaluated using $\Phi^{n-1}, \Phi^{n-2},$ and $u^{n,0}$. Since the time step $\Delta t$ is typically on the order $h/|u|$ (representing constant CFL numbers), the condition number of the system matrix behaves as $O(1)$ (Elman et al., 2005) and simple iterative solvers can be employed. Due to the non-symmetry, we choose a BiCGStab solver (Saad, 2003), preconditioned by the diagonal of the mass matrix $M$ to account for the different scaling in the non-uniform mesh. Typical iteration counts for the advection equation are between 5 and 20 for a relative tolerance of $10^{-8}$.

The projection systems for the normal vector $n$ and the curvature $\kappa$, steps 3, 5, 6 in the algorithm, are schematically of the form

$$
(M + \gamma K)X = R_X,
$$

(12)

where $X$ denotes a selected component of the nodal values of $n$ or the scalar field $\kappa$, $K$ denotes the stiffness matrix, and $\gamma$ is the amount of smoothing in the projection. The vector $R_X$ contains the evaluated weak forms $\int_{\Omega} \varphi^\Phi \frac{\partial}{\partial x_i} \tilde{\Phi} \, dx$ for the $i$-th component of the projected level set gradient and $\int_{\Omega} \nabla \varphi^\Phi n \, dx$ for the curvature computation, respectively. The magnitude of $\gamma$ is approximately $4h_I^2$, where $h_I$ denotes the maximum element size around the interface. With this choice, the final condition number of the matrix behaves similarly to that of a mass matrix, $O(1)$. Thus, a conjugate gradient method preconditioned by $\text{diag}(M)$ is used. The tolerance is set to $10^{-4}$ for algorithm step 3 and $10^{-7}$ for steps 5 and 6. Mesh sizes in our complex applications are not exactly uniform around the interface despite the same level of refinement and the smallest element size determining the condition number for stiffness matrices can be up to a factor of three smaller than $h_I$. Thus, the local conditioning is affected and iteration numbers between
20 and 100 are observed, about two to four times more than for plain mass matrices.

Finally, the equation system for the level set reinitialization, step 4 in the algorithm, is of the form

\[(M + \Delta \tau \varepsilon \tilde{K}(n)) \delta \Phi = R_R, \tag{13}\]

where \(\tilde{K}(n)\) denotes the (degenerate) stiffness matrix with diffusion along the direction \(\tilde{n}^n\) only and \(R_R\) is the residual of the reinitialization. We set the pseudo time step to \(\Delta \tau = \frac{1}{d^2} h_{\text{min,LS}},\) where \(d = 2, 3\) is the spatial dimension and \(h_{\text{min,LS}}\) is the minimum mesh size in the level set mesh. For this case, the matrix is dominated by the mass matrix and a conjugate gradient method, preconditioned by \(\text{diag}(M)\), is suitable. Typical iteration counts are between 10 and 35.

Turning to the Navier–Stokes equations, after time discretization and linearization, the following block system in velocity and pressure arises:

\[
\begin{pmatrix}
A & B^T \\
B & 0
\end{pmatrix}
\begin{pmatrix}
\delta U \\
\delta P
\end{pmatrix} =
\begin{pmatrix}
R_u \\
R_p
\end{pmatrix}, \tag{14}\]

where \(A = \frac{3}{2\Delta t} M_\rho + C_\rho(u) + \frac{1}{Re} K_\mu\) is a sum of a mass matrix, a convection matrix, and a stiffness matrix. Matrix \(B\) results from the term \(\int_\Omega \nabla \cdot \varphi_i \varphi_j^p \, dx.\) According to equation (1), the mass and convection matrices depend on the density contrast and the stiffness matrix on the viscosity contrast between the two fluids. The particular form of the convection matrix depends on the selected linearization method. For the fully implicit Newton method which is used in this work, it is the result of the following weak form:

\[C_\rho(u)_{i,j} = \int_\Omega \rho^* \varphi_i \cdot \left[ u \cdot \nabla \varphi_j + \varphi_j \cdot \nabla u + \frac{1}{2} u \nabla \cdot \varphi_j + \varphi_j \nabla \cdot u \right] \, dx.\]

The block system (14) is of saddle point structure and solved by an iterative GMRES solver (Saad, 2003). For preconditioning, a block-triangular operator \(P^{-1}\) is applied from the right (Elman et al., 2005), defined by

\[
P = \begin{pmatrix}
A & B^T \\
0 & -S
\end{pmatrix} \iff P^{-1} = \begin{pmatrix}
A^{-1} & A^{-1}B^TS^{-1} \\
0 & -S^{-1}
\end{pmatrix}, \tag{15}\]

where \(S = B A^{-1} B^T\) denotes the Schur complement of block system (14). If the inverse matrices \(A^{-1}\) and \(S^{-1}\) were formed exactly, the matrix underlying the GMRES iteration would be

\[
\begin{pmatrix}
A & B^T \\
B & 0
\end{pmatrix} P^{-1} = \begin{pmatrix}
I & 0 \\
BA^{-1} & I
\end{pmatrix}.
\]
All eigenvalues of this matrix are of unit value with a minimum polynomial of degree two, for which GMRES can be shown to converge in at most two iterations (Elman et al., 2005; Benzi et al., 2005).

