Tormod Bjøntegaard

High order methods for incompressible fluid flow: Application to moving boundary problems

Thesis for the degree philosophiae doctor

Trondheim, April 2008

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics
and Electrical Engineering

NTNU

NTNU
Norwegian University of Science and Technology

Thesis for the degree philosophiae doctor

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Mathematical Sciences

ISBN 978-82-471-8950-4 (printed version)
ISBN 978-82-471-8964-1 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2007:144

Printed by NTNU-trykk

Acknowledgements

This thesis is the result of my work for the degree of Philosophiae Doctor at Department of Mathematical Sciences, Norwegian University of Science and Technology (NTNU). The project has been funded by NTNU as an RSO project under the ICT thematic area.

First, I would like to thank my supervisor, Professor Einar M. Rønquist, for the collaboration these past four years. His support and guidance has been invaluable throughout this work. I would also like to thank my colleagues for making this a friendly and nice workplace. A special mention goes to Achenef and Xavier for a respectable number of coffee breaks, Arne Morten for both relevant and non-relevant discussions, and Øystein for all those football matches. Finally, I would like to thank my family for always supporting me.

Tormod Bjøntegaard
Trondheim, April 2008

Contents

This thesis consists of an introduction and six papers:

- **Paper I**: *A high order splitting method for time-dependent domains*, by Tormod Bjøntegaard and Einar M. Rønquist. Submitted to *Computer Methods in Applied Mechanics and Engineering*.
- **Paper II**: *Imposing free-surface boundary conditions using surface intrinsic coordinates*, by Tormod Bjøntegaard. Preprint Numerics No.5/07, Department of Mathematical Sciences, Norwegian University of Science and Technology.
- **Paper III**: *Simulation of three-dimensional Bénard-Marangoni flows including deformed surfaces*, by Tormod Bjøntegaard and Einar M. Rønquist. To appear in *Communications in Computational Physics*.
- **Paper IV**: *High order methods for incompressible fluid flow: Application to free surface problems*, by Tormod Bjøntegaard and Einar M. Rønquist. In *MekIT'07*, B. Skallerud and H.I. Anderson (eds.), Tapir Akademisk Forlag, 2007.
- **Paper V**: *Accurate interface-tracking for arbitrary Lagrangian-Eulerian schemes*, by Tormod Bjøntegaard and Einar M. Rønquist. Submitted to *Journal of Computational Physics*.
- **Paper VI**: *Fast tensor-product solvers. Part I: Partially deformed three-dimensional domains*, by Tormod Bjøntegaard, Yvon Maday and Einar M. Rønquist. Submitted to *Journal of Scientific Computing*.

Introduction

Fluid flows with moving boundaries are encountered in a large number of real life situations, with two such types being fluid-structure interaction and free-surface flows. Fluid-structure phenomena are for instance apparent in many hydrodynamic applications; wave effects on offshore structures, sloshing and fluid induced vibrations, and aeroelasticity; flutter and dynamic response. Free-surface flows can be considered as a special case of a fluid-fluid interaction where one of the fluids are practically inviscid, such as air. This type of flows arise in many disciplines such as marine hydrodynamics, chemical engineering, material processing, and geophysics. The driving forces for free-surface flows may be of large scale such as gravity or inertial forces, or forces due to surface tension which operate on a much smaller scale. Free-surface flows with surface tension as a driving mechanism include the flow of bubbles and droplets, and the evolution of capillary waves.

In this work we consider incompressible fluid flow, which are governed by the incompressible Navier-Stokes equations. There are several challenges when simulating moving boundary problems numerically, and these include

- Spatial discretization
- Temporal discretization
- Imposition of boundary conditions
- Solution strategy for the linear equations

These are some of the issues which will be addressed in this introduction. We will first formulate the problem in the arbitrary Lagrangian-Eulerian framework, and introduce the weak formulation of the problem. Next, we discuss the spatial and temporal discretization before we move to the imposition of surface tension boundary conditions. In the final section we discuss the solution of the resulting linear system of equations.

1 The Navier-Stokes equations

The incompressible Navier-Stokes equations in strong form are given by

$$\begin{aligned} \rho\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right) &= \frac{\partial \sigma_{ij}}{\partial x_j} + \rho f_i, && \text{in } \Omega(t) \\ \frac{\partial u_j}{\partial x_j} &= 0, && \text{in } \Omega(t) \\ &+ \text{boundary conditions}, && \text{on } \partial\Omega(t) \\ &+ \text{initial conditions}, \end{aligned} \tag{1}$$

where

$$\sigma_{ij} = \left(-p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right)$$

is the stress tensor. Here, $\Omega(t)$ is our time-dependent computational domain, ρ is the fluid density, u_i is the i'th component of the fluid velocity, x_i is the i'th cartesian co-ordinate direction, p is the pressure, μ is the dynamic viscocity and f_i is the i'th component of the body force. Summation over repeated indices is assumed.

1.1 ALE framework

One important aspect in a numerical simulation of moving boundary problems is the choice of framework in which to formulate the problem. The two extremeties here are the pure Eulerian and Lagrangian formulations. With an Eulerian approach, the grid is fixed which obviously has the advantage that we do not need to worry about grid distortion. However, tracking a moving boundary in this framework is not straightforward, and this issue is generally dealt with by using marker methods or volume of fluid methods [52]. In a Lagrangian framework the grid nodes move with the fluid particles, and the interface or boundary is naturally tracked. However, a disadvantage with such an approach is that large surface motions may soon lead to poor grid quality. The obvious advantages and disadvantages of the pure Eulerian and Lagrangian description lead to the advent of the arbitrary Lagrangian-Eulerian(ALE) formulation, which represents a mix between the two.

An essential feature in an ALE formulation of the Navier-Stokes equations is that the time derivative is represented with respect to a fixed reference configuration, Ω_0. From the reference configuration there exists a map $\mathcal{A}(t) : \Omega_0 \rightarrow \Omega(t)$ which at any time, t, associates this reference configuration with the physical domain, $\Omega(t)$ [22].

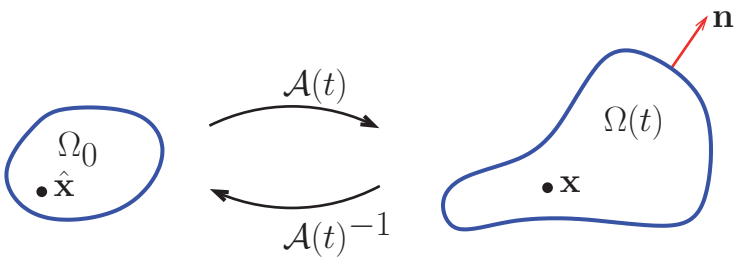


Figure 1: A mapping, $\mathcal{A}(t)$, from a fixed reference domain, Ω_0, to the physical domain, $\Omega(t)$.

Then,

$$\hat{\mathbf{w}} = \left(\frac{\partial \mathcal{A}(t)}{\partial t} \right) \Bigg|_{\hat{\mathbf{x}}}$$

is the domain velocity represented on the reference configuration, and

$$\mathbf{w} = \hat{\mathbf{w}} \circ \mathcal{A}(t)^{-1}.$$

The *non-conservative* ALE formulation of the Navier-Stokes equations in strong form now reads,

$$\begin{aligned}
\rho\left(\frac{\mathrm{D}u_i}{\mathrm{D}t}+(u_j-w_j)\frac{\partial u_i}{\partial x_j}\right) &= \frac{\partial}{\partial x_j}\left(-p\delta_{ij}+\mu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)\right)+\rho f_i, && \text{in } \Omega(t)\\
\frac{\partial u_j}{\partial x_j} &= 0, && \text{in } \Omega(t) \qquad (2)\\
&+ \text{boundary conditions} && \text{on } \partial\Omega(t)\\
&+ \text{initial conditions}
\end{aligned}$$

where

$$\frac{\mathrm{D}}{\mathrm{D}t}=\frac{\partial}{\partial t}+w_j\frac{\partial}{\partial x_j}$$

is the time derivative in the ALE frame. With no loss in generality, we will assume that $\rho = 1$.

The ALE framework was first proposed in the context of finite difference methods in the mid sixties [45, 23, 29], and has been subject to a lot of research activity since then. In the eighties the ALE formulation was used with finite element discretizations [32, 15].

In the nineties special attention was devoted to the geometric conservation law. A numerical scheme which satisfies the discrete geometric conservation law (DGCL) should be able to exactly reproduce the constant solution (under the assumtion that this is consistent with initial and boundary conditions.) The satisfaction of the DGCL is related to how the geometric factors are chosen during the time integration of an ALE scheme. Much effort has been devoted to analyze stability and accuracy properties when moving time-advancing schemes from a fixed grid to a moving grid. In a series of papers, Farhat and co-workers investigated this [37, 17, 16], and found that the satisfaction of the DGCL is advantageous for some cases. Formaggia and Nobile [21, 22] concluded that although it certainly does no harm satisfying the DGCL, its importance in the general case remains uncertain.

In the late eighties the ALE framework was first used with a spectral element discretization by Ho and Patera [30]. A spectral element discretization relies on a regular mapping from the reference domain to the physical domain in order to maintain good interpolation and quadrature properties. In order to achieve this, it is very important to have a satisfying grid on the boundary of the domain. Interior grid points may for instance be obtained by using the Gordon-Hall algorithm [25].

Upon discretization, $\mathbf{w}$ will be associated with specific grid-points and will in that context represent the *mesh-velocity*. For $\mathbf{w}$ on the boundary, $\Gamma(t) = \partial\Omega(t)$, we have the kinematic requirement that,

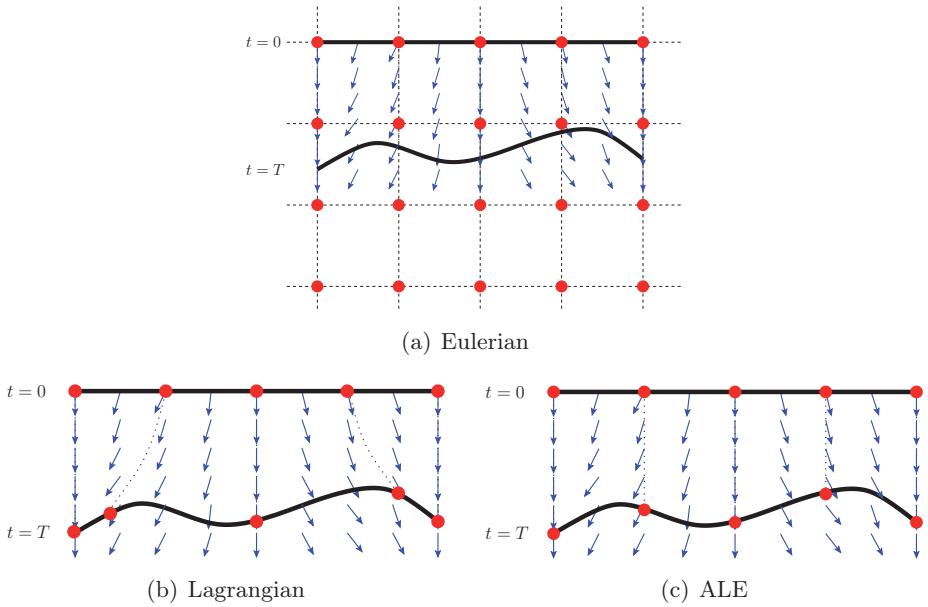
$$\mathbf{w}\cdot\mathbf{n} = \mathbf{u}\cdot\mathbf{n} \qquad \text{on } \Gamma(t).$$

However, there is significant freedom in the choice of the tangential component of $\mathbf{w}$ on the surface, as well as the mesh velocity in the interior of the domain. Once $\mathbf{w}$ is obtained, the grid position is updated by integrating

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{w}.$$

In Figure 2 we illustrate the difference between the three frameworks mentioned above. We assume that we solve a two-dimensional moving boundary problem using these frameworks, and that the front depicted in Figure 2 is part of the boundary of the two-dimensional

computational domain. The indicated velocity field will here be the solution to the underlying problem. Using an Eulerian framework, the initial front is located at the grid points, but this is clearly not the case at $t = T$. Hence, the location of the boundary is not explicitly tracked, as mentioned before. For the Lagrangian and ALE approaches we only focus on the boundary grid. For both these cases, the point distribution at $t = 0$ is good, but while both approaches are able to explicitly track the front, the Lagrangian mesh at $t = T$ doesn't have an "optimal" distribution compared to the initial mesh. For the ALE situation, a clever choice of $\mathbf{w}$ has made sure that the grid points both follow the front *and* maintain a good distribution.



(a) Eulerian

(b) Lagrangian

(c) ALE

Figure 2: The motion of a front and the underlying grid using an Eulerian, Lagrangian and ALE description.

A traditional way to update the grid on the boundary is to satisfy the kinematic requirement $\mathbf{w} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n}$ at the grid nodes in the normal direction and apply homogeneous Dirichlet or Neumann conditions in the tangential direction, and then at regular intervals perform a remeshing strategy. In this manner the grid maintains a satisfactory quality at all times, however, remeshing is computationally demanding and requires interpolation of all relevant field variables from the old distorted grid to the new grid. In a spectral element simulation of smooth transient flow where you have spectral convergence in space, it may also be desirable to get more than first order convergence in time. However, a scheme with more than first order convergence in time for fixed geometry will probably not maintain its temporal rate of convergence during the transient phase when subjected to a remeshing strategy. Hence, it would be desirable to be able to update the boundary in a manner that maintains more than first order convergence in time *and* at all times maintains a good point distribution without the need for a remeshing strategy; see Figure 2c.

A common choice to extend the mesh velocity to the interior of the computational domain is by solving an elliptic equation. In [51, 13] this is done by modelling the mesh as an elastic solid body by solving an elasticity equation. In [5] Bouffanais and Deville propose to compute the grid-velocity in the interior of each domain by solving a steady Stokes problem such that

the DGCL was automatically satisfied, and this technique was applied to moving boundary problems in [4].

1.2 Weak formulation

We will in this work consider discretizations based on a weak formulation of the problem (2). The main reasons for this choice is that the weak formulation of the problem yields a lower regularity requirement on $\mathbf{u}$ and p compared to the strong formulation, as well as convenient imposition of natural boundary conditions. The weak form offers a starting point for finite element based discretizations, which are well suited for problems in complex geometries.

We now assume that the boundary of $\Omega(t)$ is divided into two parts; $\Gamma_d(t) = \partial\Omega_d(t)$ which has prescribed Dirichlet boundary conditions and $\Gamma_\gamma(t) = \partial\Omega_\gamma(t)$ where Neumann (or stress) boundary conditions are imposed. Hence we have, $\Gamma(t) = \Gamma_d(t) \cup \Gamma_\gamma(t)$. We now introduce the function spaces:

$$
\begin{aligned}
X &= \{v(t) \in H^1(\Omega(t)),\, v(t) = 0 \qquad \text{on } \Gamma_d(t)\},\\
X_b &= \{v(t) \in H^1(\Omega(t)),\, v(t) = v_b(t) \quad \text{on } \Gamma_d(t)\},\\
Y &= \{q(t) \in L^2(\Omega(t))\},
\end{aligned}
$$

where $v_b(t)$ are given Dirichlet boundary conditions.

We multiply (2) with test functions v_i and q, and assume that the test functions, v_i, are not time-dependent on the reference domain, Ω_0,

$$\frac{\mathrm{D}v_i}{\mathrm{D}t} = 0.$$

Next, we define $\mathbf{J}$ to be the Jacobian matrix associated with the mapping $\mathcal{A}(t)$ from Ω_0 to $\Omega(t)$, with matrix elements

$$J_{ij} = \frac{\partial x_i}{\partial \hat{x}_j},$$

and apply Euler's expansion formula [1]

$$\frac{\mathrm{D}J}{\mathrm{D}t} = J\frac{\partial w_j}{\partial x_j},$$

where J is the determinant of $\mathbf{J}$.

We arrive at the *conservative* ALE formulation: Find: $u_i \in X_b$ and $p \in Y$ such that

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} v_i u_i \,\mathrm{d}V + \int_{\Omega(t)} v_i(u_j - w_j)\frac{\partial u_i}{\partial x_j}\,\mathrm{d}V - \int_{\Omega(t)} v_i u_i \frac{\partial w_j}{\partial x_j}\,\mathrm{d}V&\\
= \int_{\Omega(t)} \left(-\frac{\partial v_i}{\partial x_j}\sigma_{ij} + v_i f_i\right)\mathrm{d}V + \int_{\Gamma_\gamma(t)} v_i \sigma_{ij} n_j \,\mathrm{d}S, \quad \forall v_i \in X, \qquad (3)&\\
\int_{\Omega(t)} q\frac{\partial u_j}{\partial x_j}\,\mathrm{d}V = 0, \qquad \forall q \in Y.&
\end{aligned}
$$

We observe that we get a surface integral from integration by parts on the viscous term, and it is through this integral natural boundary conditions will be imposed.

1.3 Spatial treatment

The appropriate choice of spatial discretization for the Navier-Stokes equations is highly dependent on the problem at hand. For regular geometries a finite difference scheme may be suitable, not the least for its ease of implementation. For problems with a complex geometry, a finite element discretization may be most reasonable, while a finite volume discretization may be chosen for its conservation properties. Here, we will focus on the spectral element method. This method was first introduced by Patera in [46], and it exploits the accuracy properties of the spectral method [26] and the flexibility of the finite element method. Important analysis and convergence results for the spectral element method were presented in [40]. An essential feature of spectral element methods is that the exponential convergence of the pure spectral method is maintained as long as the solution and the geometry is infinitely smooth inside each element, *and* the geometry and the solution is C^0-continuous on the element boundaries.

We will use a spectral element method to discretize (3) in space. This involves decomposing $\Omega(t)$ into suitable elements, and on each element the unknowns are projected to polynomial spaces. We will here use an isoparametric approach where we approximate the geometry in a similar way as the unknowns.

1.3.1 The reference domain, $\hat{\Omega}$

In a spectral element setting all computations are performed on a reference domain $\hat{\Omega} = (-1,1)^d$. We assume that the reference variables are given by r^i, $i = 1, \ldots, d$, and an invertible mapping $\mathcal{F}_k(t)$ exists such that a point in the reference domain is mapped to a unique point in element k in the physical domain, $\Omega(t)$; see Figure 3.

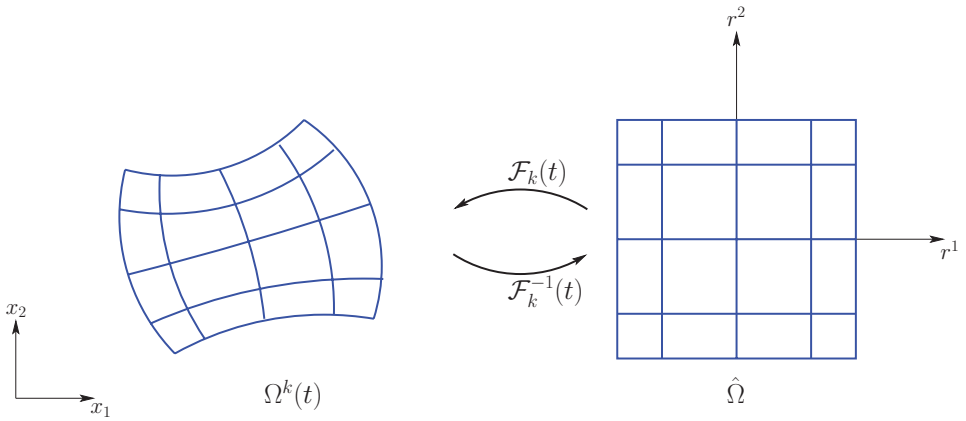


Figure 3: The reference domain, $\hat{\Omega}$, and a spectral element, Ω^k, for a two-dimensional case.

At any time t, the mapping $\mathcal{F}_k(t)$ has an associated Jacobian matrix with elements

$$J_{ij} = \frac{\partial x_i}{\partial r^j}.$$

As a nodal basis for $\mathbb{P}_N((-1,1))$ we use

$$\mathbb{P}_N((-1,1)) = \text{span}\{\ell_i(r^1)\}_{i=0}^N, \tag{4}$$

where $\ell_i(r^1)$ is an N'th degree polynomial with the property that

$$\ell_j(\xi_i) = \delta_{ij},$$

with ξ_i being the i'th Gauss-Lobatto-Legendre quadrature point. In d dimensions we use a tensor-product basis such that

$$\mathbb{P}_N(\hat{\Omega}) = \text{span} \otimes_{i=1}^{d} \{\ell_j(r^i)\}_{j=0}^{N}.$$

Hence, a two-dimensional variable expressed as an N'th order polynomial in each direction on the reference domain is expressed as

$$\varphi_N(r^1, r^2) = \sum_{i=0}^{N} \sum_{j=0}^{N} \varphi_{ij} \ell_i(r^1) \ell_j(r^2),$$

where φ_{ij} represent the *nodal* values.

We will use a staggered grid approach in the discretization of (3). This is motivated by [3, 42] where Maday and co-workers in the context of simple geometries showed that a method based on approximating the velocity components, u_i, by polynomials of degree N and the pressure, p, by polynomials of degree $N-2$ leads to a stable discretization (i.e., no spurious pressure modes). As a basis for $\mathbb{P}_{N-2}((-1,1))$ we choose

$$\mathbb{P}_{N-2}((-1,1)) = \text{span}\{\tilde{\ell}_i(r^1)\}_{i=0}^{N-2}, \tag{5}$$

where $\tilde{\ell}_i(r^1)$ is an $(N-2)$'th degree polynomial with the property that

$$\tilde{\ell}_j(\zeta_i) = \delta_{ij},$$

with ζ_i being the i'th Gauss-Legendre quadrature point. In d dimensions a tensor-product basis is given by

$$\mathbb{P}_{N-2}(\hat{\Omega}) = \text{span} \otimes_{i=1}^{d} \{\tilde{\ell}_j(r^i)\}_{j=0}^{N-2}.$$

A two-dimensional variable expressed as an $(N-2)$'th degree polynomial in each direction on the GL-grid, i.e. the pressure, is then expressed as

$$\theta_N(r^1, r^2) = \sum_{i=0}^{N-2} \sum_{j=0}^{N-2} \theta_{ij} \tilde{\ell}_i(r^1) \tilde{\ell}_j(r^2).$$

1.3.2 Discrete formulation

A discrete formulation of (3) is: Find $(u_N)_i \in X_{N_b}$ and $p_N \in Y_N$ such that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} v_i (u_N)_i \,\mathrm{d}V + \int_{\Omega(t)} v_i ((u_N)_j - w_j) \frac{\partial (u_N)_i}{\partial x_j} \,\mathrm{d}V - \int_{\Omega(t)} v_i (u_N)_i \frac{\partial w_j}{\partial x_j} \,\mathrm{d}V & \\
= \int_{\Omega(t)} \left(-\frac{\partial v_i}{\partial x_j} \sigma_{ij} + v_i f_i \right) \mathrm{d}V + \int_{\Gamma_\gamma(t)} v_i \sigma_{ij} n_j \,\mathrm{d}S, \quad \forall v_i \in X_N, & \\
\int_{\Omega(t)} q \frac{\partial (u_N)_j}{\partial x_j} \,\mathrm{d}V = 0, \qquad \forall q \in Y_N. &
\end{aligned} \tag{6}$$

We have here introduced the polynomial spaces,

$$
\begin{aligned}
X_N &= \{v \in X \mid v\big|_{\Omega^k} \circ \mathcal{F}_k(t) \in \mathbb{P}_N(\hat{\Omega}), k = 1, \ldots, K\}, \\
X_{N_b} &= \{v \in X_b \mid v\big|_{\Omega^k} \circ \mathcal{F}_k(t) \in \mathbb{P}_N(\hat{\Omega}), k = 1, \ldots, K\}, \\
Y_N &= \{v \in Y \mid v\big|_{\Omega^k} \circ \mathcal{F}_k(t) \in \mathbb{P}_{N-2}(\hat{\Omega}), k = 1, \ldots, K\}.
\end{aligned}
$$

We now transform all the integrals in (6) to the reference domain by using the mapping $\mathcal{F}_k(t)$ and the Jacobian, $\mathbf{J}$, express $(u_N)_i$, $i = 1, \ldots, d$, as N'th order polynomials and p_N as an $(N-2)$'th order polynomial on each element, and finally replace the integrals with GLL/GL-quadrature. This leads us to the semi-discrete equations [14],

$$
\begin{aligned}
\frac{\mathrm{d}(\underline{\mathbf{B}}\,\underline{\mathbf{u}})}{\mathrm{d}t} + \underline{\mathbf{C}}(\underline{\mathbf{u}}, \underline{\mathbf{w}})\underline{\mathbf{u}} &= -\underline{\mathbf{A}}\,\underline{\mathbf{u}} + \underline{\mathbf{D}}^T \underline{p} + \underline{\mathbf{E}}(\underline{\mathbf{w}})\underline{\mathbf{u}} + \underline{\mathbf{F}}. \\
-\underline{\mathbf{D}}\,\underline{\mathbf{u}} &= 0.
\end{aligned}
\tag{7}
$$

Here, $\underline{\mathbf{B}}$ is the mass matrix, $\underline{\mathbf{C}}(\underline{\mathbf{u}}, \underline{\mathbf{w}})$ is a convection operator, $\underline{\mathbf{A}}$ is the Laplacian, $\underline{\mathbf{D}}$ is the divergence operator, $\underline{\mathbf{D}}^T$ is the gradient operator, $\underline{\mathbf{E}}(\underline{\mathbf{w}})$ is the "expansion" operator involving the divergence of the grid velocity, $\underline{\mathbf{F}}$ includes all volumetric and surface forces, $\underline{u_i}$ represents the nodal values of the i'th velocity component, and $\underline{p}$ is a vector representing the nodal values of the pressure.

1.4 Temporal treatment

1.4.1 Convection/Stokes-splitting

We will now focus on the temporal discretization of the semi-discrete Navier-Stokes equations (7). One major feature of the Navier-Stokes equations is the combination of nonlinear(convection) and linear(Stokes) operators, and it is desirable to treat these operators differently. Due to the severe stability restriction the viscous term would impose on an explicit scheme, we want to treat the Stokes operator implicitly, while we wish to treat the convection term explicitly due to its nonlinear nature. In [33] a scheme was proposed which combined a backward difference scheme for the viscous term with an explicit treatment of the convection term of the Navier-Stokes equations. In [56, 24] the temporal discretization of the Navier-Stokes equations are treated in a semi-Lagrangian manner in which the convective term is incorporated in a "total derivative". These algorithms include spatial interpolation in order to find the departure points of particles such that the total derivative can be approximated.

In [41] a general framework for generating splitting schemes was proposed. Here, an operator-integration-factor(OIF) strategy was used to decouple the general initial-value problem

$$
\begin{aligned}
\frac{\mathrm{d}\underline{\mathbf{u}}}{\mathrm{d}t} &= \underline{\mathbf{M}}(t)\underline{\mathbf{u}} + \underline{\mathbf{N}}(t)\underline{\mathbf{u}} + \underline{\mathbf{f}}, \qquad 0 < t < T, \\
\underline{\mathbf{u}}(t=0) &= \underline{\mathbf{u}}_0,
\end{aligned}
$$

into two subproblems, where each subproblem may be solved using different discretizations. For instance, if we choose to treat $\underline{\mathbf{N}}(t)$ implicitly using a first order backward difference scheme, the OIF-method yields,

$$\frac{\underline{\mathbf{u}}^{n+1} - \underline{\tilde{\mathbf{u}}}^{n+1}}{\Delta t} = \underline{\mathbf{N}}(t^{n+1})\underline{\mathbf{u}}^{n+1}, \tag{8}$$

where

$$\begin{aligned} \frac{\mathrm{d}\underline{\tilde{\mathbf{u}}}}{\mathrm{d}\tau} &= \underline{\mathbf{M}}(t)\underline{\tilde{\mathbf{u}}}, \\ \underline{\tilde{\mathbf{u}}}(\tau = 0) &= \underline{\mathbf{u}}^n, \end{aligned} \tag{9}$$

and

$$\underline{\tilde{\mathbf{u}}}^{n+1} = \underline{\tilde{\mathbf{u}}}(\tau = \Delta t). \tag{10}$$

This approach may be extended to other types of discretizations of higher order. One advantage with this strategy is that we may choose different time-steps for the solution of (8) and (9). If we consider the Navier-Stokes equations, it would seem natural to apply this technique where the operator $\underline{\mathbf{N}}(t)$ plays the role of the Stokes operator, while $\underline{\mathbf{M}}(t)$ is the convection operator. We know that the eigenvalues of the convection operator have a large imaginary part, so a good choice for an explicit scheme to solve the subproblem (9) is one with a stability region which includes as much of the imaginary axis as possible. One such choice may be the explicit classical fourth-order Runge-Kutta scheme, which also has the advantage of not requiring initial data from more than one temporal configuration. Another feature when $\underline{\mathbf{M}}(t)$ is chosen to be the convection operator is that the values $\underline{\tilde{\mathbf{u}}}^{n+1}$ in (10) have a physical interpretation. These are the velocities that the particles which at time t^{n+1} are located at the grid points had at time t^n. Thus, $\frac{\underline{\mathbf{u}}^{n+1} - \underline{\tilde{\mathbf{u}}}^{n+1}}{\Delta t}$ will for this case approximate the material derivative in a Lagrangian framework, and in this setting the OIF-method can be interpreted as a semi-Lagrangian scheme. The OIF method has previously been applied to the Navier-Stokes equations and other problems in fixed geometries, but has to our knowledge not been used with moving boundary problems.

1.4.2 Velocity/pressure-splitting

If we treat the convection term explicitly and the Stokes term implicitly, (7) gives rise to the system

$$\begin{bmatrix} \underline{\mathbf{H}} & -\underline{\mathbf{D}}^T \\ -\underline{\mathbf{D}} & \underline{0} \end{bmatrix} \begin{bmatrix} \underline{\mathbf{u}}^{n+1} \\ \underline{p}^{n+1} \end{bmatrix} = \begin{bmatrix} \underline{\widetilde{\mathbf{F}}}^{n+1} \\ \underline{0} \end{bmatrix},$$

where $\underline{\mathbf{H}}$ is the Helmholtz operator and $\underline{\widetilde{\mathbf{F}}}^{n+1}$ includes all external forces and the explicitly treated terms. Using a Uzawa procedure [2, 39] to decouple this problem, we get

$$\begin{aligned} \underline{\mathbf{D}}\,\underline{\mathbf{H}}^{-1}\underline{\mathbf{D}}^T\underline{p}^{n+1} &= -\underline{\mathbf{D}}\,\underline{\mathbf{H}}^{-1}\underline{\widetilde{\mathbf{F}}}^{n+1}, \\ \underline{\mathbf{H}}\,\underline{\mathbf{u}}^{n+1} &= \underline{\mathbf{D}}^T\underline{p}^{n+1} + \underline{\widetilde{\mathbf{F}}}^{n+1}. \end{aligned} \tag{11}$$

We see that the solution of the pressure term involves nested iterations, which may be computationally expensive. This motivates a splitting of the Stokes operator, in addition to the convection/Stokes-splitting discussed above.

Two classes for decoupling the treatment of the velocity and the pressure in the Navier-Stokes equations, (1), are pressure-correction methods and velocity-correction methods [27]. The first type involves solving a Helmholtz problem for an intermediate velocity field, and next computing a pressure-correction such that the final velocity field is incompressible. Such methods were first proposed in [10, 53] and also used in [34, 55]. In velocity-correction methods [50, 33] an intermediate incompressible velocity field is first computed by treating the viscous term explicitly, and next this velocity field is incremented by treating the viscous term implicitly, leading to a Helmholtz equation. Hence, the final velocity field is not incompressible. A problem for such splitting schemes which has been subject to much research is the imposition of the correct boundary conditions for the pressure. A third class of splitting schemes are so-called *consistent splitting* schemes [28]. These are based on a weak form of the pressure Poisson equation and requires the solution of Helmholtz problems for the velocity and a Poisson equation for the pressure.

One may also use the discretized system (11) as a starting point for generating splitting schemes. Here, the velocity boundary conditions are already incorporated in the discrete operators, and no additional boundary conditions for the pressure is needed. Analysis of such splitting schemes is reported in [47, 11, 18].

1.5 Free surface boundary conditions

If we assume that we have two immiscible incompressible fluids in contact with each other at some part of the boundary, Γ_γ, and that surface tension effects are negligible, the boundary conditions are given by [36]

$$(\sigma_{ij}^2 - \sigma_{ij}^1)n_j = 0, \qquad \text{on } \Gamma_\gamma,$$

where σ_{ij}^k, $k = 1, 2$, is the stress tensor of the two fluids and $\mathbf{n}$ is the unit normal directed into medium 1. This represents a balance of viscous forces along the intersection of the two fluids. We will however focus on situations where surface tension forces *are* significant. The boundary conditions for fluid-fluid interaction for such cases take the form [36]

$$n_i(\sigma_{ij}^2 - \sigma_{ij}^1)n_j = \gamma\kappa, \qquad \text{on } \Gamma_\gamma, \tag{12}$$
$$t_i(\sigma_{ij}^2 - \sigma_{ij}^1)n_j = t_k(\nabla_s\gamma)_k, \qquad \text{on } \Gamma_\gamma. \tag{13}$$

Here, $\mathbf{t}$ is any tangent vector and κ is twice the mean curvature. (12) expresses that the normal viscous forces exerted by the two media is balanced by the product of the surface tension and the curvature, while (13) states that the tangential forces from the two media is balanced by the surface tension gradient.

Free surface boundary conditions are found by treating one of the fluids(fluid 1) as inviscid, such as air,

$$n_i\sigma_{ij}n_j = \gamma\kappa - p_a, \qquad \text{on } \Gamma_\gamma, \tag{14}$$
$$t_i\sigma_{ij}n_j = t_k(\nabla_s\gamma)_k, \qquad \text{on } \Gamma_\gamma. \tag{15}$$

Here, p_a is the atmospheric pressure. We observe that the total stress forces on the boundary are given by

$$\sigma_{ij} n_j = \gamma \kappa n_i + (\nabla_s \gamma)_i, \qquad i = 1, 2, 3, \tag{16}$$

and this is what we need to impose through the surface integral in (3).

Many articles for both two and three dimensions have been written on numerical simulation of free surface flows where the surface tension plays a significant role. However, most simulations which have used a fully deformable free surface have considered the situation with a constant surface tension, which results in only normal surface forces. Such simulations are for example reported in [30, 6, 44] for two dimensions and in [57, 13] for three dimensions. In two dimensions, the incorporation of free surface boundary conditions is quite straightforward, but in three dimensions this may be a little more cumbersome. We see from (16) that we need to be able to evaluate κ, n_i, and the surface gradient ∇_s on Γ_γ. Depending on how we represent our free surface, this may not be straightforward.

A class of phenomena where surface tension forces are significant are so-called Marangoni flows, which are flows driven by surface tension gradients [35]. Such flows can occur by heating a thin layer of fluid from below (in the presence of gravity). Temperature differences on the surface will lead to surface tension gradients, and under special conditions these tangential surface forces will induce cellular convection; see Figure 4. For such situations it is obvious that a numerical simulation needs to include tangential stress boundary conditions as this is the main driving force for the flow. There have been several numerical studies of such flows [54, 43, 7, 48], but what is common for all of them is the assumption of no normal stress forces, which is a good assumption for most purposes. If we assume a cartesian grid and a "free-surface" (free in the tangential plane) at $x_3 = \text{const}$ and $u_3 = 0$, the boundary conditions (15) are reduced to the simpler form

$$\begin{aligned} \mu \frac{\partial u_1}{\partial x_3} &= \frac{\partial \gamma}{\partial x_1}, && \text{on } \Gamma_\gamma, \\ \mu \frac{\partial u_2}{\partial x_3} &= \frac{\partial \gamma}{\partial x_2}, && \text{on } \Gamma_\gamma, \\ u_3 &= 0, && \text{on } \Gamma_\gamma, \end{aligned}$$

and these are the boundary conditions mostly used in numerical simulations. Nevertheless, in experiments of such flows, a small surface movement has been observed in which the surface is elevated and depressed at cold and hot regions, respectively. This feature is obviously not captured by numerical simulations which assume a fixed "free surface" with no normal stress forces. There are probably three reasons why these simplifications are usually made in numerical experiments; (i) the boundary conditions take the simple form described above; (ii) there is no need to worry about a moving boundary which complicates the numerical simulations; (iii) the surface deformation is typically very small for such flows.

We will next discuss an alternative way to impose (16) which was proposed by Ho and Patera [31].

1.5.1 Imposition of normal and tangential stress forces: two dimensions

Accounting for surface tension effects, the boundary conditions in two dimensions for fluid-fluid interaction are similar to the three-dimensional case

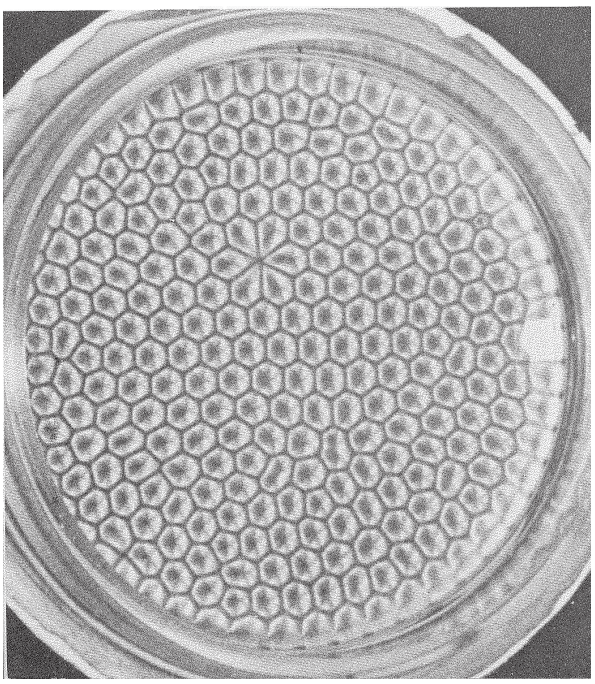


Figure 4: Hexagonal cells (top view) formed due to Bénard-Marangoni convection (taken from [35]).

$$n_i(\sigma^2_{ij} - \sigma^1_{ij})n_j = \gamma\kappa_c, \qquad \text{on } \Gamma_\gamma, \tag{17}$$

$$t_i(\sigma^2_{ij} - \sigma^1_{ij})n_j = \frac{\mathrm{d}\gamma}{\mathrm{d}s}, \qquad \text{on } \Gamma_\gamma. \tag{18}$$

where κ_c is the curvature of the free surface, γ is the surface tension coefficient, s is an arc-parameter, and $t_i = \frac{\mathrm{d}x_i}{\mathrm{d}s}$ is the i'th component of the unit tangent vector $\mathbf{t}$. The boundary conditions along a free surface become

$$n_i\sigma_{ij}n_j = \gamma\kappa_c - p_a, \qquad \text{on } \Gamma_\gamma, \tag{19}$$

$$t_i\sigma_{ij}n_j = \frac{\mathrm{d}\gamma}{\mathrm{d}s}, \qquad \text{on } \Gamma_\gamma. \tag{20}$$

Using the fact that $\kappa_c n_i = \frac{\mathrm{d}t_i}{\mathrm{d}s}$, we observe that

$$\begin{aligned}
\int_{\Gamma_\gamma} v_i\sigma_{ij}n_j\,\mathrm{d}s &= \int_{\Gamma_\gamma} v_i\left(\gamma\kappa_c n_i - p_a n_i + \frac{\mathrm{d}\gamma}{\mathrm{d}s}t_i\right)\mathrm{d}s \\
&= \int_{\Gamma_\gamma} v_i\left(\gamma\frac{\mathrm{d}t_i}{\mathrm{d}s} + \frac{\mathrm{d}\gamma}{\mathrm{d}s}t_i\right)\mathrm{d}s - \int_{\Gamma_\gamma} p_a v_i n_i\,\mathrm{d}s \\
&= \int_{\Gamma_\gamma} v_i\frac{\mathrm{d}(\gamma t_i)}{\mathrm{d}s}\,\mathrm{d}s - \int_{\Gamma_\gamma} p_a v_i n_i\,\mathrm{d}s \\
&= [\gamma v_i t_i]_a^b - \int_{\Gamma_\gamma}\gamma\frac{\mathrm{d}v_i}{\mathrm{d}s}\frac{\mathrm{d}x_i}{\mathrm{d}s}\,\mathrm{d}s - \int_{\Gamma_\gamma} p_a v_i n_i\,\mathrm{d}s.
\end{aligned} \tag{21}$$

Here, a and b denote the start and end of the free surface. The first term can be used for imposing contact angles, but will vanish if Γ_γ represents a closed surface or if we have periodic boundary conditions, while the last term is zero for $p_a = 0$. For such cases we have one single surface integral for imposition of *both* normal and tangential stress forces. We also observe that although (19) includes the curvature, which is related to the *second* derivative of $\mathbf{x}$, the resulting boundary integral only includes the *first* derivative of the position, $\mathbf{x}$.

1.5.2 Imposition of normal and tangential stress forces: three dimensions

It would be appealing to be able to derive a similar expression as (21) for the three-dimensional case. This becomes much more complicated, but in their article [31] Ho and Patera proposed a linear form for imposition of *both* tangential and normal stresses. Results from differential geometry is used, and the alternative form can be written as

$$\int_{\Gamma_\gamma} v_i \sigma_{ij} n_j \,\mathrm{d}S = \oint_{\partial\Gamma_\gamma} \gamma v_i g_i^\alpha \,\mathrm{d}n_\alpha - \int_{\Gamma_\gamma} v_{i,\alpha} \gamma g_i^\alpha \,\mathrm{d}S - \int_{\Gamma_\gamma} v_i n_i p_a \,\mathrm{d}S.$$

Here, $\mathbf{g}^\alpha, \alpha = 1, 2$, are *contravariant* tangential vectors [20] and $\mathrm{d}n_\alpha$ is an outer normal to $\partial\Gamma$. Similar to the two-dimensional case, the first integral will vanish if $\partial\Gamma_\gamma$ represents a closed surface or if we have periodic boundary conditions, while the last term has a contribution for $p_a \neq 0$.

1.6 Solution of the linear system of equations

There are two main strategies for solving the linear systems of equations in Section 1.4.2, namely by the use of direct or iterative solvers. High-order methods based on polynomial approximations typically results in large systems on the form

$$\underline{A}\,\underline{x} = \underline{b}, \tag{22}$$

where $\underline{A}$ is a dense matrix. Hence, a direct method based on Gaussian elimination of such a system would be very expensive. Also, for problems in time-dependent domains, the matrix $\underline{A}$ will be time-dependent so a new inversion would need to take place at each time-level. Hence, in the general case, using a direct solver on (22) in this setting is not a good idea, however, for special cases it is possible to construct very efficient tensor-product solvers [38]. These are solvers based on diagonalizing the operators, and they are typically applicable for solving discretized problems with constant coefficients in simple geometries where the operators are tensorizable. Despite their limited applicability such methods can also often be used successfully as preconditioners in combination with iterative solvers.

In the general case, the most common way of solving (22) will be by the use of an iterative approach, and then in particular projection based methods such as the conjugate gradient method(CG) for cases when $\underline{A}$ is symmetric positive definite. The most time-consuming part of such iterative methods are matrix-vector operations, and one feature of the tensor-product nature of the bases (4) and (5) is that matrix-vector products where the matrix originates from discretization of a two- or three-dimensional operator can be performed efficiently. Due to the tensor-product nature of the bases (4) and (5), using a *local* data representation yields the opportunity to perform such matrix operations as a series of "one-dimensional" matrix-vector products. Such an approach is often referred to as operator evaluation using sum-factorization since the full matrix is not explicitly constructed. The computational complexity for a single

operator evaluation (matrix-vector product) is typically $\mathcal{O}(N^{d+1})$ in $\mathbb{R}^d$, $d = 1, 2, 3$. For iterative methods the number of iterations is typically a function of the condition number, κ, and in particular for CG we have $\mathcal{N}_{\text{iter}} \sim \mathcal{O}(\sqrt{\kappa})$. For many systems the condition number is very large. However, this problem can be alleviated by the use of preconditioners.

In (11) we need to solve problems of the type

$$\begin{aligned} \underline{\mathbf{D}}\,\underline{\mathbf{H}}^{-1}\underline{\mathbf{D}}^T\underline{p} &= \underline{F}, \\ \underline{\mathbf{H}}\,\underline{\mathbf{u}} &= \underline{\mathbf{G}}. \end{aligned}$$

The first problem involving the Uzawa operator, $\underline{\mathbf{S}} = \underline{\mathbf{D}}\,\underline{\mathbf{H}}^{-1}\underline{\mathbf{D}}^T$, results in nested iterations, as mentioned before, and solving such problems is obviously costly. In [39] it was shown that $\underline{\tilde{\mathbf{B}}}$ is a good preconditioner for $\underline{\mathbf{S}}$ for small Reynold's numbers. Here, $\underline{\tilde{\mathbf{B}}}$ is the mass matrix defined on the Gauss-Legendre grid. This is a diagonal preconditioner which is obviously very easy to invert. For large Reynold's numbers $\underline{\mathbf{S}}$ is spectrally close to $\underline{\mathbf{E}} = \underline{\mathbf{D}}\,\underline{\mathbf{B}}^{-1}\underline{\mathbf{D}}^T$ [39], with $\underline{\mathbf{E}}$ often being denoted as the consistent pressure Poisson operator due to its similarities with the standard Poisson operator. We may also encounter this operator when using the splitting schemes from Section 1.4.2. Several preconditioners for $\underline{\mathbf{S}}$ and $\underline{\mathbf{E}}$ where proposed in [8, 49, 12, 19].

A simple preconditioner for $\underline{\mathbf{H}}$ would be to use its diagonal, since $\underline{\mathbf{H}}$ in many cases will be diagonal dominant. Other preconditioners for $\underline{\mathbf{H}}$ and $\underline{\mathbf{A}}$ are found in [9].

2 Summary of papers

2.1 Paper I: A high order splitting method for time-dependent domains

We present a high order convection/Stokes splitting method for the Navier-Stokes equations in time-dependent domains, which is based on the OIF-method [41] and can be interpreted as a semi-Lagrangian method. The splitting method is based on an arbitrary Lagrangian-Eulerian formulation, and first, second, and third order temporal convergence and spectral spatial convergence is verified for a model problem in a fixed domain with artificially imposed grid velocity in the interior of the domain. Our method is compared with other methods in the literature, and the spatial regularity requirement on the grid velocity is addressed.

2.2 Paper II: Imposing free-surface boundary conditions using surface intrinsic coordinates

We consider the surface integral proposed in [31] for weak imposition of both normal and tangential surface tension boundary conditions in three dimensions. Basic concepts from differential geometry is introduced which is used to derive quantities such as surface gradient and the divergence operator using surface intrinsic coordinates, while concepts such as the mean curvature is illuminated. In the final section we start from the strong form of the free surface boundary conditions in the normal and tangential direction and by the use of the results from differential geometry we arrive at the proposed integral.

