NUMERICAL MATHEMATICS: Theory, Methods and Applications
Numer. Math. Theor. Meth. Appl., Vol. **1**, No. 3, pp. 245-274 (2008)

Design of Finite Element Tools for Coupled Surface and Volume Meshes

Daniel Köster[1], Oliver Kriessl[2] and Kunibert G. Siebert[*,2]

[1] *Numerical Analysis and Computational Mechanics Group, University of Twente, P. O. Box 217, 7500 AE Enschede, The Netherlands.*
[2] *Institut für Mathematik, Universität Augsburg, Universitätsstraße 14, D-86159 Augsburg, Germany.*

Received 13 January, 2008; Accepted (in revised version) 25 March, 2008

Abstract. Many problems with underlying variational structure involve a coupling of volume with surface effects. A straight-forward approach in a finite element discretization is to make use of the surface triangulation that is naturally induced by the volume triangulation. In an adaptive method one wants to facilitate "matching" local mesh modifications, i.e., local refinement and/or coarsening, of volume and surface mesh with standard tools such that the surface grid is always induced by the volume grid. We describe the concepts behind this approach for bisectional refinement and describe new tools incorporated in the finite element toolbox ALBERTA. We also present several important applications of the mesh coupling.

AMS subject classifications: 65N30, 65N50, 65Y15
Key words: Adaptive finite element methods, scientific software, software design.

1. Introduction

A great variety of problems in science and engineering are modeled mathematically by means of a system of partial differential equations (PDEs) closed with suitable initial, boundary, or interface conditions. The PDEs are defined on a domain in space or space-time and in many applications the shape of the domain may also be unknown beforehand, and must be determined as part of the solution. In addition, the problems under consideration involve a coupling of surface and bulk effects. The mathematical description may reflect this in that the PDEs contain some unknowns defined on a spacial domain Ω as well as other unknowns defined on a lower-dimensional manifold $\Gamma \subset \bar{\Omega}$, for instance the domain boundary $\partial\Omega$. In Section 2 we give several examples of such problems.

These problems may be numerically solved using various discretization schemes and techniques. In this paper we will focus on finite element discretizations. Most finite element methods for time-dependent problems do not mesh the space-time domain, but

[*]Corresponding author. *Email addresses:* kosterdtp@ewi.utwente.nl (D. Köster), siebert@math.uni-augsburg.de (K. G. Siebert)

employ a suitable time discretization for converting the time-dependent problem into a sequence of stationary problems. This allows us to restrict ourselves to spacial domains. Furthermore, we only consider simplicial grids but the derived methods directly carry over to other types of meshes.

During the last decades, adaptive finite elements have become a well-established tool for the numerical solution of boundary value problems, see the monographs [1, 5, 59] and the references therein. Adaptivity is designed to use computational resources more efficiently. In higher space dimensions some problems may only be solvable in reasonable time using adaptive methods. Adaptive finite element methods employ an iteration of the form

$$\mathsf{SOLVE} \rightarrow \mathsf{ESTIMATE} \rightarrow \mathsf{MARK} \rightarrow \mathsf{REFINE/COARSEN}$$

for adapting the finite element mesh to the solution of the underlying problem. Given a grid, procedure SOLVE computes the discrete solution and ESTIMATE computes an a posteriori error estimate, which is an upper bound for the error in some given norm in terms of the discrete solution and data of the PDE. Usually, the estimator is built from element error indicators, which are used in MARK for selecting elements subject to refinement and/or coarsening. In the last step, refinement and/or coarsening algorithms locally refine and/or coarsen the grid based on the decisions taken in MARK, see for instance [49] for a more detailed description. For elliptic problems the above adaptive loop is well analyzed with respect to convergence [20, 41, 42] and optimal cardinality [9, 14, 53].

The finite element discretization of problems involving bulk and surface effects is done by triangulating Ω as well as Γ and defining finite element spaces on both triangulations. The different spaces are then used for approximating bulk quantities respectively surface quantities. The surface triangulation is naturally defined by collecting the faces of elements of the bulk triangulation that lie on Γ, i.e., the surface grid is the trace of the volume grid. Since bulk and surface effects interact, we need restrictions of bulk quantities to the surface, naturally introducing the concept of trace spaces. Some applications may also require prolongations of surface quantities to the bulk. For standard Lagrange finite element discretizations both tasks are facilitated by an injective mapping connecting surface degrees of freedom (DOFs) with bulk degrees of freedom. Such a mapping in combination with corresponding finite element bases for bulk and surface then exactly realizes the finite element space on Γ as the trace space of the bulk space defined over Ω.

Coupled meshes are easily handled if the meshes do not change during a computation. In the setting of adaptive methods the problems of coupling grids become inherently more complex. If an adaptive method requires a change of any of the involved meshes, we might lose the useful property that the surface grids were originally defined as the collection of bulk faces. Without this property the transfer of data between bulk and surface meshes becomes much more difficult. In this scenario, after each mesh change one would have to somehow reconstruct the connection of bulk elements with surface elements, a cumbersome, possibly costly process. The aforementioned mapping of DOFs would no longer exist and would have to established from scratch.

It is thus highly desirable that the coupling is maintained automatically during local mesh modifications, i.e., refinement and coarsening of relatively small patches of elements.

We will present techniques of simultaneous bulk/surface mesh adaption fulfilling the property that surface mesh elements are always faces of bulk elements. With this property, data transfer between meshes will always remain simple and efficient. However, when adapting both meshes simultaneously, the bulk mesh will require a matching refinement of the surface mesh, and vice versa. It is not clear that the $(d-1)$-dimensional surface mesh, if defined simply as the intersection of bulk elements with Γ, is actually what a given coded refinement strategy for $(d-1)$-dimensional meshes would yield if applied to the surface mesh independently.