Approximations to $A^{-1}$ and $S^{-1}$ are used in our realization. The condition number of the velocity matrix $A$ depends on the size of the time step relative to the size of the velocity and the strength of the viscous term. The time step size $\Delta t$ is of order $h/|u|$ and Reynolds numbers is moderate $Re \leq 50$, such that either the mass matrix term or the viscous term dominate. For the former case, we use an incomplete LU decomposition (ILU) (Saad, 2003) as an approximation of $A^{-1}$, whereas one V-cycle of an algebraic multigrid preconditioner (AMG) based on the software package ML (Tuminaro and Tong, 2000; Gee et al., 2006) is used for the latter case, both provided through the Trilinos library (Heroux et al., 2005). This choice will be evaluated in Sec. 5.5 below. For the Schur complement approximation, discretized differential operators on the pressure space are utilized (Elman et al., 2005). For the time-dependent incompressible Navier–Stokes equations, the action of $S^{-1}$ is approximated by the sum

$$S^{-1} = \frac{3}{2\Delta t}K_{p,\rho}^{-1} + M_{p,\mu}^{-1},$$

where

$$(K_{p,\rho})_{i,j} = \int_{\Omega} \nabla \varphi_{i}^{p} \frac{1}{\rho^{*}} \nabla \varphi_{j}^{p} \, dx \quad \text{ (pressure Laplace matrix)},$$

$$(M_{p,\mu})_{i,j} = \int_{\Omega} \varphi_{i}^{p} \frac{Re}{\mu^{*}} \varphi_{j}^{p} \, dx \quad \text{ (pressure mass matrix)}.$$  

The pressure Laplace and mass matrices are scaled by the inverse density and viscosity, respectively, in order to take up the action of the velocity mass and stiffness operators (Cahouet and Chabard, 1988; Benzi et al., 2005). As boundary conditions for the pressure Laplacian, Neumann boundary conditions are imposed on Dirichlet boundaries for the velocity, and homogeneous Dirichlet conditions are imposed on velocity Neumann boundaries (e.g. outflow) (Turek, 1999). For the augmented pressure elements $Q_{1}^{+}$, the discontinuous ansatz space necessitates the inclusion of face integrals for a consistent discrete Laplace operator. We realize this by a symmetric interior penalty discretization (Arnold et al., 2002), again weighted by the inverse density values. In the implementation of (16), approximations to $M_{p,\mu}^{-1}$ and $K_{p,\rho}^{-1}$ are used instead of exact inverses. For the Laplacian, we choose a V-cycle of the ML-AMG preconditioner with Chebyshev smoothing of degree three (Gee et al., 2006). For the mass matrix inverse, two different strategies are necessary for the Taylor–Hood and augmented Taylor–Hood case, respectively, in order to guarantee convergence that is independent of the mesh size yet
cheap to apply. For the former, an ILU is sufficient, whereas the condition number of the mass matrix in the ATH case scales as \( h^{-2} \) which is adequately approximated by a V-cycle of ML-AMG. For \( Q_1^+ \), a near-null space of two vectors needs to be specified for ML (Gee et al., 2006), including the continuous \( Q_1 \) and discontinuous \( Q_0 \) parts as separate components.

3.1 Implementation

Our solver is implemented in C++ based on the \texttt{deal.II} finite element library (Bangerth et al., 2007, 2015). The code is fully parallelized by domain decomposition with MPI using a framework that has been shown to scale to tens of thousands of processors (Bangerth et al., 2011), realizing adaptive mesh hierarchies on forests of octrees via the \texttt{p4est} library (Burstedde et al., 2011). Distributed matrix algebra and the domain decomposition additive Schwarz methods for the extension of ILU methods to the parallel case as well as the aforementioned ML-AMG is provided by the Trilinos library (Heroux et al., 2005).

Most of the iterative solvers described above spend the bulk of the computing time in matrix-vector products. A particular feature of our algorithms is the use of fast matrix-free methods from Kormann and Kronbichler (2011); Kronbichler and Kormann (2012) for matrix-vector products. For quadratic finite elements and systems of partial differential equations such as the system matrix of the incompressible Navier–Stokes equations linearized by a Newton method, the matrix-free kernels can be up to ten times as fast as matrix-based ones. Similar observations were made by May et al. (2014) in the context of the Stokes equations. The framework also enables the computation of the residual vectors \( R \) in the linear systems (11)–(14) at a cost similar to one matrix-vector product of the respective system. Due to their small costs, all timings reported below include residual computations in the solver time. Besides increasing the performance of matrix-vector products, the matrix-free approach obviates matrix assembly (aside from matrices needed for preconditioners), enabling very efficient Newton–Krylov solvers (Brown, 2010; Kronbichler and Kormann, 2012). In particular for the variable-coefficient matrices of the Navier–Stokes matrix and the level set reinitialization equation, avoiding matrix assembly already saves up to one third of the global run time.