2.3 Paper III: Simulation of three-dimensional Bénard-Marangoni flows including deformed surfaces

In this paper we present three-dimensional simulations of Bénard-Marangoni flows using a coupled thermal-fluid model. The governing equations are discretized using spectral elements in space and the operator splitting approach in time which was proposed in Paper I. Compared to previous simulations, we do not assume a fixed "free surface", and the boundary conditions are imposed by the use of surface intrinsic coordinates as proposed in [31] and derived in Paper II. We report results which are in agreement with previous experimental data and numerical simulations. In addition, we present a numerical prediction of the surface deflection, which to our knowledge is new.

2.4 Paper IV: High order methods for incompressible fluid flow: Application to free surface problems

In this paper the splitting approach proposed in Paper I is used with three different problems. First, we solve the Navier-Stokes equations in a fixed two-dimensional domain with artificially imposed grid velocity in the interior. For this problem we demonstrate first, second, and third order temporal convergence and spectral spatial convergence. Second, we revisit the simulation of Bénard-Marangoni convection with free surface boundary conditions which were discussed in Paper III. Third, we consider a two-dimensional simulation which models transportation of fresh water in a fabric container. Here, a flexible container with a fluid of density ρ_{in} is immersed in a fluid with density $\rho_o > \rho_{in}$. The dynamic response of this container is modelled by the Navier-Stokes equations with free surface boundary conditions. Compared

to the Bénard-Marangoni simulations, this problem involves large motions on the boundary, and the necessity for good ways to track the boundary is apparent.

2.5 Paper V: Accurate interface-tracking for arbitrary Lagrangian-Eulerian schemes

When solving moving boundary problems using an ALE approach, the boundary is explicitly tracked through the kinematic condition. In this paper we focus on surfaces of two-dimensional domains and propose a new framework for tracking interfaces which is able to follow the interface to second and third order accuracy *and* maintain a good mesh quality without the need for remeshing. The method relies on the solution of several convection problems along the interface and no spatial interpolation is needed. We have proposed two different point distribution strategies, and we have verified first, second and third order temporal accuracy for selected two-dimensional model problems with known analytical solutions. Comparisons with common approaches are reported.

2.6 Paper VI: Fast tensor-product solvers. Part I: Partially deformed three-dimensional domains

A fast tensor-product solver is proposed for solving partial differential equations in three-dimensional domains which are deformed in a plane and invariant in the third direction. We choose the xy-plane to be deformed, and the solution algorithm then exploits the tensor-product feature between the xy-plane and the z-direction. It is applicable for problems with variable coefficients as long as these can be expressed as a separable function with respect to the variation in the xy-plane and the z-direction. Compared to previous work exploiting this tensor-product feature, we are here not restricted to periodic boundary conditions in the z-direction or a tensorizable xy-plane. This solver can also be used as an efficient preconditioner for Helmholtz and Poisson operators for problems which have domains which are "close" to being tensorizable between a plane and a third direction. Such a case is the simulation of Bénard-Marangoni flows, where very small deformations is observed along a free surface. This solver may for instance be very suitable to use as a preconditioner for large scale Bénard-Marangoni simulations.

References

[1] R. Aris. *Vectors, Tensors, and the Basic Equations of Fluid Mechanics.* Dover Publications, Inc., 1962.

[2] K. Arrow, L. Hurwicz, and H. Uzawa. *Studies in Linear and Nonlinear Programming.* Stanford University Press, CA, 1958.

[3] C. Bernardi, Y. Maday, and B. Métivet. Spectral Approximation of the Periodic-Nonperiodic Navier-Stokes Equations. *Numerische Mathematik*, 51:655–700, 1987.

[4] N. Bodard, R. Bouffanais, and M.O. Deville. Solution of moving-boundary problems by the spectral element method. *Applied Numerical Mathematics (2007),doi:10.1016/j.apnum.2007.04.009*, 2007.

[5] R. Bouffanais and M.O. Deville. Mesh update techniques for free-surface flow solvers using spectral elements. *Journal of Scientific Computing*, 27(1-3):137–149, 2006.

[6] H. Braess and P. Wriggers. Arbitrary Lagrangian Eulerian finite element analysis of free surface flow. *Computer Methods in Applied Mechanics and Engineering*, 190:95–109, 2000.

[7] E. Bucchignani and D. Mansutti. A Numerical Modeling of Rayleigh-Marangoni Steady Convection in a Non-Uniform Differentially Heated 3D Cavity. *Journal of Scientific Computing*, 20(1):115–136, 2004.

[8] J. Cahouet and J.P. Chabard. Mulit-domains and multi-solvers finite element approach for the stokes problem. In R.P. Shaw, editor, *Proceedings of the Fourth International Symposium on Innovative Numerical Methods in Engineering*, pages 317–322. Springer-Verlag, New York, 1986.

[9] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods, Fundamentals in Single Domains*. Springer, 2006.

[10] A.J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.

[11] W. Couzy. *Spectral Element Discretization of the Unsteady Navier-Stokes Equations and Its Iterative Solution on Parallel Computers*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 1995.

[12] W. Couzy and M.O. Deville. Spectral-Element Preconditioners for the Uzawa Pressure Operator Applied to Incompressible Flows. *Journal of Scientific Computing*, 9(2):107–122, 1994.

[13] W. Dettmer and D. Perić. A computational framework for free surface fluid flows accounting for surface tension. *Computer Methods in Applied Mechanics and Engineering*, 195:3038–3071, 2006.

[14] M.O. Deville, P.F. Fischer, and E.H. Mund. *High-Order Methods for Incompressible Fluid Flow*. Cambridge University Press, 2002.

[15] J. Donea, S. Giuliani, and J.P Halleux. An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33:689–723, 1982.

[16] C. Farhat and P. Geuzaine. Design and analysis of robust ALE time-integrators for the solution of unsteady flow problems on moving grids. *Computer Methods in Applied Mechanics and Engineering*, 193:4073–4095, 2004.

[17] C. Farhat, P. Geuzaine, and C. Grandmont. The Discrete Geometric Conservation Law and the Nonlinear Stability of ALE Schemes for the Solution of Flow Problems on Moving Grids. *Journal of Computational Physics*, 174:669–694, 2001.

[18] P.F. Fischer. An Overlapping Schwarz Method for Spectral Element Solution of the Incompressible Navier-Stokes Equations. *Journal of Computational Physics*, 133:84–101, 1997.

[19] P.F. Fischer, N.I. Miller, and H.M. Tufo. An overlapping Schwarz method for spectral element simulation of three-dimensional incompressible flows. In P.Bjørstad and M.Luskin, editors, *Parallel Solution of Partial Differential Equations*, pages 158–180. Springer, Berlin, 2000.

[20] W. Flügge. *Tensor Analysis and Continuum Mechanics*. Springer, Berlin, 1972.

[21] L. Formaggia and F. Nobile. A stability analysis for the arbitrary Lagrangian Eulerian formulation with finite elements. *East-West Journal of Numerical Mathematics*, 7(2):105–131, 1999.

[22] L. Formaggia and F. Nobile. Stability analysis of second-order time accurate schemes for ALE-FEM. *Computer Methods in Applied Mechanics and Engineering*, 193:4097–4116, 2004.

[23] R.M. Franck and R.B.Lazarus. Mixed Eulerian-Lagrangian method. *Methods in Computational Physics*, 3:47–67, 1964. Alder E., Fenbach and Rosenberg Editions (eds).

[24] F.X. Giraldo. The Lagrange-Galerkin Spectral Element Method on Unstructured Quadrilateral Grids. *Journal of Computational Physics*, 147:114–146, 1998.

[25] W.J. Gordon and C.A. Hall. Construction of curvilinear co-ordinate systems and applications to mesh generation. *International Journal for Numerical Methods in Engineering*, 7:461–477, 1973.

[26] D. Gottlieb and S.A. Orszag. *Numerical Analysis of Spectral Methods: Theory and Applications*, volume 26 of *Regional Conference Series in Applied Mathematics*. SIAM, 1977.

[27] J.L. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195:6011–6045, 2006.

[28] J.L. Guermond and J. Shen. A new class of truly consistent splitting schemes for incompressible flows. *Journal of Computational Physics*, 192:262–276, 2003.

[29] C.W. Hirt, A.A. Amsden, and J.L Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14:227–253, 1974.

[30] L.W. Ho and A.T. Patera. A Legendre spectral element method for simulation of unsteady incompressible viscous free-surface flows. *Computer Methods in Applied Mechanics and Engineering*, 80:355–366, 1990.

[31] L.W. Ho and A.T. Patera. Variational formulation of three-dimensional viscous free-surface flows: Natural imposition of surface tension boundary conditions. *International Journal for Numerical Methods in Fluids*, 13:691–698, 1991.

[32] T.J.R. Hughes, W.K. Liu, and T.K Zimmermann. Lagrangian-Eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 29:329–349, 1981.

[33] G.E. Karniadakis, M. Israeli, and S.A. Orszag. High-Order Splitting Methods for the Incompressible Navier-Stokes Equations. *Journal of Computational Physics*, 97:414–443, 1991.

[34] J. Kim and P. Moin. Application of a fractional-step method to incompressible Navier-Stokes equations. *Journal of Computational Physics*, 59:308–323, 1985.

[35] E. L. Koschmieder. *Bénard Cells and Taylor Vortices*. Cambridge University Press, 1993.

[36] L.D. Landau and E.M. Lifshitz. *Fluid Mechanics*. Course of Theoretical Physics, Volume 6. Butterworth-Heinemann, 1987.

[37] M. Lesoinne and C. Farhat. Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer Methods in Applied Mechanics and Engineering*, 134:71–90, 1996.

[38] R.E. Lynch, J.R. Rice, and D.H. Thomas. Direct Solution of Partial Difference Equations by Tensor Product Methods. *Numerische Mathematik*, 6:185–199, 1964.

[39] Y. Maday, D. Meiron, A.T. Patera, and E.M. Rønquist. Analysis of iterative methods for the steady and unsteady stokes problem: application to spectral element discretizations. *SIAM Journal on Scientific Computing*, 14(2):310–337, 1993.

[40] Y. Maday and A.T. Patera. Spectral element methods for the Navier-Stokes equations. *in: A.K. Noor, J. T. Oden (Eds.), State of the Art Surveys in Computational Mechanics, ASME, New York*, pages 71–143, 1989.

[41] Y. Maday, A.T. Patera, and E.M. Rønquist. An Operator-Integration-Factor Splitting Method for Time-Dependent Problems: Application to Incompressible Fluid Flow. *Journal of Scientific Computing*, 5(4):263–292, 1990.

[42] Y. Maday, A.T. Patera, and E.M. Rønquist. The $\mathbb{P}_N \times \mathbb{P}_{N-2}$ method for the approximation of the stokes problem. Technical Report 92009, Department of Mechanical Engineering, Massachusetts Institute of Technology, 1992.

[43] M. Medale and P. Cerisier. Numerical simulation of Bénard-Marangoni convection in small aspect ratio containers. *Numerical Heat Transfer, Part A*, 42:55–72, 2002.

[44] S.E. Navti, K. Ravindran, C. Taylor, and R.W. Lewis . Finite element modelling of surface tension effects using a lagrangian-eulerian kinematic description. *Computer Methods in Applied Mechanics and Engineering*, 147:41–60, 1997.

[45] W.F. Noh. A time-dependent two-space dimensional coupled Eulerian-Lagrangian code. *Methods in Computational Physics*, 3:117–179, 1964. Alder E., Fenbach and Rosenberg Editions (eds).

[46] A.T. Patera. A Spectral Element Method for Fluid Dynamics: Laminar Flow in a Channel Expansion. *Journal of Computational Physics*, 54:468–488, 1984.

[47] J.B. Perot. An analysis of the fractional step method. *Journal of Computational Physics*, 108:51–58, 1993.

[48] J.W. Peterson. A Numerical Investigation of Bénard Convection in Small Aspect Ratio Containers. Master's thesis, The University of Texas at Austin, 2004.

[49] E.M. Rønquist. A domain decomposition method for elliptic boundary value problems: Appliciation to unsteady incompressible fluid flow. In *Proceedings of the Fifth International Symposium on Domain Decompostion Methods for Partial Differential Equations (symposium held in Norfolk, Virginia, May 1991)*, pages 545–557. SIAM, 1992.

[50] M. Israeli S.A. Orszag and M.O. Deville. Boundary Conditions for Incompressible Flows. *Journal of Scientific Computing*, 1(1):75–111, 1986.

[51] P.A. Sackinger, P.R. Schunk, and R.R. Rao. A Newton-Raphson Pseudo-Solid Domain Mapping Technique for Free and Moving Boundary Problems: A Finite Element Implementation. *Journal of Computational Physics*, 125:83–103, 1996.

[52] R. Scardovelli and S. Zaleski . Direct numerical simulation of free-surface and interfacial flow. *Annual Review of Fluid Mechanics*, 31:567–603, 1999.

[53] R. Temam. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires ii. *Archive for Rational Mechanics and Analysis*, 33(5):377–385, 1969.

[54] A. Thess and S.A. Orszag. Surface-tension-driven Bénard convection at infinite Prandtl number. *Journal of Fluid Mechanics*, 283:201–230, 1995.

[55] L.J.P. Timmermans, P.D. Minev, and F.N. Van de Vosse. An Approximate Projection Scheme for Incompressible Flow Using Spectral Elements. *International Journal for Numerical Methods in Fluids*, 22:673–688, 1996.

[56] D. Xiu and G.E. Karniadakis. A Semi-Lagrangian High-Order Method for Navier-Stokes Equations. *Journal of Computational Physics*, 172:658–684, 2001.

[57] H. Zhou and J.J. Derby. An assessment of a parallel, finite element method for three-dimensional, moving-boundary flows driven by capillarity for simulation of viscous sintering. *International Journal for Numerical Methods in Fluids*, 36:841–865, 2001.

Paper I

A high order splitting method for time-dependent domains

Tormod Bjøntegaard and Einar M. Rønquist.

Submitted to *Computer Methods in Applied Mechanics and Engineering.*

A high order splitting method for time-dependent domains

Tormod Bjøntegaard and Einar M. Rønquist

Norwegian University of Science and Technology,
Department of Mathematical Sciences, NO-7491 Trondheim, Norway

April 30, 2008

Abstract

We present a temporal splitting scheme for the semi-discrete convection-diffusion equation and the semi-discrete incompressible Navier-Stokes equations in time-depedent geometries. The proposed splitting scheme can be considered as an extension of the OIF-method proposed in [22] in the sense that it can be interpreted as a semi-Lagrangian method for time-dependent domains. The semi-discrete equations are derived from an arbitrary Lagrangian-Eulerian (ALE) formulation of the governing equations, and are discretized in space using high order spectral elements. The proposed splitting scheme has been tested numerically on model problems with known analytical solutions, and first, second, and third order convergence in time has been obtained. We also show that it is not necessary for the interior mesh velocity to be obtained through the use of an elliptic solver. Numerical tests show that it is sufficient that the mesh velocity is regular within each spectral element and only C^0-continuous across element boundaries; this is consistent with the theoretical results presented in [9]. In addition, the mesh velocity should be regular in the time direction.

Keywords: Time-dependent domains; ALE-formulation; operator splitting; spectral elements

1 Introduction

Numerical solution of the Navier-Stokes equations in time-dependent geometries has found wide-spread use in science and engineering, both in the context of basic understanding of fluid flow phenomena, as well as for predictive purposes in engineering. A powerful framework is provided by the arbitrary Lagrangian-Eulerian (ALE) formulation [13, 16, 6, 15]. Even though this framework is quite mature and is currently used in many commercial codes, it is still a subject of active research; e.g., see [15].

Part of the current research in ALE methods is related to time integration. One issue is the importance of satisfying the so-called geometric conservation law [18, 12, 10, 7]. The

conclusion is not quite clear for general Navier-Stokes problems. One complicating factor in all this effort is the fact that it is not easy to measure and verify the overall temporal accuracy during a transient simulation. This is partially due to the lack of analytical solutions for moving boundary problems, in particular, for general free surface problems where both normal and tangential stress boundary conditions are imposed.

The evolution of the surface of a time-dependent domain is typically determined via a kinematic condition which says that the normal domain velocity must coincide with the normal fluid velocity along the surface. Assuming that this surface evolution can be tracked in an accurate and efficient way, it still remains to extend the surface deformation to the interior of the domain. A smooth extension of the mesh velocity to the interior is most commonly used, e.g., using an harmonic extension, a Stokes solver, or an elasticity solver. However, other possible choices do not seem to have been fully explored; see [9] for a theoretical discussion of this issue.

The purpose of this paper is to present recent results on developing high order splitting methods for problems in time-dependent domains. Our long term goal is to be able to study large-scale free surface applications like three-dimensional Bénard-Marangoni convection including deformed surfaces [3], or problems involving fluids enclosed in flexible membranes on much larger length scales than typically associated with surface-tension-dominated effects. The latter problem is motivated by the transportation of fresh water using elastic fabric containers; see [19, 2].

In Section 2, we first present the governing equations for incompressible fluid flow and heat transfer problem in time-dependent domains. The ALE-formulation presented in Section 3 is the natural point of departure for the spatial discretization. In Section 4, we present a set of semi-discrete equations based on the spectral element method, however, any finite element method can in principle be used for the spatial discretization.

In Section 5, we present an operator splitting method for the temporal treatment of the convection-diffusion problem. The approach represents an extension of the OIF-method proposed in [22] to time-dependent domains. We conclude this section by showing numerical results for a two-dimensional test problem involving a moving front.

In Section 6, we discuss the proposed splitting scheme in the context of solving incompressible fluid flow problems. The splitting scheme represents a convection-Stokes splitting, which can also be interpreted as a semi-Lagrangian scheme. We present numerical evidence of first, second, and third order convergence in time for a three-dimensional ALE test problem with a known analytical solution. The issue of global regularity requirement for the mesh velocity is also illuminated.

In Section 7, we conclude our study and comment on future extensions.

2 Governing equations: strong form

In the following we consider the numerical solution of unsteady fluid flow and heat transfer problems in time-dependent domains. Specifically, we consider the incompressible Navier-Stokes equations and the convection-diffusion equation in a domain $\Omega(t)$,

$$\frac{\partial u_j}{\partial x_j} = 0, \qquad \text{in } \Omega(t), \tag{1}$$

$$\rho\left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j}\right) = \frac{\partial \sigma_{ij}}{\partial x_j} + f_i, \qquad \text{in } \Omega(t), \qquad i = 1, 2, 3, \tag{2}$$

$$\rho\, c_p\left(\frac{\partial \Theta}{\partial t} + u_j \frac{\partial \Theta}{\partial x_j}\right) = k\, \frac{\partial^2 \Theta}{\partial x_j \partial x_j} + g, \qquad \text{in } \Omega(t). \tag{3}$$

In (1) and (2), u_i is the i-th component of the fluid velocity in an inertial reference frame, x_j is the j-th coordinate, f_i is the i-th component of a volumetric body force, and ρ is the density of the fluid. Summation over repeated indices is assumed. The stress tensor σ_{ij} is here given as

$$\sigma_{ij} = -p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \qquad i, j = 1, 2, 3, \tag{4}$$

where p is the pressure, μ is the dynamic viscosity, and δ_{ij} is the Kronecker delta symbol. In (3), Θ is the temperature, c_p is the specific heat capacity, k is the thermal conductivity, and g is a volumetric heat source.

We consider here the stress formulation for incompressible fluid flow because our intended use of the proposed splitting scheme is to be able to accurately simulate time-dependent free surface flows with very general boundary conditions (including thermo-capillary effects). The particular boundary conditions used in the numerical tests in this paper will be discussed later.

3 ALE-formulation

In this section we briefly discuss the governing equations for fluid flow and heat transfer in time-dependent domains. In particular, we follow the arbitrary Lagrangian-Eulerian (ALE) framework which represents a powerful starting point for the numerical approximation of such problems [13, 16, 6, 15].

One of the key ingredients in the ALE-framework is the introduction of a domain velocity $\mathbf{w}$. Following closely the notation of [10] and [5], we can regard the time-dependent domain $\Omega(t)$ as a mapping $\mathcal{A}$ of a reference configuration $\Omega(t_0)$, e.g., the domain at an earlier

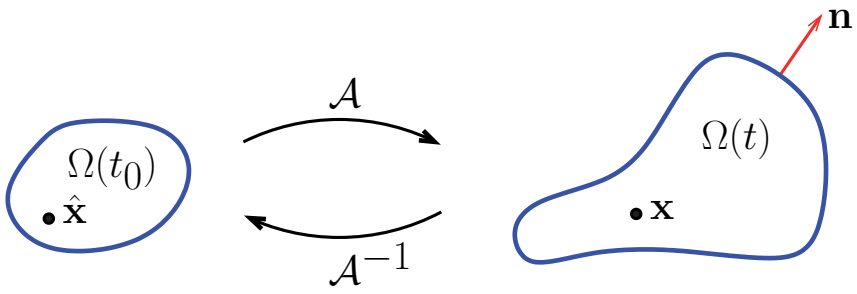


Figure 1: The time-dependent domain $\Omega(t)$ can be regarded as a one-to-one mapping $\mathcal{A}$ of a reference configuration $\Omega(t_0)$. The outward unit normal vector along $\partial\Omega(t)$ is denoted as $\mathbf{n}$.

time t_0; see Figure 1. We assume that $\mathcal{A}$ is a continuous and one-to-one mapping, i.e., a unique point $\hat{\mathbf{x}}$ in $\Omega(t_0)$ maps to a unique point $\mathbf{x}$ in $\Omega(t)$,

$$\begin{aligned} &\mathcal{A} \in C^0(\overline{\Omega(t_0)}, t), \\ &\mathbf{x} = \mathcal{A}(\hat{\mathbf{x}}, t). \end{aligned}$$

In particular,

$$\partial\Omega(t) = \mathcal{A}(\partial\Omega(t_0), t).$$

The domain velocity $\mathbf{w}$ at a point $\mathbf{x}$, corresponding to a particular location $\hat{\mathbf{x}}$ in the reference configuration, can then be defined as

$$\mathbf{w} = \left(\frac{\partial \mathcal{A}}{\partial t}\right)\Bigg|_{\hat{\mathbf{x}}} \circ \mathcal{A}^{-1}(\mathbf{x}, t).$$

If $\mathbf{u}$ represents the fluid velocity in $\Omega(t)$, it follows from the continuum hypothesis that

$$\mathbf{w} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n} \quad \text{ along } \partial\Omega(t). \tag{5}$$

The condition (5) is called the kinematic condition.

The ALE-formulation can be derived from the weak form of the governing equations using an Eulerian framework, and then apply the Reynolds' transport theorem and Euler's expansion formula [1]. Following this approach, the ALE-formulation of the heat transfer problem can be expressed as: Find Θ (the temperature) $\in X \subset H^1(\Omega)$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}(v, \Theta) + c(v, \Theta) - e(v, \Theta) = -a_\Theta(v, \Theta) + (v, g) + \ell_\Theta(v), \qquad \forall v \in X, \tag{6}$$

where we have defined the following bilinear forms,

$$(v, \phi) = \int_{\Omega(t)} v\phi \, \mathrm{d}V, \tag{7}$$

$$c(v, \phi) = \int_{\Omega(t)} v(u_j - w_j)\frac{\partial \phi}{\partial x_j} \, \mathrm{d}V, \tag{8}$$

$$e(v, \phi) = \int_{\Omega(t)} v\phi\frac{\partial w_j}{\partial x_j} \, \mathrm{d}V, \tag{9}$$

$$a_\Theta(v, \phi) = \int_{\Omega(t)} k \, \frac{\partial v}{\partial x_j}\frac{\partial \phi}{\partial x_j} \, \mathrm{d}V, \tag{10}$$

as well as the linear form

$$\ell_\Theta(v) = \int_{\partial\Omega(t)} v\frac{\partial \Theta}{\partial n} \, \mathrm{d}S. \tag{11}$$

With no loss in generality, we have set ρc_p equal to unity, and we have assumed homogeneous Dirichlet boundary conditions for Θ along part of, or all of, the boundary $\partial\Omega(t)$.

One advantage with the form (6) is that the time-derivative appears outside the integral over $\Omega(t)$; this will prove very useful for the subsequent numerical treatment. Second, the contribution from convection appears in two terms: a standard convection term where a relative convection velocity $(\mathbf{u} - \mathbf{w})$ appears (see (8)), as well as an "expansion" term involving the divergence of the domain velocity (see (9)). The first term on the right hand side represents the standard diffusion term resulting from integration by parts, while the third term represents the associated surface term allowing for a convenient imposition of flux boundary conditions; as usual, this term vanishes wherever essential boundary conditions are prescribed. The second term on the right hand side represents a prescribed heat source (which we assume is square integrable).

A similar procedure for the fluid problem yields the ALE-formulation of the incompressible Navier-Stokes equations: find $\mathbf{u} \in X \subset (H^1(\Omega))^3$ and $p \in Y \subset L^2(\Omega)$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{v}, \mathbf{u}) + c(\mathbf{v}, \mathbf{u}) - e(\mathbf{v}, \mathbf{u}) = -a_\sigma(\mathbf{v}, \mathbf{u}) + d(p, \mathbf{v}) + (\mathbf{v}, \mathbf{f}) + \ell_\sigma(\mathbf{v}), \qquad \forall \mathbf{v} \in X, \tag{12}$$

$$d(q, \mathbf{u}) = 0, \qquad \forall q \in Y, \tag{13}$$

where we have introduced the bilinear forms

$$(\mathbf{v}, \mathbf{u}) = \int_{\Omega(t)} v_i u_i \, \mathrm{d}V, \tag{14}$$

$$c(\mathbf{v}, \mathbf{u}) = \int_{\Omega(t)} v_i (u_j - w_j) \frac{\partial u_i}{\partial x_j} \, \mathrm{d}V, \tag{15}$$

$$e(\mathbf{v}, \mathbf{u}) = \int_{\Omega(t)} v_i u_i \frac{\partial w_j}{\partial x_j} \, \mathrm{d}V, \tag{16}$$

$$a_\sigma(\mathbf{v}, \mathbf{u}) = \int_{\Omega(t)} \mu \frac{\partial v_i}{\partial x_j} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \mathrm{d}V, \tag{17}$$

$$d(q, \mathbf{u}) = \int_{\Omega(t)} q \frac{\partial u_j}{\partial x_j} \, \mathrm{d}V, \tag{18}$$

as well as the linear form

$$\ell_\sigma(v) = \int_{\partial\Omega(t)} v_i \sigma_{ij} n_j \, \mathrm{d}S. \tag{19}$$

With no loss in generality, we have here set ρ equal to unity. In the above definitions of the bilinear and linear forms, summation over repeated indices is assumed. Similar to the heat transfer problem, we have assumed homogeneous Dirichlet boundary conditions for the velocity $\mathbf{u}$ along part of, or all of, the boundary $\partial\Omega(t)$.

The linear form (19) follows from integration by parts of the term $\int_{\Omega(t)} v_i \frac{\partial \sigma_{ij}}{\partial x_j} \, \mathrm{d}V$; this surface term allows for a convenient imposition of stress boundary conditons (both normal and tangential), while the surface term vanishes wherever essential velocity boundary conditions are prescribed.

4 Semi-discrete problem

The weak form presented above will be our point of departure for the spatial and temporal discretization. Our goal is to achieve high order accuracy both in the space and time. We start with a brief discussion of the spatial discretization which will be based on spectral elements [21]. Following this approach, we decompose the domain into disjoint elements and approximate all the field variables as high-order polynomials within each element. Appropriate C^0-continuity conditions are imposed across interelement boundaries for second-order problems as considered here. We assume that a high-order, tensor-product nodal

basis is used. Following this approach, we arrive at a system of semi-discrete equations for the heat transfer problem (6) on the form

$$\frac{\mathrm{d}}{\mathrm{d}t}(\underline{\mathbf{B}}\,\underline{\Theta}) + \underline{\mathbf{C}}\,\underline{\Theta} = -\underline{\mathbf{A}}\,\underline{\Theta} + \underline{\mathbf{E}}\,\underline{\Theta} + \underline{\mathrm{b}}, \tag{20}$$

$$\frac{\mathrm{d}\underline{\mathbf{x}}}{\mathrm{d}t} = \underline{\mathbf{w}}. \tag{21}$$

Equation (20) represents the semi-discrete convection-diffusion equation derived from the ALE-formulation, while equation (21) represents the system of ordinary differential equations governing the mesh evolution. All the lower case symbols represent vectors of nodal values at a particular time: $\underline{\Theta}$ represents the temperature, $\underline{\mathbf{x}}$ represents the coordinates of the grid points, $\underline{\mathbf{u}}$ and $\underline{\mathbf{w}}$ represent the fluid velocity and mesh velocity, respectively, and $\underline{\mathrm{b}}$ represents the known data (source term and boundary conditions). Furthermore, $\underline{\mathbf{B}}$ represents the time-dependent mass matrix derived from (7), which in our case is diagonal because of the fact that the quadrature points and nodal points that we use within each spectral element coincide [21]. The matrix $\underline{\mathbf{A}}$ represents the discrete Laplacian derived from (10), which is time-dependent because the computational domain is time-dependent. The matrix $\underline{\mathbf{C}}$ represents the convection operator derived from (8); this will again depend on time through the time-dependent computational domain, but also via the fluid velocity $\underline{\mathbf{u}}$ and the mesh velocity $\underline{\mathbf{w}}$. Finally, the matrix $\underline{\mathbf{E}}$ represents the discrete "expansion" term associated with the bilinear form (9).

In a similar way we can derive a set of semi-discrete equations for the incompressible fluid flow problem (12)-(13); these equations can be expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}(\underline{\mathbf{B}}\,\underline{\mathbf{u}}) + \underline{\mathbf{C}}\,\underline{\mathbf{u}} = -\underline{\mathbf{A}}_\sigma\,\underline{\mathbf{u}} + \underline{\mathbf{D}}^T\underline{p} + \underline{\mathbf{E}}\,\underline{\mathbf{u}} + \underline{\mathbf{b}}, \tag{22}$$

$$\underline{\mathbf{D}}\,\underline{\mathbf{u}} = \underline{0}, \tag{23}$$

$$\frac{\mathrm{d}\underline{\mathbf{x}}}{\mathrm{d}t} = \underline{\mathbf{w}}. \tag{24}$$

Here, $\underline{\mathbf{B}}$ represents the time-dependent mass matrix derived from (14) (i.e., for the vector case), $\underline{\mathbf{C}}$ represents the time-dependent convection operator derived from (15) (which is now nonlinear and couples all the velocity components), $\underline{\mathbf{A}}_\sigma$ represents the symmetric, viscous operator derived from (17) (which couples all the velocity components), $\underline{\mathbf{D}}$ represents the discrete divergence operator derived from (18), while $\underline{\mathbf{D}}^T$ is the corresponding discrete gradient operator, $\underline{\mathbf{E}}$ represents the discrete "expansion" operator derived from (16) (i.e., for the vector case), and $\underline{\mathbf{b}}$ is a known right hand side derived from (14) and (19).

We remark that other finite-element-based discretization methods could also have been used for the spatial discretization; the resulting semi-discrete equations could still be expressed on the form (20)-(21), or (22)-(24), and thus the following discussion regarding the temporal treatment also applies to such discretizations.

5 A convection-diffusion splitting scheme

Our goal is here to propose a high order temporal splitting scheme for (20)-(21); with high order we shall here mean higher than first order, in particular, second and third order convergence in time. Our point of departure will be the Operator-Integration-Factor (OIF) procedure proposed in [22]. The computational approach we propose in this paper can be viewed as an extension of the OIF-method to time-dependent domains.

5.1 Fixed domain

Before we discuss the details of the new scheme, let us first revisit some aspects of the original OIF-approach applied to the convection-diffusion problem. First, we assume a fixed domain Ω. Following a standard Eulerian description, the semi-discrete equations for the convection-diffusion problem can be expressed as

$$\underline{\mathbf{B}}\,\frac{\mathrm{d}\underline{\Theta}}{\mathrm{d}t} + \underline{\mathbf{C}}\,\underline{\Theta} = -\underline{\mathbf{A}}\,\underline{\Theta} + \underline{\mathrm{b}}, \tag{25}$$

where now all the discrete spatial operators are time-independent. Although the OIF-method presented in [22] offers a quite general framework for deriving temporal splitting methods, this method applied to the particular convection-diffusion problem (25) can also be interpreted as a particular semi-Lagrangian method. Specifically, a first order splitting scheme reads

$$\underline{\mathbf{B}}\left(\frac{\underline{\Theta}^{n+1} - \underline{\widetilde{\Theta}}^{n+1}}{\Delta t}\right) = -\underline{\mathbf{A}}\,\underline{\Theta}^{n+1} + \underline{\mathrm{b}}^{n+1}, \tag{26}$$

where the expression inside the parentheses on the left hand side represents a first order approximation to the total derivative $\mathrm{D}\Theta/\mathrm{D}t$ at time t^{n+1}. With this interpretation $\underline{\widetilde{\Theta}}^{n+1}$ represents the values of Θ at time t^n for those fluid particles which at time t^{n+1} coincide with the *fixed* grid points used in a pure Eulerian formulation; note that the position these fluid particles had at time t^n do *not* coincide with the grid points.

An attractive aspect with the OIF-method is the fact that it is possible to find the values $\underline{\widetilde{\Theta}}^{n+1}$ only by using information at the fixed grid points. This is in contrast to other semi-Lagrangian schemes where the positions of the fluid particles at earlier times first have to be computed by following the characteristics backwards in time, and then the solution at an earlier time needs to be interpolated at these points; see [23, 11]. In the OIF-method [22], however, the values $\underline{\widetilde{\Theta}}^{n+1}$ are found by solving the following pure convection problem defined in the interval t^n to t^{n+1},

$$\underline{\mathbf{B}}\,\frac{\mathrm{d}\underline{\widetilde{\Theta}}}{\mathrm{d}t} = -\underline{\mathbf{C}}\,\underline{\widetilde{\Theta}}, \qquad \underline{\widetilde{\Theta}}^{n} = \underline{\Theta}^{n}, \qquad t^n \le t \le t^{n+1}. \tag{27}$$

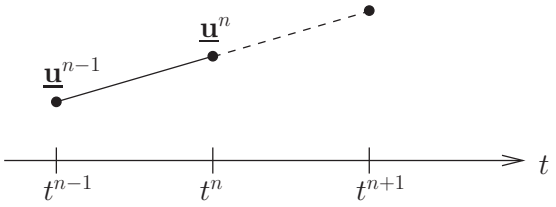


Figure 2: The convecting velocity field is approximated linearly in time in the interval $[t^{n-1}, t^{n+1}]$ for a second order OIF-scheme. In particular, the convecting velocity field is constructed as the linear interpolant between t^{n-1} and t^n, and as the linear extrapolant between t^n and t^{n+1}.

In essence, the values $\underline{\widetilde{\Theta}}^{n+1}$ we are interested in are being convected to the fixed grid points through the solution of (27).

The extension to second order in time is quite natural. We now use a second-order (Backward Differentiation) approximation of the total derivative $\mathrm{D}\Theta/\mathrm{D}t$ at time t^{n+1},

$$\underline{\mathbf{B}} \left(\frac{\frac{3}{2}\underline{\Theta}^{n+1} - 2\underline{\widetilde{\Theta}}^{n+1} + \frac{1}{2}\underline{\widetilde{\widetilde{\Theta}}}^{n+1}}{\Delta t} \right) = -\underline{\mathbf{A}}\,\underline{\Theta}^{n+1} + \underline{\mathrm{b}}^{n+1}. \tag{28}$$

The values of Θ at times t^n and t^{n-1} for those fluid particles which at time t^{n+1} coincide with the fixed grid points are now found by solving the two pure convection problems

$$\underline{\mathbf{B}}\,\frac{\mathrm{d}\underline{\widetilde{\Theta}}}{\mathrm{d}t} = -\underline{\mathbf{C}}\,\underline{\widetilde{\Theta}}, \qquad \underline{\widetilde{\Theta}}^{n} = \underline{\Theta}^{n}, \qquad t^n \le t \le t^{n+1}, \tag{29}$$

$$\underline{\mathbf{B}}\,\frac{\mathrm{d}\underline{\widetilde{\widetilde{\Theta}}}}{\mathrm{d}t} = -\underline{\mathbf{C}}\,\underline{\widetilde{\widetilde{\Theta}}}, \qquad \underline{\widetilde{\widetilde{\Theta}}}^{n-1} = \underline{\Theta}^{n-1}, \qquad t^{n-1} \le t \le t^{n+1}. \tag{30}$$

We now discuss the temporal discretization of the pure convection problems defined above. First, the discrete convection operator $\underline{\mathbf{C}}$ depends on the given convection velocity. In practice, this velocity field is typically coming from a Navier-Stokes solver and is only known at discrete times t^n, t^{n-1}, etc. We therefore approximate the convection velocity in time by using a polynomial interpolant/extrapolant. Specifically, for a first order approximation, see (26) and (27), the convection velocity is set equal to $\underline{\mathbf{u}}^n$ for the entire time interval $t^n \le t \le t^{n+1}$ in (27). For a second order approximation, the convection velocity used in (29) and (30) is approximated linearly between t^{n-1} and t^{n+1}; see Figure 2.

In order to solve the single initial value problem (27), or the two initial value problems (29) and (30), we use the classical fourth order explicit Runge-Kutta scheme (ERK4). An explicit scheme avoids the need to solve non-symmetric (and in the case of solving the Navier-Stokes equations, nonlinear) equation systems. Second, the ERK4 scheme is attractive to use since it is accurate and the associated stability region encloses a significant

part of the imaginary axis. The time step used for these "inner" convection problems can be the same as for the "outer" diffusion problem, however, this is not a requirement. On the other hand, we are required to honor the Courant stability criterion and we also need to honor a final integration time equal to t^{n+1}.

Finally, we mention that the OIF-approach presented above may also be extended to third order in time. In this case, we use a third order Backward Differentiation approximation of the total derivative, and we need to solve three separate convection problems. Furthermore, the given convection field is now constructed as a second order interpolant/extrapolant in the interval between t^{n-2} and t^{n+1}.

5.2 Time-dependent domain

We now consider the extension of the OIF-approach to time-dependent domains. In particular, we consider the solution of (20) and (21). As mentioned earlier, all the discrete spatial operators are now time-dependent.

In order to more easily apply the OIF formalism to this problem, we first define the new variable

$$\underline{\Phi} = \underline{\mathbf{B}}\,\underline{\Theta}. \tag{31}$$

The convection-diffusion equation (20) can then be expressed as

$$\frac{\mathrm{d}\underline{\Phi}}{\mathrm{d}t} + \underline{\mathbf{C}}\,\underline{\mathbf{B}}^{-1}\underline{\Phi} = -(\underline{\mathbf{A}} - \underline{\mathbf{E}})\,\underline{\mathbf{B}}^{-1}\underline{\Phi} + \underline{\mathrm{b}}. \tag{32}$$

Next, we apply the OIF-method to the system (32). Treating the expansion term as part of the "outer" problem, a first order splitting scheme can be expressed as

$$\left(\frac{\underline{\mathbf{B}}^{n+1}\underline{\Theta}^{n+1} - \underline{\widetilde{\Phi}}^{n+1}}{\Delta t}\right) = -(\underline{\mathbf{A}}^{n+1} - \underline{\mathbf{E}}^{n+1})\,\underline{\Theta}^{n+1} + \underline{\mathrm{b}}^{n+1}. \tag{33}$$

The values $\underline{\widetilde{\Phi}}^{n+1}$ are obtained through the solution of the following ("inner") convection problem,

$$\frac{\mathrm{d}\underline{\widetilde{\Phi}}}{\mathrm{d}t} = -\underline{\mathbf{C}}\,\underline{\mathbf{B}}^{-1}\,\underline{\widetilde{\Phi}}, \qquad \underline{\widetilde{\Phi}}^{n} = \underline{\mathbf{B}}^{n}\underline{\Theta}^{n}, \qquad t^n \le t \le t^{n+1}. \tag{34}$$

Finally, we solve the system (21) using an explicit multi-step scheme; for a first order approximation in time, we simply use a first order Adams-Bashforth scheme (i.e, Euler Forward),

$$\left(\frac{\underline{\mathbf{x}}^{n+1} - \underline{\mathbf{x}}^{n}}{\Delta t}\right) = \underline{\mathbf{w}}^{n}. \tag{35}$$

We now make a few remarks concerning this splitting scheme. Similar to the standard OIF-scheme for fixed domains, we use a Backward-Differentiation (BD) scheme for the

"outer" problem. This is mainly due to the convenience of only having to evaluate all the associated operators on the right hand side at time level t^{n+1}. The associated "Operator Integrating Factor" is defined to be the identity operator at t^{n+1} [22], and the use of a BD-scheme thus avoids having to solve additional "inner" problems.

Second, we note that the introduction of the new variable Φ in (31) necessitates a modified "inner" convection problem in (34) compared to the corresponding problem (27) for time-independent domains. This modification is necessary due to the fact that the mass matrix is time-dependent.

Third, the discrete operator $\underline{\mathbf{C}}\,\underline{\mathbf{B}}^{-1}$ in (34) depends on the convecting velocity field $\mathbf{u}$, on the mesh velocity $\mathbf{w}$, and on the computational domain Ω. All these quantities are approximated as constants, and equal to the the corresponding values at time t^n for a first order splitting scheme.

Fourth, similar to the system (27), the system (34) is solved using ERK4. An important observation here is the following. In the ALE formulation the convection operator $\underline{\mathbf{C}}$ is derived from the associated bilinear form (8). This form includes an "effective" convection velocity $\mathbf{u} - \mathbf{w}$. However, using integration-by-parts, we can easily show that this form is skew-symmetric, i.e.,

$$c(v, \phi) = -c(\phi, v). \tag{36}$$

This follows by noticing that the boundary integral over $\partial\Omega$ vanishes either due to essential boundary conditions or due to the kinematic condition (5). Hence, we are guaranteed that all the eigenvalues of the matrix $\underline{\mathbf{C}}\,\underline{\mathbf{B}}^{-1}$ are pure imaginary, and the ERK4 scheme is thus appropriate to use.

We now discuss the choice of including the "expansion" term $\underline{\mathbf{E}}^{n+1}\,\underline{\Theta}^{n+1}$ in the outer problem (33). First, this term is derived from the associated bilinear form (9). This bilinear form is symmetric, but has a "coefficient" $\nabla \cdot \mathbf{w}$ which can be either positive or negative definite depending on whether the domain is locally expanding or contracting. The matrix $\underline{\mathbf{E}}$ is therefore symmetric, but the definiteness is not determined. If we include the term $\underline{\mathbf{E}}\,\underline{\Theta}$ in the "inner" problem, the discrete operator in (34) changes from $\underline{\mathbf{C}}\,\underline{\mathbf{B}}^{-1}$ to $(\underline{\mathbf{C}} - \underline{\mathbf{E}})\,\underline{\mathbf{B}}^{-1}$. In this case, we cannot guarantee that all the eigenvalues will remain inside the absolute stability region of ERK4 (or inside the stability region of other explicit time integration schemes) since some eigenvalues may end up with positive real parts.

We remark that the "expansion" term vanishes if we insist on having a divergence free mesh velocity. Such a constraint has been proposed as a way to honor the so-called Geometric Conservation Law (GCL); see [12, 7, 10]. For example, the work presented in [4] honors the GCL condition through the computation of a divergence free mesh velocity, which can be achieved through the solution of a Stokes problem. However, since the significance of the GCL condition is not quite clear for general problems, we will here not assume such a constraint, and we therefore have to treat the additional term appropriately. Our goal with this study is also to gain more insight into the global regularity requirements for the mesh velocity.

From (33) it follows that we need to solve a system of equations for Θ^{n+1} of the form

$$\left(\underline{\mathbf{A}}^{n+1} + \frac{1}{\Delta t}\,\underline{\mathbf{B}}^{n+1} - \underline{\mathbf{E}}^{n+1}\right)\Theta^{n+1} = \underline{\mathrm{b}}^{n+1}, \tag{37}$$

where $\underline{\mathrm{b}}^{n+1}$ represents a known right hand side (we assume that we already have solved the "inner" problem for $\widetilde{\Phi}^{n+1}$). The first two terms inside the parentheses represent the discrete Helmholtz operator. As mentioned earlier, the matrix $\underline{\mathbf{E}}$ is symmetric (and, in our case, diagonal), but we cannot guarantee the definiteness of this matrix. On the other hand, by combining the last two terms inside the parantheses in (37), we conclude that we are guaranteed positive definiteness if

$$(1 - \Delta t \nabla \cdot \mathbf{w}) > 0.$$

However, from the basic definition of the divergence, this is the same as saying that the *relative* change in a small volume element in one single time step should be less than one. This is typically always true: a local volume element will generally not double in size in a single time step. Hence, we can assume that the system matrix in (37) is symmetric and positive definite, and that we can use iterative solvers for such systems, typically, the conjugate gradient method.

The extension to second and third order in time is quite similar to the corresponding extension for fixed domains. For example, a second order splitting scheme will read

$$\left(\frac{\frac{3}{2}\underline{\mathbf{B}}^{n+1}\underline{\Theta}^{n+1} - 2\underline{\widetilde{\Phi}}^{n+1} + \frac{1}{2}\underline{\widetilde{\widetilde{\Phi}}}^{n+1}}{\Delta t}\right) = (-\underline{\mathbf{A}}^{n+1} + \underline{\mathbf{E}}^{n+1})\,\underline{\Theta}^{n+1} + \underline{\mathrm{b}}^{n+1}. \tag{38}$$

where

$$\frac{\mathrm{d}\underline{\widetilde{\Phi}}}{\mathrm{d}t} = -\underline{\mathbf{C}}\,\underline{\mathbf{B}}^{-1}\underline{\widetilde{\Phi}}, \qquad \underline{\widetilde{\Phi}}^{n} = (\underline{\mathbf{B}}\,\underline{\Theta})^{n}, \qquad t^n \le t \le t^{n+1}, \tag{39}$$

$$\frac{\mathrm{d}\underline{\widetilde{\widetilde{\Phi}}}}{\mathrm{d}t} = -\underline{\mathbf{C}}\,\underline{\mathbf{B}}^{-1}\underline{\widetilde{\widetilde{\Phi}}}, \qquad \underline{\widetilde{\widetilde{\Phi}}}^{n-1} = (\underline{\mathbf{B}}\,\underline{\Theta})^{n-1}, \qquad t^{n-1} \le t \le t^{n+1}. \tag{40}$$

In addition, the system (21) is now solved using a second order Adams-Bashforth scheme,

$$\left(\frac{\underline{\mathbf{x}}^{n+1} - \underline{\mathbf{x}}^{n}}{\Delta t}\right) = \frac{3}{2}\,\underline{\mathbf{w}}^{n} - \frac{1}{2}\,\underline{\mathbf{w}}^{n-1}. \tag{41}$$

A special remark is required when solving the "inner" convection problems (39) and (40). For a *fixed* geometry, we recall that we need to use a first order polynomial approximation in time for the convecting velocity field; see Figure 2. For *time-dependent* geometries, we need to use a first order approximation in time for the convecting velocity

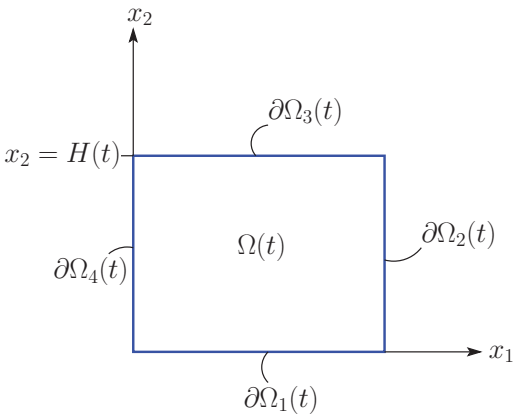


Figure 3: The rectangular domain used for the moving front problem. The height of the domain is given as $H(t)$.

field, for the mesh velocity, as well as for the geometry. Again, the approximations are based on linear interpolations/extrapolations of the values at t^{n-1} and t^n.