The main focus of this paper is a solution of these issues for bisectional refinement, which we describe in detail in Section 3. We first recall the refinement by bisection of conforming triangulations in one, two, and three space dimensions and introduce the concept of submeshes. In addition, we prove that the bisectional refinement of a surface mesh induced by the refinement the bulk mesh coincides with the natural refinement of the surface mesh. As an outcome, both meshes can be refined (and coarsened) by the standard algorithm. Having one and only one refinement/coarsening scheme for all meshes of a given dimension simplifies code, and is therefore desirable.

In general, we aim for a "dimensionless programming" approach, i.e., the software must hide dimension-dependent code from the user as much as possible. This includes for instance mesh refinement, coupling of surface and volume grids, etc. The user may thus create elegant general code valid for all dimensions. The software should also take care of all bookkeeping details concerning submeshes automatically, leaving the user free to concentrate efforts on the implementation of the underlying problem. This is the philosophy of the finite element toolbox ALBERTA [48, 49]. Basic details about the implementation of the coupling within ALBERTA are given in Section 4 as well as numerical results from three prominent applications of coupled surface volume problems that are introduced next.

2. Example problems

There are many interesting applications that couple bulk and surface effects. We present three characteristic model applications from free surface flow, growth of an epitaxial layer, and minimal surfaces. With these examples we want to show different couplings that naturally occur. Numerical results for these applications are presented in Section 4. Other important problems are for instance dendritic growth [8, 47], fluid-structure interaction [16, 30], and morphological changes in stressed epitaxial films [4, 52]. In addition to problems where the problem involves surface/bulk coupling, we also refer to mixed or hybrid finite element methods which introduce surface/bulk coupling in the numerical scheme, for instance when using Lagrange multipliers on the boundary for imposing boundary/coupling conditions [12].

2.1. Fluid flow with free capillary surfaces

Consider a droplet of an incompressible Newtonian fluid freely suspended in d-dimensional space. The fluid volume is surrounded by another medium or vacuum and solely in-

fluenced in its motion by the action of surface tension at the interface to the other medium. The standard mathematical description of this effect uses the incompressible Navier-Stokes equations together with capillary boundary conditions at the interface:

$$\frac{\partial \boldsymbol{v}}{\partial t} + (\nabla \boldsymbol{v})\boldsymbol{v} = \nabla \cdot \Sigma(\boldsymbol{v}, p) \qquad \text{in } \Omega(t),\ t \in (0,T), \tag{2.1a}$$

$$\nabla \cdot \boldsymbol{v} = 0 \qquad \text{in } \Omega(t),\ t \in (0,T), \tag{2.1b}$$

$$\Sigma(\boldsymbol{v}, p)\boldsymbol{n} = \sigma(d-1)\boldsymbol{\kappa} \qquad \text{on } \Gamma(t),\ t \in (0,T), \tag{2.1c}$$

$$\boldsymbol{v} \cdot \boldsymbol{n} = V \qquad \text{on } \Gamma(t),\ t \in (0,T), \tag{2.1d}$$

$$\boldsymbol{v} = \boldsymbol{v}_0, \quad \Omega(0) = \Omega_0 \qquad \text{for } t = 0 \tag{2.1e}$$

with the Newtonian stress tensor $\Sigma(\boldsymbol{v}, p)$ defined as

$$\Sigma_{ij}(\boldsymbol{v}, p) = -p\delta_{ij} + \eta\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right). \tag{2.1f}$$

The quantities that appear are the free boundary of the fluid domain $\Gamma = \partial\Omega$, the fluid velocity $\boldsymbol{v}$ and pressure p. Constant parameters that describe the fluid are the dynamic viscosity coefficient η and the surface tension σ. On the moving boundary Γ we use the outer unit normal $\boldsymbol{n}$, the vector of curvature $\boldsymbol{\kappa}$, and the normal velocity V, i.e., the velocity of Γ into direction $\boldsymbol{n}$. The vector $\boldsymbol{\kappa}$, by definition, has magnitude H, the mean curvature of Γ, and points in the direction $\boldsymbol{n}$. The system is closed by initial conditions $\boldsymbol{v}_0$ and Ω_0 for the velocity $\boldsymbol{v}$ and the domain Ω.

The interaction between bulk and surface in this problem is given in one direction by the fluid velocity $\boldsymbol{v}$, whose normal component at the domain boundary Γ prescribes the motion of Γ itself, see (2.1d). In the other direction, the surface curvature $\boldsymbol{\kappa}$, which is totally determined by the position of Γ, defines the surface tension that exerts stress on the fluid volume, as described by (2.1c). For the case of water droplets suspended in air it is well known that the surface tension force will seek to pull the droplet into a spherical shape corresponding to a minimum of the surface energy potential. For more details on the physical aspects we refer to [37]. A detailed study of numerical discretization aspects of this problem is presented in [7,34,35].

This mutual interaction makes the problem well suited as an example of coupled surface and bulk effects, and give rise to interesting new flow situations, such as the flow in an oscillating droplet. A finite element method for the problem should be designed in such a way that the surface effects and the surface-bulk coupling are easily discretized numerically. Namely, it is useful to have a surface triangulation automatically induced by the volume discretization, as presented later.