Of course, matrix-free operator evaluation only helps matrix-vector products and not other components in linear solvers. For example, the ILU preconditioners used for the approximations of the inverse of the velocity matrix still need explicit knowledge of matrix entries in order to build the factorization. To limit the cost of the matrix-based velocity ILU, we choose a simplified matrix that only contains the mass matrix, the convective con-
tribution of a Picard iteration, and the velocity Laplacian. In this case, the velocity matrix is block-diagonal except for boundary conditions that mix different velocity components as, e.g., conditions that require the velocity to be tangential or normal to non-Cartesian boundaries. In our solver, we use one and the same matrix for representing all three velocity blocks, increasing performance of the velocity ILU by a factor of 2 because matrix-vector multiplies are performed on three vectors at once. Since this matrix and the factorization are only used as preconditioners, they need to be updated only infrequently when the densities and viscosities have shifted too much. For the case an AMG operator is used for the velocity, matrix-free evaluation can be used to a larger extent: On the finest and most expensive level of the multilevel hierarchy, a Chebyshev smoother involving only matrix-vector products and vector updates (Adams et al., 2003) can be realized. It only needs matrix diagonals and is thus possible to realize without explicitly forming a matrix. The matrix-free evaluation is supplemented with a matrix representation for computing the coarser hierarchies of the AMG, where the matrix drops coupling between velocity components. Similar techniques were also analyzed in Kronbichler et al. (2012); May et al. (2014).

4 The test problem: Flow in a microfluidic chip

In this article, we use a computational model to study the dynamics of a microfluidic chip that enables high-throughput separation of bacteria from human blood cells. The flow in these chips is characterized by low Reynolds numbers, i.e., a laminar flow behavior, which makes it possible to handle and analyze a single particle at a time. The device schematics are shown in Fig. 2. Three inlets including sample fluid, protecting, and acting sheath flow are joined in the active area. Three collectors for small particles, large particles, and waste are connected to the main channel in downstream direction. Through this configuration, large particles are deflected from the original streamline while the path of small particles remains almost unchanged. Experiments (Wu et al., 2009) successfully demonstrated the separation of bacteria (Escherichia coli) from human red blood cells (erythrocytes) through this robust and novel microfluidic process. As a result, a fractionation of two differently sized particles into two subgroups was obtained. The separated cells were proven to be viable. This sorting was based on the size of these bio-particles with size ratio between 3 and 8.

In Fig. 3, we show the partitioning of the computational domain for 6 cores where each contiguously colored segment correspond to a core’s sub-domain. The visualization in Fig. 3 shows only the left half of the computational domain, cut along the x direction in order to visualize the refined
mesh around the interface (see subsection 2.4). We emphasize that the aim of this paper is not to investigate the effectiveness of our method to predict the sorting between particles with respect to their size but to rather present the parallel performance of the selected algorithms. Therefore, we only show results for one single configuration. We consider a particle with radius \( r = 0.25 \) centered at \((x_c, y_c, z_c) = (0.5, 0.5, 0.5)\). The viscosity and the density ratios are set to 20 and 10, respectively. The non-dimensional Weber number is set to \( \text{We} = 3 \) and the Reynolds number is \( \text{Re} = 1.5 \) (measured by the sample channel diameter and sample inflow velocity).

Fig. 4 shows snapshots of a particle moving through the microfluidic device as well as the flow pattern. The particle position and shape are visualized by the level set field with the particle colored in red. The streamlines passing from the sample inlet to the main channel are also depicted in Fig. 4. As expected, the particle deflects to the left due to the strong acting flow and enters the main channel from the curved trajectory. In the main channel, the bubble velocity as about a factor of 30 larger than at the inlet. Fig. 4(b) also shows how the velocity field is modified by the rising particle in the main channel. The trend of the predicted streamline and shape of the particle show a good agreement with experimental results in Wu et al. (2009). In addition, our numerical model characterizes the flow mechanism in the device with much more detail and predicts various features of the particles reasonably well.

5 Performance results

5.1 Experimental setup

The tests are performed on two parallel high-performance computers, Tintin and SuperMUC. Tintin is an AMD Bulldozer-based compute server at the Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX). Each compute server of Tintin consists of two octocore Opteron 6220 processors running at 3 GHz. Tintin provides a total of 2560 CPU cores (160 compute nodes with dual CPUs) and the nodes are interconnected with QDR Infiniband. The cluster is operated with Scientific Linux, version 6.4, and the GCC compiler version 4.8.0 has been used for compilation.

SuperMUC is operated by the Leibniz Supercomputing Center and uses 147,456 cores of type Intel Xeon Sandy Bridge-EP (Xeon E5-2680, 2.7 GHz). This cluster has a peak performance of about 3 Petaflops. The nodes are interconnected with an Infiniband FDR10 technology and contain 16 cores each. The GCC compiler version 5.1 has been used for compilation.
Table 1: Details of the 3D triangulation. TH and ATH designate respectively the Taylor–Hood and the Augmented Taylor–Hood element pairs. Dofs: Degrees of freedom.