Finally, the approach presented above can readily be extended to a third order splitting scheme. Similar to the fixed geometry case, we need to solve three separate ("inner") convection problems. In addition, the convecting velocity field, the mesh velocity, as well as the geometry are now all constructed as second order interpolants/extrapolants in the interval between t^{n-2} and t^{n+1}. The problem (21) is in this case solved using a third order Adams-Bashforth scheme.

5.3 Numerical results

One difficulty with assessing the accuracy of a discretization scheme in time-dependent domains is the lack of analytical solutions, especially for problems in general domains. We now present a two-dimensional convection-diffusion problem we have designed in order to verify the proposed computational approach in the context of a moving front. Specifically, we consider the solution of the time-dependent convection-diffusion equation

$$\frac{\partial \Theta}{\partial t} + \mathbf{u} \cdot \nabla \Theta = \nabla^2 \Theta + g, \quad \text{in } \Omega(t),$$

where $\Omega(t) = (0,1) \times (0, H(t))$ is the rectangular domain depicted in Figure 3. The boundary conditions are

$$\begin{aligned} \Theta &= 0, && \text{on } \partial\Omega_1(t),\ \partial\Omega_3(t), \\ \frac{\partial \Theta}{\partial n} &= 0, && \text{on } \partial\Omega_2(t),\ \partial\Omega_4(t). \end{aligned}$$

We impose a two-dimensional convecting velocity field $\mathbf{u} = (u_1, u_2)$, with

$$u_1(x_1, x_2, t) = \pi \sin\left(\frac{2\pi x_2}{H(t)}\right) \sin^2(\pi x_1) \sin(t),$$
$$u_2(x_1, x_2, t) = -H(t)\pi \sin(2\pi x_1) \sin^2\left(\frac{\pi x_2}{H(t)}\right) \sin(t),$$

and we choose the right hand side $g(x_1, x_2, t)$ such that the exact solution $\Theta(x_1, x_2, t)$ of the two-dimensional convection-diffusion equation is given as

$$\Theta(x_1, x_2, t) = \sin\left(\frac{\pi x_2}{H(t)}\right).$$

In order to mimic a "melting-front" problem (a Stefan problem), we assume that the speed of the front is determined through the condition

$$\frac{\mathrm{d}H}{\mathrm{d}t} = -\frac{\partial \Theta}{\partial x_2}\bigg|_{x_2 = H(t)}. \tag{42}$$

From the above information we can derive an analytical solution for the front,

$$H(t) = \sqrt{2\pi t + H_0^2}, \tag{43}$$

where H_0 is the "height" of the domain at time $t = 0$.

We first discretize the domain Ω using two spectral elements, Ω_1 and Ω_2. Next, we solve the semi-discrete equations using the splitting method proposed in Section 5.2. Note that we never use our knowledge about the exact solution (43) to advance the front. Instead, we use (42) to compute the x_2-component of the grid velocity, w_2, along the front directly from the numerical solution. This grid velocity is then extended to the interior of the domain by requiring that: (i) the x_1-component of the grid velocity is zero, i.e., $w_1 = 0$ in Ω; (ii) the x_2-component of the grid velocity is extended smoothly from the front to the interior of the domain; specifically, $w_2 \in \mathbb{P}_1(\Omega)$, i.e., w_2 varies linearly with x_2 in Ω.

The linear extension used here can also be regarded as an harmonic extension of the grid velocity along the boundary to the interior of the domain. Such a regular extension of the grid velocity is quite common to use in the context of the ALE-formulation. Another common approach is to use an elasticity solver [14, 8] or a Stokes solver [4].

We now integrate the governing equations until a final time, T, where we compare the numerical solution with the exact solution. The initial and final domain is depicted in Figure 4. Note that we start the simulation with two equal-sized spectral elements and that these will remain equal-sized due to the linearly varying grid velocity in the x_2-direction. In Figure 5 we show the temporal and spatial convergence behavior. The discretization error is measured in the energy-norm by first mapping the solution back to the initial configuration, and then performing the error calculation over $\Omega(t_0)$. When the temporal

error is dominating, we clearly see first, second, and third order convergence. When the spatial error is dominating, we obtain exponential convergence until we reach the temporal error level, at which point increasing degree, N, of the polynomial approximation within each spectral element does not have any effect.

Let us now revisit this problem, but this time change the global regularity of the mesh velocity. In particular, let us extend the mesh velocity derived from the numerical solution along the front to the interior of the domain in the following way: As earlier, we set the x_1-component equal to zero, i.e., $w_1 = 0$. However, this time we extend the x_2-component such that $w_2 = 0$ in Ω_1 and $w_2 \in \mathbb{P}_1(\Omega_2)$. Hence, w_2 is very regular within each spectral element (in fact, piecewise linear), but w_2 is globally only $C^0(\Omega)$; see Figure 6. The x_2-component of the grid velocity has a sharp jump in the x_2-derivative along the element boundary. Again, we integrate the equations until a final time T and compare with our earlier results. We limit our comparison to the second order splitting scheme, and when the temporal error is dominating. The convergence results are shown in Figure 7. We observe that it is not necessary to require the ALE mapping to be $C^\infty(\Omega)$, or even $C^1(\Omega)$; in this case, C^0-continuity of the mesh velocity suffices. These results are in agreement with the theoretical analysis and comments given in [9].

6 A convection-Stokes splitting scheme

For the case with a fixed domain, the treatment of (22)-(23) is precisely the OIF-method described in [22]. The extension to time-dependent domains follows a similar approach as for the convection-diffusion equation. The "inner" convection problems are treated similarly to the scalar case. However, we remark that the convection problems now represent nonlinear problems, even in fixed geometries. The "outer" problem can be expressed as (e.g., for a second order splitting scheme):

$$\left(\frac{\frac{3}{2}\underline{\mathbf{B}}^{n+1}\underline{\mathbf{u}}^{n+1} - 2\underline{\widetilde{\mathbf{\Psi}}}^{n+1} + \frac{1}{2}\underline{\widetilde{\widetilde{\mathbf{\Psi}}}}^{n+1}}{\Delta t} \right) = (-\underline{\mathbf{A}}_\sigma^{n+1} + \underline{\mathbf{E}}^{n+1})\,\underline{\mathbf{u}}^{n+1} + \underline{\mathbf{D}}^T \underline{p}^{n+1} + \underline{\mathbf{b}}^{n+1}, \tag{44}$$

$$\underline{\mathbf{D}}\,\underline{\mathbf{u}}^{n+1} = 0, \tag{45}$$

where $\underline{\mathbf{\Psi}} = \underline{\mathbf{B}}\,\underline{\mathbf{u}}$ corresponds to the transformation (31) for the convection-diffusion case. The Stokes system (44)-(45) can be solved for the velocity $\underline{\mathbf{u}}^{n+1}$ and $\underline{p}^{n+1}$ via a standard Uzawa decoupling procedure; e.g., see [20]. In the following, we will focus on such a pure convection-Stokes decoupling approach; the alternative is to also include a pressure-velocity decoupling in the Stokes operator.

We remark that the operator splitting scheme presented here, i.e., the convection-Stokes splitting scheme for the semi-discrete Navier-Stokes equations, or the convection-diffusion splitting scheme presented in the previous section, will include temporal splitting errors at steady state (for problems where a steady state solution exists). This is similar to the OIF-method applied to problems in fixed geometries. The reason for this can be understood by

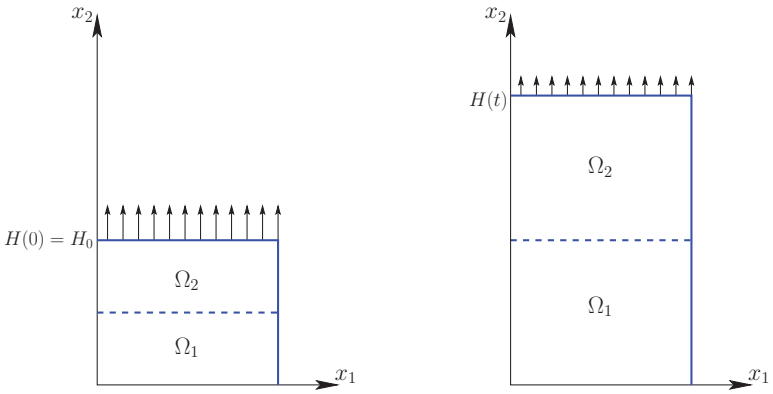


Figure 4: The initial computational domain (left) and the final computational domain (right). The two spectral elements used to solve this problem are denoted as Ω_1 and Ω_2. The mesh velocity computed along the front is extended linearly to zero in the interior of the domain.

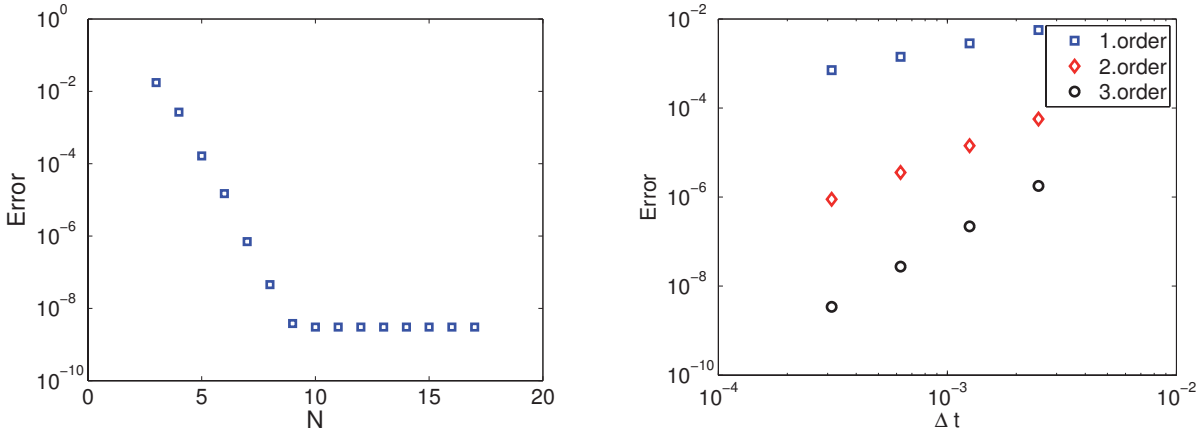


Figure 5: Discretization error in the energy-norm for the moving front test problem. The left plot shows the spatial discretization error as a function of the polynomial degree, N, when the spatial error is dominating. The right plot shows the temporal discretization error as a function of the time step, Δt, when the temporal error is dominating. Results are reported for the first, second, and third order splitting scheme discussed in Section 5.2.

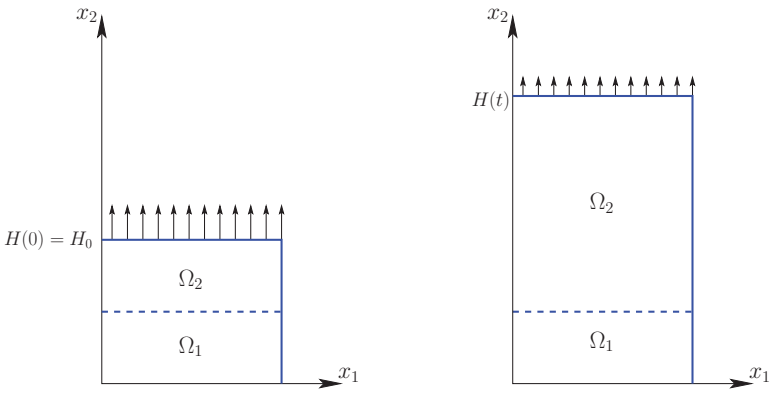


Figure 6: The initial computational domain (left) and the final computational domain (right). The two spectral elements used to solve this problem are denoted as Ω_1 and Ω_2. The mesh velocity computed along the front is here extended linearly to zero in the interior of Ω_2, while the mesh velocity in Ω_1 is identically zero during the entire simulation.

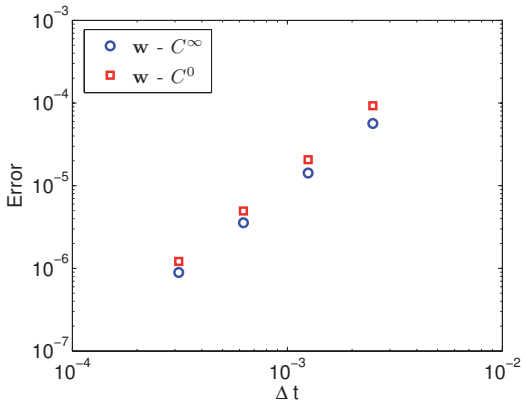


Figure 7: Discretization error in the energy-norm at a final time, T, for the moving front problem. A second order splitting scheme has been used, and the temporal error is dominating. The convergence results using a globally regular mesh velocity (in fact, $w_2 \in C^\infty(\Omega)$) is compared with using a mesh velocity which varies linearly within each spectral element, but is globally only C^0-continuous (with a significant jump in the derivative across the element boundary).

interpreting the splitting scheme as a semi-Lagrangian scheme were the total derivative is approximated along the characteristics in the upwind direction via a first, second, or third order backward differentiation scheme. In particular, the left-hand side of (38) and the left hand side of (44) both represent a streamline-upwind approximation to the convective term.

Note also that, similar to the convection-Stokes splitting presented in [22] for fixed geometries, no interpolation between the grid points is needed in order to determine the necessary field values along the characteristics; the solution of the "inner" convection sub-problems will give us the necessary information only using values at the current and previous time steps, and only using the nodal values (i.e., the values at the grid points) at the current and previous time steps.

6.1 Numerical results

We now verify our discretization approach by solving the three-dimensional Navier-Stokes equations in a cube. The domain boundary is fixed at all times, however, we specify an artificial time-periodic mesh velocity in the interior. The mesh velocity is a function of both space (all the coordinates) and time, and is zero on the domain boundary. We also specify a forcing function in the momentum equations by requiring that the following analytic solution satisfies the incompressible Navier-Stokes equations:

$$u_1(x_1, x_2, x_3, t) = \frac{\pi}{5} \sin^2(\pi x_1) \sin(2\pi x_2) \sin(2\pi x_3) \sin(t), \tag{46}$$

$$u_2(x_1, x_2, x_3, t) = -\frac{\pi}{10} \sin(2\pi x_1) \sin^2(\pi x_2) \sin(2\pi x_3) \sin(t), \tag{47}$$

$$u_3(x_1, x_2, x_3, t) = -\frac{\pi}{10} \sin(2\pi x_1) \sin(2\pi x_2) \sin^2(\pi x_3) \sin(t), \tag{48}$$

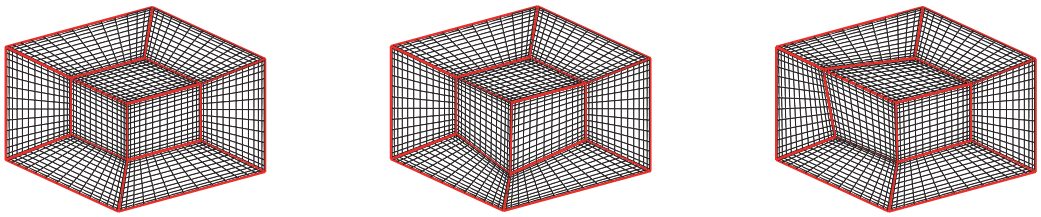
$$p(x_1, x_2, x_3, t) = \cos(\pi x_1) \sin(\pi x_2) \sin(\pi x_3) \sin(t). \tag{49}$$

Note that the prescribed mesh velocity is only C^0-continuous in space. Inside each spectral element, the mesh velocity is analytic in both space and time. However, the gradient of the mesh velocity is not continuous across element boundaries; in fact, the mesh velocity is identically zero in one of the spectral elements. On the other hand, we impose a mesh velocity which is very regular in time; see Figure 8. Finally, we remark that the mesh velocity is not divergence free. The convergence results in Figure 9 show the anticipated behavior: first, second, and third order convergence in time, and exponential convergence in space for problems with analytic solutions and data.

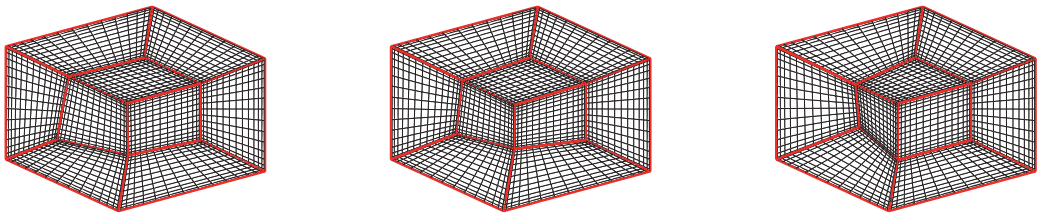
Even though we are using isoparametric spectral elements, the geometry representation is in this case effectively built upon using a trilinear approximation within each element; this is due to the fact that all element edges (internal and external) are straight. The mesh velocity is here constructed in the following way: we first define some non-trivial motion of the internal element *vertices*. We then define the mesh velocity in such a way that a numerical integration of (24) using Adams-Bashforth methods results in straight

spectral element edges. One reason for doing this is to ensure a perfect distribution of all the spectral element nodes during the simulation.

The test reported here is obviously a very artificial one; the natural choice is to use a fixed geometry since the external boundary of the cube is fixed. However, aside from providing information about the discretization error using the ALE formulation and a splitting approach, this test also allows us to repeat the numerical experiment using a fixed geometry and compare the discretization errors. In Figure 10 we compare the temporal and spatial errors when using a fixed geometry (i.e., $\mathbf{w} = \mathbf{0}$), and when imposing the artificial mesh velocity depicted in Figure 8. For example, for a fixed Δt, this plot indicates "the price" we have to pay for using an ALE-formulation where we could have expressed the governing equations in a fixed geometry. For the particular test problem we have chosen here, our numerical results indicate that this "price" is about one order or magnitude for a second order scheme and about two orders of magnitude for a third order scheme.



(a) $t = 0$ (b) $t = 0.2$ (c) $t = 0.4$



(d) $t = 0.6$ (e) $t = 0.8$ (f) $t = 1.0$

Figure 8: The three-dimensional box used in the convergence study of the ALE scheme. The domain is decomposed into seven spectral elements, one in the middle of the domain and one connected to each of the six faces of the box. The external boundary of the box is fixed. However, we specify a mesh velocity in the interior of the cube which is a function of both space and time (periodic in time). The plot indicates the grid-configuration at a few time levels for four of the spectral elements during one single period. The exact flow solution in the domain is given by (46)-(49).

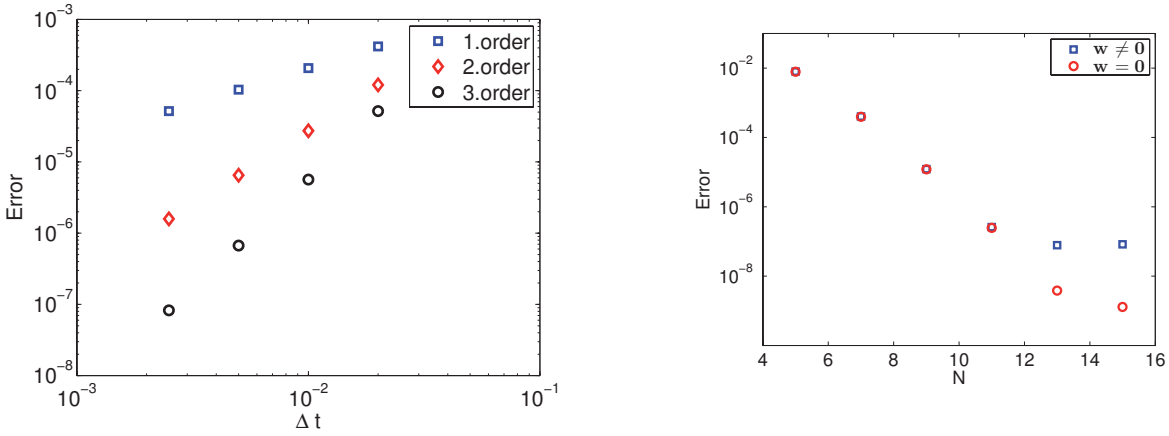


Figure 9: The left plot depicts the discretization error in the energy norm at time $T = 3$ as a function of the time step, Δt, for a first, second, and third order temporal splitting scheme; the spatial error is here subdominant the temporal error. The right plot depicts the discretization error as a function of the polynomial degree, N, used in each spectral element; the temporal error is here subdominant the spatial error for $N < 12$.

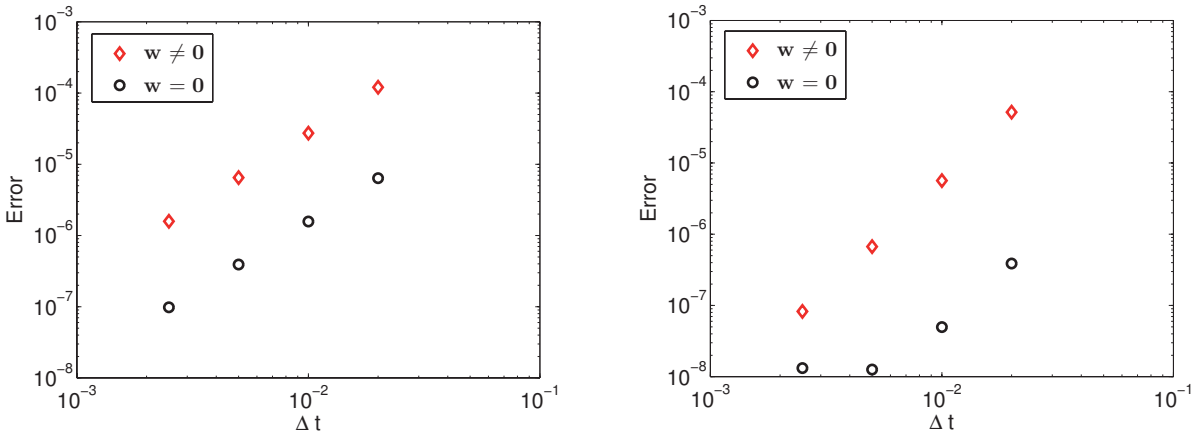


Figure 10: A comparison of the discretization error in the energy norm at time $T = 3$ with and without imposing an artificial mesh velocity in the interior of the domain; see Figure 8. The left plot shows the temporal error for the second order OIF splitting scheme, while the right plot shows the corresponding results for a third order scheme. The spatial discretization error is here on the order of 10^{-8}.

6.2 Comparison with other schemes

We now compare the OIF splitting approach for the convection-Stokes splitting with two other schemes: the scheme discussed in [14] (the HP scheme), and a slightly modified version of the scheme discussed in [17] (the KIO scheme) and used for moving boundary problems in [4]. The difference between the KIO scheme and the scheme used in this work is that we do not split the Stokes operator into a separate pressure step and a separate viscous step.

Similar to the OIF-scheme, both these schemes treat the convection term explicitly, while the Stokes operator is treated implicitly using a backward difference scheme. With reference to the semi-discrete equations (24), these two schemes can be expressed succinctly as

$$\frac{1}{\Delta t}\sum_{k=0}^{s}\beta_k(\underline{\mathbf{B}}\,\underline{\mathbf{u}})^{n+1-k} = -(\underline{\mathbf{A}}_\sigma\,\underline{\mathbf{u}})^{n+1} + (\underline{\mathbf{D}}^T\underline{p})^{n+1} + \sum_{k=0}^{s}\alpha_k((\underline{\mathbf{C}}+\underline{\mathbf{E}})\underline{\mathbf{u}})^{n-k} + \underline{\mathbf{b}}^{n+1}. \quad (50)$$

Here, s is the order of the scheme. The treatment of the divergence constraint and the mesh evolution is similar to the OIF-scheme.

In [14], α_k, $k = 0, 1, 2$, where chosen to be modified AB-coefficients for a second order scheme, while in [17], α_k, $k = 0, 1$, were chosen such that they correspond to a linear extrapolation of the convection term at time t^{n+1}. In Table 1 we also give the corresponding coefficients for a third order approach.

In Figure 11 we report numerical results showing a comparison of the three methods. We see that all methods give the correct order, and also give very similar results (with the OIF scheme proposed in this work giving a marginally better constant).

7 Conclusions

We have presented a temporal splitting scheme for the semi-discrete convection-diffusion equation and the semi-discrete incompressible Navier-Stokes equations in time-dependent domains. The proposed splitting scheme can be considered as an extension of the OIF-method proposed in [22] in the sense that it can be interpreted as a semi-Lagrangian method for time-dependent domains. The computational approach has been tested numerically on model problems with known analytical solutions, for which first, second, and third order convergence has been obtained. Based on the experience so far, the results for the OIF splitting scheme compares favorably with two alternative approaches proposed in the literature. The approach has been used successfully in the context of simulating three-dimensional Bénard-Marangoni flows with a deformable free surface [3].

We remark that all the splitting schemes discussed in this paper for the Navier-Stokes equations fundamentally focus on two types of splittings: (i) splitting the treatment of the geometry evolution from the treatment of the interior Navier-Stokes calculation; and (ii)

		First order	Second order	Third order
BD	β_0	1	$\frac{3}{2}$	$\frac{11}{6}$
	β_1	-1	-2	-3
	β_2	0	$\frac{1}{2}$	$\frac{3}{2}$
	β_3	0	0	$-\frac{1}{3}$
HP	α_0		$\frac{8}{3}$	$\frac{15}{4}$
	α_1		$-\frac{7}{3}$	$-\frac{21}{4}$
	α_2		$\frac{2}{3}$	$\frac{13}{4}$
	α_3		0	$-\frac{3}{4}$
KIO	α_0		2	3
	α_1		-1	-3
	α_2		0	1
	α_3		0	0

Table 1: Backward Differentiation (BD) coefficients β_k, together with the coefficients α_k; see (50). The coefficients α_k for the HP scheme are taken from [14] (second order scheme only), while the coefficients for the KIO scheme are taken from [17].

splitting the treatment of the (ALE) convection operator in the fluid problem from the Stokes operator. We have not considered splitting the Stokes operator itself into a pressure step and a viscous step; this would have introduced an additional splitting error. Instead, we have solved the unsteady Stokes problem via a standard Uzawa algorithm, which does not correspond to a rediscretization of the Stokes operator, but rather a decoupling algorithm.

The numerical results show that the mesh velocity introduced in the ALE-formulation does not have to be globally smooth in space. In the context of the isoparametric spectral element discretization used in this study, it is sufficient that the mesh velocity is regular within each spectral element, and C^0-continuous across element boundaries. Hence, the extension of the mesh velocity from the boundary to the interior does not necessarily have to be obtained through the use of an elliptic solver which is commonly the case. This observation is consistent with the theoretical discussion and comments in [9]. This could allow for the use of faster and more flexible ways of updating the mesh velocity at each time step; we plan to explore this opportunity in a future study. Note that, in order to obtain high order temporal accuracy, the mesh velocity should be regular in the time direction.

The proposed splitting method does not appear to satisfy the Geometric Conservation Law. Based on the tests we have done so far, we have thus not been able to conclude to what extent the quality of a general Navier-Stokes solution will improve if the GCL condition is satisfied. We remark that the mesh velocity used in this study is not divergence free.

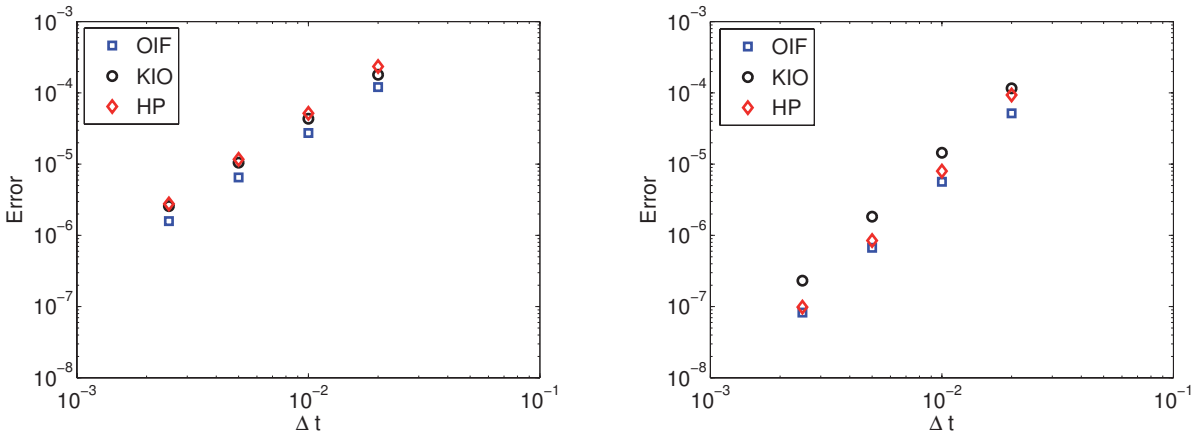


Figure 11: A comparison of the temporal error in the energy norm as a function of the time step, Δt, for the following second order (left plot) and third order (right plot) schemes: the OIF scheme presented in this paper, the HP splitting scheme discussed in [14] (also extended to third order in this study), and the KIO splitting scheme discussed in [17] (here used in a slightly modified form). For all these results, the spatial error is subdominant the temporal error.

Despite the encouraging results obtained in this study, more quantitative comparisons still need to be done in the context of free surface problems with larger and more complex deformations (i.e., free surfaces with significant curvature). An obvious challenge is to derive analytical solutions in more complex time-dependent domains. However, we believe it is necessary to obtain quantitative results for more complex problems in order to discriminate the quality of different ALE-solvers used in a more realistic setting. For example, controlling the accuracy of a moving interface with a varying (mean) curvature is a non-trivial task which we feel has not yet been treated satisfactorily in the literature; the interface-tracking issue is currently under investigation in a separate study.

Acknowledgment

The authors would like to thank the Norwegian University of Science and Technology for the financial support of this project.

References

[1] R. Aris. *Vectors, Tensors, and the Basic Equations of Fluid Mechanics.* Dover Publications, Inc., 1962.

[2] T. Bjøntegaard and E.M. Rønquist. High order methods for incompressible fluid flow: Application to free surface problems. In B. Skallerud and H. I. Andersson, editors, *MekIT'07*. Tapir Akademisk Forlag, 2007.

[3] T. Bjøntegaard and E.M. Rønquist. Simulation of three-dimensional Bénard-Marangoni flows including deformed surfaces. *Communications in Computational Physics*, to appear.

[4] N. Bodard, R. Bouffanais, and M.O. Deville. Solution of moving-boundary problems by the spectral element method. *Applied Numerical Mathematics (2007),doi:10.1016/j.apnum.2007.04.009*, 2007.

[5] D. Boffi and L. Gastaldi. Stability and geometric conservation laws for ALE formulations. *Computer Methods in Applied Mechanics and Engineering*, 193:4717–4739, 2004.

[6] J. Donea S. Giuliani and J.P Halleux. An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions. *Computer Methods in Applied Mechanics and Engineering*, 33:689–723, 1982.

[7] C. Farhat and P. Geuzaine. Design and analysis of robust ALE time-integrators for the solution of unsteady flow problems on moving grids. *Computer Methods in Applied Mechanics and Engineering*, 193:4073–4095, 2004.

[8] C. Farhat, M. Lesoinne, and N. Maman. Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation, geometry conservation and distributed solution. *International Journal for Numerical Methods in Fluids*, 21:807–835, 1995.

[9] L. Formaggia and F. Nobile. A stability analysis for the arbitrary Lagrangian Eulerian formulation with finite elements. *East-West Journal of Numerical Mathematics*, 7(2):105–131, 1999.

[10] L. Formaggia and F. Nobile. Stability analysis of second-order time accurate schemes for ALE-FEM. *Computer Methods in Applied Mechanics and Engineering*, 193:4097–4116, 2004.

[11] F.X. Giraldo. The Lagrange-Galerkin Spectral Element Method on Unstructured Quadrilateral Grids. *Journal of Computational Physics*, 147:114–146, 1998.

[12] H. Guillard and C. Farhat. On the significance of the geometric conservation law for flow computations on moving meshes. *Computer Methods in Applied Mechanics and Engineering*, 190:1467–1482, 2000.

[13] C.W. Hirt, A.A. Amsden, and J.L Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14:227–253, 1974.

[14] L.W. Ho and A.T. Patera. A Legendre spectral element method for simulation of unsteady incompressible viscous free-surface flows. *Computer Methods in Applied Mechanics and Engineering*, 80:355–366, 1990.

[15] A. Huerta and A. Rodríguez-Ferran (eds.). The Arbitrary Lagrangian-Eulerian Formulation. *Computer Methods in Applied Mechanics and Engineering*, 193(39-41):4073–4456, 2004.

[16] T.J.R. Hughes, W.K. Liu, and T.K Zimmermann. Lagrangian-Eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 29:329–349, 1981.

[17] G.E. Karniadakis, M. Israeli, and S.A. Orszag. High-Order Splitting Methods for the Incompressible Navier-Stokes Equations. *Journal of Computational Physics*, 97:414–443, 1991.

[18] M. Lesoinne and C. Farhat. Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer Methods in Applied Mechanics and Engineering*, 134:71–90, 1996.

[19] G. Løland and J.V. Aarsnes. Fabric as construction material for marine applications. *Hydroelasticity in Marine Technology*, pages 275–286, 1994.

[20] Y. Maday, D. Meiron, A.T. Patera, and E.M. Rønquist. Analysis of iterative methods for the steady and unsteady stokes problem: application to spectral element discretizations. *SIAM Journal on Scientific Computing*, 14(2):310–337, 1993.

[21] Y. Maday and A.T. Patera. Spectral element methods for the Navier-Stokes equations. *in: A.K. Noor, J. T. Oden (Eds.), State of the Art Surveys in Computational Mechanics, ASME, New York*, pages 71–143, 1989.

[22] Y. Maday, A.T. Patera, and E.M. Rønquist. An Operator-Integration-Factor Splitting Method for Time-Dependent Problems: Application to Incompressible Fluid Flow. *Journal of Scientific Computing*, 5(4):263–292, 1990.

[23] D. Xiu and G.E. Karniadakis. A Semi-Lagrangian High-Order Method for Navier-Stokes Equations. *Journal of Computational Physics*, 172:658–684, 2001.

Paper II

Imposing free-surface boundary conditions using surface intrinsic coordinates

Tormod Bjøntegaard.

Preprint Numerics No.5/07, Department of Mathematical Sciences,
Norwegian University of Science and Technology.

Imposing free-surface boundary conditions using surface intrinsic coordinates

Tormod Bjøntegaard

April 24, 2008

We consider here the incompressible Navier-Stokes equations in three dimensions subject to free surface boundary conditions along part or all of the domain boundary. In [2] a surface integral for weak imposition of both normal and tangential surface tension boundary conditions was proposed. This integral was based on describing the free surface using surface-intrinsic coordinates. In this paper we derive the proposed surface integral using results from differential geometry. A key ingredient in this derivation is an expression for the curvature-normal product, and most of the paper will be devoted to deriving this expression.

1 Introduction

We consider here the incompressible Navier-Stokes equations in three dimensions subject to free surface boundary conditions along part or all of the domain boundary. One critical aspect of the numerical approximation of such problems is the incorporation of these boundary conditions. This is related to the fact that the shape of the free surface is generally unknown, and the normal and tangential stresses in the presence of surface tension depend on the local curvature and possibly the surface gradient of the surface tension. The free surface boundary conditions can be expressed as

$$
\begin{aligned}
F_n &= n_i \sigma_{ij} n_j = \gamma \kappa, \\
F_t &= t_i \sigma_{ij} n_j = t_k (\nabla_s \gamma)_k,
\end{aligned}
$$

where $n_i, i = 1, 2, 3$, are the components of the outward unit normal vector ($\mathbf{n}$), $t_i, i = 1, 2, 3$, are the components of a tangent vector ($\mathbf{t}$), F_n is the normal component of the stress force, F_t is a tangential component of the stress force in the direction of $\mathbf{t}$, σ_{ij} is the stress tensor, γ is the surface tension and κ is twice the mean curvature. Here, $\nabla_s \gamma$ represents the surface gradient of the surface tension and $t_k(\nabla_s \gamma)_k$ represents the component of the surface gradient in the direction of $\mathbf{t}$. Summation over repeated indices is assumed.

In order to naturally incorporate the free surface boundary conditions, we need to use the stress formulation of the Navier-Stokes equations. For the numerical treatment we will use the corresponding weak formulation of the free surface problem. This involves multiplying

the governing equations with suitable test functions and integrating over our computational domain, Ω. Integration by parts on the viscous term yields the surface integral

$$\int_{\Gamma_\gamma} v_i \sigma_{ij} n_j \, \mathrm{d}S, \tag{1.1}$$

where $\Gamma_\gamma = \partial\Omega_\gamma$ is the free surface, v_i is a test function, and $\sigma_{ij} n_j$ are the total stress forces in the i'th direction; it is through this integral the imposition of the free-surface boundary conditions will be done. In [2] Ho and Patera proposed an alternative surface integral for weak imposition of both normal and tangential surface tension boundary conditions by the use of surface intrinsic coordinates. This alternative form reads,

$$-\int_{\Gamma_\gamma} \gamma \frac{\partial v_i}{\partial r^\alpha} g_i^\alpha \, \mathrm{d}S, \tag{1.2}$$

where r^1 and r^2 are surface parameters, and $\mathbf{g}^1$ and $\mathbf{g}^2$ are two vectors spanning the tangent plane which will be introduced later. To our knowledge a complete derivation of this surface integral has not been done, and the aim of this paper is to derive in detail all the necessary steps in order to go from (1.1) to (1.2). We remark that the total stress forces

$$\mathbf{F} = \mathbf{F}_n + \mathbf{F}_t = \gamma \mathbf{n} \kappa + \nabla_s \gamma,$$

where $\mathbf{F}_n$ is the normal force and $\mathbf{F}_t$ represents the tangential force. Thus, we see that in order to derive (1.2) we need to find an expression for the curvature normal product and the surface gradient in the context of surface intrinsic coordinates.

In Section 2 we will go through some basic concepts in differential geometry. This is by no means an exhaustive introduction to differential geometry, but more of a preliminary covering of the necessary tools needed in the later part of this paper. In Section 3 we cover some necessary quantities that we will need, like surface divergence, surface gradient and the mean curvature for a general surface, and in Section 4 we start from (1.1) and derive (1.2).

2 Differential geometry: preliminaries

The introduction to differential geometry in this section is mainly influenced by [1], both in contents and notation.

2.1 Cartesian coordinate system

A vector in the cartesian coordinate system, $x^{i'}$, can be written on the form,

$$\mathbf{V} = V_1 \mathbf{i}_{x^{1'}} + V_2 \mathbf{i}_{x^{2'}} + V_3 \mathbf{i}_{x^{3'}}.$$

Here, $\mathbf{i}^{i'}, i = 1, 2, 3$, are an orthonormal basis for $\mathbb{R}^d$ (Figure 2.1) such that $\mathbf{i}_{x^{m'}} \cdot \mathbf{i}_{x^{n'}} = \delta_{m'n'}$. As an example, the inner-product between two vectors written in this notation can be expressed as,

$$\mathbf{V} \cdot \mathbf{u} = V_1 u_1 + V_2 u_2 + V_3 u_3,$$

since the orthogonality makes the cross-terms zero.

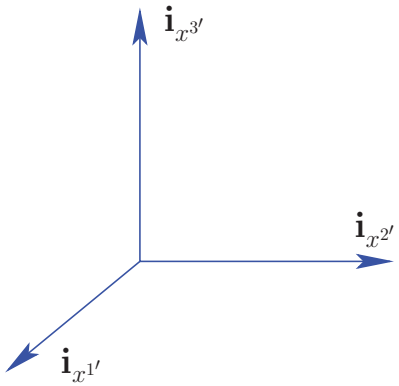


Figure 2.1: Cartesian coordinate system with orthonormal base vectors.

In the following, we will use $x^{i'}$ for the cartesian coordinate system and x^i or x^{i^*} for a general coordinate system.

2.2 General coordinate system

If we instead consider another basis for $\mathbb{R}^d$,

$$\mathbb{R}^d = \text{span}\{\mathbf{g}_i\}_{i=1}^d,$$

a vector in $\mathbb{R}^3$ can be written

$$\mathbf{V} = V^1 \mathbf{g}_1 + V^2 \mathbf{g}_2 + V^3 \mathbf{g}_3.$$

For this basis, we have no restriction on orthogonality between the base vectors nor that they have unit length. The only requirement is that they are linearly independent. The inner-product between two vectors in $\mathbb{R}^3$, $\mathbf{V}$ and $\mathbf{u}$, expressed in this basis reads

$$\begin{aligned}\mathbf{V}\cdot\mathbf{u} = &V^1u^1\mathbf{g}_1\cdot\mathbf{g}_1 + V^1u^2\mathbf{g}_1\cdot\mathbf{g}_2 + V^1u^3\mathbf{g}_1\cdot\mathbf{g}_3\\ &+ V^2u^1\mathbf{g}_2\cdot\mathbf{g}_1 + V^2u^2\mathbf{g}_2\cdot\mathbf{g}_2 + V^2u^3\mathbf{g}_2\cdot\mathbf{g}_3\\ &+ V^3u^1\mathbf{g}_3\cdot\mathbf{g}_1 + V^3u^2\mathbf{g}_3\cdot\mathbf{g}_2 + V^3u^3\mathbf{g}_3\cdot\mathbf{g}_3.\end{aligned}$$

We see that all nine entries are included, since the basis vectors in general are not orthogonal.

2.3 The base-vectors

We will consider a point, P, with an associated position vector, $\mathbf{p}$. A small increment, $\mathrm{d}\mathbf{p}$, may be expressed as

$$\mathrm{d}\mathbf{p} = \frac{\partial\mathbf{p}}{\partial x^i}\mathrm{d}x^i, \tag{2.1}$$

where x^i $(i = 1,\ldots,d)$ denotes the reference frame, see Figure 2.2. However, if we assume

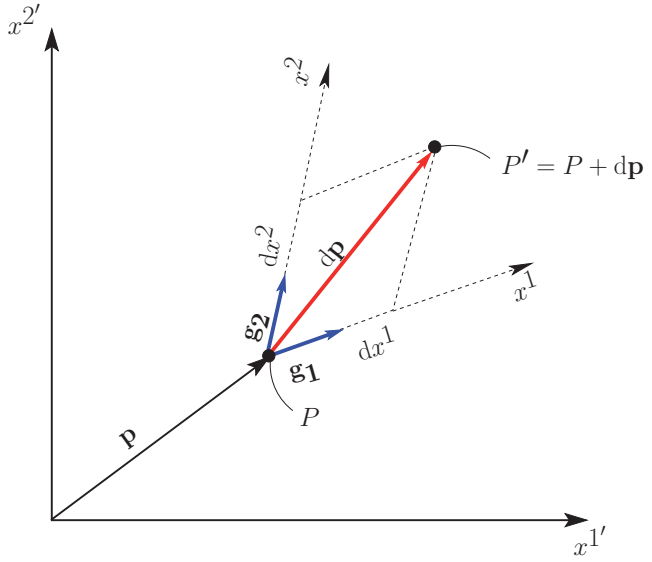


Figure 2.2: A differential expressed in a general coordinate system in two dimensions.

that each base-vector, $\mathbf{g}_i$, points in the same directions as x^i, we may choose $\{\mathbf{g}_i\}$ such that

$$\mathrm{d}\mathbf{p} = \mathbf{g}_i\mathrm{d}x^i, \tag{2.2}$$

and thus, an expression for the i'th base-vector is

$$\mathbf{g}_i = \frac{\partial\mathbf{p}}{\partial x^i}. \tag{2.3}$$

We now introduce another basis for $\mathbb{R}^d$,

$$\mathbb{R}^d = \mathrm{span}\{\mathbf{g}^i\}_{i=1}^d,$$

with the property that $\mathbf{g}^m\cdot\mathbf{g}_n = \delta_{mn} = \delta^m_n$, the Kronecker delta. This is displayed in two dimensions in Figure 2.3.

A vector in $\mathbb{R}^3$ may now be expressed as

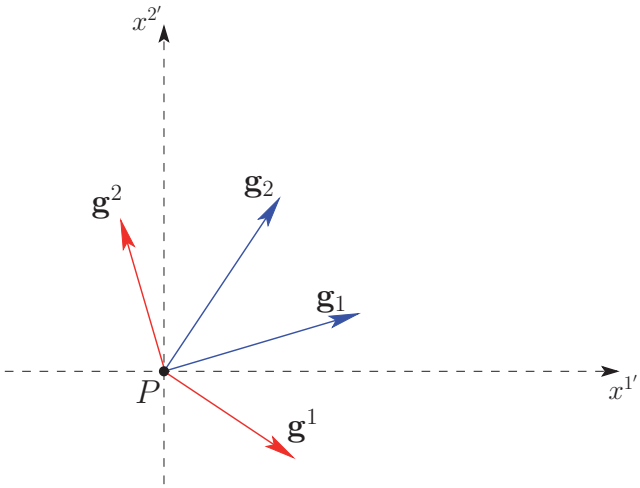


Figure 2.3: Covariant and contravariant basis vectors associated with the point P in two dimensions.

$$\mathbf{V} = V_1\mathbf{g}^1 + V_2\mathbf{g}^2 + V_3\mathbf{g}^3.$$

If we return to the example of the inner-product of two vectors, $\mathbf{u}$ and $\mathbf{v}$, where $\mathbf{u}$ is expressed in the basis $\{\mathbf{g}_i\}$ and $\mathbf{v}$ is expressed in $\{\mathbf{g}^j\}$, we get the simplified formula,

$$\mathbf{u} \cdot \mathbf{v} = u^1 v_1 + u^2 v_2 + u^3 v_3 = u^i v_i.$$

In literature, $\mathbf{g}_i$ is often denoted as the i'th *covariant* base-vector, and $\mathbf{g}^j$ as the j'th *contravariant* base-vector.