2.2. Liquid phase epitaxial growth

We consider the following part of the production process of a wafer for an infra-red detector: Inside a heated furnace a substrate is dipped into a melting pot which is filled with a compound of molten materials. By reducing the temperature a thin single crystalline

film begins to grow onto the surface of the substrate to form an epitaxial layer of about twenty microns.

The mathematical model for this problem has to account for the following effects: convection in the melt, which arises due to buoyant forces induced by temperature and concentration changes, impact of the flow on the composition and temperature distribution of layer and melt. This is described by the incompressible Navier-Stokes equations in Boussinesq approximation, conservation of energy, and conservation of mass. Precise information about the thickness and composition of the epitaxial layer has to be extracted from temperature, concentrations, and the phase diagram that describes the solidification process. For the transition between liquid phase $\Omega_\ell(t)$ and solid phase $\Omega_s(t)$ we thus apply a sharp interface model, i.e., the interface $\Gamma(t)$ is assumed to be always a $(d-1)$ dimensional manifold. For the complete derivation of the model we refer to [36] and [32].

Denoting by $\Omega(t) = \Omega_\ell(t) \cup \Omega_s(t)$ the union of liquid and solid phase, and using the same notation as in (2.1) we obtain for $t > 0$ the following nonlinear system of PDEs for velocity $\boldsymbol{v}$ and pressure p in the melt, and temperature θ and concentrations c_1, c_2 in layer and melt:

$$\frac{\partial \boldsymbol{v}}{\partial t} + (\nabla \boldsymbol{v})\boldsymbol{v} - \nabla \cdot \Sigma(\boldsymbol{v}, p) = -\boldsymbol{f}(\theta, c_1, c_2) \qquad \text{in } \Omega_\ell(t), \tag{2.2a}$$

$$\nabla \cdot \boldsymbol{v} = 0 \qquad \text{in } \Omega_\ell(t), \tag{2.2b}$$

$$\frac{\partial \theta}{\partial t} + \boldsymbol{v} \cdot \nabla \theta - \nabla \cdot (D_0 \nabla \theta) = 0 \qquad \text{in } \Omega(t), \tag{2.2c}$$

$$\frac{\partial c_i}{\partial t} + \boldsymbol{v} \cdot \nabla c_i - \nabla \cdot (D_i \nabla c_i) = 0 \qquad \text{in } \Omega(t),\ i = 1, 2, \tag{2.2d}$$

where

$$\boldsymbol{f}(\theta, c_1, c_2) = \Big(\beta_0(\theta - \bar{\theta}) + \sum_{i=1,2} \beta_i (c_i - \bar{c}_i)\Big)\boldsymbol{g}$$

is the buoyant force from the Boussinesq approximation with material constants β_i, $i = 0, 1, 2$, depending on the average temperature $\bar{\theta}$, average concentrations $\bar{c}_i$, $i = 1, 2$, and the vector of gravity $\boldsymbol{g}$. Furthermore, D_i, $i = 0, 1, 2$, are the diffusion parameters, which are constant in liquid and solid phase, and we have set $\boldsymbol{v} = \boldsymbol{0}$ in $\Omega_s(t)$.

The solidification process depends on the local temperature and local concentrations, i.e., the melting temperature and the composition of the layer depend on the concentrations in the melt. Whereas the temperature is assumed to be continuous across the interface, concentrations may jump and hereafter we denote for $i = 1, 2$ by c_i^ℓ, c_i^s the concentrations of the melt respectively layer. This coupling is described for a specific melt by a phase diagram. For our application it is given by the following nonlinear equations on the interface $\Gamma(t)$ (cf. [29]):

$$\theta = 1.36 - 0.56 c_2^\ell + 0.65 c_1^\ell - \frac{1.03 c_1^\ell}{1 - c_2^\ell}, \tag{2.2e}$$

$$c_1^s = \frac{0.11 c_1^\ell}{1 - 0.78 c_1^\ell - c_2^\ell} \quad \text{and} \quad c_2^s = \frac{1}{2}. \tag{2.2f}$$

In addition, temperature and concentrations satisfy the Stefan conditions on $\Gamma(t)$:

$$\left[D_0 \frac{\partial \theta}{\partial \boldsymbol{\nu}}\right]_s^\ell = -V \quad \text{and} \quad \left[D_i \frac{\partial c_i}{\partial \boldsymbol{\nu}}\right]_s^\ell = -\left[c_i\right]_s^\ell V \quad i = 1, 2, \tag{2.2g}$$

where $[\cdot]_s^\ell$ denotes the jump between liquid and solid, and V the interface velocity in normal direction. We finally assume no-slip boundary conditions for the velocity on $\Gamma(t)$. The problem is completed by initial conditions for flow, domains, temperature, and concentrations, as well as by additional boundary conditions at the border of the melting pot.

When discretizing (2.2), the computational domain given by the geometry of the melting pot is fixed, whereas the liquid part shrinks and the solid part grows. Since the thickness of the emerging layer is of special interest in the fabrication process a precise determination of the interface velocity is essential, which requires a high grid resolution near the interface. Furthermore, we use a triangulation of the melting pot such that the initial interface between liquid and solid part is represented by a surface triangulation.

This approach has several benefits. Using additional bulk basis functions connected to the surface grid, we can easily extend standard continuous finite element spaces to account for jumps at the interface without using a discontinuous Galerkin method. Such extended spaces are the appropriate ones for a discretization of (2.2d) and are preferable to a discontinuous Galerkin method, which produces for the precision a larger number of DOFs. Solving the discrete nonlinear system by a Newton method, the assemblage of the linearized system requires the computation of several integrals over the interface related to the phase diagram (2.2e), (2.2f), and the Stefan condition (2.2g). This can efficiently be done by assemblage routines working directly on the interface triangulation.