<table>
<thead>
<tr>
<th>Active cells</th>
<th>Velocity dofs</th>
<th>Pressure dofs</th>
<th>Level set</th>
<th>max(h)/min(h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>607 104</td>
<td>TH 15 013 701</td>
<td>TH 642 178</td>
<td>16 726 994</td>
<td>0.0209 / 0.0117</td>
</tr>
<tr>
<td>ATH 15 013 701</td>
<td>ATH 1 249 282</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2 Strong (fixed-size) scalability tests

For fixed problem size, we record the computational time spent on up to $N = 2048$ cores. The statistics of the grid are shown in Table 1. An adaptively refined mesh similar to the one in Fig. 3 but with one more level of global refinements is used in these tests. Fig. 5 displays the computation time for 65 time steps versus the core count. This time frame includes at least one dynamic adaptation in the mesh and is therefore representative for the behavior of the solver over longer time intervals. We have verified that the testing time is long enough to keep run time variations negligible. A logarithmic scaling of axes is used. The results show a considerable speedup if computational resources are increased for both the TH and ATH solvers. We observe an almost perfect linear scaling from 32 to 512 cores. When using 512 cores, the global run time is reduced by a factor of 11.0 for the TH discretization (respectively 10.3 for ATH) as compared to the run time at 32 cores. Due to the more involved linear system (more iterations, slightly more unknowns), the computational cost is higher for the ATH finite element pair.

The parallel efficiency of these runs is presented in Fig. 6. The ratios are displayed as compared to the elapsed time on 32 processors. An efficiency of 1 indicates perfect isogranular efficiency, which is reached or even slightly exceeded at 64 cores (due to cache effects). However, we observe a degradation of the performance at the tail end of the plot for 1024 and 2048 cores, where the scaling appears saturated. For the 1024 core case, the number of unknowns per core for the Navier–Stokes part is only around 15 000 and the cost of communication rather than computation becomes dominant below that threshold. From Fig. 7, we observe that at 1024 cores the Navier–Stokes solver consumes most of the global computational time, indicating significantly worse scaling than for the level set components.

More detailed explanations for the slowdown in the Navier–Stokes solver are given in Sec. 5.5 below, including algorithmic reasons. The strong scalability tests reveal an overall very good scaling of the chosen algorithms and implementation: The parallel efficiency of both discretizations remains well above 60% when going from 32 to 512 cores on Tintin (TH: 69%, ATH:
<table>
<thead>
<tr>
<th>Refinement 1</th>
<th>Refinement 2</th>
<th>Refinement 3</th>
<th>Refinement 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active cells</td>
<td>8 800</td>
<td>70 400</td>
<td>563 200</td>
</tr>
<tr>
<td>Velocity dofs</td>
<td>234 651</td>
<td>1 782 099</td>
<td>13 884 195</td>
</tr>
<tr>
<td>Pressure dofs (ATH)</td>
<td>19 609</td>
<td>1 486 17</td>
<td>1 157 233</td>
</tr>
<tr>
<td>Pressure dofs (TH)</td>
<td>10 809</td>
<td>78 217</td>
<td>549 033</td>
</tr>
<tr>
<td>Level-set dofs</td>
<td>255 025</td>
<td>1 969 849</td>
<td>1 548 297</td>
</tr>
</tbody>
</table>

Table 2: Number of degrees of freedom (dofs) for the weak scalability tests component for the Augmented Taylor–Hood (ATH) and Taylor–Hood (TH) element at different refinement levels.

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>4</th>
<th>32</th>
<th>256</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall-clock time (TH/\textit{Tintin}) [s]</td>
<td>469</td>
<td>828</td>
<td>1210</td>
<td>2770</td>
</tr>
<tr>
<td>Wall-clock time (ATH/\textit{Tintin}) [s]</td>
<td>414</td>
<td>817</td>
<td>1400</td>
<td>5030</td>
</tr>
<tr>
<td>Wall-clock time (TH/\textit{SuperMUC}) [s]</td>
<td>156</td>
<td>206</td>
<td>238</td>
<td>305</td>
</tr>
<tr>
<td>Wall-clock time (ATH/\textit{SuperMUC}) [s]</td>
<td>177</td>
<td>247</td>
<td>343</td>
<td>709</td>
</tr>
</tbody>
</table>

Table 3: Wall-clock times for the Augmented Taylor–Hood (ATH) and Taylor–Hood (TH) element when increasing the problem size and the number of cores with the same factor (weak scaling).

64%). Fig. 7 also demonstrates that the time spent in support functionality such as the adaptive mesh refinement, assembling the preconditioner, or output analysis, remains below 4%.

5.3 Weak scalability tests

In this subsection, we assess the weak scaling of our solver on the \textit{Tintin} and \textit{SuperMUC} systems, respectively. For the weak scaling study, we simultaneously increase the problem size and the number of cores while keeping the problem size per core constant (the arithmetic load per processor in terms of degrees of freedom is constant, disregarding possible increases in solver iterations). Table 2 lists the problem sizes used in this test. We increase the core count in steps of 8 from 4 to 2048 to reflect isotropic refinement of the base mesh. The smallest problem contains 8 800 cells and the largest one 4 505 600. The measured run times for the two sets of tests are displayed in Table 3. The run time is also plotted in Fig. 8, but broken down into different parts of the algorithm.