2.3.1 The metric tensor

Since both the covariant and contravariant base-vectors represent a basis for $\mathbb{R}^d$, the covariant vector, $\mathbf{g}_i$, can be expressed in terms of contravariant base-vectors,

$$\mathbf{g}_i = g_{ij}\mathbf{g}^j.$$

Here, g_{ij} are the components of the *covariant metric tensor*. Similarly, we have

$$\mathbf{g}^i = g^{ij}\mathbf{g}_j.$$

By the use of the orthogonality relation, we get

$$\mathbf{g}_i \cdot \mathbf{g}_j = g_{ik}\mathbf{g}^k \cdot \mathbf{g}_j = g_{ij} = g_{ji},$$
$$\mathbf{g}^i \cdot \mathbf{g}^j = g^{ik}\mathbf{g}_k \cdot \mathbf{g}^j = g^{ij} = g^{ji},$$

and

$$\delta_i^j = \mathbf{g}_i \cdot \mathbf{g}^j = g_{ik}\mathbf{g}^k \cdot g^{jl}\mathbf{g}_l = g_{ik}g^{jl}\mathbf{g}^k \cdot \mathbf{g}_l = g_{ik}g^{jk}.$$

Thus, if the covariant components of the metric tensor is known, it is straightforward to find the contravariant components.

2.4 Transformation between coordinate systems

We now assume that we have two reference frames, $\{x_i\}_{i=1}^d$ and $\{x_i^*\}_{i=1}^d$, with two corresponding sets of covariant base-vectors, $\mathbf{g}_i$ and $\mathbf{g}_{i^*}$ (see Figure 2.4). We also have contravariant vectors which satisfy the relations

$$\mathbf{g}_i \cdot \mathbf{g}^j = \delta_i^j,$$
$$\mathbf{g}_{i^*} \cdot \mathbf{g}^{j^*} = \delta_{i^*}^{j^*}.$$

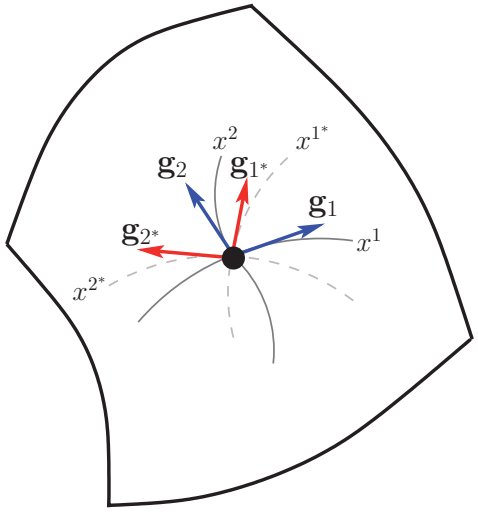


Figure 2.4: Two sets of base-vectors in two dimensions.

These four sets of base-vectors span the same space, and can therefore be expressed in terms of each other. Thus, we can write,

$$\mathbf{g}_{i^*} = \beta_{i^*}^j \mathbf{g}_j,$$
$$\mathbf{g}^{i^*} = \beta_j^{i^*} \mathbf{g}^j.$$

Notice that the summation is over the superscript of β_i^j for the first case, and the subscript for the second case. Further,

$$\delta_{i^*}^{k^*} = \mathbf{g}_{i^*} \cdot \mathbf{g}^{k^*} = \beta_{i^*}^j \mathbf{g}_j \cdot \beta_l^{k^*} \mathbf{g}^l = \beta_{i^*}^j \beta_l^{k^*} \mathbf{g}_j \cdot \mathbf{g}^l = \beta_{i^*}^j \beta_l^{k^*} \delta_j^l$$
$$\Downarrow$$
$$\beta_{i^*}^j \beta_j^{k^*} = \delta_{i^*}^{k^*}, \tag{2.4}$$

and in the same manner we have

$$\beta_i^{j^*} \beta_{j^*}^k = \delta_i^k. \tag{2.5}$$

2.4.1 The components, $\beta_i^{j^*}$

As before, we may write a small vector, $\mathrm{d}\mathbf{s}$, in terms of the covariant base-vectors,

$$\mathrm{d}\mathbf{s} = \mathbf{g}_i \, \mathrm{d}x^i = \beta_i^{k^*} \mathbf{g}_{k^*} \, \mathrm{d}x^i,$$

by the use of two reference frames, $\{x_i\}_{i=1}^d$ and $\{x_i^*\}_{i=1}^d$. The same vector may also be expressed as

$$\mathrm{d}\mathbf{s} = \mathbf{g}_{k^*} \, \mathrm{d}x^{k^*} = \mathbf{g}_{k^*} \frac{\partial x^{k^*}}{\partial x^i} \, \mathrm{d}x^i.$$

Thus,

$$\beta_i^{k^*} \mathbf{g}_{k^*} \, \mathrm{d}x^i = \mathbf{g}_{k^*} \frac{\partial x^{k^*}}{\partial x^i} \, \mathrm{d}x^i$$

should be valid for all $\mathrm{d}\mathbf{s}$. By systematically choosing,

$$\begin{aligned} \mathrm{d}x^1 = 1, \quad \mathrm{d}x^2 = 0, \quad \mathrm{d}x^3 = 0, \\ \mathrm{d}x^1 = 0, \quad \mathrm{d}x^2 = 1, \quad \mathrm{d}x^3 = 0, \\ \mathrm{d}x^1 = 0, \quad \mathrm{d}x^2 = 0, \quad \mathrm{d}x^3 = 1, \end{aligned}$$

we find that

$$\beta_i^{k^*} \mathbf{g}_{k^*} = \mathbf{g}_{k^*} \frac{\partial x^{k^*}}{\partial x^i}.$$

Taking the inner-product of both sides with $\mathbf{g}^{j^*}$, we get

$$\begin{aligned} \beta_i^{k^*} \mathbf{g}_{k^*} \cdot \mathbf{g}^{j^*} &= \mathbf{g}_{k^*} \cdot \mathbf{g}^{j^*} \frac{\partial x^{k^*}}{\partial x^i}, \\ \Downarrow & \\ \beta_i^{j^*} &= \frac{\partial x^{j^*}}{\partial x^i}. \end{aligned}$$

2.5 The permutation tensor

We wish to be able to evaluate cross-products in general coordinates. This is closely related to a tensor often called the *permutation tensor*, and in order to derive this tensor we will first introduce the *permutation symbols*.

2.5.1 The permutation symbols

The permutation symbols, $e_{ijk} = e^{ijk}$, are defined by

$e_{ijk} = +1,$	if i, j, k is a cyclic sequence $(1,2,3;\ 2,3,1$ or $3,1,2$ for the 3×3 case.)
$e_{ijk} = -1,$	if i, j, k is a anticyclic sequence $(3,2,1;\ 2,1,3$ or $1,3,2$ for the 3×3 case.)
$e_{ijk} = 0,$	if i, j, k is acyclic (two or more subscripts are equal.)

2.5.2 The determinant

We define a^2 to be the determinant of a matrix a_i^j:

$$a^2 = |a_i^j| = \begin{vmatrix} a_1^1 & a_2^1 & a_3^1 \\ a_1^2 & a_2^2 & a_3^2 \\ a_1^3 & a_2^3 & a_3^3 \end{vmatrix}.$$

By the use of the permutation symbols, this can also be written as

$$a^2 = a_1^i a_2^j a_3^k e_{ijk},$$

or

$$a^2 = a_l^1 a_m^2 a_n^3 e^{lmn}.$$

2.5.3 The permutation tensor

We now define the permutation tensor connected to the cartesian coordinate system, $x^{i'}$, as

$$\varepsilon_{i'j'k'} = e_{i'j'k'}. \tag{2.6}$$

If we wish to transform this tensor to another coordinate system, x^i, we may do this in the same way as introduced earlier,

$$\varepsilon_{ijk} = \varepsilon_{i'j'k'} \beta_i^{i'} \beta_j^{j'} \beta_k^{k'} = e_{i'j'k'} \beta_i^{i'} \beta_j^{j'} \beta_k^{k'}.$$

However, we recognize this as $|\beta_i^{i'}|$ if $\{i, j, k\}$ is a cyclic sequence and $-|\beta_i^{i'}|$ if $\{i, j, k\}$ is an anticyclic sequence,

$$\begin{aligned} \varepsilon_{ijk} &= |\beta_i^{i'}|, \qquad \text{if } i, j, k \text{ is cyclic,} \\ \varepsilon_{ijk} &= -|\beta_i^{i'}|, \qquad \text{if } i, j, k \text{ is anticyclic.} \end{aligned} \tag{2.7}$$

From (2.4), we know that

$$\beta_{i'}^j \beta_j^{k'} = \delta_{i'}^{k'},$$

thus,

$$\begin{gathered} |\beta_{i'}^j \beta_j^{k'}| = 1, \\ \Downarrow \\ |\beta_{i'}^j| = \Delta, \qquad |\beta_j^{i'}| = \frac{1}{\Delta}, \end{gathered}$$

where Δ is currently unknown. We define g^2 to be the determinant of the metric tensor,

$$g^2 = |g_{ij}| = \begin{vmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{vmatrix}.$$

Since,

$$g_{ij}g^{jk} = \delta_i^k,$$

we have

$$\begin{aligned} |g_{ij}g^{jk}| &= 1, \\ &\Downarrow \\ |g_{ij}||g^{jk}| &= 1, \\ &\Downarrow \\ |g^{jk}| &= \frac{1}{g^2}. \end{aligned}$$

We transform the metric tensor to the cartesian reference frame,

$$g_{i'j'} = g_{ij}\beta_{i'}^i\beta_{j'}^j,$$

which leads to

$$(g')^2 = |g_{i'j'}| = |g_{ij}\beta_{i'}^i||\beta_{j'}^j| = |g_{ij}||\beta_{i'}^i||\beta_{j'}^j| = g^2\Delta\Delta. \tag{2.8}$$

For the cartesian coordinate system, we know that

$$g' = 1,$$

since the metric tensor is simply the identity matrix, and this leads to

$$\frac{1}{\Delta} = g.$$

Thus, we find that the tensor $|\beta_i^{i'}|$, which is used for expressing a base vector in a general coordinate system in terms of base vectors in the cartesian coordinate system, has the determinant g. We arrive at,

$$\begin{aligned} &\varepsilon_{ijk} = |\beta_i^{i'}| = g, && \text{if } i,j,k \text{ is cyclic,} \\ &\varepsilon_{ijk} = -|\beta_i^{i'}| = -g, && \text{if } i,j,k \text{ is anticyclic,} \\ &\varepsilon_{ijk} = 0, && \text{if } i,j,k \text{ is acyclic.} \end{aligned} \tag{2.9}$$

2.6 Representation of a three-dimensional surface

We will use surface intrinsic coordinates for describing three-dimensional surfaces. Such surfaces will in general be described by two surface parameters and written on the form $x^{i'} = f_i(r^1, r^2)$, $i = 1, 2, 3$. We are free to choose the way we describe the surface in terms of r^1 and r^2, but it is natural to choose r^1 and r^2 as variables on a pre-determined reference domain, which is connected to the physical surface through a one-to-one mapping. The covariant base-vectors are then given by

$$\mathbf{g}_i = \frac{\partial \mathbf{p}}{\partial r^i}.$$

Note that, independent of the choice of r^1 and r^2, the covariant base-vector, $\mathbf{g}_i$, will at each point on the surface point in the mapped direction of r^i while $\mathbf{g}^i$ will generally have a component in both the r^1 and r^2 direction.

In the following we will use this approach, where r^1 and r^2 are variables on a reference domain, $\hat{\Gamma}$ (see Figure 2.5.) Here, we see that a line on the reference domain in which r^1 varies and r^2 is constant corresponds to a curve on the physical surface, Γ, through the mapping $\mathcal{F}$. The vector $\mathbf{g}_1$ will be a tangent to this curve at all grid points, and similarly $\mathbf{g}_2$ will be the tangent vector along a curve on Γ corresponding to constant r^1 and varying r^2 on the reference domain. Thus, $\{\mathbf{g}_1, \mathbf{g}_2\}$ and $\{\mathbf{g}^1, \mathbf{g}^2\}$ will both span the tangent plane at all points on the surface. For the third direction, we will in the following choose $\mathbf{g}_3 = \mathbf{g}^3 = \mathbf{n}$, such that

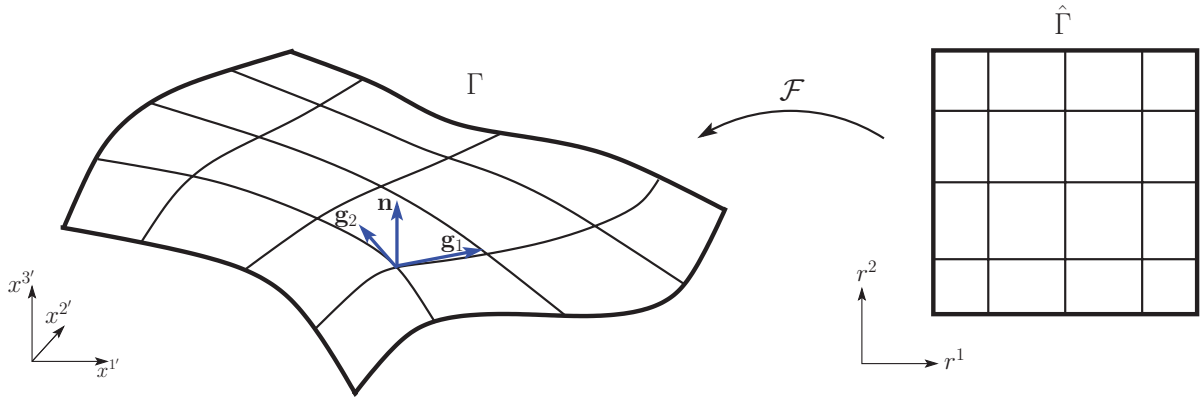


Figure 2.5: Mapping between reference domain and physical domain for a surface in three dimensions.

the third base vector is orthogonal to the tangent vectors as well as normalized,

$$\begin{aligned} \mathbf{g}_3 \cdot \mathbf{g}_i &= 0, \qquad i = 1, 2, \\ \sqrt{\mathbf{g}_3 \cdot \mathbf{g}_3} &= 1, \\ &\Downarrow \\ \mathbf{g}_3 &= \mathbf{n}. \end{aligned}$$

Note that for this choice of $\mathbf{g}_3$ we get,

$$g^2 = \begin{vmatrix} g_{11} & g_{12} & 0 \\ g_{21} & g_{22} & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{vmatrix}.$$

2.7 Cross product

2.7.1 Cross product between base vectors

If we choose $\mathbf{g}_{i'}$ as the orthonormal basis in the cartesian coordinate system,

$$\mathbf{g}_{i'} = \mathbf{i}_{x^{i'}}, \qquad i = 1, 2, 3,$$

we know that for this case

$$\mathbf{g}_{i'} \times \mathbf{g}_{j'} = \varepsilon_{i'j'k'}\mathbf{g}^{k'} \tag{2.10}$$

is valid. Here $\varepsilon_{i'j'k'}$ is the permutation tensor associated with the cartesian coordinate system, defined in (2.6) and $\mathbf{g}^{3'} = \mathbf{g}_{3'} = \mathbf{i}_{x^{3'}}$. For general coordinate directions, we may now write

$$\begin{aligned}\mathbf{g}_i \times \mathbf{g}_j &= (\beta_i^{i'}\mathbf{g}_{i'}) \times (\beta_j^{j'}\mathbf{g}_{j'}) \\ &= \beta_i^{i'}\beta_j^{j'}(\mathbf{g}_{i'} \times \mathbf{g}_{j'}),\end{aligned}$$

which by the use of (2.10) can be written as

$$\begin{aligned}&= \beta_i^{i'}\beta_j^{j'}\varepsilon_{i'j'k'}\mathbf{g}^{k'} \\ &= \beta_i^{i'}\beta_j^{j'}\beta_k^{k'}\varepsilon_{i'j'k'}\mathbf{g}^{k} \\ &\Downarrow \\ \mathbf{g}_i \times \mathbf{g}_j &= \varepsilon_{ijk}\mathbf{g}^{k}.\end{aligned} \tag{2.11}$$

2.7.2 Area-element on the surface

We now denote s^1 and s^2 to be surface coordinates pointing in the same directions as r^1 and r^2, respectively, such that $s^1 = s^1(r^1)$ and $s^2 = s^2(r^2)$, see Figure 2.6. We are interested in expressing an area-element, $\mathrm{d}S$, on a general three-dimensional surface in terms of the reference variables, r^1 and r^2.

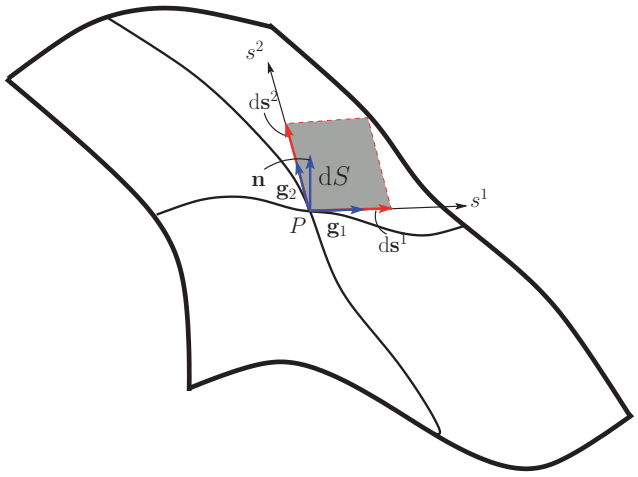


Figure 2.6: A surface element.

We assign $\mathrm{d}\mathbf{s}^1$ to be a vector originating from P associated with a small increment $\mathrm{d}r^1$ on the reference domain, and similarily $\mathrm{d}\mathbf{s}^2$ is the vector associated with the increment $\mathrm{d}r^2$. Thus, $\mathrm{d}\mathbf{s}^1$ points in the direction of $\mathbf{g}_1$ and $\mathrm{d}\mathbf{s}^2$ points in the direction of $\mathbf{g}_2$ and we have the relation

$$\begin{aligned} \mathrm{d}\mathbf{s}^1 \times \mathrm{d}\mathbf{s}^2 &= \mathrm{d}S\mathbf{n}, \\ &\Downarrow \\ \mathrm{d}S &= |\,\mathrm{d}\mathbf{s}^1 \times \mathrm{d}\mathbf{s}^2|. \end{aligned} \tag{2.12}$$

We also have

$$\begin{aligned} \mathrm{d}\mathbf{s}^1 &= \frac{\partial \mathbf{s}^1}{\partial r^1}\,\mathrm{d}r^1 = \frac{\partial \mathbf{p}}{\partial r^1}\,\mathrm{d}r^1 = \mathbf{g}_1\,\mathrm{d}r^1, \\ \mathrm{d}\mathbf{s}^2 &= \frac{\partial \mathbf{s}^2}{\partial r^2}\,\mathrm{d}r^2 = \frac{\partial \mathbf{p}}{\partial r^2}\,\mathrm{d}r^2 = \mathbf{g}_2\,\mathrm{d}r^2. \end{aligned}$$

Inserted into (2.12), we get

$$\begin{aligned} \mathrm{d}S &= |(\mathbf{g}_1\,\mathrm{d}r^1 \times \mathbf{g}_2\,\mathrm{d}r^2)|, \\ &= |\mathbf{g}_1 \times \mathbf{g}_2|\,\mathrm{d}r^1\,\mathrm{d}r^2. \end{aligned}$$

From (2.11) and our choice of $\mathbf{g}^3 = \mathbf{n}$, we get

$$\begin{aligned} |\mathbf{g}_1 \times \mathbf{g}_2| &= |\varepsilon_{123}\mathbf{n}|, \\ &= |\varepsilon_{123}||\mathbf{n}|, \\ &= \varepsilon_{123}, \\ &\Downarrow \\ \mathrm{d}S &= g\,\mathrm{d}r^1\,\mathrm{d}r^2, \end{aligned} \tag{2.13}$$

by the use of (2.9) and the fact that g is always positive.

2.7.3 Cross-product between two general vectors

The cross product between two general vectors expressed in a covariant basis

$$\begin{aligned} \mathbf{a} &= a^i\mathbf{g}_i, \\ \mathbf{b} &= b^j\mathbf{g}_j, \end{aligned}$$

becomes

$$\begin{aligned} \mathbf{a} \times \mathbf{b} &= a^i\mathbf{g}_i \times b^j\mathbf{g}_j, \\ &= a^i b^j \mathbf{g}_i \times \mathbf{g}_j, \\ &= a^i b^j \varepsilon_{ijk}\mathbf{g}^k. \end{aligned} \tag{2.14}$$

2.7.4 Normal vector of a line element

We consider a line element vector, $\mathrm{d}\mathbf{s}$, associated with a curve, C, on an arbitrary curved surface, S. In Figure 2.7 this curve is the boundary of the surface, but in general, C could be anywhere on S. We will consider a point P on C where $\mathrm{d}\mathbf{s}$ is a tangent vector, $\mathbf{g}^3 = \mathbf{n}$ is the normal to S at P and $\mathrm{d}\mathbf{n}$ is an outer normal to C. We express $\mathrm{d}\mathbf{s}$ in terms of covariant base-vectors and $\mathrm{d}\mathbf{n}$ in terms of contravariant base-vectors,

$$\begin{aligned} \mathrm{d}\mathbf{s} &= \mathrm{d}r^\alpha \mathbf{g}_\alpha, \\ \mathrm{d}\mathbf{n} &= \mathrm{d}n_\beta \mathbf{g}^\beta. \end{aligned} \tag{2.15}$$

Then, the outer normal to C at P may be written

$$\begin{aligned} \mathrm{d}\mathbf{n} &= \mathrm{d}\mathbf{s} \times \mathbf{g}^3 \\ &= \mathrm{d}r^\alpha (\mathbf{g}_\alpha \times \mathbf{g}_3) \\ &= \mathrm{d}r^\alpha \varepsilon_{\alpha 3 \beta} \mathbf{g}^\beta. \end{aligned} \tag{2.16}$$

Comparing (2.15) and (2.16) leads to

$$\begin{aligned} \mathrm{d}n_\beta &= \mathrm{d}r^\alpha \varepsilon_{\alpha 3 \beta}, \\ \mathrm{d}n_\alpha &= \varepsilon_{\beta 3 \alpha}\, \mathrm{d}r^\beta, \\ &\Downarrow \\ &= \tilde{\varepsilon}_{\alpha\beta}\, \mathrm{d}r^\beta, \end{aligned} \tag{2.17}$$

where $\tilde{\varepsilon}_{12} = g$, $\tilde{\varepsilon}_{21} = -g$ and $\tilde{\varepsilon}_{11} = \tilde{\varepsilon}_{22} = 0$. Since $\mathbf{g}^3$ is of unit length, we observe that $|\,\mathrm{d}\mathbf{n}| = |\,\mathrm{d}\mathbf{s}|$.

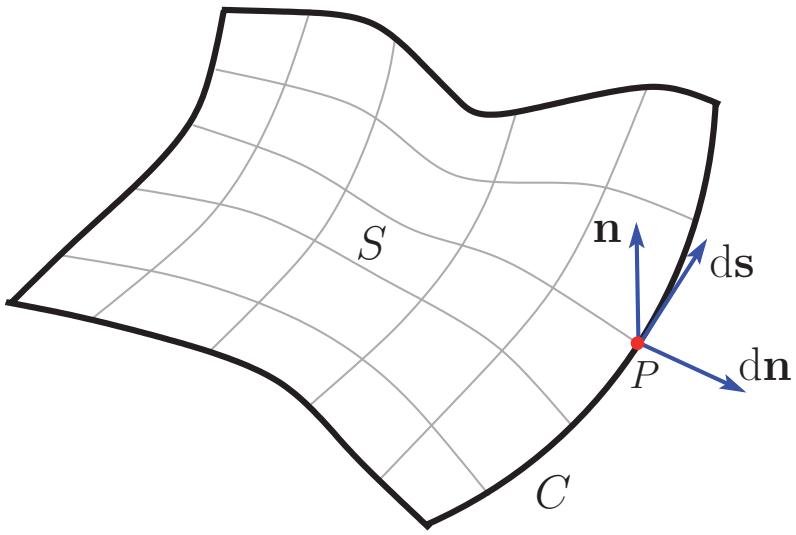


Figure 2.7: Outward normal to the point P on the curve C.

2.8 Summary

In summary, we have some results and definitions which will be used in the following sections:

$$\mathbf{g}_\alpha = \frac{\partial \mathbf{p}}{\partial r^\alpha}, \tag{2.18}$$

$$\mathbf{g}_\alpha \cdot \mathbf{g}_\beta = g_{\alpha\beta}, \tag{2.19}$$

$$g_{\alpha\gamma} g^{\gamma\beta} = \delta_\alpha^\beta, \tag{2.20}$$

$$\mathbf{g}^\alpha = g^{\alpha\beta} \mathbf{g}_\beta, \tag{2.21}$$

$$g = \sqrt{[\det(g_{\alpha\beta})]}, \tag{2.22}$$

$$b_{\alpha\beta} = \mathbf{n} \cdot \frac{\partial \mathbf{g}_\alpha}{\partial r^\beta} = -\mathbf{g}_\alpha \cdot \frac{\partial \mathbf{n}}{\partial r^\beta} \tag{2.23}$$

The relation between the contravariant and covariant metric tensor is given by

$$\begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} g^{11} & g^{12} \\ g^{21} & g^{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} g^{11} & g^{12} \\ g^{21} & g^{22} \end{bmatrix} = \frac{1}{g^2} \begin{bmatrix} g_{22} & -g_{12} \\ -g_{21} & g_{11} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \frac{1}{g^2} \begin{bmatrix} g_{22} & -g_{12} \\ -g_{21} & g_{11} \end{bmatrix}$$

$$\Downarrow$$

$$g^{11} = \frac{1}{g^2} g_{22} \tag{2.24}$$

$$g^{12} = -\frac{1}{g^2} g_{12} \tag{2.25}$$

$$g^{21} = g^{12} \tag{2.26}$$

$$g^{22} = \frac{1}{g^2} g_{11} \tag{2.27}$$

3 Mean curvature and related operators

The main objective in this section is to find an expression for the curvature-normal product. In order to do this we will derive expressions for some important quantities like gradient, divergence and principal directions related to the mean curvature in the context of a general coordinate system. The contents of this section is mainly influenced by [4] and [3], while the notation is mostly the same as used in [1] and [3].

3.1 Surface divergence of a vector in general coordinates

If we have a vector, $\mathbf{F} = \mathbf{F}(r^1, r^2)$, expressed in general coordinates, the divergence at a given point is given by [4]

$$\nabla_s \cdot \mathbf{F} = \frac{1}{g^2}\mathbf{g}_1 \cdot \left(g_{22}\frac{\partial \mathbf{F}}{\partial r^1} - g_{12}\frac{\partial \mathbf{F}}{\partial r^2} \right) + \frac{1}{g^2}\mathbf{g}_2 \cdot \left(g_{11}\frac{\partial \mathbf{F}}{\partial r^2} - g_{12}\frac{\partial \mathbf{F}}{\partial r^1} \right). \tag{3.1}$$

We now wish to find another expression for $\nabla_s \cdot \mathbf{F}$. Expressing $\mathbf{F}$ in covariant base-vectors, we may write $\mathbf{F} = a\mathbf{g}_1 + b\mathbf{g}_2 + c\mathbf{n}$. Thus, the divergence may be written as

$$\nabla_s \cdot \mathbf{F} = \nabla_s \cdot (a\mathbf{g}_1) + \nabla_s \cdot (b\mathbf{g}_2) + \nabla_s \cdot (c\mathbf{n}).$$

By the help of (3.1), we get

$$\begin{aligned}
\nabla_s \cdot (a\mathbf{g}_1) &= \frac{1}{g^2}\mathbf{g}_1 \cdot \left[g_{22}\left(\frac{\partial a}{\partial r^1}\mathbf{g}_1 + a\frac{\partial \mathbf{g}_1}{\partial r^1} \right) - g_{12}\left(\frac{\partial a}{\partial r^2}\mathbf{g}_1 + a\frac{\partial \mathbf{g}_1}{\partial r^2} \right) \right] \\
&+ \frac{1}{g^2}\mathbf{g}_2 \cdot \left[g_{11}\left(\frac{\partial a}{\partial r^2}\mathbf{g}_1 + a\frac{\partial \mathbf{g}_1}{\partial r^2} \right) - g_{12}\left(\frac{\partial a}{\partial r^1}\mathbf{g}_1 + a\frac{\partial \mathbf{g}_1}{\partial r^1} \right) \right] \\
&= \frac{1}{g^2}\left[g_{22}\frac{\partial a}{\partial r^1}\left(\mathbf{g}_1 \cdot \mathbf{g}_1 \right) + g_{22}a\left(\mathbf{g}_1 \cdot \frac{\partial \mathbf{g}_1}{\partial r^1} \right) - g_{12}\frac{\partial a}{\partial r^2}\left(\mathbf{g}_1 \cdot \mathbf{g}_1 \right) - g_{12}a\left(\mathbf{g}_1 \cdot \frac{\partial \mathbf{g}_1}{\partial r^2} \right) \right] \\
&+ \frac{1}{g^2}\left[g_{11}\frac{\partial a}{\partial r^2}\left(\mathbf{g}_2 \cdot \mathbf{g}_1 \right) + g_{11}a\left(\mathbf{g}_2 \cdot \frac{\partial \mathbf{g}_1}{\partial r^2} \right) - g_{12}\frac{\partial a}{\partial r^1}\left(\mathbf{g}_1 \cdot \mathbf{g}_2 \right) - g_{12}a\left(\mathbf{g}_2 \cdot \frac{\partial \mathbf{g}_1}{\partial r^1} \right) \right]
\end{aligned}$$

To proceed, we need the following quantities (note that $\frac{\partial \mathbf{g}_1}{\partial r^2} = \frac{\partial \mathbf{g}_2}{\partial r^1}$),

$$\begin{aligned}
\mathbf{g}_1 \cdot \frac{\partial \mathbf{g}_1}{\partial r^1} &= \frac{1}{2}\frac{\partial (\mathbf{g}_1 \cdot \mathbf{g}_1)}{\partial r^1} &&= \frac{1}{2}\frac{\partial g_{11}}{\partial r^1}, \\
\mathbf{g}_1 \cdot \frac{\partial \mathbf{g}_1}{\partial r^2} &= \frac{1}{2}\frac{\partial (\mathbf{g}_1 \cdot \mathbf{g}_1)}{\partial r^2} &&= \frac{1}{2}\frac{\partial g_{11}}{\partial r^2}, \\
\mathbf{g}_2 \cdot \frac{\partial \mathbf{g}_1}{\partial r^2} &= \mathbf{g}_2 \cdot \frac{\partial \mathbf{g}_2}{\partial r^1} = \frac{1}{2}\frac{\partial (\mathbf{g}_2 \cdot \mathbf{g}_2)}{\partial r^1} &&= \frac{1}{2}\frac{\partial g_{22}}{\partial r^1}, \\
\mathbf{g}_2 \cdot \frac{\partial \mathbf{g}_1}{\partial r^1} &= \frac{\partial (\mathbf{g}_1 \cdot \mathbf{g}_2)}{\partial r^1} - \mathbf{g}_1 \cdot \frac{\partial \mathbf{g}_2}{\partial r^1} &&= \frac{\partial g_{12}}{\partial r^1} - \frac{1}{2}\frac{\partial g_{11}}{\partial r^2}.
\end{aligned}$$

Thus,

$$
\begin{aligned}
\nabla_s \cdot (a\mathbf{g}_1) &= \frac{1}{g^2}\left[g_{22}\frac{\partial a}{\partial r^1}g_{11} + g_{22}a\left(\frac{1}{2}\frac{\partial g_{11}}{\partial r^1}\right) - g_{12}\frac{\partial a}{\partial r^2}g_{11} - g_{12}a\left(\frac{1}{2}\frac{\partial g_{11}}{\partial r^2}\right)\right] \\
&+ \frac{1}{g^2}\left[g_{11}\frac{\partial a}{\partial r^2}g_{12} + g_{11}a\left(\frac{1}{2}\frac{\partial g_{22}}{\partial r^1}\right) - g_{12}\frac{\partial a}{\partial r^1}g_{12} - g_{12}a\left(\frac{\partial g_{12}}{\partial r^1} - \frac{1}{2}\frac{\partial g_{11}}{\partial r^2}\right)\right] \\
&= \frac{1}{g^2}\overbrace{(g_{11}g_{22} - g_{12}^2)}^{g^2}\frac{\partial a}{\partial r^1} + \frac{a}{2g^2}\left(g_{11}\frac{\partial g_{22}}{\partial r^1} + g_{22}\frac{\partial g_{11}}{\partial r^1} - 2g_{12}\frac{\partial g_{12}}{\partial r^1}\right), \\
&= \frac{\partial a}{\partial r^1} + \frac{a}{2g^2}\left(g_{11}\frac{\partial g_{22}}{\partial r^1} + g_{22}\frac{\partial g_{11}}{\partial r^1} - 2g_{12}\frac{\partial g_{12}}{\partial r^1}\right), \\
&= \frac{\partial a}{\partial r^1} + \frac{a}{2g^2}\frac{\partial}{\partial r^1}(g_{11}g_{22} - g_{12}^2), \\
&= \frac{1}{g}\frac{\partial (ga)}{\partial r^1},
\end{aligned}
$$

where $g = \sqrt{g_{11}g_{22} - g_{12}^2}$. Similarly, we have

$$
\nabla_s \cdot (b\mathbf{g}_2) = \frac{1}{g}\frac{\partial (gb)}{\partial r^2},
$$

and we will later show that

$$
\nabla_s \cdot (c\mathbf{n}) = -\kappa c,
$$

where κ is twice the mean curvature. This gives us another formula for the divergence of a vector, $\mathbf{F} = a\mathbf{g}_1 + b\mathbf{g}_2 + c\mathbf{n}$,

$$
\nabla_s \cdot \mathbf{F} = \frac{1}{g}\left[\frac{\partial (ga)}{\partial r^1} + \frac{\partial (gb)}{\partial r^2}\right] - \kappa c. \tag{3.2}
$$

3.1.1 Example

As a simple example we will consider the surface of a sphere with radius $r = 1$.

We will choose basis vectors

$$
\begin{aligned}
\mathbf{g}_1 &= -\sin\theta \mathbf{i}_{x^{1'}} + \cos\theta \mathbf{i}_{x^{2'}} + 0\mathbf{i}_{x^{3'}}, \\
\mathbf{g}_2 &= \cos\phi\cos\theta \mathbf{i}_{x^{1'}} + \cos\phi\sin\theta \mathbf{i}_{x^{2'}} - \sin\phi \mathbf{i}_{x^{3'}},
\end{aligned}
$$

where $\{\mathbf{i}_{x^{i'}}\}_{i=1}^3$ as usual are the standard basis vectors in the cartesian coordinate system. We observe that

$$
\begin{aligned}
g_{12} &= \mathbf{g}_1 \cdot \mathbf{g}_2 = 0, \\
g_{11} &= \mathbf{g}_1 \cdot \mathbf{g}_1 = 1, \\
g_{22} &= \mathbf{g}_2 \cdot \mathbf{g}_2 = 1, \\
&\Downarrow \\
g &= 1.
\end{aligned} \tag{3.3}
$$

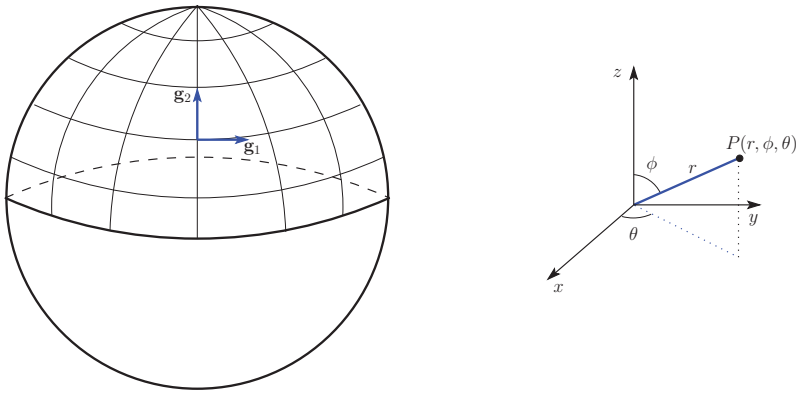


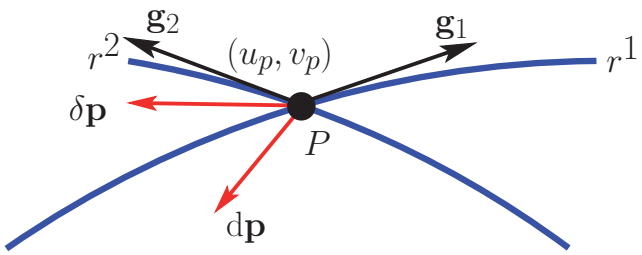
Figure 3.1: Sphere with orthonormal base-vectors.

Inserted into (3.2), we find that the surface divergence to a vector $\mathbf{F} = a\mathbf{g}_1 + b\mathbf{g}_2$ for this case is given by

$$\nabla_s \cdot \mathbf{F} = \frac{\partial a}{\partial r^1} + \frac{\partial b}{\partial r^2},$$

which is what we would expect.

3.2 Surface gradient of a scalar field



We now consider a scalar function, $\phi(r^1, r^2)$, on a surface, S, parameterized by the reference variables r^1 and r^2. We assume that $\phi(r^1, r^2) = C$ is a *level curve* on the surface and that P is a point on this curve. If $(\delta r^1, \delta r^2)$ is a small displacement from P such that

$$\delta\mathbf{p} = \mathbf{g}_1 \delta r^1 + \mathbf{g}_2 \delta r^2 \tag{3.4}$$

is a tangent to this curve, then

$$\phi_{,1}\delta r^1 + \phi_{,2}\delta r^2 = 0, \tag{3.5}$$

where $\phi_{,1} = \frac{\partial \phi}{\partial r^1}$ and $\phi_{,2} = \frac{\partial \phi}{\partial r^2}$.

We consider another displacement $(\mathrm{d}r^1, \mathrm{d}r^2)$, with associated displacement vector,

$$\mathrm{d}\mathbf{p} = \mathbf{g}_1 \,\mathrm{d}r^1 + \mathbf{g}_2 \,\mathrm{d}r^2, \tag{3.6}$$

and find that the inner-product between $\delta\mathbf{p}$ and $\mathrm{d}\mathbf{p}$ is given by

$$\begin{aligned}\mathrm{d}\mathbf{p}\cdot\delta\mathbf{p} &= \Big(\mathbf{g}_1\,\mathrm{d}r^1+\mathbf{g}_2\,\mathrm{d}r^2\Big)\cdot\Big(\mathbf{g}_1\delta r^1+\mathbf{g}_2\delta r^2\Big)\\ &= g_{11}\,\mathrm{d}r^1\delta r^1+g_{12}(\,\mathrm{d}r^1\delta r^2+\,\mathrm{d}r^2\delta r^1)+g_{22}\,\mathrm{d}r^2\delta r^2.\end{aligned}$$

We now assume that $\mathrm{d}\mathbf{p}$ and $\delta\mathbf{p}$ are perpendicular, which leads to the relation

$$g_{11}\frac{\mathrm{d}r^1}{\mathrm{d}r^2}\frac{\delta r^1}{\delta r^2}+g_{12}\left(\frac{\mathrm{d}r^1}{\mathrm{d}r^2}+\frac{\delta r^1}{\delta r^2}\right)+g_{22}=0. \tag{3.7}$$

From (3.5) we see that

$$\frac{\delta r^1}{\delta r^2}=-\frac{\phi_{,2}}{\phi_{,1}}.$$

We insert this into (3.7) and find that

$$\frac{\mathrm{d}r^1}{\mathrm{d}r^2}=\frac{g_{22}\phi_{,1}-g_{12}\phi_{,2}}{g_{11}\phi_{,2}-g_{12}\phi_{,1}}. \tag{3.8}$$

The displacement vector, $\delta\mathbf{p}$, is a tangent to the curve $\phi(r^1,r^2)=C$, and we know that the surface gradient points in a normal direction to this curve along the surface. Thus, from (3.8) we see that the vector

$$\mathbf{V}=k(g_{22}\phi_{,1}-g_{12}\phi_{,2})\mathbf{g}_1+k(g_{11}\phi_{,2}-g_{12}\phi_{,1})\mathbf{g}_2$$

is parallel to $\nabla_s\phi$. In order to determine k, we require that

$$\mathbf{V}\cdot\frac{\mathbf{g}_1}{\sqrt{g_{11}}}=\frac{\partial\phi}{\partial r^1}\frac{\partial r^1}{\partial s^1}=\frac{1}{\sqrt{g_{11}}}\phi_{,1},$$

where s_1 is an arc-length coordinate which runs in the same "direction" as the reference variable, r^1. We find that $k=\frac{1}{g^2}$, so the gradient of the scalar function $\phi(r^1,r^2)$ may be written

$$\nabla_s\phi=\frac{(g_{22}\phi_{,1}-g_{12}\phi_{,2})}{g^2}\mathbf{g}_1+\frac{(g_{11}\phi_{,2}-g_{12}\phi_{,1})}{g^2}\mathbf{g}_2, \tag{3.9}$$

$$\begin{aligned}&=(g^{11}\phi_{,1}+g^{12}\phi_{,2})\mathbf{g}_1+(g^{12}\phi_{,1}+g^{22}\phi_{,2})\mathbf{g}_2,\\ &=\phi_{,\alpha}\mathbf{g}^\alpha,\end{aligned} \tag{3.10}$$

where we have used (2.21) in the last step, and $\alpha=1,2$.

3.2.1 Example

By using the same example as for the surface divergence, (see Figure 3.1), where the geometric factors are given by (3.3), we get the simplified formula

$$\nabla_s\phi=\frac{\partial\phi}{\partial r^1}\mathbf{g}_1+\frac{\partial\phi}{\partial r^2}\mathbf{g}_2,$$

which again seems reasonable.

3.3 Curvature of a curve

We will consider a curve, C, spanning three dimensions given by

$$\mathbf{p}(s) = \begin{bmatrix} x^{1'}(s) \\ x^{2'}(s) \\ x^{3'}(s) \end{bmatrix},$$

where $x^{i'}$ is the i'th cartesian coordinate and s is an arc-length variable along the curve. Such a curve is depicted in Figure 3.2. Here, we have used

- **Normal plane**, N

  The plane spanned by all vectors normal to the unit tangent vector $\mathbf{t}(s) = \dot{\mathbf{p}} = \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}s}$ at the point P.

- **Osculating plane**, O

  The plane spanned by $\mathbf{t} = \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}s}$ and $\dot{\mathbf{t}} = \frac{\mathrm{d}^2\mathbf{p}}{\mathrm{d}s^2}$.

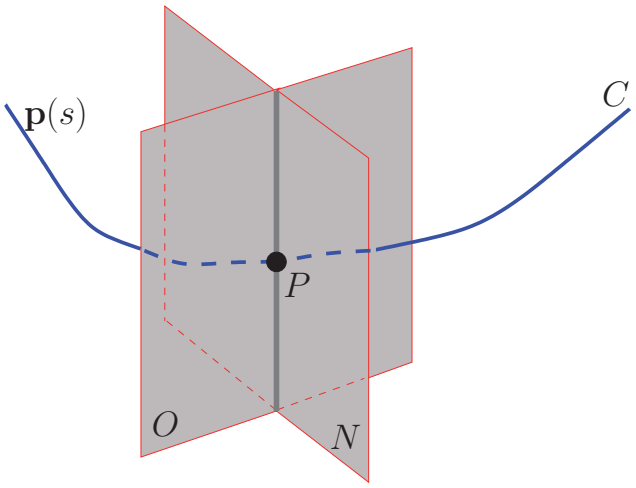


Figure 3.2: A curve in three dimensions. N is the normal plane and O is the osculating plane.

We observe that the vector

$$\mathbf{n}_c = \frac{\dot{\mathbf{t}}(s)}{|\dot{\mathbf{t}}(s)|}$$

is a unit normal to C. This vector lies in both the normal plane, N, and in the osculating plane, O, and therefore points in the direction of the line of intersection between N and O. The curvature of the curve C at the point $P(s)$ is given by

$$\begin{aligned} \kappa_c &= |\dot{\mathbf{t}}(s)|, \\ &\Downarrow \\ \kappa_c \mathbf{n}_c &= \ddot{\mathbf{p}}(s). \end{aligned} \tag{3.11}$$

Figure 3.3 shows a plot of the curve C projected to the osculating plane at the point P. The point M at a distance $\rho = \frac{1}{\kappa_c}$ from P in the direction of $\mathbf{n}_c$ is called the *centre of curvature*.

The circle in the osculating plane with centre M and radius ρ is called the *circle of curvature* of C at P.

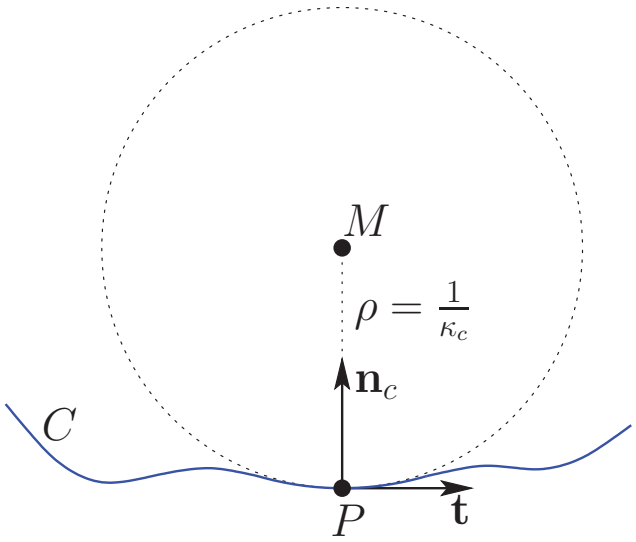


Figure 3.3: The curve, C, projected to the osculating plane at P.

We observe that the circle of curvature is only dependent on $\dot{\mathbf{t}}$ at the point P, such that any curve through P with the same local behavior will have the same circle of curvature.