2.3. Minimal surfaces

We finally consider an application concerning minimal surfaces that appear for instance in nature as soap films or soap bubbles. We focus on the classical Plateau problem which is one of the oldest problems in mathematical analysis and can be described as follows. Given a Jordan curve Γ in $\mathbb{R}^n$, $n \geq 3$, we want to find a *disc-type* minimal surface S spanning Γ, i.e., among all surfaces spanning Γ, S is of minimal area. We may parameterize S by a function $\boldsymbol{u} : B \to \mathbb{R}^n$ over the unit disc $B := \{x \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1\}$. If $S = \boldsymbol{u}(\bar{B})$ is spanning Γ with minimal area, then $\boldsymbol{u}$ satisfies the following nonlinear PDE system

$$\Delta \boldsymbol{u} = 0 \qquad \text{in } B, \tag{2.3a}$$

$$\left|\frac{\partial \boldsymbol{u}}{\partial x_1}\right|^2 - \left|\frac{\partial \boldsymbol{u}}{\partial x_2}\right|^2 = \frac{\partial \boldsymbol{u}}{\partial x_1} \cdot \frac{\partial \boldsymbol{u}}{\partial x_2} = 0 \qquad \text{in } B, \tag{2.3b}$$

$$\boldsymbol{u} : \partial B \to \Gamma \qquad \text{is one to one.} \tag{2.3c}$$

Conversely, a solution $\boldsymbol{u}$ of (2.3) parameterizes a surface S spanning Γ, where the surface area of S is stationary. Thus, equation (2.3) can be considered as the Euler-Lagrange equations for the classical Plateau problem and for a solution $\boldsymbol{u}$ of (2.3) we call $\boldsymbol{u}(B)$

minimal surface. The strong nonlinearity of the problem is hidden in (2.3b), asking $\boldsymbol{u}$ to be conformal, as well as in (2.3c), asking $\boldsymbol{u}_{|\partial B}$ to be a monotone parameterization of Γ. Classical results together with references for minimal surfaces can be found in the books [17, 18, 43].

Following Struwe [55], we reformulate problem (2.3) and determine a weak solution as follows. We denote by S^1 the unit circle in $\mathbb{R}^2$ and distinguish it from ∂B. Fixing one smooth parameterization $\gamma\colon S^1 \to \Gamma$, we look for a bijective mapping $s\colon \partial B \to S^1$ such that s is stationary for the energy

$$E(s) := \frac{1}{2}\int_B \left|\nabla \Phi(\gamma \circ s)\right|^2. \tag{2.4}$$

Hereafter, $\Phi(\gamma \circ s)$ denotes the harmonic extension of boundary values $\gamma \circ s\colon \partial B \to \Gamma$. This means, we are looking for a parameterization $\gamma \circ s$ such that the harmonic extension $\boldsymbol{u} = \Phi(\gamma \circ s)$ is stationary for the Dirichlet integral. The surface $\boldsymbol{u}(\bar{B})$ is then a minimal surface, i.e., $\boldsymbol{u}$ is a solution of (2.3), compare with [55].

Dziuk and Hutchinson used the above reformulation for a finite element discretization of the classical Plateau problem employing piecewise linear finite elements for both the approximation of $\boldsymbol{u}$ and s [25]. For this discretization they derived optimal order a priori error estimates [26] and, later on, Dörfler and Siebert performed an a posteriori error analysis and designed an adaptive method for computing minimal surfaces [21].

The above discretization is primarily a finite element method for computing an approximation S to the parameterization $s\colon \partial B \to S^1$ on a 1d triangulation of ∂B. For accessing the corresponding Dirichlet energy (2.4) one has to compute a discrete harmonic extension of boundary data $\gamma \circ S$, which can be easily done by a finite element method for the Laplace equation over a 2d triangulation of B. Obviously, the finite element method over the 2d domain could be replaced by a boundary element method defined on the surface triangulation [2, 28]. But in contrast to a boundary element method the presented finite element approach can directly be generalized to the more complex problem of looking for surfaces of constant mean curvature spanning Γ, compare for instance [27].

3. Coupling bulk and surface meshes

We now describe our concept of coupling bulk and surface grids. In the following $\Omega \subset \mathbb{R}^d$ is a bounded domain triangulated by some initial conforming triangulation $\mathcal{T}_0$ and we restrict ourself to simplicial meshes that are created by bisectional refinement from $\mathcal{T}_0$. We first recall the refinement by bisection and then couple meshes of different dimensions. The basic concepts of triangulations and finite element spaces are taken from [10, 11, 15, 49].

3.1. Refinement by bisection

The algorithm for creating local refinements of a given triangulation using bisection of single elements relies on tagging for any element one of its edges as its *refinement edge*.

Hereafter, an element is a simplex, i.e., a line segment in 1d, a triangle in 2d, and a tetrahedron in 3d. In 1d, an "edge" is the element itself. Any element is refined into two elements by cutting this edge at its midpoint. There are several possibilities of choosing such a refinement edge for a simplex, one example is to use the longest edge; Mitchell [40] compared different approaches in 2d. We assume that refinement edges are assigned for all elements of $\mathscr{T}_0$. The refinement algorithm then prescribes the refinement edges of the two children such that shape regularity of any refinement of $\mathscr{T}_0$ is ensured.