We notice that the global run time increases slightly as the mesh is refined, with the most pronounced increase between 4 and 32 cores where memory bandwidth limitations within the node contribute substantially. Broken down into the solver components, the computing time for the level set parts only increases mildly. This effect is clearly visible from Fig. 9 where the
Navier–Stokes shows the worst scaling in both Navier–Stokes discretization. For instance, the parallel efficiency of the Navier–Stokes solver for the TH case degrades to 20% at 256 cores where all other parts achieve more than 40% of parallel efficiency. On SuperMUC, the weak scalability appears more favorable. Fig. 10 shows a combined strong and weak scalability study, starting from refinement level 2 and going to level 5 with approximately one billion degrees of freedom in the Navier–Stokes system. The level set components (LSC, Reinit) scale almost perfectly up to 16,384 cores. As also observed from Fig. 7, this behavior leads to the fact that the level set components that dominate the computational time at small problem sizes and low processor counts become very modest. Note that the level set mesh was chosen by a factor of three finer than the Navier–Stokes mesh, giving the best overall accuracy for a given computational input.

### 5.4 Node performance

In this subsection we detail the computational behavior within the shared memory region of a node in order to quantify the balance of computationally bound components versus memory bandwidth bound components. To this end, we again consider strong scaling tests on Tintin and SuperMUC. The problem setting is displayed in Table 4, using a mesh with one level of adaptive refinement around the interface.

Table 5 collects the global execution time of both the TH and ATH discretizations on Tintin and SuperMUC, including a breakdown of times into the major solver components such as level set computations, reinitialization, and Navier–Stokes solver. On each system, we notice an improvement of more than a factor of 10 in the global compute time when going from 1 to 16 cores. Its worth to point out that each component runs at least three times as fast on SuperMUC than on Tintin. Moreover, the various components of the algorithm behave differently. The Navier–Stokes part takes respectively 23% and 15% of the global computational time on SuperMUC and Tintin using the TH version of the algorithm, and 40% and 30% of the time for ATH. The share of level set computations is between 35% and 27% for the TH and ATH cases on both systems. On Tintin, reinitialization uses about 30 – 36% of the compute time while on SuperMUC this percentage

<table>
<thead>
<tr>
<th>Active cells</th>
<th>Velocity dofs</th>
<th>Pressure dofs</th>
<th>Level set</th>
<th>max(h)/min(h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TH 83 588</td>
<td>TH 2 128 323</td>
<td>TH 93 095</td>
<td>ATH 2 128 323</td>
<td>2 349 479</td>
</tr>
</tbody>
</table>

Table 4: Details of the 3D triangulation used for a strong scaling test within one node.
Table 5: Comparison of run times measured on both SuperMUC and Tintin systems through a strong scaling test where the problem described in Table 4 is considered. For each version of the solver, the table reports the total wall-clock time (Global) and the timing for the major components of the solver such as, Level set computations (LSC), reinitialization (Reinit) and the Navier–Stokes solver (NSSv) as the core count increases. All run times are presented in seconds.

is reduced to 18 – 20%, illustrating the more effective vectorization on the processors of the latter system.

Table 5 also shows that components which involve only matrix-free evaluation such as the reinitialization algorithm scale considerably better within one node than algorithms which involve sparse matrix kernels: ILU methods are used within Navier–Stokes preconditioner and the normal vector computation within the LSC part is based on sparse matrix-vector products with 3 vectors processed simultaneously rather than the matrix-free evaluation due to its improved overall performance.

5.5 Scaling of Navier–Stokes solver

Now we consider the Navier–Stokes linear solver in more detail. This is the most important aspect of this work since the level set solver shows better scaling and its cost can be adjusted as desired by choosing the level of additional refinement beyond the Navier–Stokes mesh.

Fig. 11 shows weak (different lines in the same color) and strong (along connected lines) scaling in the components of the Navier–Stokes solver individually. The matrix-vector products scale almost perfectly. For weak scaling from 32 to 16384 processors, the time for a matrix-vector product goes from 5.3 ms to 5.65 ms, i.e., about 94% parallel efficiency. In terms of arithmetic performance, the matrix-vector product reaches between 30% and 40% of the theoretical arithmetic peak on SuperMUC. For instance,
the matrix-vector product reaches 191 TFlops out of 603 TFlops possible at 32,768 cores. This number is based on the approximately 14,900 floating point operations that are done per cell in the linearized Navier–Stokes operator for our implementation (or about 600 operations per degree of freedom). The slight deviation from ideal scaling observed in Fig. 11 is due to an increase in the number of linear iterations for larger problem sizes and larger processor counts. The ILU preconditioner time per application scales even slightly better than matrix-vector products because the no-overlap policy in the additive Schwarz operation removes coupling and thus operations when more processors are used; on the other hand, this is responsible for the considerable increase in linear iterations as shown in Table 6 below.