3.4 Orthogonal curves on a surface

In Section 3.2 we found that (3.7) must be satisfied for the two directions $\frac{\mathrm{d}r^1}{\mathrm{d}r^2}$ and $\frac{\delta r^1}{\delta r^2}$ to be orthogonal. We will later encounter equations on the form

$$a_1 \left(\frac{\mathrm{d}r^1}{\mathrm{d}r^2}\right)^2 + a_2 \left(\frac{\mathrm{d}r^1}{\mathrm{d}r^2}\right) + a_3 = 0, \tag{3.12}$$

where the solutions represent two directions on a surface associated with a point, P. If we assume that $\frac{\mathrm{d}r^1}{\mathrm{d}r^2}$ and $\frac{\delta r^1}{\delta r^2}$ are the two solutions of (3.12), we find that

$$\frac{\mathrm{d}r^1}{\mathrm{d}r^2} + \frac{\delta r^1}{\delta r^2} = -\frac{a_2}{a_1},$$
$$\frac{\mathrm{d}r^1}{\mathrm{d}r^2}\frac{\delta r^1}{\delta r^2} = \frac{a_3}{a_1}.$$

Combining this with (3.7), we get the required relation for orthogonality,

$$g_{11}a_3 - g_{12}a_2 + g_{22}a_1 = 0. \tag{3.13}$$

3.4.1 Example

If we again consider a situation with orthonormal base vectors, for instance the case in Figure 3.1, we get the required relation,

$$a_3 + a_1 = 0.$$

Choosing $a_1 = -a_3$ in (3.12), we get the two solutions,

$$\frac{\mathrm{d}r^1}{\mathrm{d}r^2} = -\frac{a_2}{2a_3} + \frac{\sqrt{a_2^2 + 4a_3^2}}{2a_3},$$
$$\frac{\delta r^1}{\delta r^2} = -\frac{a_2}{2a_3} - \frac{\sqrt{a_2^2 + 4a_3^2}}{2a_3}.$$

Defining the two vectors,

$$\mathbf{V}_1 = \left(-\frac{a_2}{2a_3} + \frac{\sqrt{a_2^2 + 4a_3^2}}{2a_3} \right) \mathbf{g}_1 + \mathbf{g}_2,$$
$$\mathbf{V}_2 = \left(-\frac{a_2}{2a_3} - \frac{\sqrt{a_2^2 + 4a_3^2}}{2a_3} \right) \mathbf{g}_1 + \mathbf{g}_2,$$

we find that

$$\mathbf{V}_1 \cdot \mathbf{V}_2 = 0,$$

which shows that the two directions are orthogonal.

3.5 Principal directions and mean curvature

We now wish to find an expression for the mean curvature at a point on a surface. In Section 3.3 we showed that the curvature at the point P of a curve C with the parametric representation $\mathbf{p}(s)$ is given by

$$\kappa_c \mathbf{n}_c = \ddot{\mathbf{p}}(s),$$

where $\mathbf{n}_c$ is the unit normal to C which points in the direction of the intersection of the osculating plane and the normal plane, and s is an arc-length variable. If we now set $\mathbf{n}_c = \mathbf{n}$, where $\mathbf{n}$ is the unit normal to a surface S at P, we get the formula

$$\kappa_c \mathbf{n} = \ddot{\mathbf{p}}(s). \tag{3.14}$$

This formula will give us the curvature for all curves $\mathbf{p}(s)$ on S for which the intersection between the osculating plane and the normal plane points in the same direction as $\mathbf{n}$. This leads us to a type of curves on S called *normal sections*:

- **Normal section**

  A normal section is a *plane curve* associated with a general curve on S which passes through P. A normal section is defined by the intersection of S and a plane containing the normal $\mathbf{n}$ of S at P and a tangent vector, $\mathbf{t}$, to the curve. The normal section at P will then automatically have $\mathbf{t}$ as a tangent vector and $\mathbf{n}$ as a principal normal. For such a curve (3.14) will be the formula for the curvature. We will denote the curvature of a normal section by κ_n.

An example of a normal section is displayed in Figure 3.4.

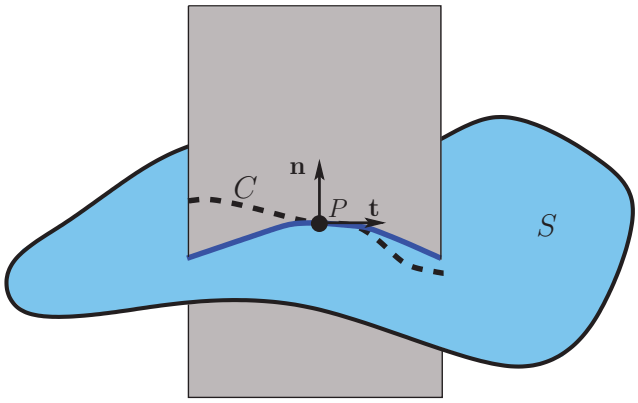


Figure 3.4: A normal section associated with a curve, C, passing through the point, P..

3.5.1 Curvature of a normal section

We now wish to derive another expression for the curvature of a normal section which involves tensors. From (3.14) we have

$$\begin{aligned}
\kappa_n \mathbf{n} = \ddot{\mathbf{p}}(s) &= \frac{\mathrm{d}^2\mathbf{p}}{\mathrm{d}s^2} \\
&= \frac{\partial^2 \mathbf{p}}{\partial r^\alpha \partial r^\beta}\dot{r}^\alpha \dot{r}^\beta + \frac{\partial \mathbf{p}}{\partial r^\alpha}\ddot{r}^\alpha \\
&\Downarrow \\
\kappa_n &= \left(\frac{\partial^2 \mathbf{p}}{\partial r^\alpha \partial r^\beta}\cdot \mathbf{n}\right)\dot{r}^\alpha \dot{r}^\beta \\
&= \left(\frac{\partial \mathbf{g}_\alpha}{\partial r^\beta}\cdot \mathbf{n}\right)\dot{r}^\alpha \dot{r}^\beta .
\end{aligned} \tag{3.15}$$

By the use of (2.23), we find that

$$\kappa_n = b_{\alpha\beta}\dot{r}^\alpha \dot{r}^\beta .$$

If we now assume that we parameterize the curve C by a parameter t instead of the arc-length, s, we find,

$$\dot{r}^\alpha = \frac{\mathrm{d}r^\alpha}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}s} = \frac{r^{\alpha'}}{s'}.$$

Thus, we may write

$$\kappa_n = \frac{b_{\alpha\beta}r^{\alpha'}r^{\beta'}}{(s')^2} \tag{3.16}$$

We know that

$$\mathrm{d}s^2 = \mathrm{d}\mathbf{p}\cdot\mathrm{d}\mathbf{p} = g_{\alpha\beta}\,\mathrm{d}r^\alpha\,\mathrm{d}r^\beta$$

and thus

$$(s')^2 = g_{\alpha\beta} r^{\alpha'} r^{\beta'}.$$

(3.16) now becomes

$$\begin{aligned} \kappa_n &= \frac{b_{\alpha\beta} r^{\alpha'} r^{\beta'}}{g_{\alpha\beta} r^{\alpha'} r^{\beta'}} \\ &= \frac{b_{\alpha\beta}\, \mathrm{d}r^{\alpha}\, \mathrm{d}r^{\beta}}{g_{\alpha\beta}\, \mathrm{d}r^{\alpha}\, \mathrm{d}r^{\beta}}. \end{aligned} \tag{3.17}$$

Thus, (3.17) gives us an expression for the curvature of a normal section whose tangent direction is given by $(\mathrm{d}r^1, \mathrm{d}r^2)$.

3.5.2 Principal directions

Twice the mean curvature is given by

$$\kappa = \kappa_{n_{\min}} + \kappa_{n_{\max}},$$

where $\kappa_{n_{\min}}$ and $\kappa_{n_{\max}}$ are the extremas for κ_n when we consider all possible curves on S passing through a point, P. We now wish to find which directions on S for which κ_n has its extremas. In (3.17) we have a formula to find the curvature at a point P of the *normal section*. In Figure 3.5, we see that for a general vector, $\mathbf{v} = \mathrm{d}r^1\mathbf{g}_1 + \mathrm{d}r^2\mathbf{g}_2$, the ratio $\frac{\mathrm{d}r^1}{\mathrm{d}r^2}$ defines a direction on the surface. Thus, we wish to find the directions, $\frac{\mathrm{d}r^1}{\mathrm{d}r^2}$, such that $\frac{\partial \kappa_n}{\partial s} = 0$, where s is a variable in the angular direction, see Figure 3.6. We see that a vector in the angular direction may be expressed

$$\delta\mathbf{s} = \delta r^1 \mathbf{g}_1 + \delta r^2 \mathbf{g}_2.$$

Thus, if we require that

$$\begin{aligned} \frac{\partial \kappa_n}{\partial l^1} &= 0, \\ \frac{\partial \kappa_n}{\partial l^2} &= 0, \end{aligned} \tag{3.18}$$

where $l^{\alpha} = \mathrm{d}r^{\alpha}$, we will also have $\frac{\partial \kappa_n}{\partial s} = 0$.

(3.17) may be written as

$$(b_{\alpha\beta} - \kappa_n g_{\alpha\beta}) l^{\alpha} l^{\beta} = 0. \tag{3.19}$$

If we set

$$a_{\alpha\beta} = b_{\alpha\beta} - \kappa_n g_{\alpha\beta},$$

differentiating (3.19) yields

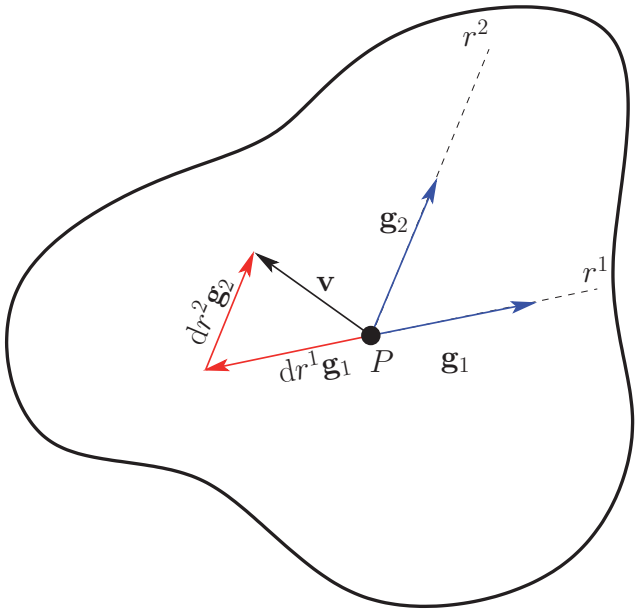


Figure 3.5: A vector, $\mathbf{v}$, in a general direction on a surface.

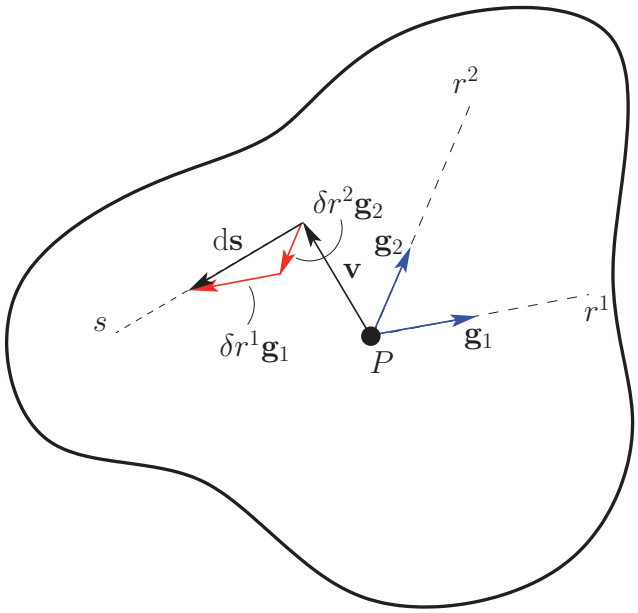


Figure 3.6: A vector, $d\mathbf{s}$, in the angular direction on a surface.

$$\frac{\partial}{\partial l^\gamma}(a_{\alpha\beta}l^\alpha l^\beta) = a_{\alpha\beta}\left(\frac{\partial l^\alpha}{\partial l^\gamma}l^\beta + l^\alpha\frac{\partial l^\beta}{\partial l^\gamma}\right)$$
$$= a_{\alpha\beta}(\delta^\alpha_\gamma l^\beta + l^\alpha\delta^\beta_\gamma) = a_{\gamma\beta}l^\beta + a_{\alpha\gamma}l^\alpha = (a_{\gamma\alpha} + a_{\alpha\gamma})l^\alpha, \qquad \gamma = 1, 2.$$

(Note here that $\frac{\partial}{\partial l^\gamma}$ denotes differentiation with respect to the direction $\mathrm{d}r^\gamma$ on the surface. Thus, the point P is constant, and therefore $\frac{\partial b_{\alpha\beta}}{\partial l^\gamma} = 0$ and $\frac{\partial g_{\alpha\beta}}{\partial l^\gamma} = 0$.) $a_{\alpha\beta}$ is symmetric, and we get the two equations

$$\begin{aligned}(b_{\alpha1} - \kappa_n g_{\alpha1})\,\mathrm{d}r^\alpha = 0,\\ (b_{\alpha2} - \kappa_n g_{\alpha2})\,\mathrm{d}r^\alpha = 0.\end{aligned} \tag{3.20}$$

If we eliminate κ_n from (3.20), we end up with the second order equation

$$(g_{11}b_{12} - g_{12}b_{11})\left(\frac{\mathrm{d}r^1}{\mathrm{d}r^2}\right)^2 + (g_{11}b_{22} - g_{22}b_{11})\left(\frac{\mathrm{d}r^1}{\mathrm{d}r^2}\right) + (g_{12}b_{22} - g_{22}b_{12}) = 0. \tag{3.21}$$

We see that (3.21) satisfies (3.13) with

$$\begin{aligned}a_1 &= (g_{11}b_{12} - g_{12}b_{11}),\\ a_2 &= (g_{11}b_{22} - g_{22}b_{11}),\\ a_3 &= (g_{12}b_{22} - g_{22}b_{12}),\end{aligned}$$

and thus we have shown that the two principal curves are orthogonal.

3.5.3 Mean curvature

From (3.20) we find that

$$\begin{aligned}\frac{\mathrm{d}r^1}{\mathrm{d}r^2} &= \frac{(\kappa_n g_{21} - b_{21})}{(b_{11} - \kappa_n g_{11})},\\ \frac{\mathrm{d}r^1}{\mathrm{d}r^2} &= \frac{(\kappa_n g_{22} - b_{22})}{(b_{12} - \kappa_n g_{12})}.\end{aligned}$$

Eliminating $\frac{\mathrm{d}r^1}{\mathrm{d}r^2}$ leads to the second order equation

$$(g_{11}g_{22} - g_{12}^2)\kappa_n^2 + (2g_{12}b_{12} - b_{11}g_{22} - g_{11}b_{22})\kappa_n + (b_{11}b_{22} - b_{12}^2) = 0, \tag{3.22}$$

from which we obtain $\kappa_{n_{max}}$ and $\kappa_{n_{min}}$. Twice the mean curvature is given by

$$\begin{aligned}\kappa &= \kappa_{n_{max}} + \kappa_{n_{min}}\\ &= \frac{2(g_{22}b_{11} - 2g_{12}b_{12} + g_{11}b_{22})}{2g^2}\\ &= b_{11}g^{11} + 2b_{12}g^{12} + b_{22}g^{22}.\end{aligned} \tag{3.23}$$

3.5.4 Surface divergence of the unit normal

From (3.1) and (2.23), we find that the divergence of the unit normal may be expressed as

$$\begin{aligned}
\nabla_s \cdot \mathbf{n} &= \frac{1}{g^2}\mathbf{g}_1 \cdot \left(g_{22}\frac{\partial \mathbf{n}}{\partial r^1} - g_{12}\frac{\partial \mathbf{n}}{\partial r^2} \right) + \frac{1}{g^2}\mathbf{g}_2 \cdot \left(g_{11}\frac{\partial \mathbf{n}}{\partial r^2} - g_{12}\frac{\partial \mathbf{n}}{\partial r^1} \right) \\
&= \frac{1}{g^2}\left(g_{22}\mathbf{g}_1 \cdot \frac{\partial \mathbf{n}}{\partial r^1} - g_{12}\mathbf{g}_1 \cdot \frac{\partial \mathbf{n}}{\partial r^2} + g_{11}\mathbf{g}_2 \cdot \frac{\partial \mathbf{n}}{\partial r^2} - g_{12}\mathbf{g}_2 \cdot \frac{\partial \mathbf{n}}{\partial r^1} \right) \\
&= -\frac{(g_{22}b_{11} - g_{12}b_{12} + g_{11}b_{22} - g_{12}b_{21})}{g^2} \\
&= -\frac{(g_{11}b_{22} - 2g_{12}b_{12} + g_{22}b_{11})}{g^2} \\
&= -(b_{11}g^{11} + 2b_{12}g^{12} + b_{22}g^{22}) \\
&= -\kappa.
\end{aligned}$$

3.6 The surface Laplacian of the position vector, $\nabla_s^2\mathbf{p}$

If we use (3.9) with (3.2), we find

$$\begin{aligned}
\nabla_s^2 p_i &= \nabla_s \cdot \nabla_s p_i \\
&= \frac{1}{g}\frac{\partial}{\partial r^1}\left(\frac{g_{22}\frac{\partial p_i}{\partial r^1} - g_{12}\frac{\partial p_i}{\partial r^2}}{g} \right) + \frac{1}{g}\frac{\partial}{\partial r^2}\left(\frac{g_{11}\frac{\partial p_i}{\partial r^2} - g_{12}\frac{\partial p_i}{\partial r^1}}{g} \right) \\
&= \frac{1}{g}\frac{\partial}{\partial r^1}\left(\frac{g_{22}g_{1_i} - g_{12}g_{2_i}}{g} \right) + \frac{1}{g}\frac{\partial}{\partial r^2}\left(\frac{g_{11}g_{2_i} - g_{12}g_{1_i}}{g} \right) \\
&= \frac{1}{g}\left(\frac{\partial}{\partial r^1}\left(\frac{g_{22}}{g} \right) g_{1_i} + \left(\frac{g_{22}}{g} \right)\frac{\partial g_{1_i}}{\partial r^1} - \frac{\partial}{\partial r^1}\left(\frac{g_{12}}{g} \right) g_{2_i} - \left(\frac{g_{12}}{g} \right)\frac{\partial g_{2_i}}{\partial r^1} \right) \\
&+ \frac{1}{g}\left(\frac{\partial}{\partial r^2}\left(\frac{g_{11}}{g} \right) g_{2_i} + \left(\frac{g_{11}}{g} \right)\frac{\partial g_{2_i}}{\partial r^2} - \frac{\partial}{\partial r^2}\left(\frac{g_{12}}{g} \right) g_{1_i} - \left(\frac{g_{12}}{g} \right)\frac{\partial g_{1_i}}{\partial r^2} \right). \qquad (3.24)
\end{aligned}$$

Thus, we need other expressions for $\frac{\partial \mathbf{g}_1}{\partial r^1}$, $\frac{\partial \mathbf{g}_2}{\partial r^1}$, $\frac{\partial \mathbf{g}_1}{\partial r^2}$ and $\frac{\partial \mathbf{g}_2}{\partial r^2}$. We may show that

$$\begin{aligned}
\frac{\partial \mathbf{g}_1}{\partial r^1} &= a_1\mathbf{g}_1 + b_1\mathbf{g}_2 + c_1\mathbf{n} \\
\frac{\partial \mathbf{g}_2}{\partial r^1} = \frac{\partial \mathbf{g}_1}{\partial r^2} &= a_2\mathbf{g}_1 + b_2\mathbf{g}_2 + c_2\mathbf{n} \\
\frac{\partial \mathbf{g}_2}{\partial r^2} &= a_3\mathbf{g}_1 + b_3\mathbf{g}_2 + c_3\mathbf{n}
\end{aligned}$$

Taking the inner-product of $\mathbf{g}_1$, $\mathbf{g}_2$ and $\mathbf{n}$ with these three equations leads to

$$\begin{aligned}
a_1 &= \frac{(g_{22}\frac{\partial g_{11}}{\partial r^1} - 2g_{12}\frac{\partial g_{12}}{\partial r^1} + g_{12}\frac{\partial g_{11}}{\partial r^2})}{2g^2} \\
b_1 &= \frac{(2g_{11}\frac{\partial g_{12}}{\partial r^1} - g_{11}\frac{\partial g_{11}}{\partial r^2} - g_{12}\frac{\partial g_{11}}{\partial r^1})}{2g^2} \\
c_1 &= b_{11} \\
a_2 &= \frac{(g_{22}\frac{\partial g_{11}}{\partial r^2} - g_{12}\frac{\partial g_{22}}{\partial r^1})}{2g^2} \\
b_2 &= \frac{(g_{11}\frac{\partial g_{22}}{\partial r^1} - g_{12}\frac{\partial g_{11}}{\partial r^2})}{2g^2} \\
c_2 &= b_{12} \\
a_3 &= \frac{(2g_{22}\frac{\partial g_{12}}{\partial r^2} - g_{22}\frac{\partial g_{22}}{\partial r^1} - g_{12}\frac{\partial g_{22}}{\partial r^2})}{2g^2} \\
b_3 &= \frac{(g_{11}\frac{\partial g_{22}}{\partial r^2} - 2g_{12}\frac{\partial g_{12}}{\partial r^2} + g_{12}\frac{\partial g_{22}}{\partial r^1})}{2g^2} \\
c_3 &= b_{22}
\end{aligned}$$

Inserted into (3.24), we see that all the tangential components cancel, and we end up with

$$\begin{aligned}
\nabla_s^2 p_i &= \frac{(g_{11}b_{22} - 2g_{12}b_{12} + g_{22}b_{11})}{g^2} n_i \\
&= \kappa n_i.
\end{aligned}$$

3.7 Curvature-normal product

From (3.24) we have that

$$\kappa n_i = \frac{1}{g}\frac{\partial}{\partial r^1}\left(\frac{g_{22}\frac{\partial p_i}{\partial r^1} - g_{12}\frac{\partial p_i}{\partial r^2}}{g}\right) + \frac{1}{g}\frac{\partial}{\partial r^2}\left(\frac{g_{11}\frac{\partial p_i}{\partial r^2} - g_{12}\frac{\partial p_i}{\partial r^1}}{g}\right).$$

Finally, by the use of (2.24)-(2.27) and (2.21)

$$\begin{aligned}\kappa n_i &= \frac{1}{g}\frac{\partial}{\partial r^1}\left(g\left(\frac{g_{22}g_{1_i}-g_{12}g_{2_i}}{g^2}\right)\right)+\frac{1}{g}\frac{\partial}{\partial r^2}\left(g\left(\frac{-g_{12}g_{1_i}+g_{11}g_{2_i}}{g^2}\right)\right)\\ &= \frac{1}{g}\frac{\partial}{\partial r^1}(g(g^{11}g_{1_i}+g^{12}g_{2_i}))+\frac{1}{g}\frac{\partial}{\partial r^2}(g(g^{21}g_{1_i}+g^{22}g_{2_i}))\\ &= \frac{1}{g}(gg_i^{\alpha})_{,\alpha}. \end{aligned} \tag{3.25}$$

4 Derivation of the surface integral

We have now obtained all the necessary expressions in order to find another expression for (1.1) by using surface intrinsic coordinates. By the use of (2.13), (3.25) and (3.10),

$$\begin{aligned}
\int_\Gamma v_i \sigma_{ij} n_j \,\mathrm{d}S &= \int_\Gamma v_i(\gamma n_i \kappa + (\nabla_s \gamma)_i)\,\mathrm{d}S \\
&= \int_{\hat{\Gamma}} v_i(\gamma g^{-1}(g g_i^\alpha)_{,\alpha} + \gamma_{,\alpha} g_i^\alpha) g \,\mathrm{d}r^1\,\mathrm{d}r^2 \\
&= \int_{\hat{\Gamma}} v_i(\gamma g_{i,\alpha}^\alpha + \gamma g^{-1} g_{,\alpha} g_i^\alpha + \gamma_{,\alpha} g_i^\alpha) g \,\mathrm{d}r^1\,\mathrm{d}r^2 \\
&= \int_{\hat{\Gamma}} v_i(\gamma g g_{i,\alpha}^\alpha + \gamma g_{,\alpha} g_i^\alpha + \gamma_{,\alpha} g g_i^\alpha)\,\mathrm{d}r^1\,\mathrm{d}r^2 \\
&= \int_{\hat{\Gamma}} v_i(\gamma g g_i^\alpha)_{,\alpha}\,\mathrm{d}r^1\,\mathrm{d}r^2 \\
&= \oint_{\partial\hat{\Gamma}} \gamma v_i g_i^\alpha \,\mathrm{d}n_\alpha - \int_{\hat{\Gamma}} v_{i,\alpha} \gamma g_i^\alpha g \,\mathrm{d}r^1\,\mathrm{d}r^2.
\end{aligned}$$

where $\mathrm{d}n_\alpha = \tilde{\varepsilon}_{\alpha\beta}\,\mathrm{d}r_\alpha$, and $\tilde{\varepsilon}_{11} = \tilde{\varepsilon}_{22} = 0, \tilde{\varepsilon}_{12} = -\tilde{\varepsilon}_{21} = g$.

For the cases we will consider, $\oint_{\partial\hat{\Gamma}} \gamma v_i g_i^\alpha \,\mathrm{d}n_\alpha = 0$, such that

$$\int_\Gamma v_i \sigma_{ij} n_j \,\mathrm{d}S = -\int_{\hat{\Gamma}} v_{i,\alpha} \gamma g_i^\alpha g \,\mathrm{d}r^1\,\mathrm{d}r^2.$$

This is the same integral as proposed in [2].

Acknowledgement
I would like to thank Professor Einar Rønquist for proposing to work on this topic. I would also like to thank Norwegian University of Science and Technology for the financial support of this project.

References

[1] W. Flügge. *Tensor Analysis and Continuum Mechanics.* Springer, Berlin, 1972.

[2] L.W. Ho and A.T. Patera. Variational formulation of three-dimensional viscous free-surface flows: Natural imposition of surface tension boundary conditions. *International Journal for Numerical Methods in Fluids*, 13:691–698, 1991.

[3] E. Kreyszig. *Differential Geometry.* Dover Publications, Inc., 1991.

[4] C.E. Weatherburn. *Differential Geometry of Three Dimensions.* Cambridge University Press, 1927.

Paper III

Simulation of three-dimensional Bénard-Marangoni flows including deformed surfaces

Tormod Bjøntegaard and Einar M. Rønquist.

To appear in *Communications in Computational Physics*.

Is not included due to copyright

Paper IV

High order methods for incompressible fluid flow: Application to free surface problems

Tormod Bjøntegaard and Einar M. Rønquist.

In *MekIT'07*, B. Skallerud and H.I. Anderson (eds.), Tapir Akademisk Forlag, 2007.

High order methods for incompressible fluid flow: Application to free surface problems

Tormod Bjøntegaard and Einar M. Rønquist
Faculty of Information Technology, Mathematics and Electrical Engineering
The Norwegian University of Science and Technology

Summary We present an Arbitrary Lagrangian Eulerian formulation for solving free surface problems including the effect of both normal and tangential stresses. The discretization is based on combining spectral elements in space with an operator splitting approach in time. The computational framework gives exponential convergence in space, and first, second, or third order convergence in time for selected test problems with analytic solutions and data. We present simulation results for two applications. First, we consider a two-dimensional model problem associated with the transportation of fresh water in flexible containers. Second, we present simulation results for three-dimensional Bénard-Marangoni convection, incorporating both normal and tangential stresses along the free surface. The numerical prediction of the free surface deflection for such problems is completely new, and demonstrates the advantages of the particular weak form used in this study, as well as the advantages of using surface intrinsic coordinates for the imposition of free surface boundary conditions.

Introduction

Numerical solution of the Navier-Stokes equations in time-dependent geometries has found wide-spread use in science and engineering, both in the context of basic understanding of fluid flow phenomena, as well as for predictive purposes in engineering. A powerful framework is provided by the Arbitrary Lagrangian Eulerian (ALE) formulation. Even though this framework is quite mature and is currently used in many commercial codes, it is still a subject of active research; e.g., see [11].

Part of the current research in ALE methods is related to time integration. One issue is the importance of satisfying the so-called geometric conservation laws [14, 9, 8, 5]. The conclusion is still not quite clear for general Navier-Stokes problems. One complicating factor in all this effort is the fact that it is not easy to measure and verify the overall temporal accuracy during a transient simulation. This is partially due to the lack of analytical solutions for moving boundary problems, in particular, for general free surface problems where both normal and tangential stress boundary conditions are imposed.

Special issues arise when trying to apply high order spectral methods to free surface problems. Because the free surface is part of the overall time-dependent solution, it is, in general, nontrivial to update the domain boundary in such a way that the distribution of the surface points remains optimal from the initial time to the final time. This is due to the fact that the free surface displacement is fundamentally linked to the normal fluid velocity along the surface (”front-tracking”), while the tangential displacement of surface points is less well defined and is often treated in a more *ad hoc* fashion (e.g., by imposing homogeneous Dirichlet or Neumann conditions).

Assuming the evolution of the time-dependent free surface can be tracked in an accurate and efficient way, it still remains to extend the mesh velocity along the free surface to the interior of the domain. A smooth extension to the interior is most commonly used, e.g., using an harmonic extension or an elasticity solver. However, a thorough study (both theoretical and numerical) of the regularity requirements of the mesh velocity appears to be lacking.

Finally, the imposition of general free surface boundary conditions in three dimensions, both normal and tangential stresses, has not been fully explored. This is particularly true for problems

involving a variable surface tension, which necessitate an efficient and accurate calculation of simultaneous curvature and surface gradient effects using surface intrinsic coordinates [10].

The purpose of this paper is to present a few selected results from an ongoing research effort initiated with the purpose of addressing some of the above-mentioned issues. To this end, we first present numerical results which demonstrate first, second and third order splitting error in time, in combination with exponential convergence in space for analytic solution and data. We then present simulation results for two applications. The first problem is a two-dimensional free surface model problem which is motivated by the transportation of fresh water using elastic fabric containers. The second problem involves the simulation of three-dimensional Bénard-Marangoni convection [2, 3, 19, 12]. Based on the method proposed in [10], we will demonstrate a very powerful and convenient way to impose both normal and tangential stresses along general deformed surfaces. The free surface deformation associated with the formation of three-dimensional hexagonal cells has previously only been studied experimentally [4, 12] or analyzed analytically using linear stability analysis [20]; the numerical simulation of the free surface deformation presented in this study is new.

Governing equations

Strong formulation

We consider first the governing equations for incompressible fluid flow in two space dimensions; the extension to three dimensions will be discussed later. The conservation of linear momentum and the conservation of mass can be expressed as

$$\rho\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right) = \frac{\partial \sigma_{ij}}{\partial x_j} + f_i \quad \text{in } \Omega, \quad i = 1, 2, \tag{1}$$

$$\frac{\partial u_j}{\partial x_j} = 0 \qquad \text{in } \Omega, \tag{2}$$

where ρ is the density of the fluid, u_i is the i-th component of the fluid velocity in an inertial reference frame, x_j is the j-th coordinate, and f_i is the i-th component of a given body force. Summation over repeated indices is assumed. The stress tensor σ_{ij} for a viscous Newtonian fluid is given as

$$\sigma_{ij} = -p\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \qquad i, j = 1, 2, \tag{3}$$

where p is the pressure and μ is the dynamic viscosity.

We assume that the boundary $\partial\Omega$ can be split into two parts: $\partial\Omega_0$ where homogeneous velocity boundary conditions are specified, and $\partial\Omega_\gamma$ where free surface boundary conditions are specified, i.e.,

$$u_i = 0 \qquad \text{on } \partial\Omega_0, \tag{4}$$

$$n_i\sigma_{ij}n_j = \gamma\kappa - p_o \qquad \text{on } \partial\Omega_\gamma, \tag{5}$$

$$t_i\sigma_{ij}n_j = \frac{\mathrm{d}\gamma}{\mathrm{d}s} \qquad \text{on } \partial\Omega_\gamma. \tag{6}$$

Here, γ is the surface tension, κ is the local curvature, p_o is the external pressure, n_i and t_i are the i-th component of the unit normal vector and the unit tangent vector along the free surface, respectively, and s is the arclength along the free surface.

The domain Ω is, in general, time-dependent, i.e., $\Omega = \Omega(t)$. We assume that both Ω and u_i, $i = 1, 2$ are known at the initial time $t = 0$.

Weak formulation

We first introduce the function spaces X and Y defined as

$$X = \{v \in H^1(\Omega), \quad v = 0 \text{ on } \partial\Omega_0\},$$
$$Y = q \in L^2(\Omega).$$

The governing equations can then be expressed as: Find $u_i \in X$, $i = 1, 2$ and $p \in Y$ such that

$$\rho \int_{\Omega(t)} \left(v_i \frac{\partial u_i}{\partial t} + v_i u_j \frac{\partial u_i}{\partial x_j} \right) \mathrm{d}\Omega = \int_{\Omega(t)} \left(-\frac{\partial v_i}{\partial x_j} \sigma_{ij} + f_i v_i \right) \mathrm{d}\Omega + I_\gamma(v_i) \qquad \forall v_i \in X, \tag{7}$$

$$\int_{\Omega(t)} q \frac{\partial u_j}{\partial x_j} = 0 \qquad \forall q \in Y, \tag{8}$$

where

$$I_\gamma(v_i) = \int_{\partial\Omega_\gamma(t)} v_i \sigma_{ij} n_j \, \mathrm{d}s \tag{9}$$

is the surface integral resulting from integration by parts of the volume term $\int_{\Omega(t)} v_i \frac{\partial \sigma_{ij}}{\partial x_j} \, \mathrm{d}\Omega$.

ALE formulation

Let us now briefly recall the main ingredients in deriving the Arbitrary Lagangian Eulerian formulation from (7) and (8). First, we introduce a mesh velocity with components w_i, $i = 1, 2$, in order to describe the time-dependent evolution of the domain $\Omega(t)$ both in the interior and on the boundary, i.e.,

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = w_i, \quad i = 1, 2. \tag{10}$$

Second, we can exploit Reynolds transport theorem [1] in order to move the differentiation with respect to time outside the volume integral; this will prove very useful for the subsequent numerical treatment since Ω is time-dependent. Finally, we exploit Euler's expansion formula [1] to arrive at the following ALE-formulation of the momentum equations (7): Find $u_i \in X$, $i = 1, 2$ and $p \in Y$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} v_i u_i \, \mathrm{d}\Omega = \int_{\Omega(t)} \left(-\frac{\partial v_i}{\partial x_j} \sigma_{ij} + v_i f_i \right) \mathrm{d}\Omega + I_\gamma(v_i)$$
$$- \int_{\Omega(t)} \left(v_i [u_j - w_j] \frac{\partial u_i}{\partial x_j} - v_i u_i \frac{\partial w_j}{\partial x_j} \right) \mathrm{d}\Omega \qquad \forall v_i \in X. \tag{11}$$

We remark that the last integral in (11) represents all the convective contributions.

The free surface boundary conditions (5) and (6) are imposed via the surface integral $I_\gamma(v_i)$ given in (9). Using (5) and (6) and exploiting the fact that

$$\kappa \, n_i = \frac{\mathrm{d}t_i}{\mathrm{d}s}, \tag{12}$$

we can derive a particularly convenient form of this surface integral:

$$\begin{aligned} I_\gamma(v_i) &= \int_{\partial\Omega_\gamma(t)} v_i \sigma_{ij} n_j \,\mathrm{d}s \\ &= \int_{\partial\Omega_\gamma(t)} v_i \left(\gamma\kappa n_i - p_o n_i + \frac{\mathrm{d}\gamma}{\mathrm{d}s} t_i \right) \,\mathrm{d}s \\ &= \int_{\partial\Omega_\gamma(t)} v_i \left(\gamma \frac{\mathrm{d}t_i}{\mathrm{d}s} + \frac{\mathrm{d}\gamma}{\mathrm{d}s} t_i \right) \,\mathrm{d}s - \int_{\partial\Omega_\gamma(t)} p_o v_i n_i \,\mathrm{d}s \\ &= \int_{\partial\Omega_\gamma(t)} v_i \frac{\mathrm{d}(\gamma t_i)}{\mathrm{d}s} \,\mathrm{d}s - \int_{\partial\Omega_\gamma(t)} p_o v_i n_i \,\mathrm{d}s \\ &= [\gamma v_i t_i]_a^b - \int_{\partial\Omega_\gamma(t)} \gamma \frac{\mathrm{d}v_i}{\mathrm{d}s} t_i \,\mathrm{d}s - \int_{\partial\Omega_\gamma(t)} p_o v_i n_i \,\mathrm{d}s \\ &= -\int_{\partial\Omega_\gamma(t)} \gamma \frac{\mathrm{d}v_i}{\mathrm{d}s} \frac{\mathrm{d}x_i}{\mathrm{d}s} \,\mathrm{d}s - \int_{\partial\Omega_\gamma(t)} p_o v_i n_i \,\mathrm{d}s \end{aligned} \tag{13}$$

Here, a and b denote the start and end of the free surface, i.e., these points represent the surface of the free surface; we remark that this boundary term comes from another integration-by-parts and will here vanish due to the boundary conditions (4) (note that this term will also vanish if the free surface represents a closed surface, or for periodic boundary conditions).

We thus end up with two contributions. The first integral in (13) includes the contribution from both normal and tangential stresses; the integrand has a form similar to a one-dimensional Laplacian and allows for a variable surface tension. Another advantage of this form comes from the fact that the regularity requirement on the geometry representation has been lowered through integration-by-parts (only the first derivative of x_i appears even though this term includes curvature effects). The second integral in (13) represents the normal stresses due to the external pressure p_o.

In addition to the boundary conditions, we must also impose a kinematic condition along $\partial\Omega_\gamma$:

$$w_j n_j = u_j n_j. \tag{14}$$

This condition says that the normal velocity of the free surface must coincide with the normal fluid velocity along the free surface ("fronttracking").

Extension to three dimensions

The ALE formulation for three-dimensional problems can be expressed similar to (11) and (8), the only difference being that we now have three momentum equations instead of two, i.e., $i = 1, 2, 3$. What is different in the three-dimensional case is the expression for the surface integral $I_\gamma(v_i)$. This is due to the fact that we now have two tangent vectors associated with each point along the free surface, and that these tangent vectors are not necessarity orthogonal. The curvature-normal product (12) now also include two principle curvature directions, from which we can derive the mean curvature. By using tools from differential geometry [22, 6, 13], it is possible to derive an expression for $I_\gamma(v_i)$ which is fundamentally similar to (13); see [10].

Discretization

Our starting point for the numerical discretization is the ALE formulation presented above. The domain $\Omega(t)$ is first decomposed into spectral elements [16]; following this spatial discretization procedure, the fluid velocity, the mesh velocity, and the geometry are all approximated as

Nth order polynomials in each element (isoparametric representation), while the pressure is approximated as polynomials of degree $N-2$ within each element. This represents a stable discretization and results in a spatial discretization error which depends on the regularity of the solution and the data.

For the temporal discretization we have extended the convection-Stokes splitting approach presented in [17] (the OIF-method) to time-dependent geometries. This will allow us to obtain first, second or third order convergence in time; the technical details of this approach will be reported elsewhere.

The boundary conditions are imposed via the surface integrals given in (13), and the kinematic condition (14) is imposed. The treatment of the tangential mesh velocity along the free surface is done in a way that will ensure a good point distribution for all times. The mesh velocity along the free surface can be extended to the interior of the domain using different methods; we will comment more on the possibilities and restrictions regarding the mesh velocity in a forthcoming article.

Convergence results for an ALE test problem

We first verify our discretization approach by solving the two-dimensional Navier-Stokes equations in a circular domain. The domain boundary is fixed at all times, however, we specify an artificial time-periodic mesh velocity in the interior; the mesh velocity is a function of both space and time, and is zero on the domain boundary; see Figure 1. We also specify a forcing function in the momentum equations by requiring that the following analytic solution satisfies the incompressible Navier-Stokes equations (here expressed in polar coordinates):

$$u_r(r,\theta,t) = \frac{1}{5}\sin^2(\pi r)\sin(\theta)\sin(t), \tag{15}$$

$$u_\theta(r,\theta,t) = \frac{1}{5}\sin(\pi r)(2\pi r\cos(\pi r)+\sin(\pi r))\cos(\theta)\sin(t), \tag{16}$$

$$p(r,\theta,t) = \sin^2(\pi r)\sin(t). \tag{17}$$

The convergence results in Figure 2 show the expected behavior.

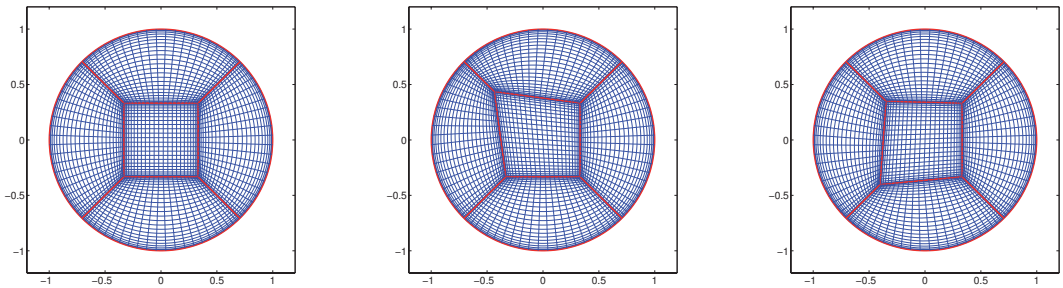
Additional numerical convergence results will be reported elsewhere.

Transportation of fresh water in a fabric container

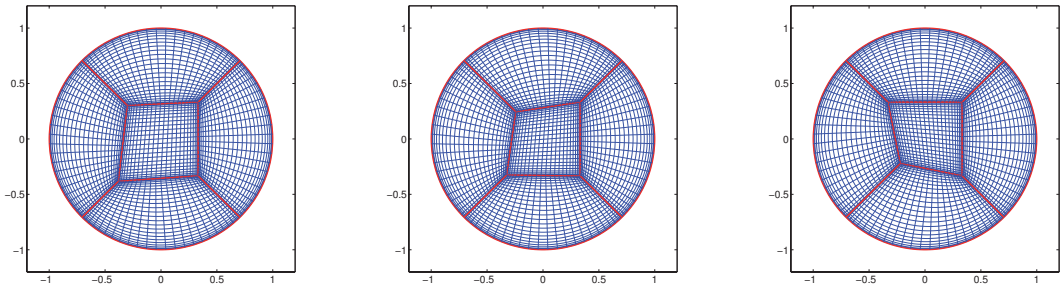
Consider the case where we have a fabric container filled with a fluid of density ρ_{in} (e.g., fresh water). We wish to predict the dynamic and static response of this container when it is put in a fluid with density $\rho_o > \rho_{in}$ (e.g., salt water). This example is motivated by the potential use of deformable fabric containers for transportation of fresh water; see [15] as well as Figure 3.

In our simplified model problem, we assume that the container initially has the form of a circle where half of the container is immersed in the denser fluid and the other half is above; see Figure 4. The dynamic response is modeled by the Navier-Stokes equations inside the container subject to free-surface boundary conditions along the flexible membrane; the forces acting on the membrane is here modeled via standard surface tension effects. In our simplified model, we set the external pressure p_o equal to the atmospheric pressure p_a if a point on the membrane is above the denser fluid, and equal to the hydrostatic pressure if a point on the membrane is adjacent to the denser fluid, i.e.,

$$\begin{aligned} p_o &= p_a && x_2 \geq 0, \\ p_o &= p_a - \rho_o g x_2 && x_2 < 0, \end{aligned}$$



(a) t=0 (b) t=0.2 (c) t=0.4



(d) t=0.6 (e) t=0.8 (f) t=1

Figure 1: The boundary of the domain is a circle which is time-independent. However, we specify a mesh velocity in the interior which is a function of both space and time, and which is zero on the external boundary. The plot shows the grid-configuration at a few time levels during one single period.

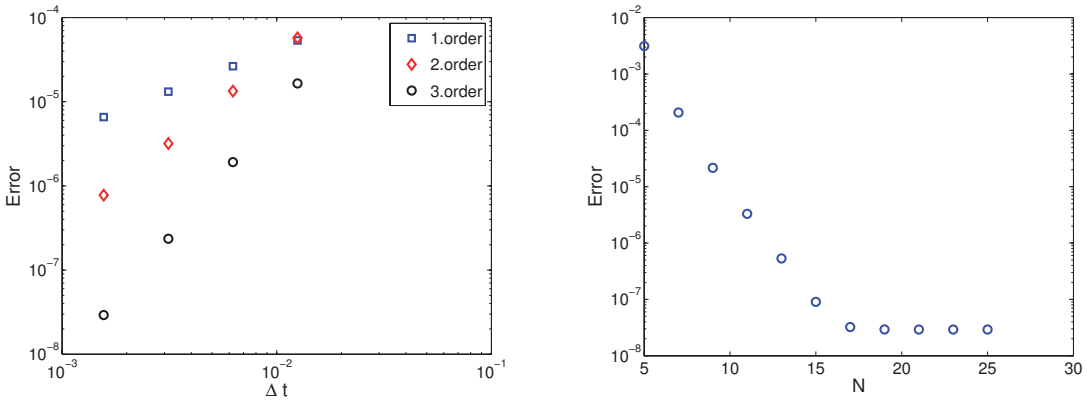


Figure 2: The left plot depicts the discretization error (energy norm) as a function of the time step, Δt, for a first, second, and third order temporal splitting scheme; the spatial error is here subdominant the temporal error. The right plot depicts the discretization error as a function of the polynomial degree, N, used in each spectral element; the temporal error is here subdominant the spatial error for $N < 15$.

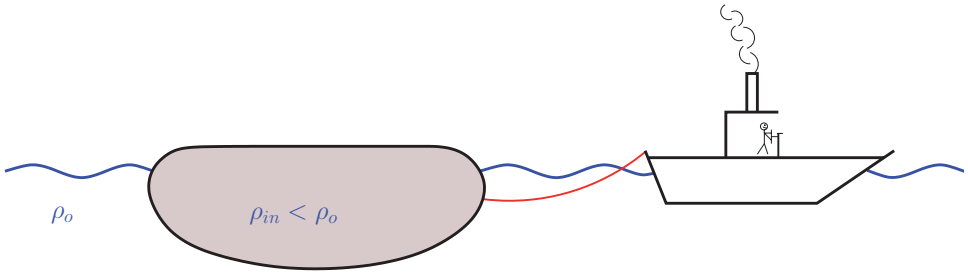
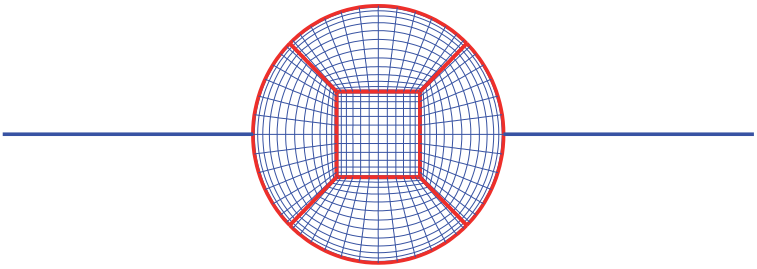


Figure 3: Transportation of fresh water using a deformable fabric container.