For describing the inheritance of refinement edges from parent to children during refinement we follow the concept of Kossaczký [33]. We adopt the convention that all vertices of an element are given fixed *local indices* $0,\cdots,d$. We assume that the refinement edges are set as those edges between local indices 0 and 1 on the initial triangulation $\mathscr{T}_0$. During refinement, the new vertex numbers, and thereby the refinement edges, for the newly created child simplices are prescribed by the refinement algorithm in the following way. In all dimensions, the index of the newly generated vertex at the midpoint of this edge has the highest local index for both children. This already fixes in 1d the numbers of the other two vertices, compare Fig. 1 (left). For 2d and 3d the numbering of vertices on the children are shown in Fig. 1 (right) respectively Fig. 2. In 1d and 2d this numbering is the same for all refinement levels. In 3d, one has to make some special arrangements: the numbering of the second child's vertices depends on the *type* of the element. There exist three different element types 0, 1, and 2. The type of the elements on $\mathscr{T}_0$ can be prescribed (usually type 0 tetrahedron). The type of the refined tetrahedra is recursively given by the definition that the type of a child element is ((parent's type + 1) modulo 3). In Fig. 2 we used the following convention: for the index set {1,2,2} on `child[1]` of a tetrahedron of type 0 we use the index 1 and for a tetrahedron of type 1 and 2 the index 2.

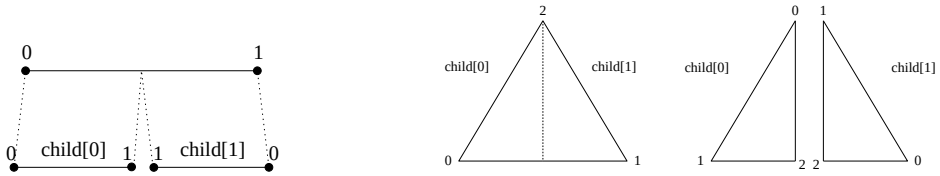

Figure 1: Numbering of nodes on parent and children for intervals and triangles.

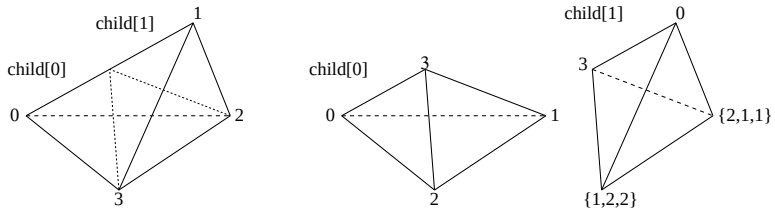

Figure 2: Numbering of nodes on parent and children for tetrahedra. Note that child 1 vertex numbering depends partially on the three cases of parent type.

Relying on the above algorithm all possible refinements of $\mathscr{T}_0$, i.e., shape and position of any possible element, any possible conforming refinement, etc., are totally determined

by the local vertex numbering on $\mathscr{T}_0$ plus a prescribed element type on $\mathscr{T}_0$ in 3d. Furthermore, a successive refinement of every macro element only produces a small number of similarity classes, guaranteeing shape regularity. Consider the special macro triangulations of a (unit) square in 2d and cube in 3d decomposed into two triangles resp. six tetrahedra, such that the common edge is the refinement edge for all macro elements. In this situation the above algorithm guarantees that the longest edge will always be the refinement edge for any element.

Up to now we have described the refinement of a single element. Refinement of a conforming triangulation can either be done in an iterative or recursive way. In the iterative variant all selected elements are bisected and this in general results in a non-conforming triangulation. In order to maintain conformity additional elements have to be refined until the resulting triangulation is conforming [6].

The recursive variant, which we use latter on, avoids non-conforming situations by only allowing refinement of a selected element if its refinement edge is the refinement edge for *all* elements that share this edge. The set of these elements is called the refinement patch. If for all patch elements the common edge is the refinement edge, then the entire patch is refined at the same time by inserting one new vertex in the midpoint of the common refinement edge and bisecting every element of the patch. This process is called the *atomic refinement operation* and the resulting triangulation is always a conforming one. In the other situation there is an element in the patch whose refinement edge is not the common edge. Such a neighbour is *not compatibly divisible* and we first perform the atomic refinement operation at the neighbour's refinement edge. In 2d the child of such a neighbour at the common edge is then compatibly divisible; in 3d such a neighbour has to be bisected at most three times and the resulting tetrahedron at the common edge is then compatibly divisible. The recursive refinement algorithm now reads

Algorithm 3.1 (Recursive refinement of one simplex).

```
subroutine recursive_refine(T, 𝒯)
  do
    𝒜 := {T' ∈ 𝒯; T' is not compatibly divisible with T}
    for all T' ∈ 𝒜 do
      recursive_refine(T', 𝒯);
    end for
  until 𝒜 = ∅

  𝒜 := {T' ∈ 𝒯; T' is element at the refinement edge of T}
  for all T' ∈ 𝒜
    bisect T' into T'_0 and T'_1
    𝒯 := 𝒯\{T'} ∪ {T'_0, T'_1}
  end for
```

The refinement of a given triangulation $\mathscr{T}$ where some or all elements are marked for refinement is then performed by

Algorithm 3.2 (Recursive refinement algorithm)**.**

```
subroutine refine(𝒯)
  for all T ∈ 𝒯 do
    if T is marked for refinement
      recursive_refine(T, 𝒯)
    end if
  end for
```

Relying on recursion, we have to ensure that the recursion terminates. Termination of the recursion hinges on the distribution of the refinement edges on $\mathscr{T}_0$ and we call a distribution admissible, if recursion terminates for any element of any refinement of $\mathscr{T}_0$. An arbitrary choice of refinement edges may not be admissible, see [40,49] for an example. In 2d, tagging the "longest edge" as refinement edge for all elements in $\mathscr{T}_0$ is admissible [40]. In 3d, the situation is more complex and it is not clear that there exists an admissible distribution of refinement edges for arbitrary $\mathscr{T}_0$. Allowing for a possible refinement of $\mathscr{T}_0$, Kossaczký proved the existence of an admissible distribution on the possibly refined grid [33].