We emphasize that a scalar ILU matrix representation for the velocity preconditioner and appropriate sparse matrix kernels is used in order to improve performance by a reduced memory transfer of almost a factor of three. Despite this optimization, the matrix-vector product for the Navier–Stokes matrix is only approximately 1.3 to 1.5 times as expensive. If sparse matrices were used for the Navier–Stokes matrix, those would be by a factor of 6 to 10 more expensive than the chosen matrix-free approach, and slow down the whole solver by a factor of between two and three for moderate processor counts. Fig. 12 shows a comparison of the proposed matrix-free solver with matrix-based variants for the problem with 14.4m and 114m degrees of freedom, respectively. The improved matrix-vector products clearly improve the overall performance. Fig. 12 also adds the time to recompute the system matrix in each Newton step in order to compare with the total cost of the matrix-based solver, using standard assembly routines from deal.II (Bangerth et al., 2015). In this metric, we see a three to fourfold improvement of the presented algorithms to large scales despite the close-to-ideal parallel scaling of matrix assembly. Moreover, the runs at 64 and 512 cores, respectively, did not complete for the matrix-based solver because the matrices and factorizations did not fit into the 2GB memory per core available on SuperMUC. Obviously, we observe better strong scaling for the matrix-based solver because the component with the worst scaling, the pressure AMG, is the same in both cases.

The cost in the pressure Poisson inversion by the ML-AMG and the vector operations scale considerably worse than the matrix-vector products since these involve global communication. In particular the pressure Poisson operator is affected by the small local problem sizes: The strong scaling shown in Fig. 11 goes from approximately 9,000 degrees of freedom per core down to 550 degrees of freedom per core with the obvious implications on the communication cost. As also observed in Sundar et al. (2012), scaling of algebraic multilevel preconditioners to the largest core counts represents a serious difficulty. In addition to bottlenecks while solving, the setup phase
for AMG takes an increasing time with large processor counts as represented by the increasing “NSSt” times in Fig. 10. We note that the numbers presented here are averages over timings taken separately on each core (without synchronization that would disturb the time measurements at that size). Operations that only involve nearest-neighbor communication can advance asymmetrically, leading to idle times in later stages. This is partly responsible for the superlinear scaling in vector operations in the early stages of strong scaling (another factor is cache effects in the Gram–Schmidt orthogonalization of GMRES).

Finally, we detail the iteration numbers and cost per linear iteration for the Navier-Stokes solver. Table 6 lists the average number of linear iterations per time step (accumulated over the nonlinear iteration) and compares the execution time consumed by one linear iteration in the Navier–Stokes Solver part (NSSv) for strong scaling according to Sec. 5.2 above. We observe that the time per linear iteration starts growing from 1024 cores onwards for both element pairs, indicating the overwhelming communication, mostly due to the pressure Poisson (AMG) preconditioner. This is illustrated prominently in Fig. 7 which shows the share of run time of the main solver components. At 2048 CPU cores, the Navier–Stokes solver requires 70% of the global run time. On the other hand, the suboptimal scaling below 512 cores can mostly be attributed to the increase in linear iterations due to the additive Schwarz realization of the velocity ILU.

Fig. 13 shows the number of iterations for a weak scaling experiment. As in Table 6, we observe a degradation of the ILU preconditioner as the problem size grows. This is expected at constant time step sizes because the diffusive term increases in strength as the mesh is refined. The iteration number for a velocity AMG preconditioner, on the other hand, remains almost constant as the mesh is refined, indicating algorithmic optimality. We note, however, that one linear iteration with the velocity AMG preconditioner is more than twice as expensive as with the velocity ILU, see also Fig. 12. Hence, the AMG preconditioner yields a faster time to solution only

<table>
<thead>
<tr>
<th>Element</th>
<th>Number of cores</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>TH</td>
<td>Total time elapsed [s]</td>
<td>1570</td>
<td>825</td>
<td>595</td>
<td>412</td>
<td>246</td>
<td>637</td>
<td>612</td>
</tr>
<tr>
<td></td>
<td>Lin. iterations/time step</td>
<td>30.2</td>
<td>31.5</td>
<td>34.9</td>
<td>37.4</td>
<td>39.5</td>
<td>43.4</td>
<td>35.8</td>
</tr>
<tr>
<td></td>
<td>Time/linear iteration [s]</td>
<td>0.79</td>
<td>0.4</td>
<td>0.26</td>
<td>0.16</td>
<td>0.09</td>
<td>0.22</td>
<td>0.26</td>
</tr>
<tr>
<td>ATH</td>
<td>Total time elapsed [s]</td>
<td>6200</td>
<td>3510</td>
<td>2030</td>
<td>1300</td>
<td>797</td>
<td>874</td>
<td>2820</td>
</tr>
<tr>
<td></td>
<td>Lin. iterations/time step</td>
<td>49.6</td>
<td>55.1</td>
<td>56.3</td>
<td>57.1</td>
<td>56.8</td>
<td>43.3</td>
<td>55.1</td>
</tr>
<tr>
<td></td>
<td>Time/linear iteration [s]</td>
<td>1.92</td>
<td>0.97</td>
<td>0.55</td>
<td>0.35</td>
<td>0.21</td>
<td>0.31</td>
<td>0.78</td>
</tr>
</tbody>
</table>

Table 6: Linear iteration in the Navier–Stokes solver in a strong scaling setting (on Tintin). We present the time in seconds required to perform a single linear iteration.
for the largest case with 16,384 cores. Similar observations were made when using an inner BiCGStab solver preconditioned by ILU: Even though the iteration number of the outer GMRES solver go down, the increase in cost for the inner solver outweighs the gain. The poor quality of the ILU only affects the solution at large processor counts (or larger CFL numbers), whereas the quality of the Schur complement approximation dominates at smaller core counts, even in the constant CFL case. Fig. 13 also includes the minimum and maximum number of linear iterations taken in a specific time step. During the first time steps and whenever the mesh is refined/coarsened, the quality of the extrapolation according to the algorithm in Sec. 2.5 is reduced and more linear iterations are necessary. Likewise, due to the interface motion and the shift in material properties, the preconditioner quality degrades over time, until a heuristic strategy forces the re-computation of the preconditioner.