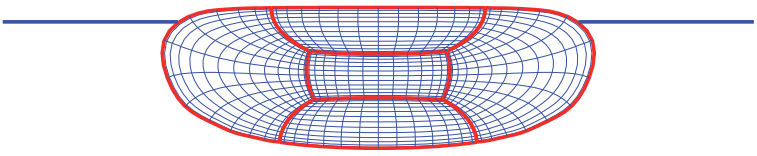
where g represents gravity ($g > 0$). Finally, the volumetic body force associated with the momentum equations inside the container is also due to gravity and given by $f_1 = 0$, $f_2 = -\rho_{in} g$.

The solution to this problem can be described in terms of two non-dimensional numbers, e.g., the Capillary number $Ca = \rho_{in}\nu^2/\gamma L$ (viscous forces relative to surface tension forces), and by the Stokes number $St = Bo/Ca$ (gravitational forces relative to viscous forces). Here, $Bo = \rho_{in} g L^2/\gamma$ is the Bond number (gravitational forces relative to surface tension forces), and L is a characteristic length scale. Note that the non-dimensionalization of this problem is not unique and depends on which forces are dominating.

We discretize the governing equations and the boundary conditions as described earlier, and solve for the dynamic behavior of the fabric container. Typically, the container is completely submerged during part of a single period of a damped oscillating behavior. In Figure 4 we show the initial and final shape of the fabric container for $St = 5$ and $Ca = 7$.



(a) Initial condition



(b) Steady state

Figure 4: Initial and final shape of the fabric container for the case $St = 5$ and $Ca = 7$; five spectral elements are used to solve this problem. The horizontal line corresponds to $x_2 = 0$.

In Figure 5 we compare the shape of the container at steady state for different Capillary numbers and for a fixed Stokes number $St = 5$.

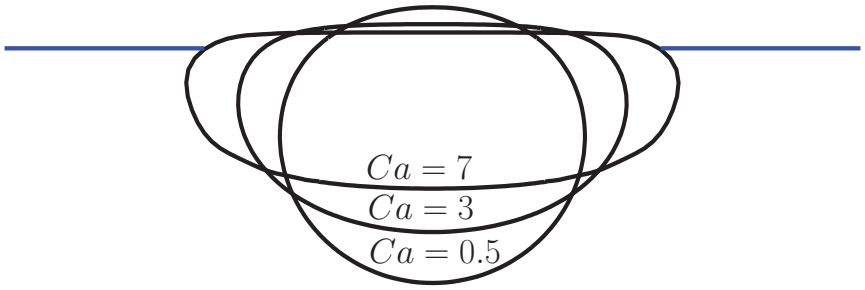


Figure 5: The shape of the fabric container at steady state is a strong function of the Capillary number (i.e., the elastic properties of the membrane). The horizontal line corresponds to $x_2 = 0$.

Simulation of three-dimensional Bénard-Marangoni convection

We now present simulation results which exploit many benefits of the proposed computational approach. In particular, we consider Bénard-Marangoni convection in a horizontal fluid layer heated from below and with a free surface on the top. The terms "top" and "bottom" here refer to the case when gravitational forces are present (e.g., on Earth with gravity pointing downwards); however, we also consider zero-gravity conditions.

Bénard-Marangoni convection has been studied extensively experimentally, theoretically, and computationally. One of the intriguing features with this problem is the formation of hexagonal convection cells from random initial conditions. This formation can originate from different effects: it can be caused by small density variations due to the fact that the density is a function of the temperature, or it can be due to variations in the surface tension due to the fact that the surface tension is a function of the temperature, or both effects can be present at the same time. In all cases, these effects are incorporated into a coupled fluid-thermal model via the linearizations

$$\rho(T) = \rho_0(1 - \beta(T - T_0)), \tag{18}$$
$$\gamma(T) = \gamma_0(1 - \tau(T - T_0)), \tag{19}$$

where ρ is the density, γ is the surface tension, T is the temperature, T_0 is a characteristic temperature, and ρ_0, β, γ_0, and τ are constants.

The governing equations for the fluid problem are the incompressible Navier-Stokes equations as described earlier. In the case of non-zero gravity, the body forces are the buoyancy forces due to the small density variations given by (18) (the "Boussinesq approximation"), while the effect of surface tension variation is included through the surface integral I_γ with a variable surface tension given by (19). The behavior of this system is governed by the following non-dimensional numbers: the Rayleigh number $Ra = g\beta\Delta T L^3/\kappa\nu$, the Marangoni number $Ma = \gamma_0\tau\Delta T d/\mu\kappa$, the capillary number $Ca = \mu\kappa/\gamma_0 L$, and the Prandtl number $Pr = \nu/\kappa$. Here, L is the thickness of the fluid layer, ΔT is the temperature difference between the top and the

bottom surface in the purly conductive regime, and ν and κ are the momentum and thermal diffusivities, respectively.

An issue which has been studied extensively, both experimentally and theoretically, is the small deformation of the free surface in the presence of hexagonal cells. Depending on whether buoyancy effects or surface tension gradient effects are dominating, the free surface is either elevated or depressed at the centers of the cells. Previous computational studies of Bénard-Marangoni convection has been done, however, all these studies assume a fixed and undeformed "free" surface. In the present study, we include all the physical effects, combining both normal and tangential stresses along the free surface via the surface integral I_γ. We are therefore able to compute the associated surface deflection over each hexagonal cell.

We now report some of the results obtained by our computational approach. The first results are for the case with an infinite Prandtl number and zero Rayleigh number. In steady state, this corresponds to the limit of solving the steady Stokes equations in zero gravity conditions. The computatational domain is a three-dimensional box, with periodic boundary conditions specified along the "vertical" sides. The initial condition for the velocity is zero, while the temperature (or, more precisely, the deviation from a purely conductive temperature profile) is set to be a random field at time $t = 0$. This particular case has been studied computationally in [21] and [18], but then with a fixed and flat "free" surface. The results in Figure 6 depict the velocity vectors and temperature distribution over the free surface at steady state (top view); we clearly see the presence of hexagonal cells. In Figure 7 we also show the deformation of the free surface over a single cell; the depressed free surface at the center of the cell is consistent with the fact that there are no buoyancy effects. Figure 8 depicts the same results seen from above; we clearly see that the results are independent of the particular surface discretization, indicating the advantages of expressing the surface integral I_γ using surface intrinsic coordinates. Finally, in Figure 9, we report the maximum surface deflection as a function of the Capillary number; the results are consistent with earlier theoretical results based on linear stability analysis; see [20].

Finally, we show the steady state results for a case where both buoyancy effects and surface gradient effects are present; the particular values of the non-dimensional numbers correspond to the properties of silicon oil. The domain has the shape of a hexagon (top view). The boundary conditions for the fluid problem are solid walls on the bottom and along the vertical sides, and free surface conditions on the top surface. The boundary conditions for the thermal problem are homogeneus Dirichlet condition on the bottom surface and adiabatic conditions along the remaining sides. The initial condition for the velocity is zero, while the temperature (or, more precisely, the deviation from a purely conductive temperature profile) is set to be a random field at time $t = 0$. Figure 10 depicts the temperature contours and the free surface deflection at steady state. The formation of seven cells is in qualitative agreement with the numerical and experimental results presented in [18] using a flat "free" surface.

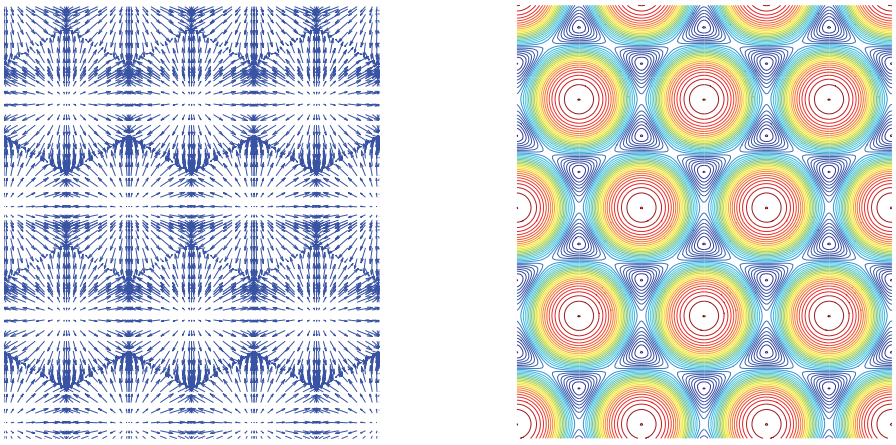


Figure 6: Numerical results for the three-dimensional Bénard-Marangoni convection problem. The left plot shows the velocity vectors, while the right plot depicts the temperature distribution over the free surface at steady state (top view) for the case with $Ma = 90$, $Ra = 0$, $Pr = \infty$, and $Ca = 3 \cdot 10^{-4}$.

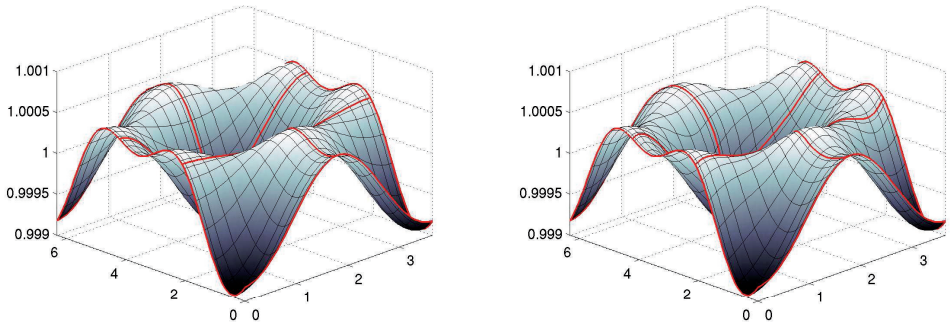


Figure 7: Free surface elevation over a single periodic structure using two different spectral element grids: a grid with straight sides (left) and a grid with deformed sides (right) inside the periodic structure. Here, $Ma = 90$, $Ra = 0$, $Pr = \infty$, and $Ca = 3 \cdot 10^{-4}$.

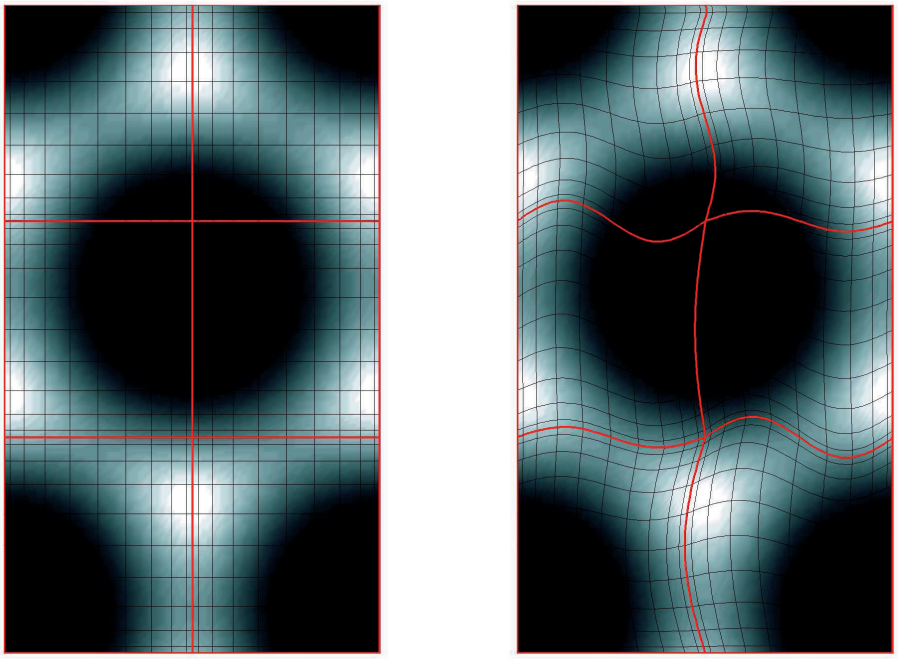


Figure 8: Similar results as in Figure 7, but now showing the top view.

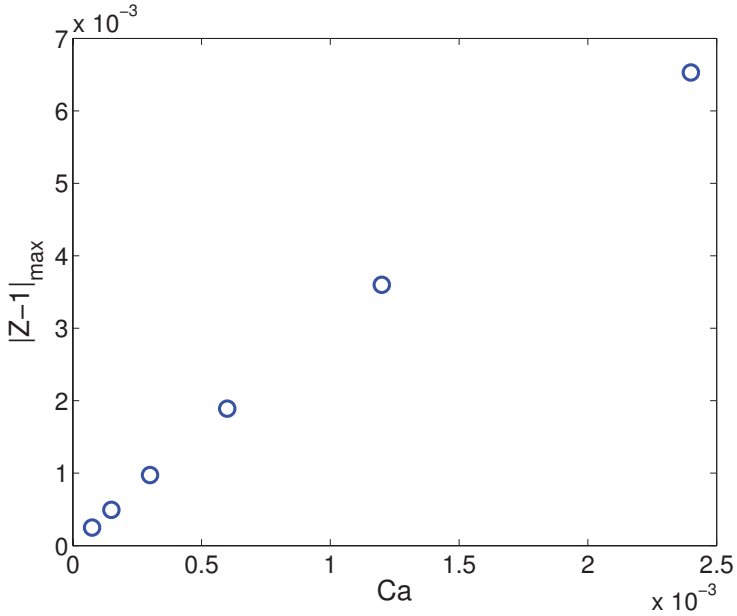
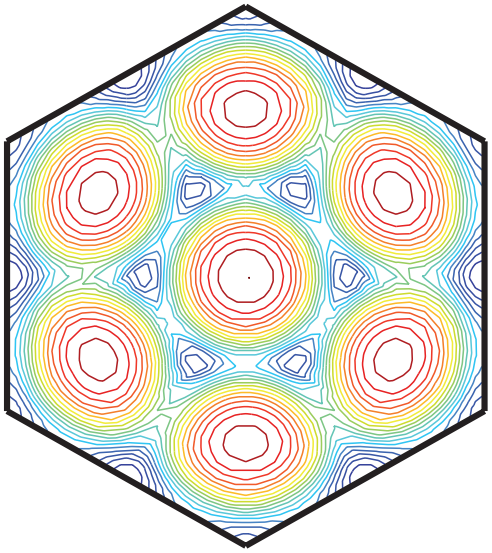
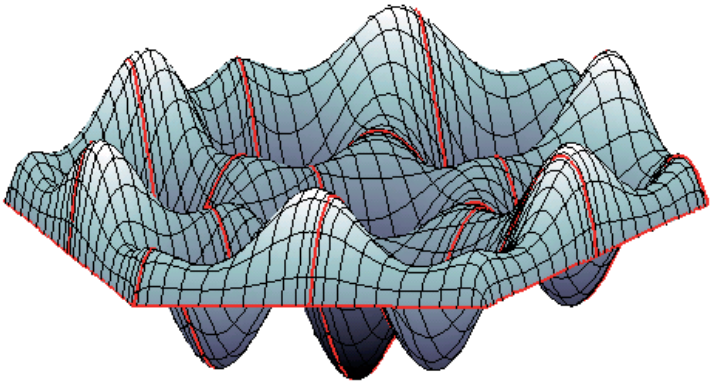


Figure 9: Numerical results for the three-dimensional Bénard-Marangoni convection problem: maximum surface deflection as a function of the Capillary number for the case with $Ma = 90$, $Ra = 0$, and $Pr = \infty$. Periodic boundary conditions are specified along the "vertical" sides.



(a)



(b)

Figure 10: Three-dimensional simulation results for the Bénard-Marangoni convection problem in a hexagonal domain for the case $Ma = 105$, $Ra = 48$, $Pr = 890$, and $Ca = 3 \cdot 10^{-4}$ (corresponding to silicon oil): (a) temperature contours (top view); and (b) free surface deflection.

References

[1] R. Aris. *Vectors, Tensors, and the Basic Equations of Fluid Mechanics*. Dover Publications, 1962.

[2] H. Bénard. Les tourbillons cellulaires dans une nappe liquide transportant de la chaleur par convection en régime permanent. *Annales de Chimie et de Physique*, 23:62–144, 1901.

[3] M. J. Block. Surface tension as the cause of Bénard cells and surface deformation in a liquid film. *Nature*, 178:650–651, 1956.

[4] P. Cerisier, C. Jamond, J. Pantaloni, and J. C. Charmet. Deformation de la surface libre en convection de Bénard-Marangoni. *J. Physique*, 45:405–411, 1984.

[5] C. Farhat and P. Geuzaine. Design and analysis of robust ALE time-integrators for the solution of unsteady flow problems on moving grids. *Computer Methods in Applied Mechanics and Engineering*, 193:4073–4095, 2004.

[6] W. Flügge. *Tensor Analysis and Continuum Mechanics*. Springer, Berlin, 1972.

[7] L. Formaggia and F. Nobile. A stability analysis for the arbitrary Lagrangian Eulerian formulation with finite elements. *East-West Journal of Numerical Mathematics*, 7(2):105–131, 1999.

[8] L. Formaggia and F. Nobile. Stability analysis of second-order time accurate schemes for ALE-FEM. *Computer Methods in Applied Mechanics and Engineering*, 193:4097–4116, 2004.

[9] H. Guillard and C. Farhat. On the significance of the geometric conservation law for flow computations on moving meshes. *Computer Methods in Applied Mechanics and Engineering*, 190:1467–1482, 2000.

[10] L.-W. Ho and A. T. Patera. Variational formulation of three-dimensional viscous free-surface flows: Natural imposition of surface tension boundary conditions. *International Journal for Numerical Methods in Fluids*, 13:691–698, 1991.

[11] A. Huerta and A. Rodríguez-Ferran (eds.). The Arbitrary Lagrangian-Eulerian Formulation. *Computer Methods in Applied Mechanics and Engineering*, 193(39-41):4073–4456, 2004.

[12] E. L. Koschmieder. *Bénard Cells and Taylor Vortices*. Cambridge University Press, 1993.

[13] E. Kreyszig. *Differential Geometry*. Dover Publications, 1991.

[14] M. Lesoinne and C. Farhat. Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer Methods in Applied Mechanics and Engineering*, 134:71–90, 1996.

[15] G. Løland and J. V. Aarsnes. Fabric as construction material for marine applications. *Hydroelasticity in Marine Technology*, pages 275–286, 1994.

[16] Y. Maday and A. T. Patera. Spectral element methods for the Navier-Stokes equations. *in: A.K. Noor, J. T. Oden (Eds.), State of the Art Surveys in Computational Mechanics, ASME, New York*, pages 71–143, 1989.

[17] Y. Maday, A. T. Patera, and E. M. Rønquist. An Operator-Integration-Factor Splitting Method for Time-Dependent Problems: Application to Incompressible Fluid Flow. *Journal of Scientific Computing*, 5(4):263–292, 1990.

[18] M. Medale and P. Cerisier. Numerical simulation of Bénard-Marangoni convection in small aspect ratio containers. *Numerical Heat Transfer, Part A*, 42:55–72, 2002.

[19] J. R. A. Pearson. On convection cells induced by surface tension. *Journal of Fluid Mechanics*, 4:489–500, 1958.

[20] L. E. Scriven and C. V. Sternling. On cellular convection driven by surface-tension gradients: effects of mean surface tension and surface viscosity. *Journal of Fluid Mechanics*, 19:321–340, 1964.

[21] A. Thess and S. A. Orszag. Surface-tension-driven Bénard convection at infinite Prandtl number. *Journal of Fluid Mechanics*, 283:201–230, 1995.

[22] C. E. Weatherburn. *Differential Geometry of Three Dimensions*. Cambridge University Press, 1927.

Paper V

Accurate interface-tracking for arbitrary Lagrangian-Eulerian schemes

Tormod Bjøntegaard and Einar M. Rønquist.

Submitted to *Journal of Computational Physics*.

Accurate interface-tracking for arbitrary Lagrangian-Eulerian schemes

Tormod Bjøntegaard [a] and Einar M. Rønquist [a]

[a] *Norwegian University of Science and Technology, Department of Mathematical Sciences, Trondheim, Norway*

Abstract

We present a new method for tracking an interface immersed in a given velocity field. The method is particularly relevant to the simulation of unsteady free surface problems using the arbitrary Lagrangian-Eulerian (ALE) framework. The new method has been constructed with two goals in mind: (i) to be able to accurately follow the interface; and (ii) to maintain a good point distribution for the grid points along the interface. The method combines information from a pure Lagrangian approach with information from an ALE approach. No integration backwards in time is needed; instead, the new method relies of the solution of several pure convection problems along the interface in order to obtain the relevant information. The new method offers flexibility in terms of how an "optimal" point distribution should be defined. We have been able to verify first, second, and third order temporal accuracy for the new method by solving two-dimensional model problems with known solutions.

1 Introduction

The ability to accurately follow time-dependent surfaces is very important in many areas of computational science and engineering. An important class of such problems is free surface flows, with the free surface representing the interface between two fluids, e.g., air and water. Computational methods for solving such problems can typically be classified into two categories: methods which explicitly track the free surface (interface-tracking methods; e.g., [16]) and methods where the interface is more implicitly defined (e.g., level set methods [15,17,14] or volume-of-fluid methods [7]); we will here focus on the former class.

Interface-tracking methods (or sometimes also referred to as front-tracking methods) comprise a few essential steps. At any particular point in time, a velocity field is typically determined from the governing equations within

the fluid(s), e.g., by solving the Navier-Stokes equations. By integrating this velocity field, it is possible to obtain a new position of the interface.

A pure Lagrangian approach applied to an evolving interface is simply based on integrating the velocity of the fluid particles along the surface to obtain the position of the surface at a later point in time. However, in the context of a numerical approximation (e.g., using finite-element-based methods), a pure Lagrangian approach is often not a very practical approach since it typically results in large deformations of the computational domain.

In the context of free surface flows, the arbitrary Lagrangian-Eulerian (ALE) formulation of the governing equations has been very successful as a point of departure for a numerical approximation [6,3,10]. A typical approach to updating the free surface is to enforce a kinematic condition along the surface. This condition has its origin in a continuum description, and says that the normal fluid velocity has to be equal to the normal domain velocity at any point along the surface. An important consequence of this condition is the fact that a fluid particle which is present somewhere along the free surface at a particular time will also be present at the free surface at a later time.

While the kinematic condition enforces a normal condition, a tangential domain velocity also needs to be specified along the surface; a common choice is to enforce a homogeneous Dirichlet condition for the tangential component [18,1]. This choice typically reduces the deformation of the computational domain compared to a pure Lagrangian approach, however, it offers limited control over the quality of the grid used to represent the free surface. In particular, the *distribution* of the grid points along the free surface may deteriorate over time, which may ultimately result in severe loss of accuracy (or even breakdown of the simulation). This latter issue may be dealt with in various ways, e.g., through remeshing or other mesh update strategies [11,4]. However, the temporal accuracy will typically suffer using such a strategy.

The issue of a non-optimal evolution of the surface representation is particularly acute in the context of using high order finite elements or spectral elements. The reason for this is related to the fact that such methods depend on locally regular mappings between a reference domain and the corresponding physical element. If the distribution of the surface points along the free surface becomes very distorted, this mapping may not be so regular anymore, resulting in a loss of spatial accuracy. This will again affect the calculation of tangent and normal vectors, as well as the local curvature, since the computation of these quantities depends on the coupling between many surface points [9,21].

One could also imagine enforcing the kinematic condition together with a tangential component of the domain velocity in such a way that the integration

of the total domain velocity would: (i) result in an accurate representation of the free surface; and (ii) maintain a good distribution of the grid points along the surface [4,2]. An obvious challenge with this approach is how to define the overall domain velocity in such a way that not only good spatial accuracy is achieved (with no need for remeshing), but in a way that will also ensure good temporal accuracy (better than first order). The goal of this paper is to propose a way to achieve these two objectives at the same time.

The paper is organized as follows. We first discuss some key aspects associated with the two-dimensional problems we will focus on, including the notation we will use. We will only discuss the evolution of a surface when it is "immersed" in a known two-dimensional velocity field; no partial differential equation will be solved to obtain this velocity field. We will let the surface evolve in time, and different computational strategies for predicting the surface evolution will be tested and compared. In particular, we will compare two well-known methods with the new approach proposed in this work. Numerical tests will illustrate the similarities and differences between the methods, and conclusions will be made based on these. This study is part of an ongoing research project on solving partial differential equations in time-dependent geometries.

2 Two-dimensional interface-tracking

Consider first the front depicted in Figure 1. Assume that we know the front at time t^n. Assume also that a numerical approximation of the front is used, something which requires a surface parameterization. A typical way to achieve this is to use piecewise polynomial approximations (e.g., finite-element-based methods), which typically introduces grid points.

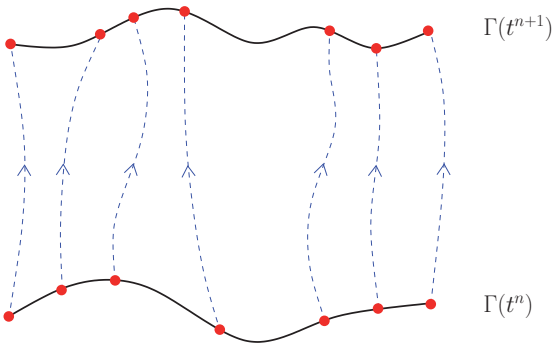


Fig. 1. A front at time t^n with an "optimal" point distribution. For example, the points can be the nodes along an edge of a deformed spectral element. The front is "immersed" in a velocity field and the particles follow the path of the dashed lines.

Assume now that we have an interface with "optimally" distributed grid points at time level t^n. At each point $\mathbf{x}$ along the surface there is an associated velocity

field $\mathbf{u}$. This velocity field can be explicitly known (as in our study here), or it can be given as the solution of an underlying partial differential equation (e.g., the solution of the Navier-Stokes equations in a free surface problem).

We assume that the velocity at a point along the surface represents the velocity of the corresponding "fluid particle". If we integrate the velocity of all the fluid particles along the surface, we obtain the position of the surface at a later time. This is what a pure Lagrangian description will give us; the motion of a particle is simply governed by the equation

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{u}(\mathbf{x}, t). \tag{1}$$

In a computational setting, we can limit the integration of (1) to the grid points, and then use the underlying surface parameterization to represent the entire surface at a later time; see Figure 1. A severe problem with this approach is obvious: we have no control over the distribution of the grid points at a later time t^{n+1}. This will again result in a loss of accuracy in the calculation of surface quantities, e.g., tangent and normal vectors, as well as the local curvature.

A great advantage with the ALE formulation is that it introduces a separate domain velocity $\mathbf{w}$ (also referred to as the grid velocity in the context of the discrete problem), which limits the deformation of the computational domain. In an ALE-framework, the position of the interface is advanced according to

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{w}(\mathbf{x}, t) \tag{2}$$

instead of the pure Lagrangian approach (1).

A continuum description dictates that $\mathbf{w} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n}$ (the kinematic condition), where $\mathbf{n}$ is the unit normal along the interface. However, no particular condition is required for the tangential component $\mathbf{w} \cdot \mathbf{t}$ of the domain velocity ($\mathbf{t}$ is unit tangent vector). A common choice is to set $\mathbf{w} \cdot \mathbf{t} = 0$ along the surface, although this is often not an optimal choice; see Figure 2 and Figure 3.

Let us also comment on the issue of temporal accuracy. Integration of (1) and (2) can be done using an explicit method; often an explicit multi-step method (e.g., Adams-Bashforth) is used [8,1]. The choice of an explicit method is often motivated by the wish to compute the velocity field separately from the treatment of the geometry. If the velocity fields ($\mathbf{u}$ and $\mathbf{w}$) are sufficiently regular, we expect to achieve higher order temporal accuracy (second and third) in terms of the *location* of individual points along the front. As mentioned above, this approach may yield limited control over the *distribution* of the

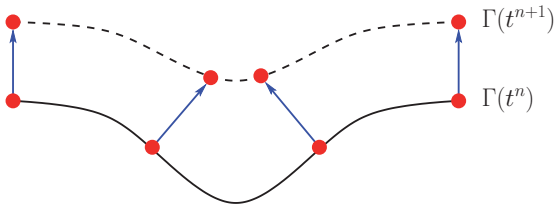


Fig. 2. A front at time t^n with an "optimal" point distribution. This interface is advanced by honoring the kinematic condition, while imposing a zero tangential grid velocity. The resulting point distribution at a later time t^{n+1} is obviously no longer optimal.

points along the front. If the point distribution is non-optimal, the resulting loss of spatial accuracy *will* affect the accuracy of surface quantities such as normal and tangent vectors, local curvature, and length/area, and this again may affect the accuracy of the interface tracking.

In order to construct a computational approach which will yield both high order temporal accuracy (e.g., second or third order), as well as good spatial accuracy in the calculation of surface quantities, we need to solve the problem of automatically obtaining a good point distribution in a satisfactory way. This problem is particularly acute in the context of using high order methods. Despite the importance of this issue, very limited discussion or results appear to be available in the literature.

We also mention a complicating factor in the development and assessment of various approaches for interface tracking: the lack of analytic solutions. In Section 4, we propose a few examples of test problems which we will use for verification purposes.

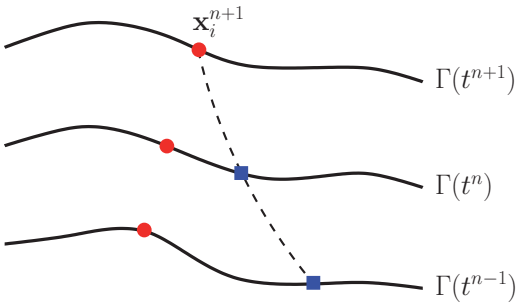


Fig. 3. The plot depicts the position of a single point along the front at three different time levels (red circles). The point moves according to (2), with $\mathbf{w} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n}$ and $\mathbf{w} \cdot \mathbf{t} = 0$. The position at time level t^{n+1} is $\mathbf{x}_i^{n+1}$. Note that the corresponding positions at time levels t^n and t^{n-1} do *not* correspond to the same fluid particle; the path of the fluid particle (e.g., a particle which moves according to (1)) ending up at position $\mathbf{x}_i^{n+1}$ at time t^{n+1} follows the dashed line.

3 The new approach

We will now propose a method which gives an accurate representation of the interface (i.e., the individual grid points end up on the correct surface), as well as the flexibility of specifying a user defined point distribution. This method will be "automatic" in the sense of fulfilling both goals in an "integrated" fashion, i.e., with no need for remeshing etc.

Our strategy is based on a combination of a pure Lagrangian approach and an ALE approach, in which both (1) and (2) are integrated in order to find the new grid points. One major feature of the new algorithm is that we search for the fluid particle at time t^n which ends up along some prescribed direction at time t^{n+1}. Thus, this will invole an iteration process for each grid point.

In order to make the algorithm concrete, we will limit our discussion to the case where the entire front $\Gamma(t)$ corresponds to an edge in a single spectral element. However, we remark that the ideas behind the proposed method is equally applicable to the case where the front is composed of a number of low order finite elements.

Following a standard spectral element discretization [12], the front $\Gamma(t)$ is parameterized as follows: for a given $\xi \in \hat{\Gamma} = [-1, 1]$, the corresponding point $\mathbf{x}$ on $\Gamma(t)$ is given as $\mathbf{x} = (x_1, x_2) = (x_1(\xi), x_2(\xi))$. In general, any field variable φ associated with the front can be represented in terms of the reference variable ξ. In particular, an Nth order polynomial approximation φ_N of φ at time t^n can be expressed in terms of the following nodal basis:

$$\varphi_N^n(\xi) = \sum_{j=0}^{N} \varphi_j^n \ell_j(\xi). \tag{3}$$

Here, φ_j^n represents an approximation of $\varphi(\xi_j, t^n)$, ξ_j is the j-th Gauss-Lobatto Legendre (GLL) point, and $\ell_j(\xi)$ is the Nth order Lagrangian interpolant through the GLL points; as usual, $\ell_j(\xi_i) = \delta_{ij}$.

Without presenting all the details at once, we first discuss the key ingredients in a second order temporal scheme. We will later return to discuss all the details, as well as the extension to a third order temporal scheme.

We assume that the following surface variables are known:

$$\underline{x}_1^n, \underline{x}_2^n, \underline{u}_1^n, \underline{u}_2^n, \underline{u}_1^{n-1}, \underline{u}_2^{n-1}.$$

Here, $(x_i^n)_j$ is the i'th coordinate of the j'th point at time t^n and $(u_i^n)_j$ is the i'th velocity component of the j'th point at time t^n. We use underscore to denote a vector comprising *all* the nodal values associated with a field variable, i.e., $j = 0, \ldots, N$; see (3).

In Algorithm 1 we present the first version of the new algorithm. A few comments are required at this stage. First, *convergence* in this setting is related to the point *distribution*. The position computed in Step 4 represents the integration of (1) using a second order Adams-Bashforth scheme, however, we don't know if the computed grid point on $\Gamma(t^{n+1})$ (grid point j) is in a good position relative to its neighbors. Thus, we obtain convergence when we have chosen the "correct" particle, which amounts to choosing the "correct" ξ at time t^n (which we denote as ξ^n). However, we need to quantify this and we will return to this issue shortly.

Second, Step 2 in Algorithm 1 involves finding ξ^{n-1}, which is the reference coordinate at time t^{n-1} for the particle which at time t^n has the reference coordinate ξ^n; see also Figure 3. Hence, the computation of ξ^{n-1} appears to involve integration backwards in time. However, what we actually need is the *velocity* the particle in question had at time t^{n-1}, and not necessarily the position it had at time t^{n-1}. In Section 3.1 we will propose a way to compute $\hat{u}_i^2$, $i = 1, 2$, which does not involve integration backwards in time.

Third, when we have converged to the correct position $(x_i^{n+1})_j$ of a grid point j on $\Gamma(t^{n+1})$, see Step 4, we also compute the corresponding grid velocity $(w_i^n)_j$ such that an integration of (2) using a second order Adams-Bashforth method will result in the same position; this is done in Step 6. Here, $(x_i^n)_j$, $(x_i^{n+1})_j$, and $(w_i^{n-1})_j$ are known, while $(w_i^n)_j$ is unknown. The reason for computing the grid velocity in this way is that: (i) it gives consistency with a pure Lagrangian approach; (ii) we are indirectly satisfying the kinematic condition; and (iii) we are indirectly and "automatically" able to specify a tangential grid velocity such that we obtain a good point distribution.

Let us now discuss more precisely what we mean by convergence in Algorithm 1. Essentially, convergence is related to quantifying a good point distribution. We will consider two alternative strategies. The first strategy is illustrated in Figure 4. Starting from the front at time t^n, we move the end points to time level t^{n+1}. How these points are moved will be problem dependent, but it may for instance be a pure Lagrangian motion where the two end points follow the path of the fluid particles from t^n to t^{n+1} (see (1)), or the end points may move according to a standard ALE-formulation (see (2)). When this is done, we construct the chord between the two end points, and distribute grid points along this chord according to the desired distribution (e.g., a GLL distribution). Next, the normal from the chord is constructed at each of these points. Thus, using this strategy we search for the particle at time t^n which, when advanced through a pure Lagrangian motion, is located along this normal up to a given tolerance. The advantage of this approach is that we are in full control of the inner grid-points at each time-step. The disadvantage is that it may not be so easy to extend the approach to three dimensions.

Algorithm 1 First version of a second order temporal scheme

for $j = 0$ to N **do**

1. Guess ξ^n.

repeat

2. Find $\xi^{n-1}(\xi^n)$.
3. Compute

$$\begin{aligned} \hat{x}_i^1 &= x_i^n(\xi^n) \qquad i = 1, 2, \\ \hat{u}_i^1 &= u_i^n(\xi^n) \qquad i = 1, 2, \\ \hat{u}_i^2 &= u_i^{n-1}(\xi^{n-1}) \quad i = 1, 2. \end{aligned}$$

4. Compute $(x_i^{n+1})_j = \hat{x}_i^1 + \Delta t \left(\frac{3}{2}\hat{u}_i^1 - \frac{1}{2}\hat{u}_i^2 \right), \quad i = 1, 2.$
5. Update ξ^n.

until convergence

6. Compute $(w_i^n)_j$ from $(x_i^{n+1})_j = (x_i^n)_j + \Delta t \left(\frac{3}{2}w_i^n - \frac{1}{2}w_i^{n-1} \right)_j, \quad i = 1, 2.$

end for

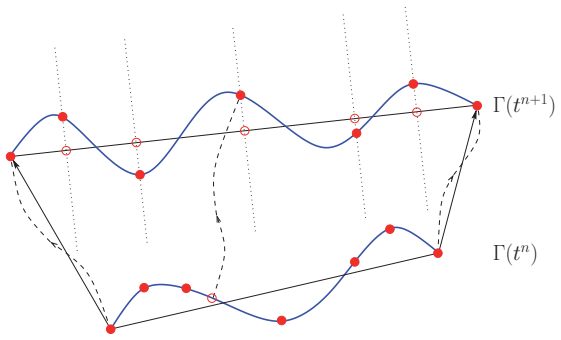


Fig. 4. Strategy 1.

The second strategy is illustrated in Figure 5. Again, the end points are first advanced to t^{n+1}. Next, for each end point, a vector which connects the end point at t^n to the end point at t^{n+1} is constructed. Then, directional vectors for the interior nodes are constructed through a linear interpolation on the reference domain, $\hat{\Gamma}$, of the end point vectors. Finally, the requirement is to find the particle at time t^n which, when advanced to t^{n+1}, is located along this interpolated vector starting at the grid point at time t^n. This strategy has the advantage that it is more easily extended to three dimensions. However, we have less explicit control over the interior grid points at each time level.

There will probably be other strategies for addressing the issue of grid distribution as well, however, this will only change the convergence condition. The rest of the algorithm which will be presented in the subsequent sections will remain the same.

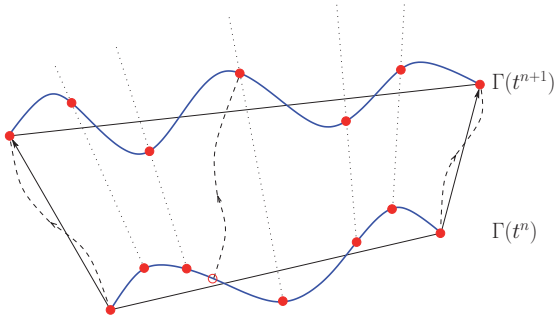


Fig. 5. Strategy 2.

3.1 Finding a fluid particle's earlier velocities

The second order scheme presented in Algorithm 1 requires information about the velocities at time levels t^n and t^{n-1} for those fluid particles along the interface which end up at the grid points $\mathbf{x}_j^{n+1}$, $j = 0, \ldots, N$, along $\Gamma(t^{n+1})$. We would like to avoid tracking the characteristics backwards in time since this requires interpolation and can be expensive for high order methods [19,5]. In addition, we need to deal with the fact that the interface Γ changes shape as a function of time.

Since we assume that the same particles remain on the surface at all time, it is sufficient to solve a one-dimensional convection problem *forward* in time. In order to explain the procedure, consider the following pure time-dependent convection problem along the interface $\Gamma(\tau)$: Find $\varphi(s, \tau)$ such that

$$\begin{aligned} \frac{\partial \varphi}{\partial \tau} + u_s \frac{\partial \varphi}{\partial s} &= 0, && \text{on } \Gamma(\tau), \\ \varphi(s, \tau = 0) &= \varphi_0(s), && \text{on } \Gamma(\tau = 0). \end{aligned}$$

Here, s is an arc-length variable and $u_s = \mathbf{u} \cdot \mathbf{t}$ is the tangential component of the fluid velocity; in all our model problems $u_s = 0$ on $\partial\Gamma(\tau)$. We can easily derive the ALE formulation of this problem: Find $\varphi(s; \tau) \in X$ such that

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\tau} \int_{\hat{\Gamma}} v \varphi J_s \, \mathrm{d}\xi + \int_{\hat{\Gamma}} v(u_s - w_s) \frac{\partial \varphi}{\partial \xi} \, \mathrm{d}\xi - \int_{\hat{\Gamma}} v \varphi \frac{\partial w_s}{\partial \xi} \, \mathrm{d}\xi = 0, \quad \forall v \in X, \\ \varphi(\xi; \tau = 0) = \varphi_0(\xi). \end{aligned} \tag{4}$$

Here, X is an appropriate function space, e.g., $X = H^{1/2}(\Gamma)$ if all the field variables correspond to the trace of H^1-functions in the adjacent fluid domains. Furthermore, $J_s = \left(\left(\frac{\partial x_1}{\partial \xi} \right)^2 + \left(\frac{\partial x_2}{\partial \xi} \right)^2 \right)^{1/2}$ is the surface Jacobian, and $w_s = \mathbf{w} \cdot \mathbf{t}$ is the tangential component of the domain velocity; note that both u_s and w_s are zero on $\partial\Gamma$ for all the model problems in our study. We also remark that we have used the same symbol φ for a field variable expressed both in the

arc-length variable and in the reference coordinate.

We discretize the convection problem (4) using a standard spectral method in space based on high order polynomials, and arrive at the following set of ordinary differential equations:

$$\begin{aligned} \frac{\mathrm{d}(\underline{B}^s\,\underline{\varphi})}{\mathrm{d}\tau} &= -\underline{C}^s(\underline{\mathbf{u}},\underline{\mathbf{w}})\,\underline{\varphi}, \\ \underline{\varphi}(\tau=0) &= \underline{\varphi}_0. \end{aligned} \tag{5}$$

Here, $\underline{B}^s$ is the surface mass matrix, $\underline{C}^s$ is the discrete convection operator along the surface (including the "surface divergence" of the domain velocity); note that both $\underline{B}^s$ and $\underline{C}^s$ are time-dependent.

If we integrate (5) from 0 to Δt with $\underline{\varphi}_0 = \underline{u}_i^{n-1}$, $i = 1, 2$, we observe that $\underline{\varphi}(\tau = \Delta t)$ will be an approximation to the i'th velocity component at time t^{n-1} of the fluid particles which at time t^n are located at the grid points along $\Gamma(t^n)$. This approach is inspired by the ideas presented in [13] in the context of constructing convection-Stokes splitting schemes.

Note that $\hat{u}_i^2$ in Algorithm 1 generally represents the velocity of a fluid particle at time t^{n-1} which does *not* coincide with a grid point along $\Gamma(t^n)$. However, by computing the velocities at time t^{n-1} of the fluid particles which coincide with the grid points of $\Gamma(t^n)$ (i.e., by solving (5)), we can use the polynomial expansion (3) to find the velocity at *any* value of the parameter ξ. Moreover, *one* particular value of ξ will now give us information about the velocity of a fluid particle at two different time levels (t^n and t^{n-1}). This is also exactly the type of information we need in our algorithm; in fact, access to this information avoids entirely the need to find $\xi^{n-1}(\xi^n)$ in Step 2 of Algorithm 1 and thus represents a significant simplification.

The approach is readily extended to velocities at earlier time levels as well. For instance, if we choose $\underline{\varphi}_0 = \underline{u}_i^{n-2}$ in (5), $\underline{\varphi}(\tau = 2\Delta t)$ will be an approximation to the i'th velocity component at time t^{n-2} of the fluid particles which at time t^n are located at the grid points along $\Gamma(t^n)$.

For the integration of (5) we have chosen the classical explicit fourth order Runge-Kutta scheme. Similar to the convection subproblem treated in [13], the convection velocities $\underline{\mathbf{u}}$ and $\underline{\mathbf{w}}$, as well as the surface geometry $\underline{\mathbf{x}}$, are each approximated as a polynomial in time of one order lower than the Adams-Bashforth scheme used in Step 4 and Step 6 of Algorithm 1. Thus, for a second order temporal scheme, a first order polynomial interpolation/extrapolation in time is used for these quantities when solving (5), while the extension to a third order scheme will use a second order polynomial approximation in time for $\underline{\mathbf{u}}$, $\underline{\mathbf{w}}$ and $\underline{\mathbf{x}}$.

To summarize this section: in Algorithm 1 we are interested in the velocities that particular fluid particles had at time level t^n and t^{n-1}. We choose $\underline{\varphi}_0 = \underline{u}_i^{n-1}$ in (5) and set $\tilde{\underline{u}}_i^n = \underline{\varphi}(\tau = \Delta t)$, $i = 1, 2$. We now have six sets of nodal values, $\underline{x}_i^n$, $\underline{u}_i^n$, and $\tilde{\underline{u}}_i^n$, $i = 1, 2$, all associated with the interface $\Gamma(t^n)$. By using the polynomial expansion (3), we have thus six polynomial approximations, and all of these are defined along $\Gamma(t^n)$. Hence, one particular value of ξ corresponds to the *same* particle. Thus, there is no longer need to compute $\xi^{n-1}(\xi^n)$ in Step 2 of Algorithm 1. We may now only focus on finding the appropriate ξ^n which will allow us to achieve convergence in Step 4.

3.2 Finding the "correct" particle

Assume that we are searching for the position of a fluid particle at time level t^n, as well as its velocities at time levels t^n and t^{n-1}, such that we can compute the position of the fluid particle at time level t^{n+1} according to Step 4 in Algorithm 1. As explained in the previous section, the problem can be reduced to finding one particular value of $\xi \in \hat{\Gamma}$.

Assume that we are currently interested in updating the information about the grid point in the middle of the reference domain $\hat{\Gamma}$ (i.e., this point is associated with the index j in Algorithm 1), and that we are searching for information about a particle located somewhere in $\hat{\Gamma}$. Finding this particle is certainly possible, but will be somewhat cumbersome if the interface Γ comprises several elements (either low order finite elements or high order spectral elements). The reason for this is that we do not *a priori* then know which element the particle belongs to at the different time levels, and some kind of search algorithm needs to be used.