Any possible refinement $\mathscr{T}$ of the initial triangulation $\mathscr{T}_0$ can be described as a collection of binary trees, where the roots of the trees are elements of $\mathscr{T}_0$, and every element has either exactly two descendents or none. The refinement $\mathscr{T}$ is simply the collection of all elements with no descendents. The tree structure is solely induced by bisection of single elements and does not depend on refining a whole triangulation iteratively or recursively. It can be exploited in computations to save memory. As an example, only the physical vertex coordinates of macro elements need to be stored. Coordinates of descendent simplices can be calculated from the hierarchy induced by the tree structure. In addition, a coarsening algorithm as the inverse operation to refinement profits enormously from having access to all possible refinements of $\mathscr{T}_0$, compare with the detailed description in [49]. When coupling volume and surface meshes we want to exploit this hierarchical structure for both triangulations.

We finish this section by a short review of existing bisectional refinement algorithms. The described approach was introduced in 2d by Rivara [45] in 1984 giving the *newest vertex* bisection (in Mitchell's notation). It was generalized to 3d by Bänsch [6] in 1991. Later on, Liu and Joe [38] as well as Arnold et al. [3] presented modified versions of the algorithm by Bänsch, mainly relaxing the assumptions on the assignment of the refinement edges for elements in $\mathscr{T}_0$. All these algorithms refine a given triangulation iteratively.

In 2d, Mitchell already described a recursive variant [40] and Kossaczký converted the 3d algorithm by Bänsch into a recursive one [33]. This recursive algorithm was then generalized to any space dimension by Maubach [39] and Traxler [58]. Using the recursive variant needs slightly stronger assumptions on the distribution of refinement edges on $\mathscr{T}_0$. On a first glance this seems to be a drawback of the recursive variant. But firstly, these assumptions permit the derefinement of any refinement $\mathscr{T}$ of $\mathscr{T}_0$ back into $\mathscr{T}_0$ which is not true in general for the iterative variant. Secondly, similar assumptions on the distribution

of refinement edges on $\mathscr{T}_0$ are used to prove a crucial complexity result for the adaptive algorithm. This result mainly states that in any iteration of an adaptive method the total number of produced elements is always proportional to the sum of the numbers of selected elements. This was first proved by Binev et al. in 2d [9] and was generalized recently by Stevenson to any dimension [54].

3.2. Submeshes

In this section, we present our concept of a subtriangulation or submesh and discuss features of the approach and present a refinement algorithm for simultaneously refining bulk and surface meshes. Our concept for submeshes is motivated by the following natural demands.

(1) A submesh $\mathscr{S}$ should be subordinate to exactly one master mesh $\mathscr{T}$ and refinement/coarsening of both meshes should be done simultaneously in such a way that the submesh property is always preserved. Adaptive methods should be usable on master and/or submesh and the automatic local refinement/coarsening should be performed fully automatically on both meshes. In addition, the underlying refinement algorithms for both meshes should be according to the bisectional algorithm presented above, no special treatment should be necessary.

(2) Besides some dimension dependent parts, like refinement, DOF administration, definition of local basis functions, etc., most tasks in a finite element code can be realized in the same fashion regardless of the problem dimension. The dimension dependent parts are usually hidden in the library, and this permits creation of general purpose code valid for all dimensions. This concept is to be maintained for submeshes. This means, submeshes should be treated as regular mesh objects, endowed with some special properties. All methods and routines for single meshes, i.e., assembly of systems, numerical quadrature, mesh adaption, etc., must also be available for submeshes. Thus, a general purpose code can be used for both the master mesh and the submesh.

(3) When coupling bulk and surface discretization one needs bulk data on the surface and vice versa. This means that we need efficient trace and prolongation operators. For finite element data such operators can directly be constructed from a suitable injective mapping of surface DOFs to bulk DOFs.

In order to meet the above design concept we use the following notion of master and submesh.

Definition 3.1 (Submesh)**.** *Let $\mathscr{T}$ be a conforming d-dimensional triangulation of a domain Ω, $d = 2,3$, the* master mesh*. A* submesh *of $\mathscr{T}$ is a $(d-1)$-dimensional triangulation $\mathscr{S}$ iff for all elements $S \in \mathscr{S}$ it holds that S is a $(d-1)$-dimensional subsimplex of some element $T \in \mathscr{T}$.*

A direct consequence is that each element $S \in \mathscr{S}$ has to be assigned and subordinate to exactly one master element $T \in \mathscr{T}$. This then in turn implies that any submesh of a

conforming master mesh is also conforming. Definition 3.1 also guarantees an injective mapping of vertices of $\mathscr{S}$ to vertices of $\mathscr{T}$, or more generally an injective mapping of DOFs defined on $\mathscr{S}$ to DOFs defined on $\mathscr{T}$ when the surface space is the trace of a corresponding bulk space.