5.6 Simulations of the full device dynamics

In Fig. 14, we show the behavior of the solver over a full simulation of the particle flowing through the microfluidic device. In this simulation, two levels of global refinement and three more levels of adaptive refinements around the interface are selected, using approximately 1.6 million elements and 43 million degrees of freedom for the velocity. This results in a mesh size of around 0.004, spanning almost three orders of magnitude compared to the global geometry. For the simulation, variable time step sizes have been used with the step size set to meet the following two goals:

- The CFL number in the advection equation should not exceed 0.6.
- Condition (8) must be fulfilled.

Up to $t \approx 1.55$, the latter condition dominates (with changes due to the different mesh size), where the variations in the time step size are due to variations in the mesh size around the bubble. At later times when the bubble reaches the main channel, the particle is accelerated considerably and the CFL condition decreases the time step size down to approximately $2.5 \cdot 10^{-5}$. In the last phase, strong forces act on the particle, which result in an increase of the number of linear iterations.

In this simulation, more than 25,000 time steps have been performed. Due to the small mesh sizes, the AMG preconditioner was used for the velocity. Overall, the mesh was adapted 3,500 times in order to follow the bubble motion. The total simulation time on 1,024 cores was 40 hours, with about 34% of time spent in the Navier–Stokes solver and 24% in the AMG preconditioner setup (which was called approximately every second
time step). 21% of the computational time was spent in the level set reini-
tialization, 16% in the other level set computations, 3% in mesh refinement 
and solution transfer functions, and the remaining 2% of time in solution 
analysis and input/output routines.

6 Conclusions

In this work, a massively parallel finite element solver for the simulation of 
multiphase flow in a microfluidic device has been presented. Spatial dis-
cretizations of the incompressible Navier–Stokes equations with inf–sup sta-
ble Taylor–Hood and augmented Taylor–Hood elements have been used, re-
spectively. For time discretization, the Navier–Stokes and the level set part 
have been split into two systems by extrapolation. Within each field, implicit 
BDF-2 time stepping has been used. First, the level set is advanced by a 
second-order extrapolated velocity field. The incompressible Navier–Stokes 
equations are evaluated with surface tension from the updated level set field 
and solved as one full velocity–pressure system with a block-triangular pre-
conditioner. Scalability of the chosen algorithms up to approximately one 
billion degrees of freedom in each of the Navier–Stokes and the level sys-
tem has been demonstrated. For fixed problem sizes, almost linear strong 
scaling has been observed as long as the local problem size is larger than 
approximately 30 000 degrees of freedom. Weak scaling tests show very good 
performance, with some degradation in the Navier–Stokes solver due to the 
non-optimal ILU preconditioner in the velocity. Better scaling is possible 
with multigrid-based preconditioners. Our experiments show that for low 
and moderate processor counts and small to moderate CFL numbers, ILU is 
the better choice in terms of computational time, in particular when the ILU 
is only based on the scalar convection–diffusion operator in velocity space. 
The picture only changes when the number of processors exceeds approxi-
mately 10 000 and for CFL numbers larger than approximately 2, when the 
better scaling in the AMG operators encourages to switch to that method. 
Future work will include the derivation of geometric multigrid components 
within the velocity operator in order to reduce the bottleneck from AMG at 
the largest core counts, using strategies similar to Sundar et al. (2012).

The key ingredient to our solvers are fast matrix-free implementations 
of matrix-vector products. Our results show an improvement of more than 
a factor of two in pure solver times over the best matrix-based algorithm. 
Taking into account also the cost of matrix assembly, the advantage grows 
in favor of our algorithm. While this improvement either decreases time to 
solution or allows for larger problems on the same computational resources, 
it reaches communication limits earlier when used in a strong scaling setting.
Detailed analysis of run times of selected solver components shows that as soon as the run time goes below approximately 0.5–1 s per time step in the Navier–Stokes solver (or approximately 0.2–0.5 s for the solution of one linear system), the communication cost in the algebraic multigrid preconditioner for the pressure Poisson operator prevents further scaling.

Funding

The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer SuperMUC at Leibniz Supercomputing Centre (LRZ, www.lrz.de) through project id pr83te. Some computations were performed on resources provided by SNIC through Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX) under project p2010002. The financial support from the eSSENCE strategic research program and the Swedish Research Council, as well as the technical assistance from UPPMAX are gratefully acknowledged.