Instead of finding the coordinate of this fluid particle (or the equivalent value of ξ), we propose to find a corresponding artificial time $\tilde{\tau}$ which convects this particle to our grid point (index j) using an artificial convecting velocity U; see Figure 6. To this end, we consider the one-dimensional convection problem

$$\begin{aligned} \frac{\partial \varphi}{\partial \tau} + U \frac{\partial \varphi}{\partial \xi} &= 0, \qquad \text{on } \hat{\Gamma}, \\ \varphi(\tau = 0) &= \varphi_0. \end{aligned} \tag{6}$$

We assume that $U = 0$ on $\partial\hat{\Gamma}$ (a similar condition was assumed for u_s in the previous section), and we will thus not specify any particular boundary condition for φ. We discretize (6) using a spectral method based on high order

polynomials and arrive at the following set of ordinary differential equations

$$\begin{aligned} \underline{\hat{B}}\frac{\mathrm{d}\underline{\varphi}}{\mathrm{d}\tau} + \underline{\hat{C}}\,\underline{\varphi} = 0, \\ \underline{\varphi}(\tau = 0) = \underline{\varphi}_0. \end{aligned} \tag{7}$$

In (7) , $\underline{\hat{B}}$ and $\underline{\hat{C}}$ are the mass matrix and discrete convection operator, respectively; the hat indicates that both matrices are associated with $\hat{\Gamma}$. The idea is that, instead of finding a suitable ξ in searching for the suitable particle, we search for a $\tilde{\tau}$ such that we obtain the information we need when we integrate (7) from $\tau = 0$ to $\tau = \tilde{\tau}$. Hence, instead of moving along the surface searching for a particle, we will sit at a fixed point and convect the pertinent information about the particle to us. Thus, the new procedure requires no backward time integration, and eliminates the need for a search and interpolation algorithm; see [20,5] in the context of semi-Lagrangian schemes.

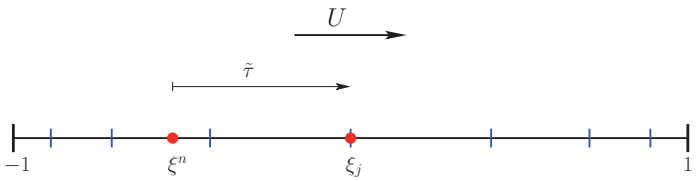


Fig. 6. Six sets of nodal values, $\underline{x}_i^n$, $\underline{u}_i^n$, and $\underline{\tilde{u}}_i^n$, $i = 1, 2$, are associated with the reference domain $\hat{\Gamma}$. By integrating (7) from $\tau = 0$ to $\tau = \tilde{\tau}$, we can artificially convect the information associated with a particular $\xi \in \hat{\Gamma}$ (i.e., associated with a particular fluid particle) and check if Step 4 in Algorithm 1 will give us a "valid" new position for a grid point along $\Gamma(t^{n+1})$.

Note that the particular value of $\tilde{\tau}$ is actually of no interest to us; all we need is $\underline{\varphi}(\tilde{\tau})$. For this reason we only need to consider a fixed domain (in our case, $\hat{\Gamma}$). There is also great flexibility in the choice of the convective velocity, U. This is because, for a specific particle somewhere along the surface, and for a reasonable choice of U, there will always be a corresponding $\tilde{\tau}$ (perhaps negative) which convects this particle to the current grid point. An easy and reasonable choice is $U(\xi) = 1 - \xi^2$ since this velocity is very smooth, it does not change sign, and it is also compatible with the condition $u_s = 0$ at $\partial\Gamma$ (which is the case for our numerical examples).

Note that $\underline{\varphi}$ in (7) is a vector with nodal values from the entire surface (corresponding to a single spectral element as in our study, or perhaps multiple finite elements). We need to carry the entire vector in the computation in order to evaluate the spatial derivative. On the other hand, we are only interested in the information convected to our current grid point. Thus, we need to solve one problem of the type (7) for each grid point.

3.3 Final version of a second order temporal scheme

The discussion from the preceding sections leads us to an improved algorithm for a second order temporal scheme; see Algorithm 2 below. Let us say a few words about how we achieve convergence in the new grid positions. In Step 1 we guess a $\tilde{\tau}$. For each grid point, we then solve (7) six times. From the resulting information, we update the new grid position as indicated in Step 3. We can check how this position compares with a "valid" position according to our chosen convergence strategy, see Figure 4 and Figure 5. A Newton-iteration can then be used to find the correct $\tilde{\tau}$, which again implies the correct ξ (or choosing the correct fluid particle). In our case, we obtain derivative information for the Newton iteration from the last stage in the explicit Runge-Kutta scheme that we use to integrate (7). For the test problems considered in this study, convergence is achieved in 2 or 3 Newton iterations.

Finally, we remark that the loop $j = 0, \ldots, N$ over the grid points includes the two end points of the interface. These two end points may be treated differently compared to the inner points; this depends on the particular information which is available for the end points. For instance, for our numerical test problems, the end points are moved in a pure Lagrangian fashion.

Algorithm 2 Final version of a second order temporal scheme

Solve (5) with $\underline{\varphi}_0 = \underline{u}_i^{n-1}$ and set $\underline{\tilde{u}}_i^n = \varphi(\tau = \Delta t)$ for $i = 1, 2$.
for $j = 0$ to N **do**
1. Guess $\tilde{\tau}$.
repeat
2. Integrate (7) from 0 to $\tilde{\tau}$ six times with $\underline{\varphi}_0 = \underline{x}_i^n$, $\underline{\varphi}_0 = \underline{u}_i^n$, $\underline{\varphi}_0 = \underline{\tilde{u}}_i^n$, for $i = 1, 2$. Define the results as $\underline{\hat{x}}_i$, $\underline{\hat{u}}_i^1$, $\underline{\hat{u}}_i^2$, respectively.
3. Compute $(x_i^{n+1})_j = (\hat{x}_i)_j + \Delta t \left(\frac{3}{2}\hat{u}_i^1 - \frac{1}{2}\hat{u}_i^2\right)_j$ for $i = 1, 2$.
4. Update $\tilde{\tau}$.
until convergence
5. Compute $(w_i^n)_j$ from $(x_i^{n+1})_j = (x_i^n)_j + \Delta t \left(\frac{3}{2}w_i^n - \frac{1}{2}w_i^{n-1}\right)_j$ for $i = 1, 2$.
end for

3.4 Extension to a third order temporal scheme

Algorithm 3 represents a third order version of this method. We see that we now have to first solve two problems of the type (5) for each velocity component. During the integration of these problems, we need to use a second order polynomial approximation in time of the grid, the fluid velocity, and the grid velocity in order to maintain a third order temporal convergence rate.

Algorithm 3 Final version of a third order temporal scheme

Solve (5) with $\underline{\varphi}_0 = \underline{u}_i^{n-1}$ and set $\tilde{\underline{u}}_i^n = \varphi(\tau = \Delta t), \quad i = 1, 2.$
Solve (5) with $\underline{\varphi}_0 = \underline{u}_i^{n-2}$ and set $\tilde{\tilde{\underline{u}}}_i^n = \varphi(\tau = 2\Delta t), \quad i = 1, 2.$
for $j = 0$ to N **do**
1. Guess $\tilde{\tau}$.
repeat
2. Integrate (7) from 0 to $\tilde{\tau}$ eight times with $\underline{\varphi}_0 = \underline{x}_i^n$, $\underline{\varphi}_0 = \underline{u}_i^n$, $\underline{\varphi}_0 = \tilde{\underline{u}}_i^n$, $\underline{\varphi}_0 = \tilde{\tilde{\underline{u}}}_i^n$, $i = 1, 2$. Define the results as $\hat{\underline{x}}_i$, $\hat{\underline{u}}_i^1$, $\hat{\underline{u}}_i^2$, $\hat{\underline{u}}_i^3$, respectively.
3. Compute $(x_i^{n+1})_j = (\hat{x}_i)_j + \Delta t \left(\frac{23}{12}\hat{u}_i^1 - \frac{4}{3}\hat{u}_i^2 + \frac{5}{12}\hat{u}_i^3 \right)_j, \qquad i = 1, 2.$
4. Update $\tilde{\tau}$.
until convergence
5. Compute $(w_i^n)_j$ from $(x_i^{n+1})_j = (x_i^n)_j + \Delta t \left(\frac{23}{12}w_i^n - \frac{4}{3}w_i^{n-1} + \frac{5}{12}w_i^{n-2} \right)_j$, $i = 1, 2.$
end for

Obviously, the overall integration scheme for integrating (5) must also be at least of third order in time. As before, we use a fourth order explicit Runge-Kutta scheme, so this will not cause any problems. The solution of (7) will be the same as for the second order scheme. In Step 3 and Step 5, we use a third order Adams-Bashforth scheme for computing the new position and the new grid velocity of each grid point, respectively.

4 Numerical results

In this section we will perform numerical experiments in order to validate and compare the different algorithms for tracking the interface. For all the test problems we will define a two-dimensional, time-dependent velocity field, and at time $t = 0$ we will specify an initial interface. With the interface "immersed" in this two-dimensional velocity field, we will then monitor:

(1) how accurately we are able to follow the exact interface; and
(2) the quality of the corresponding point distribution.

The velocity fields will be prescribed in such a way that we are able to obtain analytic solutions for the exact interfaces at all times.

4.1 Error computation

The way we compute the error, E_1, in following the interface is illustrated in Figure 7. We first compute the chord between the end points of our numerical

solution, and then compute the normal, $\mathbf{n}_c$, to this chord. For each grid point, $(\mathbf{x}^n)_j$, $j = 0, \ldots, N$, along our numerically approximated interface, we find the intersection between the analytical front and the line originating from $(\mathbf{x}^n)_j$ and moving in the $\mathbf{n}_c$-direction. We call this intersection $(\mathbf{x}^e)_j$ and compute

$$e_j = \sqrt{((x_1^e)_j - (x_1^n)_j)^2 + ((x_2^e)_j - (x_2^n)_j)^2}.$$

Finally, we define the error E_1 as

$$E_1 = \frac{1}{N+1} \sum_{j=0}^{N} e_j. \tag{8}$$

The way we will measure the grid quality is by computing the error, E_2, in the length of the interface. For a high order approximation, this measure will generally give an indication of the quality of the point distribution (although there are special situations when this is not the case). Thus, we first compute the length, S_n, of our numerically computed interface. Using GLL quadrature we compute

$$S_n = \sum_{\alpha=0}^{N} \rho_\alpha (J_s^n)_\alpha = \sum_{\alpha=0}^{N} \rho_\alpha \left(\left(\frac{\partial x_1^n}{\partial \xi} \right)^2 + \left(\frac{\partial x_2^n}{\partial \xi} \right)^2 \right)_\alpha^{1/2}, \tag{9}$$

where J_s^n is the surface Jacobian at time $T = t^n$, and ρ_α, $\alpha = 0, \ldots, N$ are the GLL quadrature weights. We then define the error

$$E_2 = |S_n - S_e|, \tag{10}$$

where S_e is the length of our exact interface.

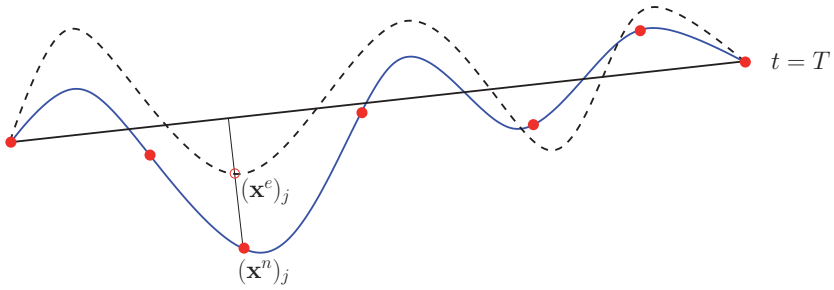


Fig. 7. The solid line corresponds to our numerically computed interface at time $T > 0$, while the dashed line represents the corresponding exact interface.

4.2 Convergence tests

In order to more clearly see the strengths and the weaknesses of the different approaches, we will first consider three examples where the velocity field is chosen such that the front keeps it shape. For these three tests, we will compare four different approaches:
(i) the new proposed method using Strategy 1 (see Figure 4);
(ii) the new proposed method using Strategy 2 (see Figure 5);
(iii) enforcing the kinematic condition as well a zero tangential grid velocity (denoted as the "Normal" method in the following);
(iv) a pure Lagrangian approach.
The fourth and last test will involve a more complex interface and a more complex velocity field; in this test we will only compare (i) and (iv).

4.3 Test 1

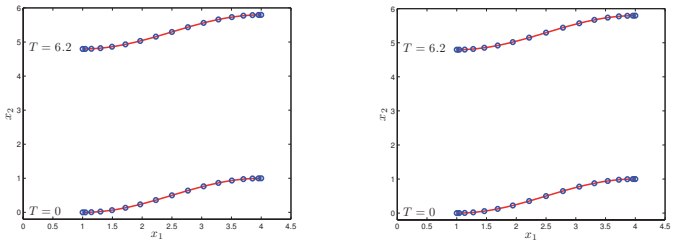
In the first example, the interface motion is only in the x_2-direction. The initial front given by

$$x_2(x_1) = \frac{1}{2}\left(1 - \cos\left(\frac{\pi(x_1 - 1)}{3}\right)\right), \qquad 1 \leq x_1 \leq 4,$$

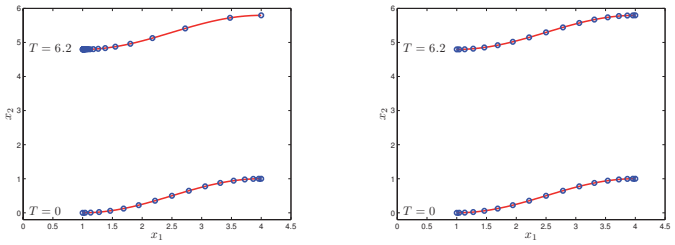
while the prescribed velocity field is given as

$$\begin{aligned} u_1(x_1, t) &= 0, \\ u_2(x_1, t) &= -\frac{1}{2} + \frac{1}{2}e^{t/4}(1 + \cos(\pi t)). \end{aligned}$$

Thus, each particle on the initial front will only move in the x_2-direction. Note that, for the Normal approach, we compute the normals *analytically* (which we can do since we know the analytical expression of the front at all times), and not numerically. A numerical computation of the normals will lead to the Normal algorithm breaking down due to the bad interpolation properties when the point distribution becomes poor. Figure 8 shows the initial point distribution and the point distribution at the final time $T = 6.2$ for the four different methods. In Figures 9 and 10 we report the errors E_1 and E_2, respectively. We observe that all four methods perform well in terms of "hitting" the front. However, for the Normal approach, the point distribution is poor at $T = 6.2$ due to the shape of the front; this again leads to a poor approximation of the length of the front. For the three other methods, the error E_2 reaches machine precision since u_2 does not depend on x_1, and we use a sufficiently large polynomial degree, N.

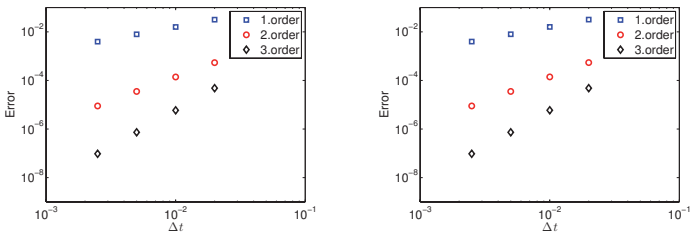


(a) Strategy 1 (b) Strategy 2

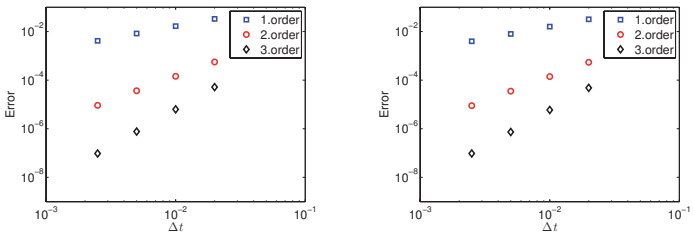


(c) Normal (d) Lagrangian

Fig. 8. The interface and the point distribution at the initial time $t = 0$ and at the final time $T = 6.2$ for Test 1 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.

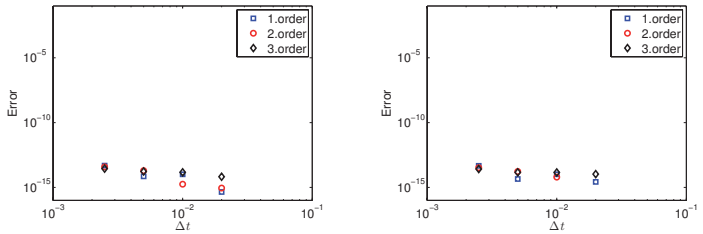


(a) Strategy 1 (b) Strategy 2



(c) Normal (d) Lagrangian

Fig. 9. The error E_1 at time $T = 6.2$ for Test 1 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.



(a) Strategy 1 (b) Strategy 2

(c) Normal (d) Lagrangian

Fig. 10. The error E_2 at time $T = 6.2$ for Test 1 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation. The computed value for the length of the interface, S_n, does not converge for the Normal method due to an incorrect point distribution.

4.4 Test 2

Next, we will consider a trivial interface, but a velocity field with a more substantial tangential contribution. The initial interface is given by

$$x_2(x_1) = 0, \qquad 1 \leq x_1 \leq 4,$$

while the prescribed velocity field is

$$u_1(x_1, t) = \frac{2}{5} \sin\left(\frac{\pi(x_1 - 1)}{3}\right),$$
$$u_2(x_1, t) = -\frac{1}{2} + \frac{1}{2} e^{t/4}(1 + \cos(\pi t)).$$

Hence, the front has no curvature, and all the normals point in the x_2-direction. Thus, $u_1(x, t)$ corresponds to a pure tangential component. Figure 11 shows the initial point distribution and the point distribution at the final time $T = 6.2$ for the four different methods. In Figures 12 we report the error E_1. In this example, Strategy 1 and Strategy 2 give identical results. We get first, second and third order temporal convergence in capturing the interface for all four methods.

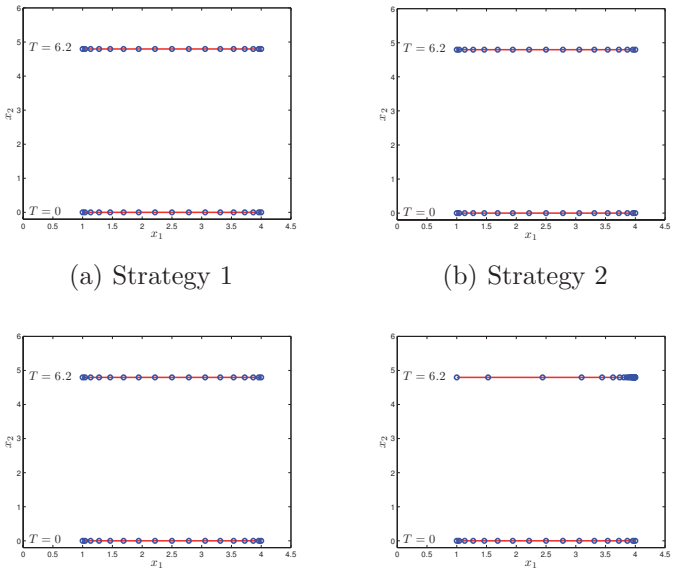
In Figure 11 we see that the pure Lagrangian approach leads to a poor point distribution. However, this is not reflected in the computation of the length due to the simplicity of the front. For this simple interface, (9) reduces to

$$S_n = \sum_{\alpha=0}^{N} \rho_\alpha \left(\frac{\partial (x_1)_N}{\partial \xi}\right)_\alpha = \int_{-1}^{1} \frac{\partial (x_1)_N}{\partial \xi} \, \mathrm{d}\xi,$$

since $\frac{\partial (x_1)_N}{\partial \xi}$ is an $(N-1)$'th degree polynomial which is integrated exactly using GLL quadrature. In addition, since

$$\int_{-1}^{1} \frac{\partial (x_1)_N}{\partial \xi} \, \mathrm{d}\xi = (x_1)_N(1) - (x_1)_N(-1),$$

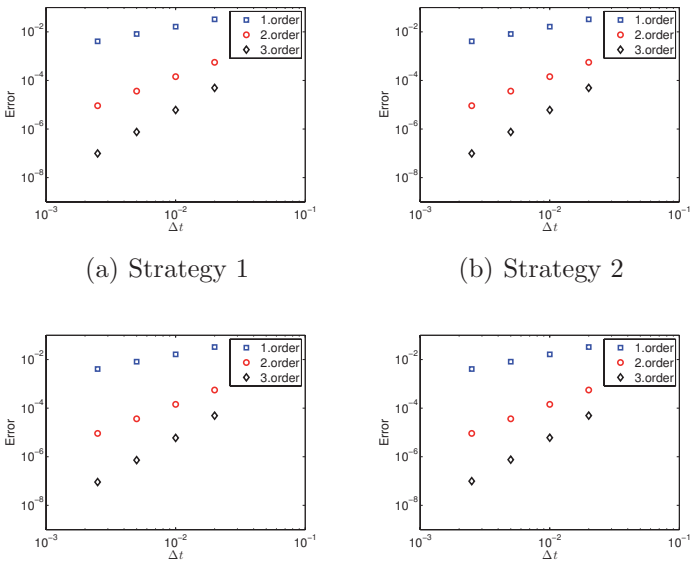
and since the end points are the same for all strategies, we achieve machine precision for all four methods regardless of the quality of the grid. It should be emphasized that this is, indeed, a very special case.



(a) Strategy 1 (b) Strategy 2

(c) Normal (d) Lagrangian

Fig. 11. The interface and the point distribution at the initial time $t = 0$ and at the final time $T = 6.2$ for Test 2 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.



(a) Strategy 1 (b) Strategy 2

(c) Normal (d) Lagrangian

Fig. 12. The error E_1 at time $T = 6.2$ for Test 2 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.

4.5 Test 3

We now consider a combination of the previous two tests. The initial front is the same as in Test 1, but we prescribe a velocity field with non-zero components in both the x_1- and the x_2-direction. In particular, we choose the same velocity field as in Test 1, but with the modification that we also add another tangential component. Hence, the interface will still keep its shape during the simulation. Our initial interface is then given by

$$x_2(x_1) = \frac{1}{2}\left(1 - \cos\left(\frac{\pi(x_1 - 1)}{3}\right)\right), \qquad 1 \leq x_1 \leq 4,$$

while the prescribed velocity field is given as

$$\begin{aligned} u_1(x_1, t) &= u_1^t, \\ u_2(x_1, t) &= -\frac{1}{2} + \frac{1}{2}e^{t/4}(1 + \cos(\pi t)) + u_2^t, \end{aligned}$$

with

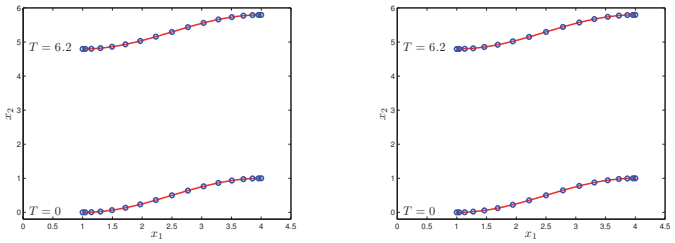
$$\begin{aligned} u_1^t &= \frac{2}{5}\sin\left(\frac{\pi(x_1 - 1)}{3}\right), \\ u_2^t &= \frac{\pi}{6}\sin\left(\frac{\pi(x_1 - 1)}{3}\right)\frac{2}{5}\sin\left(\frac{\pi(x_1 - 1)}{3}\right). \end{aligned}$$

Here, $\mathbf{u}^t = u_1^t\mathbf{e}_1 + u_2^t\mathbf{e}_2$ is a vector which points in the tangential direction of the interface. Figure 13 shows the initial point distribution and the point distribution at the final time $T = 6.2$ for the four different methods.

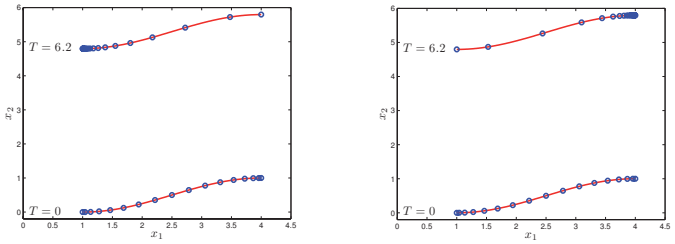
In Figures 14 and 15 we report the errors E_1 and E_2, respectively. These results are in agreement with the two previous numerical experiments. We observe that all four methods perform well in terms of "hitting" the front.

Strategy 1 and 2 still perform well with respect to both error measures. The Normal approach gives the same results as for Test 1 since the only difference with this example is the addition of a tangential velocity component. The grid quality for a pure Lagrangian approach is poor, which is what we would expect.

Another thing we observe is that the error level for E_2 (the length computation) is about a factor $10^2 - 10^3$ smaller than the error level for E_1 (the interface error). The reason for this is that the interface error is rather uniform, which again is due to the simplicity of the problem. This makes the error in the spatial *derivative* of the interface substantially smaller than the interface error, which again leads to a better approximation of the length of the front.

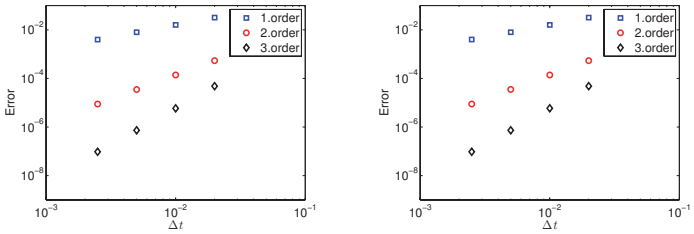


(a) Strategy 1 (b) Strategy 2

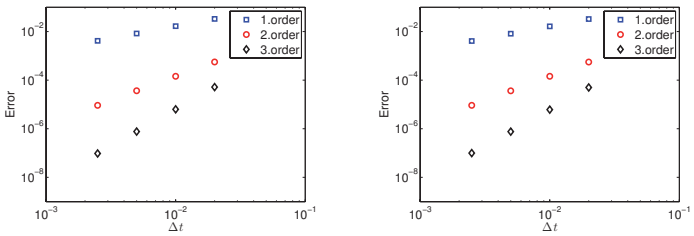


(c) Normal (d) Lagrangian

Fig. 13. The interface and the point distribution at the initial time $t = 0$ and at the final time $T = 6.2$ for Test 3 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.

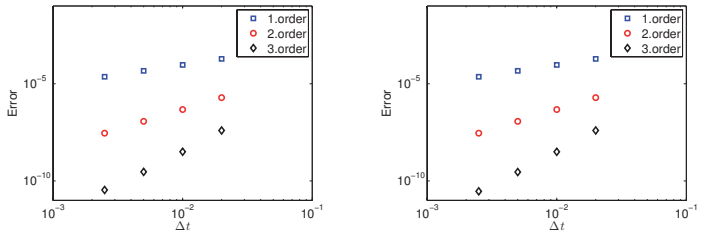


(a) Strategy 1 (b) Strategy 2



(c) Normal (d) Lagrangian

Fig. 14. The error E_1 at time $T = 6.2$ for Test 3 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation.



(a) Strategy 1 (b) Strategy 2

(c) Normal (d) Lagrangian

Fig. 15. The error E_2 at time $T = 6.2$ for Test 3 using the four different strategies. A polynomial degree $N = 16$ is used for the spectral approximation. The computed value for the length of the interface, S_n, does not converge for the Normal method and for the Lagrangian method due to an incorrect point distribution.

4.6 Test 4

The three previous test cases where all constructed to illuminate some of the strengths and weaknesses of the different methods for tracking an interface; for this reason they were chosen to be rather simple. We now consider a more complicated numerical example in order to demonstrate the applicability of the new strategy to solve more general problems. We still choose a velocity field which depends on the initial shape of the front, and in such a way that we are able to derive an analytical expression for the shape of the front at all times. A major difference from the previous test cases is that we now choose a time dependent front. In particular, we demand that the *shape* of the front is given by

$$x_2(x_1,t) = \frac{1}{4}\cos(\pi t)\left(\cos\left(\frac{\pi(x_1-1)}{3+0.4t}\right) - \cos\left(\frac{3\pi(x_1-1)}{3+0.4t}\right)\right). \tag{11}$$

Hence, the front will have a time-dependent amplitude and a time-dependent wavelength. We also wish to rotate the front in a circular motion, and in order to achieve this, the velocity field must be chosen in a careful manner. In particular, it consists of four contributions:

- an angular velocity which is responsible for a pure rotation of the initial front. The angle is computed with respect to a circle with center $(-4, 0)$, and a constant angular velocity $u_\theta = 0.1$ is imposed;
- a velocity field which accounts for the time-dependent amplitude in (11);
- a velocity field which "stretches" the front in accordance with the time-dependent wavelength in (11);
- an additional velocity field which points in the tangential direction on the front.

By adding these four contributions, we obtain a smooth, two-dimensional, time-dependent velocity field. Apart from spatial and temporal discretization errors, the initial front $x_2(x_1, t = 0)$ will keep the *shape* given by (11) when "immersed" in our velocity field; see Figure 16.

In Figure 17 we show the initial point distribution and the point distribution at the final time $T = 5$ using Strategy 1 and a Lagrangian approach. In Figures 18 and 19 we report the errors E_1 and E_2 for the two methods, respectively. We see that Strategy 1 performs well both in terms of "capturing" the front and in terms of giving the correct length. The Lagrangian approach is also able to "capture" the front, but the point distribution is poor such that the error in the length of the front is large. Also, compared with Test 3, the difference between the error levels for E_1 and E_2 is now much smaller; this is due to the time-dependent amplitude in (11), which makes the interface error much less uniform.

5 Conclusions

We have presented a new approach for tracking an interface immersed in a given velocity field. The method is particularly relevant to the simulation of unsteady free surface problems using the arbitrary Lagrangian-Eulerian framework. The new method has been constructed with two goals in mind: (i) to be able to accurately follow the interface; and (ii) to maintain a good point distribution for the grid points along the interface. The method combines information from a pure Lagrangian approach with information from an ALE approach. No interpolation of data is needed; instead, we have been able to obtain the relevant information by solving several pure convection problems along the interface. In this respect, the new approach represents a semi-Lagrangian method applied to surface information.

We have been able to construct two-dimensional model problems offering analytical expressions for both the interface as well as the prescribed velocity field in which the interface (or front) is "immersed". This has allowed us to verify and compare the temporal accuracy of different methods: the new approach, a pure Lagrangian approach, and an approach honoring the kinematic condition in the normal direction, but imposing a homogeneous Dirichlet condition for the tangential component of the grid velocity (called the Normal approach).

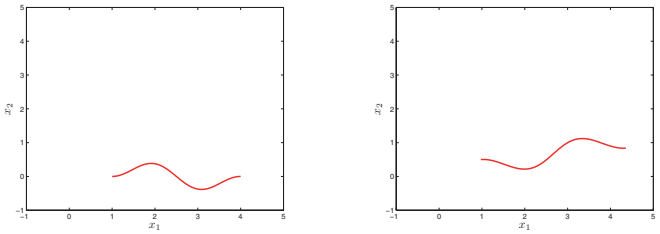
Using the new approach we have been able to achieve both of our primary objectives; in particular, we have verified first, second, and third order temporal accuracy for all four model problems. The new method is particularly important in the context of using high order spatial discretization schemes.

Both the Lagrangian approach and the Normal approach generally give a non-optimal point distribution along the interface, something which again may result in large errors in the computation of important surface quantities (e.g., normal and tangent vectors, local curvature, length etc). Such errors may, in worst case, result in a complete breakdown of the interface tracking.

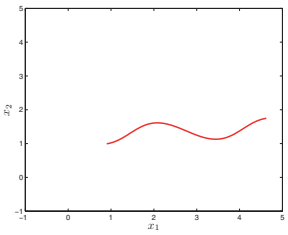
The new method should be extended to, and tested in, more general situations, in particular, by solving real free surface problems using an ALE approach, and by extending the approach to three dimensions.

Acknowledgment

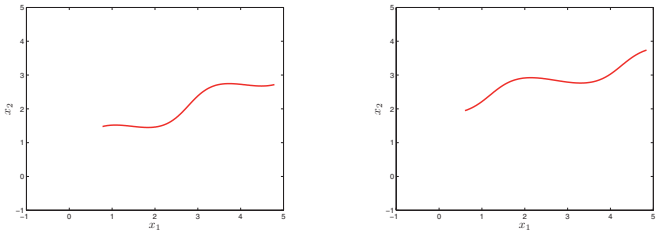
The authors would like to thank the Norwegian University of Science and Technology for the financial support of this project.



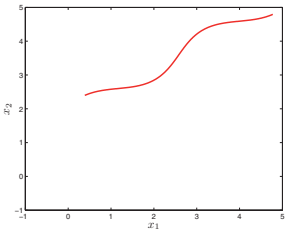
(a) $T = 0$ (b) $T = 1$



(c) $T = 2$

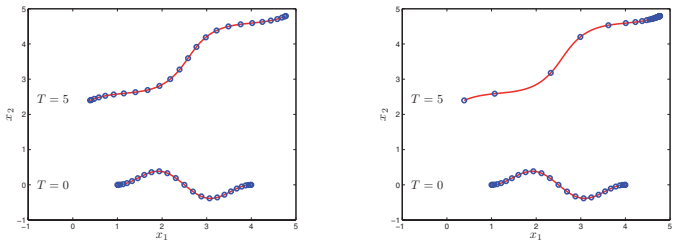


(d) $T = 3$ (e) $T = 4$



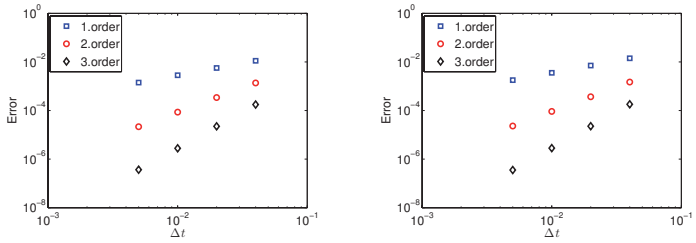
(f) $T = 5$

Fig. 16. The front $x_2(x_1, t)$ in (11) at different times when "immersed" in the prescribed velocity field.



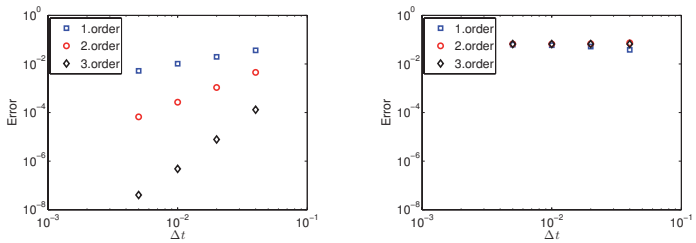
(a) Strategy 1 (b) Lagrangian

Fig. 17. The interface and the point distribution at the initial time $t = 0$ and at the final time $T = 5$ for Test 4 using Strategy 1 and the Lagrangian approach. A polynomial degree $N = 24$ is used for the spectral approximation.



(a) Strategy 1 (b) Lagrangian

Fig. 18. The error E_1 at time $T = 5$ for Test 4 using Strategy 1 and the Lagrangian approach. A polynomial degree $N = 24$ is used for the spectral approximation.



(a) Strategy 1 (b) Lagrangian

Fig. 19. The error E_2 at time $T = 5$ for Test 4 using Strategy 1 and the Lagrangian approach. A polynomial degree $N = 24$ is used for the spectral approximation. The computed value for the length of the interface, S_n, does not converge for the Lagrangian method due to an incorrect point distribution.

References

[1] R. Bouffanais, M. Deville., Mesh update techniques for free-surface flow solvers using spectral elements, Journal of Scientific Computing 27 (1-3) (2006) 137–149.

[2] W. Dettmer, D. Perić., A computational framework for free surface fluid flows accounting for surface tension, Computer Methods in Applied Mechanics and Engineering 195 (2006) 3038–3071.

[3] J. Donea S. Giuliani, J. Halleux., An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions, Computer Methods in Applied Mechanics and Engineering 33 (1982) 689–723.

[4] F. Duarte, R. Gormaz, S. Natesan., Arbitrary Lagrangian-Eulerian method for Navier-Stokes equations with moving boundaries, Computer Methods in Applied Mechanics and Engineering 193 (2004) 4819–4836.

[5] F. Giraldo., The Lagrange-Galerkin Spectral Element Method on Unstructured Quadrilateral Grids, Journal of Computational Physics 147 (1998) 114–146.

[6] C. Hirt, A. Amsden, J. Cook., An arbitrary Lagrangian-Eulerian computing method for all flow speeds, Journal of Computational Physics 14 (1974) 227–253.

[7] C. W. Hirt, B. Nichols., Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries, Journal of Computational Physics 39 (1981) 201–225.

[8] L. Ho, A. Patera., A Legendre spectral element method for simulation of unsteady incompressible viscous free-surface flows, Computer Methods in Applied Mechanics and Engineering 80 (1990) 355–366.

[9] L. Ho, A. Patera., Variational formulation of three-dimensional viscous free-surface flows: Natural imposition of surface tension boundary conditions, International Journal for Numerical Methods in Fluids 13 (1991) 691–698.

[10] A. Huerta, A. Rodríguez-Ferran (eds.)., The Arbitrary Lagrangian-Eulerian Formulation, Computer Methods in Applied Mechanics and Engineering 193 (39-41) (2004) 4073–4456.

[11] A. Johnson, T. Tezduyar., Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces, Computer Methods in Applied Mechanics and Engineering 119 (1994) 73–94.

[12] Y. Maday, A. Patera., Spectral element methods for the Navier-Stokes equations, in: A.K. Noor, J. T. Oden (Eds.), State of the Art Surveys in Computational Mechanics, ASME, New York (1989) 71–143.

[13] Y. Maday, A. Patera, E. Rønquist., An Operator-Integration-Factor Splitting Method for Time-Dependent Problems: Application to Incompressible Fluid Flow, Journal of Scientific Computing 5 (4) (1990) 263–292.

[14] S. Osher, R. Fedkiw., Level Set Methods and Dynamic Implicit Surfaces, Springer-Verlag, 2002.

[15] S. Osher, J. Sethian., Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations, Journal of Computational Physics 79 (1988) 12–49.

[16] B. Ramaswamy, M. Kawahara., Arbitraty Lagrangian-Eulerian finite element method for unsteady convective incompressible viscous free surface flow, International Journal for Numerical Methods in Fluids 7 (1987) 1053–1074.

[17] J. Sethian., Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, 1999.

[18] M. Walkey, P. Gaskell, P. Jimack, M. Kelmanson, J. Summers., Finite Element Simulation of Three-Dimensional Free-Surface Flow Problems, Journal of Scientific Computing 24 (2) (2005) 147–162.

[19] D. Xiu, G. Karniadakis., A Semi-Lagrangian High-Order Method for Navier-Stokes Equations, Journal of Computational Physics 172 (2001) 658–684.

[20] J. Xu, D. Xiu, G. Karniadakis., A Semi-Lagrangian Method for Turbulence Simulations Using Mixed Spectral Discretizations, Journal of Scientific Computing 17 (1-4) (2002) 585–597.

[21] H. Zhou, J. Derby., An assessment of a parallel, finite element method for three-dimensional, moving-boundary flows driven by capillarity for simulation of viscous sintering, International Journal for Numerical Methods in Fluids 36 (2001) 841–865.

Paper VI

Fast tensor-product solvers. Part I: Partially deformed three-dimensional domains

Tormod Bjøntegaard, Yvon Maday and Einar M. Rønquist.

Submitted to *Journal of Scientific Computing*.

Fast tensor-product solvers.

Part I: Partially deformed three-dimensional domains

Tormod Bjøntegaard, Yvon Maday, and Einar M. Rønquist

April 30, 2008

Abstract

We consider the numerical solution of partial differential equations in partially deformed three-dimensional domains in the sense that a *general* two-dimensional cross section in the xy-plane is invariant with respect to the z-direction. Earlier work has exploited such geometries by approximating the solution as a truncated Fourier series in the z-direction. In this paper we propose a new solution algorithm which also exploits the tensor-product feature between the xy-plane and the z-direction. However, the new algorithm is not limited to periodic boundary conditions, but works for general Dirichlet and Neumann type of boundary conditions. The proposed algorithm also works for problems with variable coefficients as long as these can be expressed as a separable function with respect to the variation in the xy-plane and the variation in the z-direction. For most problems where the new method is applicable, the computational cost is better or at least as good as the best iterative solvers. The new algorithm is easy to implement, and useful, both in a serial and parallel context. Numerical results are presented for three-dimensional Poisson and Helmholtz problems using both low order finite elements and high order spectral element discretizations.

Key words: fast solver, tensor-product, partial differential equation, parallel algorithm.

Suggested running head: Fast tensor-product solvers in space.

1 Introduction

Fast tensor-product solvers were proposed by Lynch, Rice and Thomas in 1964; see [10]. This is a very specialized class of solution methods with limited applicability, but they are very attractive to use whenever they are applicable. They have typically been used in the context of solving the system of algebraic equations resulting from discretized partial differential equations with constant coefficients in very simple computational domains (rectangular, hexahedral, or cylindrical domains). For example, the solution of the Poisson equation or the biharmonic equation in a cube can be done extremely efficiently using this approach; see [13, 14, 4, 1]. This class of solution methods have also been exploited in the context of constructing efficient preconditioners for iterative methods; see for example [5, 16].

Recently, the tensor-product approach has also been extended to solve elliptic problems associated with layered media; see [9]. However, this two- and three-dimensional direct solver still assumes a fully separable elliptic problem in order to reduce the multi-dimensional problem to a sequence of one-dimensional problems.

In this work we exploit the tensor-product feature to solve problems in partially tensor-product domains. In particular, we consider three-dimensional generalized cylinders (meaning with a possibly nonquadratic cross-section, and even including holes); see Figure 1. The two-dimensional cross-section may be discretized in an unstructured manner, or may be discretized in a structured manner but deformed so as to prevent fast tensor-product solvers to be used directly. The main idea is to exploit the tensor-product feature between the two-dimensional cross-section (the xy-plane) and the perpendicular z-direction.

We demonstrate the approach by solving the Poisson problem and the Helmholtz problem in selected three-dimensional domains. The proposed method results in a set of two-dimensional Helmholtz-type problems, one for each cross-sectional plane, which can be solved completely independently. The algorithm is therefore highly parallel. The two-dimensional systems can be solved either iteratively or using a direct method. Each individual two-dimensional system may also be parallelized, thus allowing for a two level parallelization: a parallelization with respect to all the two-dimensional planes, and a separate and independent parallelization of each individual plane.

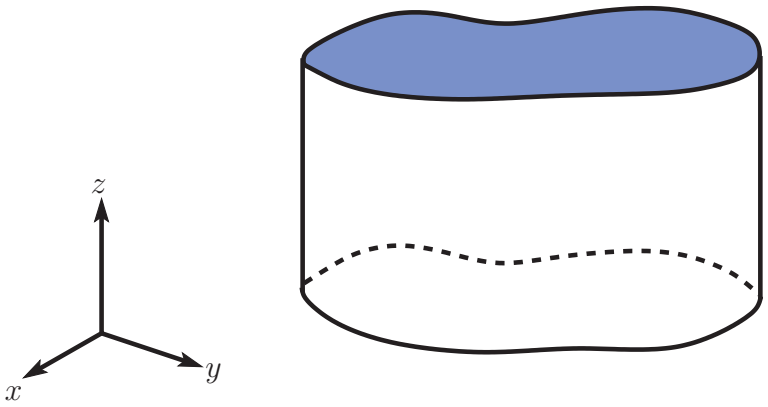


Figure 1: A two-dimensional deformed domain extended in the third direction.

2 The Poisson problem

Let Ω be a three-dimensional domain which can be considered as a "cylinder" in the sense that a two-dimensional cross section, $\mathcal{O}$, in the xy-plane is invariant with respect to the z-direction; see Figure 1. Hence, we can write

$$\Omega = \mathcal{O} \times (0, L), \tag{2.1}$$

where L is the length of the domain in the z-direction.

We consider now the Poisson problem in such a cylinder,

$$\begin{aligned} -\nabla^2 u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\mathcal{O} \times [0, L]. \end{aligned} \tag{2.2}$$

The boundary conditions on the planes $z = 0$ and $z = L$ will be discussed below.

Earlier work has exploited domains that can be expressed as (2.1). For example, the work in [3] and [7] exploit this fact in the numerical solution of the three-dimensional Navier-Stokes equations by using a spectral element discretization in the xy-plane and a truncated Fourier expansion in the z-direction. Another example is the stability analysis of three-dimensional free surface flows; see [2]. Most earlier works trying to exploit the particular structure (2.1) have been based on models incorporating periodicity in the z-direction, i.e.,

$$u(x, y, 0) = u(x, y, L).$$

The motivation for this work is to generalize this approach, in particular, with respect to prescribed boundary conditions in the z-direction, but also with respect to more general discretizations. This is done by using diagonalization techniques between the xy-plane and the z-direction.

In the following discussion, we assume homogeneous Dirichlet boundary conditions also in the z-direction. We will return to other types of boundary conditions later.

The weak formulation of the Poisson problem (2.2) is then: Find $u \in H_0^1(\Omega)$ such that

$$a(u, v) = (f, v) \quad \forall v \in H_0^1(\Omega), \tag{2.3}$$

where the bilinear form $a(u, v)$ is given as

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\Omega = \int_\Omega \left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial v}{\partial y} + \frac{\partial u}{\partial z}\frac{\partial v}{\partial z} \right) \mathrm{d}\Omega, \tag{2.4}$$

and the right hand side is given as

$$(f, v) = \int_\Omega f v \, \mathrm{d}\Omega. \tag{2.5}$$

We will assume that the given data f is square integrable over Ω.

3 Discretization

We now discuss the numerical solution of (2.3) based on the weak formulation. The particular structure of our linear differential operator and of our domain Ω allows us to express the discrete solution space X_N as

$$X_N = \mathrm{span}\,\{\phi_m(x, y)\psi_n(z)\}, \quad m = 1, \ldots, N_2, \quad n = 1, \ldots, N_1. \tag{3.1}$$

Each basis function can thus be expressed as a product of a two-dimensional function, ϕ_m, defined over $\mathcal{O}$ and a one-dimensional function, ψ_n, defined on the interval $(0, L)$. We note that, as the resolution is increased (i.e., as the discretization parameters N_2 and N_1 go to infinity), the

two-dimensional functions $\{\phi_m(x,y)\}_{m=1}^{\infty}$ will span $H_0^1(\mathcal{O})$, while the one-dimensional functions $\{\psi_n(z)\}_{n=1}^{\infty}$ will span $H_0^1((0,L))$.

The discrete problem can be stated as: Find $u_N \in X_N$ such that

$$a(u_N, v) = (f, v) \quad \forall v \in X_N. \tag{3.2}$$

The discrete solution can be expressed in tensor product form as

$$u_N(x,y,z) = \sum_{m=1}^{N_2} \sum_{n=1}^{N_1} u_{mn}\, \phi_m(x,y)\psi_n(z), \tag{3.3}$$

where u_{mn} are the basis coefficients. Note that each basis function is a separable function while u_N is not.