In the context of adaptive methods, we start with a master triangulation $\mathscr{T}_0$ of Ω and given submesh $\mathscr{S}_0$ of $\mathscr{T}_0$ and let

$$\Gamma := \bigcup_{S \in \mathscr{S}_0} S \subset \bar{\Omega}.$$

Let $\mathscr{T}$ be any triangulation of Ω derived through bisectional refinement according to the above algorithm. Then there necessarily exists a corresponding triangulation $\mathscr{S}$ of Γ, which is a submesh of $\mathscr{T}$. As we shall see below, a suitable tagging of refinement edges on $\mathscr{S}_0$ implies that $\mathscr{S}$ may also be constructed from $\mathscr{S}_0$ using the bisectional refinement algorithm. This implies that a code does not need to carry out inelegant loops over meshes to collect, manage, or even carry out ad-hoc refinement of submesh elements to maintain the submesh property. The local numbering of submesh elements follows the established rules of Section 3.1. This drastically simplifies the code for adaptive refinement, as no distinction has to be made algorithmically between refining normal meshes and refining submeshes.

Any possible refinement of $\mathscr{S}_0$ is uniquely determined by the distribution of refinement edges, and thus the initial local numbering of vertices, on the macro triangulation. The goal to construct $\mathscr{S}$ by bisectional refinement from $\mathscr{S}_0$ becomes then simply a question of how to correctly set the local vertex numbering on a submesh macro triangulation. We will show in the following how this may be done.

The case of a 2d triangulation with a 1d submesh is trivial. In the case of a 3d triangulation with a 2d submesh we refer to Table 1. In this table we deduce from the local numbering of a master element the local numbering of all faces that might be submesh elements. The numbering depends on the type and orientation of the master tetrahedron, as well as the face. Denoting by $\boldsymbol{v}_0, \cdots, \boldsymbol{v}_d \in \mathbb{R}^d$ the vertices of a simplex T, the orientation of T is given as

$$\operatorname{sign}\det \begin{bmatrix} | & & | \\ \boldsymbol{v}_1 - \boldsymbol{v}_0 & \cdots & \boldsymbol{v}_d - \boldsymbol{v}_0 \\ | & & | \end{bmatrix} \in \{\pm 1\}.$$

Hence, the orientation is determined by the local numbering of the vertices and the face with local number i is located opposite the ith local vertex. The numbering scheme is presented as a mapping from local tetrahedron vertex numbers to submesh triangle numbers. See also Fig. 3 for an illustration.

The numbering of submesh elements is chosen such that the refinement edge of a master tetrahedron always coincides with the refinement edge of a submesh triangle attached to a face along the master refinement edge. As we shall see in Lemma 3.1 this also holds for any refinement of the master element and/or submesh element. Hence, this construction of a local numbering of surface elements derived from bulk elements maintains the submesh property upon any bisectional refinement of $\mathscr{T}_0$ with a corresponding bisectional

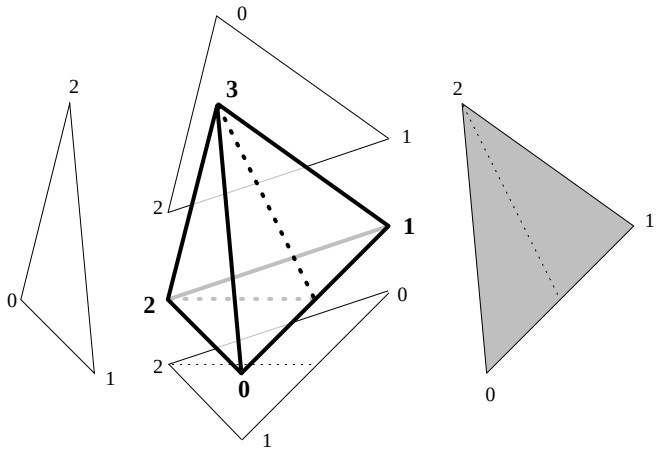

Figure 3: Local vertex numbering of submesh triangles along a 3d element of type 0 and positive orientation. As an example, the shaded triangle corresponds to the shaded box in Table 1.

Table 1: Local vertex numbering of submesh triangles along a 3d element. Each series of four numbers states how local vertices 0,1,2,3 of the tetrahedron are mapped to one of the local vertices 0,1,2 of the triangle. The shaded box corresponds to the example face given in Fig. 3.

Types	0		1, 2	
Orientation	+	−	+	−
Master face 0	-, 1, 2, 0	-, 0, 2, 1	-, 0, 1, 2	-, 1, 0, 2
Master face 1	1, -, 0, 2	0, -, 1, 2	1, -, 0, 2	0, -, 1, 2
Master face 2	0, 1, -, 2	1, 0, -, 2	0, 1, -, 2	1, 0, -, 2
Master face 3	1, 0, 2, -	0, 1, 2, -	1, 0, 2, -	0, 1, 2, -

refinement of $\mathscr{S}_0$.

A couple of important remarks are in order. First, note that the local vertex numbering of submesh triangles is oriented in such a way that the vector $\overline{01} \times \overline{02} \in \mathbb{R}^3$ always points away from the master tetrahedron, where $\overline{ab}$ is the position vector linking the vertices a and b. This is an arbitrary choice useful in applications where the domain boundary is a submesh and the code needs to be aware of which direction points outside of the domain. Another observation is that the enumeration of a submesh triangle does not differentiate between the triangle being a macro element or a child element derived by bisectional refinement. Finally, care must be taken that a submesh element is bound to precisely one master element, see the implementation goals above. This guarantees the unique numbering scheme.