References


Figure 1: To the left: The domain $\Omega$ occupied by two immiscible fluids separated by an interface $\Gamma$. $\rho_i$ and $\mu_i$ denote respectively the density and viscosity in $\Omega_i$. To the right, a 2D snapshot of the level set function $\Phi$ and an adaptive mesh is depicted.

Figure 2: The microfluidic chip.

Figure 3: Visualization of the geometric partitioning of the grid on 6 processors. The computational domain in this example is sliced at $x = 0.5$ along the $x$-direction.
(a) After a few time steps
(b) At $t = 1.6$ (range of large acceleration forces)

Figure 4: Streamline, velocity field, and the flowing particle in the device.

Figure 5: Wall-clock time for constant total work on a 3D test problem using the Taylor–Hood and Augmented Taylor–Hood element pairs on Tintin.

Figure 6: Parallel efficiency on 3D test problem using the Taylor–Hood element pairs and the Augmented Taylor–Hood at constant total work (strong scaling) on Tintin.
Figure 7: Left: Distribution of the computational time spent in various blocks of the algorithm on Tintin. LSC: Level set computations (level set advance, normal, curvature, Heaviside, force calculations, i.e., steps 2, 3, 5–8 in the algorithm in Sec. 2.5), Reinit: reinitialization (algorithm step 4), NSSv: Navier–Stokes solver including residual computation (algorithm step 9), NSSt: Navier–Stokes setup (setup and assembly of matrices, computation of preconditioners). “Others” gathers the time for output, grid adaptation, and solution interpolation between different grids.

Figure 8: Total wall-clock time and timing for the major components of the solver on Tintin for weak scaling. A log-log scale is used in these plots. LSC: Level set computation (computing Heaviside, curvature, force, normal and advance the level set), Reinit: Reinitialization step, NSSv: Navier–Stokes solver.
Figure 9: Parallel efficiency on 3D test problem at constant total work per core (weak scaling) on *Tintin*.

Figure 10: Scaling results on *SuperMUC*. Four different problem sizes at 70k, 563k, 4.5m, and 36m elements are shown for 5 processor configurations each (except for the largest size), involving between 232k and 14k Navier–Stokes dofs per core. The data points for strong scaling are connected, with breaks for weak scaling.
Figure 11: Scaling results for Navier–Stokes solver components according to the algorithm described in Sec. 3 on SuperMUC, where each set of lines is for problem sizes with 70k, 563k, 4.5m, and 36m elements, respectively. “NS mat-vec” refers to matrix-vector products with the Navier–Stokes matrix, “Vel ILU” to multiplication with the ILU approximation of the inverse velocity matrix, “Div mat-vec” multiplication by the transpose of the divergence matrix $B^T$, “Pres ILU” the inverse pressure mass matrix approximation, “Pres AMG” the inverse pressure Poisson matrix approximation, and “Vector ops” the orthogonalization work done for the GMRES iterative solver, including global communication through inner products.

Figure 12: Scaling comparison of proposed matrix-free solver (MF) with matrix-based solvers (SpMV) on SuperMUC. For the matrix-based solver, the cost of actually assembling the sparse matrices approximately twice per time step is also included. The matrix-based ILU preconditioners use two options: on the scalar convection operator and Laplacian (ILU scalar) vs. ILU for the complete linearized velocity matrix (ILU all). AMG is based on the vector Laplacian plus vector convection–diffusion operator without coupling between velocities.
Figure 13: Average number of linear iterations per time step (lines) including the minimum and maximum number of iterations (vertical bars) in a weak scaling setting. Results are shown for Taylor–Hood element with velocity ILU and velocity AMG as well as the augmented Taylor–Hood element (ATH ILU).

(a) Constant time step $\Delta t = 0.0001$  
(b) Constant CFL, $\Delta t = 0.0016 \ldots 0.0001$

Figure 14: Number of iterations per time step (including the average over 500 time steps as a bold line) and time step size over long-time simulation over more than 25 000 time steps.
Recent licentiate theses from the Department of Information Technology

2015-005  Siyang Wang: Analysis of Boundary and Interface Closures for Finite Difference Methods for the Wave Equation

2015-004  Pavol Bauer: Parallelism and Efficiency in Discrete-Event Simulation

2015-003  Fredrik Hellman: Multiscale and Multilevel Methods for Porous Media Flow Problems

2015-002  Ali Dorostkar: Developments in preconditioned iterative methods with application to glacial isostatic adjustment models

2015-001  Karl Ljungkvist: Techniques for Finite Element Methods on Modern Processors

2014-007  Ramūnas Gutkovas: Advancing Concurrent System Verification: Type based approach and tools

2014-006  Per Mattsson: Pulse-modulated Feedback in Mathematical Modeling and Estimation of Endocrine Systems

2014-005  Thomas Lind: Change and Resistance to Change in Health Care: Inertia in Sociotechnical Systems

2014-004  Anne-Kathrin Peters: The Role of Students’ Identity Development in Higher Education in Computing

2014-003  Liang Dai: On Several Sparsity Related Problems and the Randomized Kaczmarz Algorithm

2014-002  Johannes Nygren: Output Feedback Control - Some Methods and Applications