Inserting (3.3) into (3.2) and choosing test functions $v(x,y,z) = \phi_i(x,y)\psi_j(z)$, we get

$$\sum_{m=1}^{N_2} \sum_{n=1}^{N_1} \left[(\int_{\mathcal{O}} \widetilde{\nabla}\phi_i \cdot \widetilde{\nabla}\phi_m \, dx\, dy)(\int_0^L \psi_j \psi_n dz) + (\int_{\mathcal{O}} \phi_i \phi_m \, dx\, dy)(\int_0^L \psi_j' \psi_n' dz) \right] u_{mn} = g_{ij}. \tag{3.4}$$

Here, $\widetilde{\nabla}$ denotes the gradient operator in two space dimensions (in the xy-plane), and

$$g_{ij} = \int_0^L \left(\int_{\mathcal{O}} f(x,y,z)\phi_i(x,y)\, dx\, dy \right) \psi_j(z)\, dz. \tag{3.5}$$

In the case of Dirichlet boundary conditions in the z-direction, $\psi_n(0) = \psi_n(L) = 0$ for all $n = 1, \ldots, N_1$. A convenient choice of functions $\psi_n(z)$ are those functions that satisfy

$$\int_0^L \psi_j' \psi_n' dz = \lambda_n \int_0^L \psi_j \psi_n dz, \qquad \begin{array}{l} \forall j = 1, \ldots, N_1, \\ \forall n = 1, \ldots, N_1. \end{array} \tag{3.6}$$

where λ_n is a constant. Inserting (3.6) into (3.4), we arrive at

$$\sum_{m=1}^{N_2} \sum_{n=1}^{N_1} \left[\left(\int_{\mathcal{O}} \left(\widetilde{\nabla}\phi_i \cdot \widetilde{\nabla}\phi_m + \lambda_n \phi_i \phi_m \right) dx\, dy \right) \left(\int_0^L \psi_j \psi_n dz \right) \right] u_{mn} = g_{ij}. \tag{3.7}$$

On a continuous level, (3.6) is equivalent to choosing $\psi_n(z)$ to be the solution of the symmetric

eigenvalue problem

$$-\psi_n'' = \lambda_n \psi_n, \quad \psi_n(0) = \psi_n(L) = 0,$$

which is explicitly given as

$$\psi_n(z) = \sqrt{2/L}\,\sin(n\pi z/L), \tag{3.8}$$

$$\lambda_n = n^2\pi^2/L^2. \tag{3.9}$$

Note also that these eigenfunctions $\psi_n(z)$ are orthonormal,

$$\int_0^L \psi_j \psi_n dz = \delta_{jn}, \quad \forall j, n. \tag{3.10}$$

Using (3.10) with (3.7), we arrive at the systems

$$\sum_{m=1}^{N_2} \left[\int_{\mathcal{O}} \left(\widetilde{\nabla}\phi_i \cdot \widetilde{\nabla}\phi_m + \lambda_j \phi_i \phi_m \right) dx\, dy \right] u_{mj} = g_{ij}, \qquad \begin{array}{l} \forall i = 1, \ldots, N_2, \\ \forall j = 1, \ldots, N_1. \end{array} \tag{3.11}$$

We see that this approach transforms the solution of a single three-dimensional problem into the solution of N_1 two-dimensional Helmholtz-type systems.

However, other choices for $\psi_n(z)$ are also possible. For example, in the low order finite element case, we can choose the $\psi_n(z)$ to be the eigenvectors of the one-dimensional stiffness matrix with respect to the one-dimensional mass matrix; these eigenvectors are then spanned by the usual one-dimensional "hat" basis functions. Similarly, in the high order spectral (element) case, we can choose the functions $\psi_n(z)$ to be the eigenvectors of the stiffness matrix with respect to the one-dimensional mass matrix, however, now the eigenvectors would be spanned by (piecewise) high order polynomials which vanish at the end points $z = 0$ and $z = L$.

We remark that, for both the low order finite element case and the high order spectral (element) case, the discrete eigenvalues λ_n and the corresponding discrete eigenfunctions ψ_n are different from the analytical expressions (3.9) and (3.8). An expansion in the z-direction based on the analytical eigenfunctions (3.8) represents an approach similar to the Fourier expansion for periodic problems. The disadvantage with this approach is that we need to find the explicit form

for the analytical eigenfunctions (and eigenvalues) for each specific type of boundary conditions specified at $z = 0$ and $z = L$, and for each type of operator. This may be cumbersome or difficult, and the implementation will be different for each specific case. In the following, we will therefore focus on a finite element or spectral (element) discretization in the z-direction, thus allowing for a more general and flexible setting.

3.1 Algebraic system of equations

Let us now proceed with the details for arbitrary finite element discretizations, both low order and high order discretizations. We let $\phi_m(x, y)$ be any two-dimensional basis function, and let $\psi_n(z)$ be any one-dimensional basis function. The discrete space X_N and the discrete solution u_N are then given by (3.1) and (3.3), respectively. If both ϕ_m and ψ_n are nodal basis functions, the unknown basis coefficients u_{mn} will be the numerical approximation at the nodal points.

It now follows directly from (3.4) that we can express the system of algebraic equations as

$$\sum_{m=1}^{N_2}\sum_{n=1}^{N_1}(A_{im}^{2D}B_{jn}^{1D} + B_{im}^{2D}A_{jn}^{1D})\,u_{mn} = g_{ij}, \qquad \begin{array}{l}\forall i = 1,\ldots,N_2,\\ \forall j = 1,\ldots,N_1.\end{array} \tag{3.12}$$

where

$$\begin{aligned} A_{im}^{2D} &= \int_{\mathcal{O}} \widetilde{\nabla}\phi_i \cdot \widetilde{\nabla}\phi_m \, dx\, dy, \\ B_{im}^{2D} &= \int_{\mathcal{O}} \phi_i\phi_m \, dx\, dy, \end{aligned}$$

are the elements in the two-dimensional stiffness matrix and mass matrix, respectively, and

$$\begin{aligned} A_{jn}^{1D} &= \int_0^L \psi_j'\psi_n' \, dz, \\ B_{jn}^{1D} &= \int_0^L \psi_j\psi_n \, dz, \end{aligned}$$

are the elements in the one-dimensional stiffness matrix and mass matrix associated with the z-direction. The right hand side elements g_{ij} in (3.12) are given in (3.5).

In matrix form, we can write (3.12) succinctly as

$$(B^{1D} \otimes A^{2D} + A^{1D} \otimes B^{2D})\, \underline{u} = \underline{g}, \tag{3.13}$$

where we have used standard tensor product notation. Note that $\underline{u}$ is a vector of length $N_2 N_1$, and that the unknowns are numbered in such a way that all the nodes in a fixed xy-plane are numbered before proceeding to the next plane. A similar numbering scheme is used for the right hand side $\underline{g}$.

3.2 Diagonalization

We now consider fast tensor-product techniques to solve the system (3.13). To this end, we introduce the generalized eigenvalue problem

$$A^{1D} \underline{q}_j = \lambda_j\, B^{1D} \underline{q}_j, \qquad j = 1, \ldots, N_1,$$

or

$$A^{1D} Q = B^{1D} Q \Lambda. \tag{3.14}$$

Here, $\underline{q}_j$ is an eigenvector of the one-dimensional discrete Laplace operator A^{1D} with respect to the one-dimensional mass matrix B^{1D}, and λ_j is the corresponding (real and positive) eigenvalue. The columns of Q contain all the eigenvectors, while Λ is a diagonal matrix containing the eigenvalues along the diagonal.

Normalizing the eigenvectors with respect to B^{1D} we get

$$B^{1D} = Q^{-T} Q^{-1}, \tag{3.15}$$

$$A^{1D} = Q^{-T} \Lambda\, Q^{-1}. \tag{3.16}$$

From (3.13) and (3.15)-(3.16) it follows that

$$\begin{aligned} B^{1D} \otimes A^{2D} + A^{1D} \otimes B^{2D} &= Q^{-T} Q^{-1} \otimes A^{2D} + Q^{-T} \Lambda \, Q^{-1} \otimes B^{2D}, \\ &= (Q^{-T} \otimes I^{2D})(I^{1D} \otimes A^{2D} + \Lambda \otimes B^{2D})(Q^{-1} \otimes I^{2D}), \end{aligned}$$

where I^{2D} and I^{1D} are identity matrices of the same dimension as the number of degrees-of-freedom in each xy-plane and in the z-direction, respectively.

By introducing the variables

$$\begin{aligned} \underline{\tilde{u}} &= (Q^{-1} \otimes I^{2D}) \, \underline{u}, \\ \underline{\tilde{g}} &= (Q^{T} \otimes I^{2D}) \, \underline{g}, \end{aligned}$$

the algebraic problem can be written as

$$(I^{1D} \otimes A^{2D} + \Lambda \otimes B^{2D}) \, \underline{\tilde{u}} = \underline{\tilde{g}}.$$

In "semi-local" form (global numbering in the xy-plane, but local numbering for the combined xy-plane and the z-direction), this reduces to

$$\sum_{m=1}^{N_2} \sum_{n=1}^{N_1} \big[(A^{2D})_{im} \underbrace{(I^{1D})_{jn}}_{\delta_{jn}} + (B^{2D})_{im} \underbrace{(\Lambda)_{jn}}_{\lambda_j \delta_{jn}} \big] \tilde{u}_{mn} = \tilde{g}_{ij}, \qquad \begin{array}{l} i = 1, \ldots, N_2, \\ j = 1, \ldots, N_1. \end{array} \tag{3.17}$$

or

$$\sum_{m=1}^{N_2} \big[(A^{2D})_{im} + \lambda_j (B^{2D})_{im} \big] \tilde{u}_{mj} = \tilde{g}_{ij}, \qquad \begin{array}{l} i = 1, \ldots, N_2, \\ j = 1, \ldots, N_1. \end{array} \tag{3.18}$$

Here, the first index runs over the entire xy-plane, while the second index corresponds to the z-direction.

Thus, the total algorithm comprises three steps.

The first step is to transform the right hand side:

$$\tilde{g}_{ij} = \sum_{m=1}^{N_2} \sum_{n=1}^{N_1} \underbrace{(I^{2D})_{im}}_{\delta_{im}} Q^T_{jn} g_{mn} \tag{3.19}$$

$$= \sum_{n=1}^{N_1} Q^T_{jn} g_{in} = \sum_{n=1}^{N_1} g_{in} Q_{nj}, \qquad \begin{array}{l} i = 1, \ldots, N_2, \\ j = 1, \ldots, N_1. \end{array} \tag{3.20}$$

or, in matrix form,

$$\widetilde{G} \;=\; G\,Q. \tag{3.21}$$

Here, G and $\widetilde{G}$ are the given data and transformed data, respectively, in a "semi-local" data representation.

The second step is to solve the systems (3.18). Each system (for a fixed value of j) couples the degrees-of-freedom within a single xy-plane. Using a global data representation for the unknowns within each plane, the systems (3.18) can also be expressed as:

$$(A^{2D} + \lambda_j\, B^{2D})\, \underline{\tilde{u}}_j = \underline{\tilde{g}}_j, \quad j = 1, \ldots, N_1. \tag{3.22}$$

Each two-dimensional solution $\underline{\tilde{u}}_j$ forms a column in the two-dimensional solution matrix $\widetilde{U}$ following a semi-local data representation. Note that these systems represent completely decoupled two-dimensional systems which can be solved independently of each other.

The third step involves a transformation of $\widetilde{U}$ to the final solution U. Again, using a "semi-local" data representation, we get

$$u_{ij} = \sum_{n=1}^{N_1} \tilde{u}_{in} Q_{jn} = \sum_{n=1}^{N_1} \tilde{u}_{in} Q^T_{nj} \tag{3.23}$$

or, in matrix form,

$$U \;=\; \widetilde{U}\, Q^T. \tag{3.24}$$

4 Numerical results

We now present numerical results using the proposed solution algorithm to solve the resulting systems of algebraic equations. All the numerical tests we report have been implemented in MATLAB®.

4.1 Finite element discretization

We first consider the solution of the Poisson problem in a wedge-shaped domain as depicted in Figure 2. The discretization is based on linear finite elements. Each two-dimensional cross section corresponds to a sector $\pi/4$ of the unit circle, which is discretized into triangles with mesh size h. The grid spacing in the z-direction is h. Hence, our three-dimensional discretization corresponds to using prismatic linear finite elements of mesh size h.

We derive the right-hand side, f, by using the exact solution

$$u(x,y,z) = \sin\left(2\pi\left(x^2+y^2\right)\right)\sin\left(x\left(y-\tan\left(\frac{\pi}{4}\right)x\right)\pi\right)\sin\left(\frac{\pi z}{L}\right).$$

The domain length in the z-direction is $L = 1$, and homogeneous Dirichlet boundary conditions are imposed along the entire domain boundary $\partial\Omega$. Figure 3 shows the convergence results. As expected, the error between the exact solution and the finite element solution decreases as $\mathcal{O}(h^2)$ as the mesh size h decreases; the error is here measured in the discrete L^2-norm. Note that the finest grid used in the convergence study uses 90 elements in the radial direction, in the angular direction, and in the z-direction. Hence, the resulting algebraic system of equations corresponds to about 340,000 degrees-of-freedom. All the two-dimensional systems are solved directly using full LU-factorization. However, even with a simple solver that does not exploit the sparsity associated with the triangulation of each plane, the solution is readily obtained using MATLAB®.

4.2 Spectral discretization

We now demonstrate the proposed method by solving the Poisson problem in a domain Ω as depicted in Figure 4. The domain is a deformed rectangle in the xy-plane which is extended in

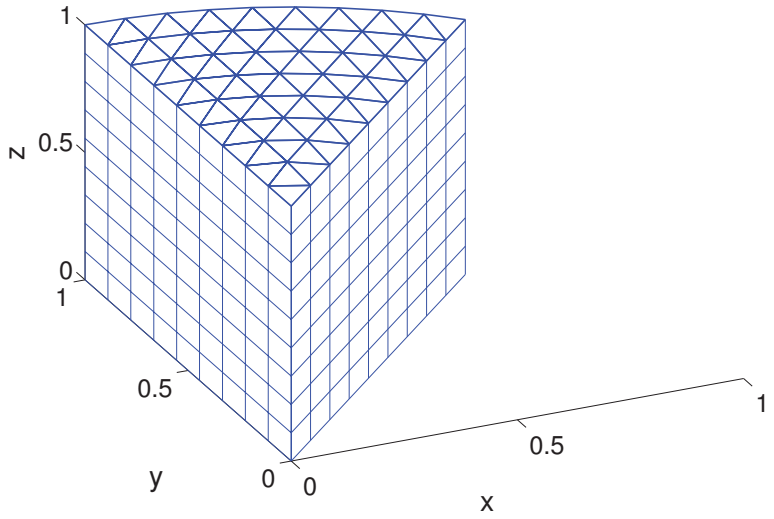


Figure 2: The finite element grid for a wedge-shaped domain.

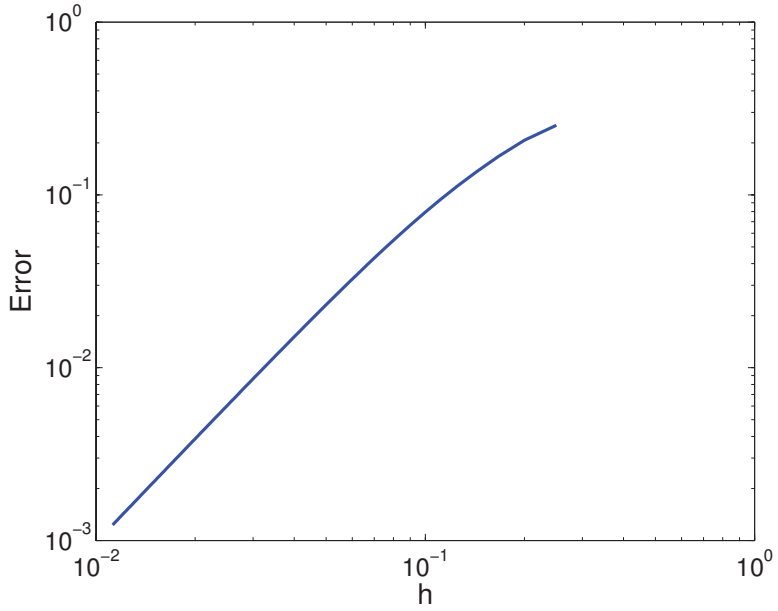


Figure 3: Relative error between the exact solution u and the numerical solution u_h of the three-dimensional Poisson problem using linear finite elements with mesh size h; the error is measured in the discrete L^2 norm. The computational grid is depicted in Figure 2.

the z-direction. The Poisson problem is discretized using a pure spectral method based on high order polynomials. Figure 4 indicates both the deformed Gauss-Lobatto Legendre grid in the xy-plane as well as a Gauss-Lobatto Legendre distribution of the nodes in the z-direction.

For the numerical experiments, we compute the right hand side, f, from the exact solution

$$u(x,y,z) = \sin\left(\frac{\pi(x - a_4\sin(\pi y))}{1 + a_2\sin(2\pi y) - a_4\sin(\pi y)}\right)\sin\left(\frac{3\pi(y + a_1\sin(2\pi x))}{1 + a_3\sin(\pi x) + a_1\sin(2\pi x)}\right)\sin\left(\frac{2\pi z}{L}\right), \tag{4.1}$$

with $a_1 = 0.08$, $a_2 = 0.10$, $a_3 = 0.12$ and $a_4 = 0.15$.

The set of algebraic equations are solved using (3.21), (3.22), and (3.24). In contrast to the finite element case, the two-dimensional problems in (3.22) are now solved iteratively using the conjugate gradient method. For the convergence result of the spectral method, we have measured the discretization error in the discrete L^2-norm. Figure 5 shows the relative error as a function of the polynomial degree, N, in each spatial direction. We see how the exponential convergence is influenced by the stopping criterion, ε, used in the conjugate gradient method for each two-dimensional problem in (3.22).

4.3 Spectral element discretization

Again, we consider the Poisson-problem with homogeneous Dirichlet boundary conditions, but the three-dimensional domain Ω is now a cylinder; see Figure 6. The two-dimensional cross section is a circle with radius equal to two. Each cross section is divided into five elements, while two layers of elements is used in the z-direction; see Figure 6.

The exact solution of the Poisson problem is

$$u(x,y,z) = \sin\left(\frac{7\pi}{4}\sqrt{(x-2)^2 + (y-2)^2} + \frac{\pi}{2}\right)\sin\left(\frac{2\pi z}{L}\right). \tag{4.2}$$

In Figure 7, we show the discretization error in the discrete L^2-error as a function of the polynomial degree, N, used to approximate the solution in each in each element. Note that the finest grid used in this convergence study corresponds to a polynomial degree $N = 29$; with ten elements, this corresponds to about 300,000 degrees-of-freedom. Similar to earlier experiments, the numerical results are obtained using MATLAB®.

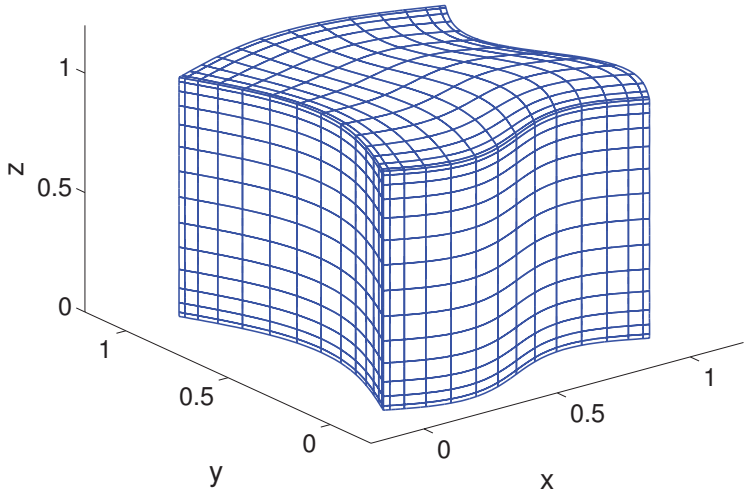


Figure 4: The domain Ω used in the numerical experiment in Section 4.2.

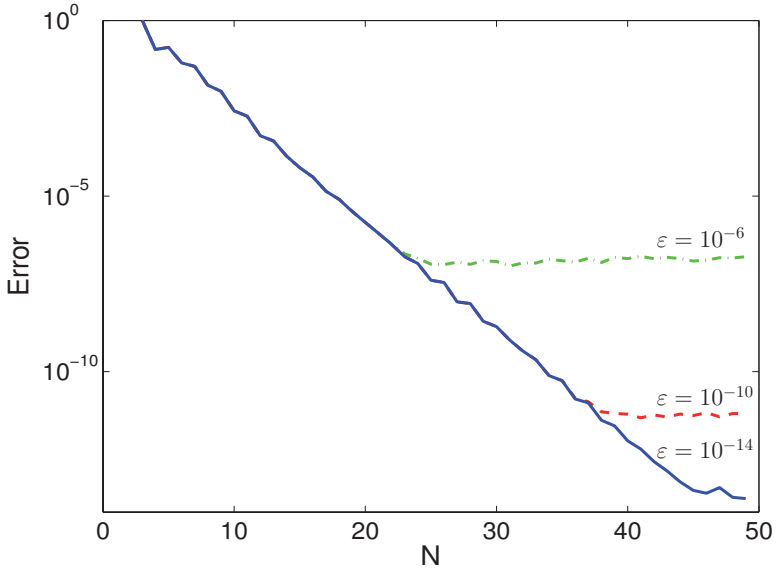


Figure 5: Relative error between the exact solution u and the numerical solution u_N of the three-dimensional Poisson problem as a function of the polynomial degree, N, in each spatial dimension. The error is measured in the discrete L^2-norm. The convergence results are shown for different choices of the stopping criterion, ε, used in the conjugate gradient iteration when solving the two-dimensional systems (3.22): $\varepsilon = 10^{-14}$, $\varepsilon = 10^{-10}$ and $\varepsilon = 10^{-6}$.

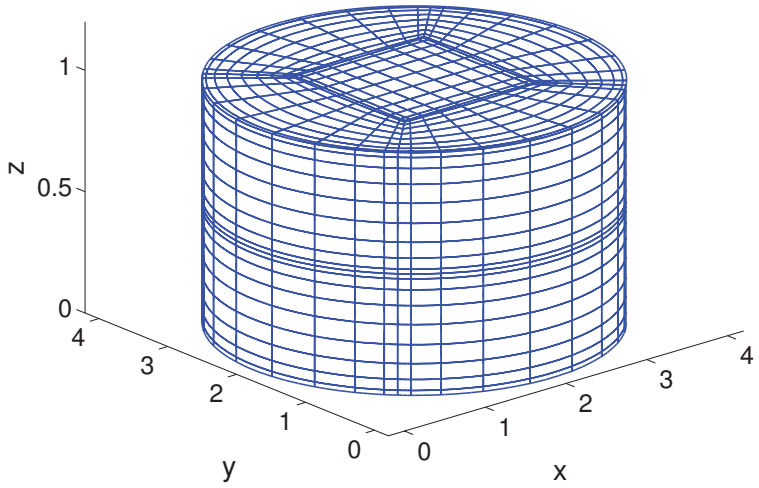


Figure 6: The computational domain, Ω, and the spectral element discretization. Two layers of elements are used in the z-direction. Each two-dimensional cross-section is divided into five spectral elements.

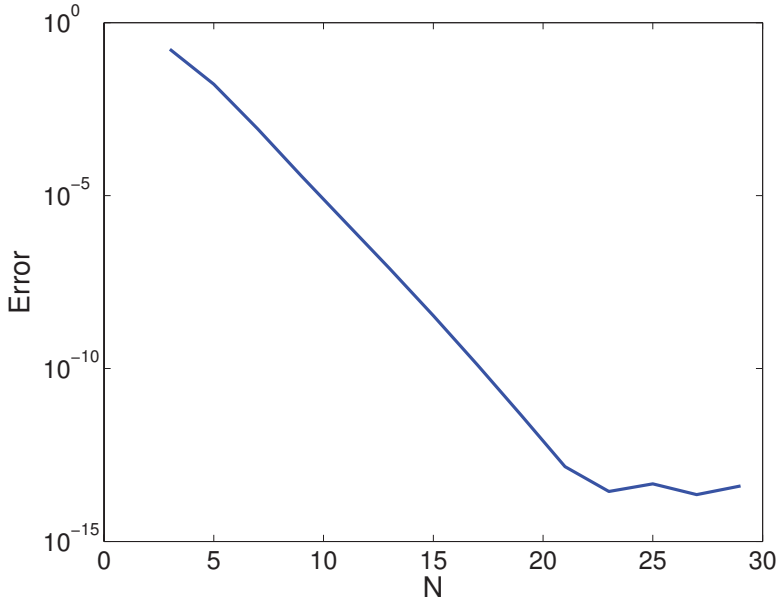


Figure 7: Relative error between the exact solution u and the numerical solution u_N of the three-dimensional Poisson problem as a function of the polynomial degree, N, in each spatial direction in each element. The error is measured in the discrete L^2-norm. The exact solution is given by (4.2). Each two-dimensional system in (3.22) is solved using conjugate gradient iteration with a tolerance $\varepsilon = 10^{-12}$.

4.4 Extension to solving the Helmholtz problem

We now extend the proposed method to solving the Helmholtz-problem

$$-\nabla^2 u + \alpha u = f \qquad \text{in } \Omega, \tag{4.3}$$

$$u = 0 \qquad \text{on } \partial\Omega, \tag{4.4}$$

where α is a positive constant. The only difference from the solution procedure for the Poisson problem is that the two-dimensional problems in (3.22) now read

$$\left(A^{2D} + (\lambda_j + \alpha)\, B^{2D}\right) \underline{\tilde{u}}_j = \underline{\tilde{g}}_j, \quad j = 1, \ldots, N_1. \tag{4.5}$$

The domain Ω and the discretization is the same as described in Section 4.2; see Figure 4. Again, we use the exact solution (4.1) to derive the right hand side, f. The parameter α is set equal to one. Figure 8 shows the convergence plot. As expected, we obtain exponential convergence similar to the Poisson problem.

4.5 Further extensions

We conclude with an example showing the extension of the proposed approach to solving problems with other types of boundary conditions and nonconstant material properties. The particular example we consider can be formulated mathematically as

$$-\nabla \cdot \kappa \nabla u = f \qquad \text{in } \Omega, \tag{4.6}$$

$$u = 0 \qquad \text{at } z = 0, \tag{4.7}$$

$$\kappa \frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\mathcal{O} \times [0, L], \tag{4.8}$$

$$\kappa \frac{\partial u}{\partial n} = q \qquad \text{at } z = L. \tag{4.9}$$

This problem describes steady heat transfer in a domain Ω with the temperature $u = 0$ prescribed on the plane $z = 0$, a heat flux, q, prescribed on the plane $z = L$, and no heat transfer through the cylinder wall. A volumetric heat source, f, is assumed given. We choose the domain Ω

and the discretization to be the same as depicted in Figure 6. The parameter κ is the thermal conductivity. As long as κ can be expressed as

$$\kappa(x, y, z) = \kappa_2(x, y) \cdot \kappa_1(z),$$

the proposed solution algorithm still applies.

We solve this test problem with $f = 0$, $q = 1$, $\kappa = 1$ in the lower half of the cylinder, and $\kappa = 2$ in the upper half of the cylinder. This problem has a simple analytical solution which only depends on z; the exact solution is piecewise linear with a jump in the first derivative at $z = 1/2$,

$$\begin{aligned} u(x, y, z) &= z \quad \text{for } 0 \le z \le \frac{1}{2}, \\ u(x, y, z) &= \frac{1}{2}z + \frac{1}{4} \quad \text{for } \frac{1}{2} \le z \le 1. \end{aligned}$$

The numerical solution is, indeed, constant in each xy-plane, and the variation in the z-direction is shown in Figure 9. We remark that using the analytic eigenfunctions (3.8) would be inappropriate for this problem due to the nonconstant thermal conductivity in the z-direction. In this sense, the proposed diagonalization procedure offers convenience in terms of handling situations with variable coefficients and other types of boundary conditions.

5 Computational complexity

We now discuss the computational cost of the proposed procedure. This includes the cost of the diagonalization step (3.14), and the solution steps (3.21), (3.22), and (3.24).

In the following, we denote the total number of elements (finite elements or spectral elements) in the domain as K, the number of elements in a two-dimensional cross-section as K_{xy}, while the number of layers of elements in the z-direction is denoted as K_z, i.e., $K = K_{xy} \cdot K_z$. In the spectral element case, we denote the polynomial order as N. Note that, in the spectral (element) case, the polynomial order need not be the same in the z-direction as in the xy-plane. With

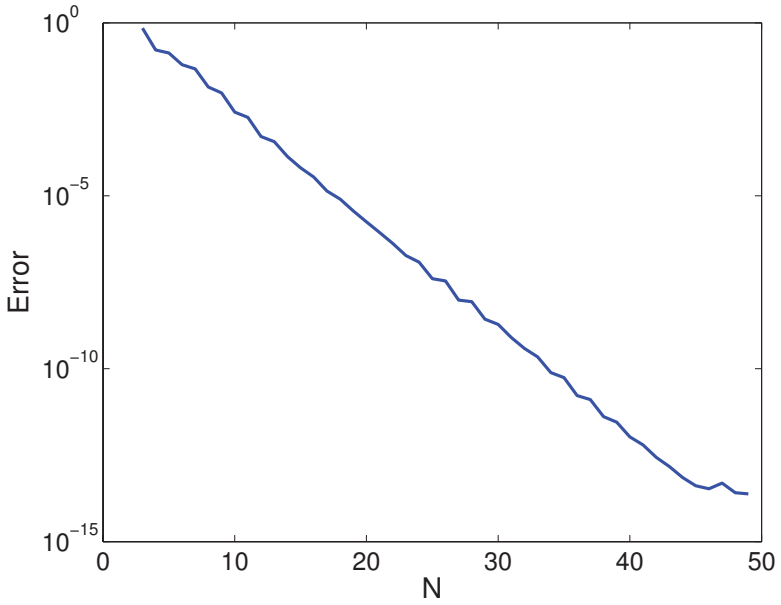


Figure 8: Relative error between the exact solution u and the numerical solution u_N of the three-dimensional Helmholtz problem (4.4) as a function of the polynomial degree, N, in each spatial direction. The error is measured in the discrete L^2-norm.

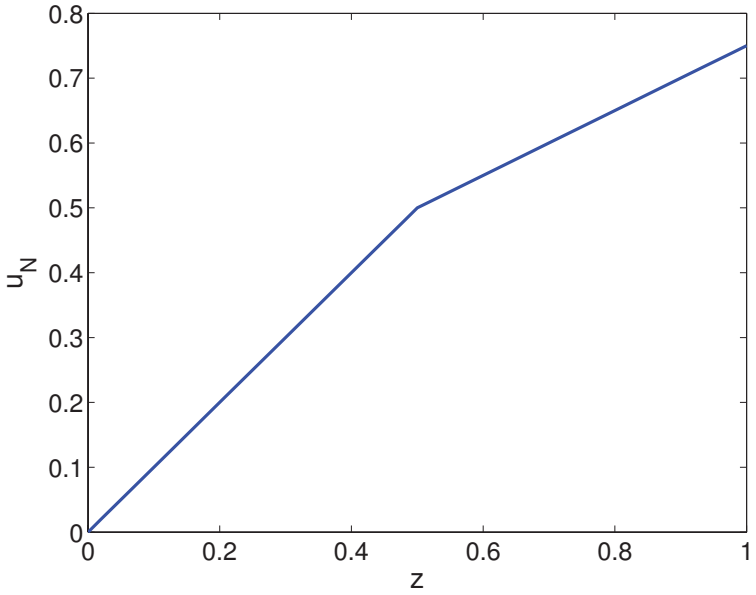


Figure 9: A plot of the variation in the numerical solution of (4.6) - (4.9) in the z-direction at a fixed point in the xy-plane. In this problem, $f = 0$ and $q = 1$. As expected, the numerical solution coincides with the exact solution.

these parameters, it follows that

$$N_2 \approx K_{xy} \qquad \text{using linear finite elements,}$$
$$N_2 \approx K_{xy} \cdot N^2 \qquad \text{using spectral elements,}$$

and

$$N_1 \approx K_z \qquad \text{using linear finite elements,}$$
$$N_1 \approx K_z \cdot N \qquad \text{using spectral elements.}$$

The exact numbers N_2 and N_1 will depend on the prescribed boundary conditions.

In order to provide a "reference" for a cost analysis of the proposed method, we first estimate the cost of performing a single operator evaluation involving the three-dimensional discrete Laplacian:

$$72\,K \qquad \text{using linear finite elements,} \tag{5.1}$$
$$12\,K\,N^4 \qquad \text{using spectral elements.} \tag{5.2}$$

In the linear finite element case, this estimate is based on an element-by-element approach, and assuming an already explicitly contructed element matrix of dimension 6×6. In the spectral element case, this estimate assumes the use of tensor product sum-factorization techniques; see [12].

We now proceed with a cost analysis of the proposed algorithm; a summary of this cost analysis is given in Table 1. The cost of solving the one-dimensional eigenvalue problem (3.14) is approximately equal to N_1^3. This estimate just states the usual fact that the cost of solving the eigenvalue problem scales like the cube of the dimension of the problem; see [6]. Let us compare this cost with the cost of performing a single operator evaluation involving the discrete Laplacian. To this end, let us first assume that $K_z \sim \sqrt{K_{xy}}$. The cost of the eigenvalue problem

	Linear finite elements	Spectral elements
N_1	K_z	$K_z N$
N_2	K_{xy}	$K_{xy} N^2$
Operator evaluation $\underline{w} = A^{3D}\underline{v}$	$72K$	$12KN^4$
Eigenvalue problem (3.14); $K_z \sim \sqrt{K_{xy}}$	K	KN^3
Transformation (3.21) and (3.24)	$4K_zK$	$4K_zKN^4$
Solution of (3.22) (M iterations)	$18MK$	$8MKN^4$
Total cost (proposed solver)	$(18M + 4K_z)K$	$(8M + 4K_z)KN^4$
Total cost (PCG, ideal 3D preconditioner)	$72MK$	$12MKN^4$

Table 1: A table showing the scalings and estimated cost for the various parts of the proposed algorithm. The total cost is also compared to an iterative solution of the full three-dimensional system, using an "ideal" preconditioner in the sense of incuring an insignificant computational cost, while giving a resolution-independent convergence rate.

is then

$$N_1^3 \sim K_z^3 \sim K \qquad \text{using linear finite elements,}$$

$$N_1^3 \sim K_z^3 N^3 \sim K N^3 \qquad \text{using spectral elements.}$$

In the linear finite element case, the cost of the eigenvalue problem will then be significantly smaller than the cost of performing a single operator evaluation (assuming we exploit all the sparsity); see (5.1). In the spectral element case, the cost of the eigenvalue problem will also be significantly less than the cost of performing a single operator evaluation since the latter scales like $\mathcal{O}(K\,N^4)$; see (5.2). If $K_z \ll \sqrt{K_{xy}}$, this conclusion will be even more true. If $K_z \gg \sqrt{K_{xy}}$, the solution of the eigenvalue problem will become more important. However, the resolution in the z-direction has to be quite extreme compared to the x- and y-direction before the eigenvalue problem will play a significant role; see below.

The transformation step (3.21) involves a matrix-matrix product. The number of floating point operations for this step is

$$2\,N_2 \cdot N_1^2 \;\approx\; 2\,K_{xy} \quad \cdot (K_z)^2 \quad = 2\,K_z \cdot K \qquad \text{using linear finite elements,} \tag{5.3}$$

$$2\,N_2 \cdot N_1^2 \;\approx\; 2\,K_{xy}\,N^2 \cdot (K_z N)^2 = 2\,K_z \cdot K\,N^4 \qquad \text{using spectral elements.} \tag{5.4}$$

Hence, for $K_z < 36$ in the linear finite elements case, and for $K_z < 6$ in the spectral element

case, the cost of (3.21) is less than the cost of a single iteration using an iterative solver for the full three-dimensional problem.

Step (3.22) involves solving approximately N_1 two-dimensional problems. The cost of these problems depends on the particular solver used. For an elliptic problem like the Poisson problem, we can use a preconditioned conjugate gradient method with a domain-decomposition-based preconditioner. In the optimal case, the cost of each two-dimensional solve will at least be a fixed number of iterations, M, times the cost of a *two-dimensional* operator evalution (or matrix-vector product); see [15]. In the linear finite element case, this lower bound can be estimated to be $18\,M\,K_{xy}$, while in the spectral element case, this lower bound is equal to $8\,M\,K_{xy}\,N^3$. The total number of floating point operations for (3.22) is then approximately equal to

$$\begin{array}{ll} 18\,M\,K & \text{using linear finite elements,} \\ 8\,M\,K\,N^4 & \text{using spectral elements.} \end{array}$$

For non-optimal preconditioners, this cost will be higher. We have here neglected the cost of the preconditioner and other aspects of the iterative solver. Note that from an implementation point of view, implementing "optimal" preconditioners in the two-dimensional case is much easier than in the three-dimensional case. Finally, note that solving two-dimensional problems has eliminated potential difficult aspect ratio problems associated with domains with a small/large length scale in the z-direction compared to a typical length scale in the xy-plane.

The above analysis only considers an iterative solution strategy for the two-dimensional problems in (3.22). This is the most practical strategy for large problems. However, it may also be possible to use a direct solution method for these two-dimensional systems even though this may not be practical for the full three-dimensional problem. We refer to the numerical test using linear finite elements in Section 4.1 as an example. If a direct method is used for the two-dimensional systems, the entire proposed solution procedure can be classified as a direct solution method.

The final transformation step (3.24) involves a matrix-matrix product at a similar cost as (3.21); see (5.3)-(5.4).

In summary, if we can neglect the cost of solving the one-dimensional eigenvalue problem,

the total cost of the proposed algorithm is then

$$\begin{aligned} (18\,M\,+\,4\,K_z)\cdot K \qquad & \text{using linear finite elements,} \\ (8\,M\,+\,4\,K_z)\cdot K\,N^4 \qquad & \text{using spectral elements.} \end{aligned}$$

We have here assumed iterative solution of the two-dimensional systems using optimal (ideal) preconditioners, and each system taking M iterations to converge.

Consider now the cost of using an iterative solver for the full three-dimensional problem with an optimal (ideal) preconditioner that also takes M iterations to converge. From (5.1) and (5.2) this cost is at least

$$\begin{aligned} (72\,M)\cdot K \qquad & \text{using linear finite elements,} \\ (12\,M)\cdot K\,N^4 \qquad & \text{using spectral elements.} \end{aligned}$$

The proposed method should therefore (conservatively) be competitive with the best (ideal) iterative solution methods if

$$\begin{aligned} K_z < 13\,M \qquad & \text{using linear finite elements,} \\ K_z < M \qquad & \text{using spectral elements.} \end{aligned}$$

For example, if $M = 20$, we can have up to $K_z = 260$ layers of linear finite elements, and $K_z N = 200$ nodes in the z-direction in the spectral element case (assuming $N = 10$).

Again, this is a conservative estimate. For most problems where the proposed algorithm is applicable, the computational complexity will be smaller than the best iterative solution method for the full three-dimensional problem. Additional issues in favor of the proposed approach are: elimination of the aspect ratio problem discussed earlier; the number of iterations, M, is typically larger for a full three-dimensional problem compared to solving a problem in a two-dimensional cross section; the number of iterations, M, is not a constant, but a function of the polynomial degree N in the case of using spectral elements; easier implementation (two-dimensional versus three-dimensional solvers); easier parallelization (see the next section).

6 Parallel processing

The solution step (3.22) suggests a natural way to parallelize this algorithm. One way is to simply solve one (or more) two-dimensional problem(s) on a single processor. This will imply that all the two-dimensional problems can be solved in parallel *without any communication*. The implementation of this step will be unchanged from a serial version.

If we assume that the eigenvector matrix Q is stored on all the processors, step (3.21) can be done by first storing the right hand side g_{ij} for some values of i in the xy-plane, and for all values of j along the z-direction. The transformation of the given data as given by (3.20) or (3.21) can then be done in parallel *without any communication*.

The transformed data $\widetilde{G}$ can now be "transposed" so that $\tilde{g}_{ij}$ will be available on each processor for *all* values of the index i in the xy-plane, and for one or more values of j. This "transpose" operation will require global communication.

The two-dimensional systems (3.22) can now be solved in parallel in a completely decoupled fashion, and the solution $\widetilde{U}$ will be distributed so that $\tilde{u}_{ij}$ will be available on each processor for *all* values of the index i in the xy-plane, and for one or more values of j. This distribution is again "transposed" so that $\tilde{u}_{ij}$ will be available on each processor for some values of i in the xy-plane, and for all values of j along the z-direction. The final solution U is then obtained from (3.24) by performing the matrix-matrix product in parallel *without any communication*.

Hence, the total communication cost for the algorithm is limited to the two "transpose" operations which imply an all-to-all-type of communication pattern. The parallel implementation as described above should be compared with a more standard domain decomposition approach which typically implies a parallel implementation of the preconditioned conjugate gradient method applied to the full three-dimensional system. The advantage with the proposed method is that it is has low computational complexity and it is easy to implement both in a serial and a parallel context.

It is interesting to notice that an iterative solution of the independent two-dimensional systems will enjoy both the convergence rate and ease of implementation associated with two-dimensional systems. As explained above, we can solve each two-dimensional system on a single processor. However, in the case that each two-dimensional system is also very large, we can parallelize

the solution of each two-dimensional system as well (in addition to the parallelization across two-dimensional planes). For such large problems, the proposed algorithm would offer a way to exploit computer platforms with many processors.

7 Conclusions

We have presented a new solution algorithm for problems in computational domains which can be expressed as a tensor-product between a general two-dimensional domain in the xy-plane and the z-direction. The new algorithm allows general boundary conditions to be specified in the z-direction. It can also be used to solve problems with variable coefficients as long as these can be expressed as a separable function with respect to the variation in the xy-plane and the variation in the z-direction.

For most problems where the proposed method is applicable, the computational complexity is better or at least as good as the best available iterative solvers. An attractive feature with the proposed method is that it eliminates the aspect ratio problem associated with domains which have a small length scale in the z-direction compared to a typical length scale in the xy-plane. The method also allows for an easy implementation, both in a serial and a parallel context.

We have demonstrated the new algorithm by solving selected Poisson- and Helmholtz-type problems. However, we remark that the method is equally applicable for three-dimnensional geometries with other two-dimensional topologies, e.g., a planar region with a certain number of holes, and for solving other partial differential equations.

8 Future work

Future work will include the application of the method presented here to simulate three-dimensional Bénard-Marangoni convection [8]. Previous numerical results for this problem have assumed a fixed and undeformed free surface [11]. However, it is known that the (unknown) free surface will be slightly deformed for this type of problems. In the context of solving the governing equations (the incompressible Navier-Stokes equations and the energy equation) numerically using a splitting approach, the computational problem is reduced to solving a Helmholtz-type equation

for the velocity and a Poisson-type equation for the pressure at each time step. The fast tensor-product solver presented here could function as a perfect preconditioner for these elliptic solves; for small free surface deformations, the convergence should be very rapid. The solver should also be insensitive to the potentially large aspect ratios associated with the computational domains for this class of problems. Note that the two-dimensional cross-sections may be quite general and may not be possible to express in tensor-product form [11].

Future work will also extend the approach presented here to include fast tensor-product solvers for a combined space-time treatment. In a second paper (Part II), we will discuss a tensor-product solver for the pure spectral case. This is part of an ongoing research effort to extend the possibilities for parallel processing in the time direction, thus allowing for an overall increased speedup for the simulation of evolution problems described by partial differential equations.

Finally, we mention an open area for the possible application of fast tensor-product solvers, namely, the approximate solution of the Boltzmann equation. For this type of problems, we have 6 independent variables in three physical space dimensions: 3 velocity directions and 3 physical coordinate directions. A "cross-section" in the xyz-"plane" is here invariant with respect to all the velocity directions. This could potentially be exploited in the construction of tensor-product bases and fast solvers.

References

[1] Bjørstad, P.E. and Tjøstheim, B.P. (1997). Efficient algorithms for solving a fourth-order equation with the spectral Galerkin method. *SIAM J. Sci. Comput.*, **18**(2), 621-632.

[2] Carvalho, M.S. and Scriven, L.E. (1999). Three-dimensional stability analysis of free surface flows: Application to forward deformable roll coating. *J. Comput. Phys.*, **151**(2), 534-562.

[3] Chu, D., Henderson, R., and Karniadakis, G.E. (1992). Parallel spectral-element-Fourier simulation of turbulent flow over riblet-mounted surfaces. *Theoretical and Computational Fluid Dynamics*, **3**, 219-229.

[4] Couzy, W. and Deville, M.O. (1995). A fast Schur complement method for the spectral element discretization of the incompressible Navier-Stokes equations. *J. Comput. Phys.*, **116**, 135-142.

[5] Fischer, P.F. (1997). An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations. *J. of Comput. Phys.*, **133**, 84-101.

[6] Golub, G. and Van Loan, C.F. (1983). *Matrix Computations*, John Hopkins.

[7] Henderson, R. (1997). Nonlinear dynamics and pattern formation in turbulent wake transition. *J. Fluid Mech.*, **353**, 65-112.

[8] Koschmieder, E.L. (1993). *Bénard Cells and Taylor Vortices*, Cambridge University Press.

[9] Kwan, Y.-Y. and Shen, J. (to appear). An efficient direct parallel spectral-element solver for separable elliptic problems. *J. Comput. Phys.*, DOI: 10.1016/j.jcp2007.02.013.

[10] Lynch, R.E., Rice, J.R., and Thomas, D.H. (1964). Direct solution of partial differential equations by tensor product methods. *Numer. Math.*, **6**, 185-199.

[11] Medale, M. and Cerisier, P. (2002). Numerical simulation of Bénard-Marangoni convection in small aspect ratio containers. *Numer. Heat Transfer, Part A*, **42**, 55-72.

[12] Maday, Y. and Patera, A.T. (1989). Spectral element methods for the Navier-Stokes equations. In Noor, A.K. (ed.), *State of the Art Surveys in Computational Mechanics*, ASME, New York, 71-143.

[13] Patera, A.T. (1986). Fast direct Poisson solvers for high-order finite element discretizations in rectangularly decomposable domains. *J. Comput. Phys.*, **65**, 474-480.

[14] Shen, J. (1994). Efficient spectral-Galerkin method I. Direct solvers for the second and fourth order equations using Legendre polynomials. *SIAM J. Sci. Comput.*, **15**(6), 1489-1505.

[15] Toselli, A. and Widlund, O.B. (2004). *Domain Decomposition Methods - Algorithms and Theory*, Springer Series in Computational Mathematics, **34**.

[16] Tufo, H.M. and Fischer, P.F. (1999). Terascale Spectral Element Algorithms and Implementations. Gordon Bell Prize paper, Proc. of the ACM/IEEE SC99 Conf. on High Performance Networking and Computing. IEEE Computer Soc., CDROM.