Lemma 3.1 (Conservation of Submesh Property in 3d)**.** *Let $\mathscr{T}_0$ be a master mesh with a submesh $\mathscr{S}_0$ according to Definition 3.1. Let the local numbering of all submesh triangles be given according to Table 1 and let $\mathscr{T}$ be a triangulation obtained from $\mathscr{T}_0$ using bisectional refinement.*

Then there exists a triangulation $\mathscr{S}$ such that $\mathscr{S}$ is a submesh of $\mathscr{T}$ and $\mathscr{S}$ is constructed from $\mathscr{S}_0$ with bisectional refinement. In addition, for any pair of master and submesh elements

the local numbering coincides with Table 1.

Proof. The core of the bulk bisectional refinement procedure is the splitting of a tetrahedron into two child tetrahedra. We are specifically interested in the case of a parent tetrahedron lying along Γ. The basic idea of the proof is to examine that this case yields a valid result after the refinement of the parent tetrahedron (and possibly of the adjoining submesh triangle) and use an induction principle to complete the proof.

Table 1 shows that $0-1$ edges of tetrahedra always line up with $0-1$ edges of submesh triangles. This implies that the intersection of a refinement patch of $\mathscr{T}_0$ with Γ will always be a refinement patch of $\mathscr{S}_0$, hence it is always possible to match child submesh triangles with master mesh child tetrahedron faces. As mentioned, we examine in detail what occurs when a single tetrahedron $T \in \mathscr{T}_0$ of type $t \in \{0,1,2\}$ having orientation $p \in \{+,-\}$ bordering on a submesh element $S \in \mathscr{S}_0$ on face $f \in \{2,3\}$ is refined. There are 8 possible cases according to Table 1.

We restrict the study to the exemplary case $p = +$, $t = 0$, $f = 2$, which corresponds to the shaded entries in Fig. 3 and Table 1. The other cases are left to the reader. Splitting the tetrahedron according to Fig. 2 results in two child elements T_0, T_1. We also split the submesh triangle into two children S_0, S_1 according to Fig. 1. Since refinement edges line up as already observed, the submesh property is maintained. It remains to verify that the newly refined submesh has a numbering again given by Table 1.

The situation for T_0 is as follows: T_0 has orientation $+$ and type 1. T_0 vertex 0 faces S_0 vertex 1, T_0 vertex 2 faces S_0 vertex 0, T_0 vertex 3 faces S_0 vertex 2. S_0 lies on face 1 of T_0. A look at line 2, column 3 of Table 1 confirms this to be the correct numbering for this situation.

The situation for T_1: T_1 has orientation $+$, type 1, as well. T_1 vertex 0 faces S_1 vertex 0, T_1 vertex 1 faces S_1 vertex 1, T_1 vertex 3 faces S_1 vertex 2. S_1 lies on face 2 of T_0. This is confirmed by line 3, column 3 of the table. □

3.3. Simultaneous refinement of master mesh and submesh

As stated in the design goals of Section 3.2 we wish to automatically refine and coarsen bulk mesh and submeshes *simultaneously*, which means that the submesh property is preserved throughout an adaptive simulation. Any mesh, whether master mesh or submesh, may be refined or coarsened. Since coarsening is essentially the inverse operation to refinement, we will concentrate on the latter. When dealing with master and submeshes, one may even desire submeshes of submeshes, allowing whole hierarchies of dependent meshes starting from one top-level master mesh, see Fig. 4.

Using the iterative variant of bisectional refinement for simultaneous mesh adaptation is straight forward. Relying on the recursive variant we only want to make use of atomic refinement operations for both bulk and surface mesh. Doing this for a 2d bulk mesh coupled to a 1d surface mesh is obvious. For a 3d bulk mesh we observe that performing the atomic refinement operation in the bulk mesh means that all involved elements share the same refinement edge. Thanks to Lemma 3.1, the refinement edges of submesh elements

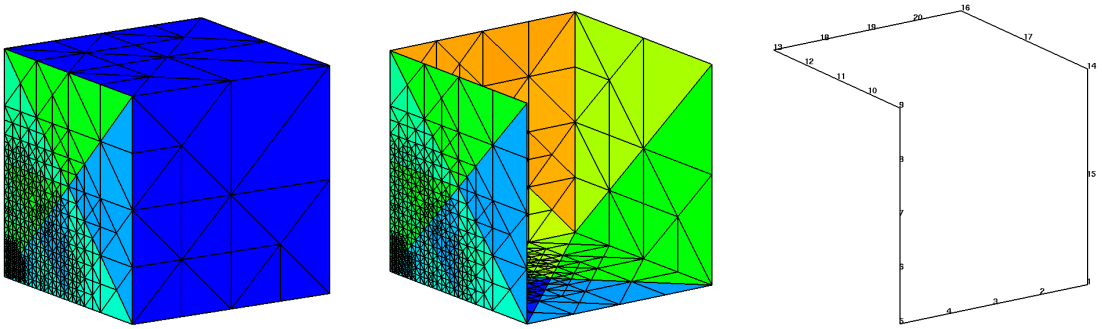

Figure 4: Entire hierarchies of submeshes may be defined.

always line up with the refinement edges of the assigned master elements. This implies that submesh elements within a refinement patch of the master mesh also build an admissible patch for the atomic refinement operation in 2d. This observation is the basis of the generalization of the Algorithm 3.2 accounting for submeshes and which we describe next.

We are going to successively update the submesh triangulation by atomic refinement operations whenever an atomic refinement operation of the master triangulation occurs in the course of Algorithm 3.1. If a top-level master mesh is to be refined, then the refinement algorithm is carried out as for single meshes. We create corresponding submesh refinement patches of the submesh elements adjoining any master patches. The submesh patches are then refined (in the process forcing in turn any refinement of subordinated submesh patches in a hierarchy).

If a submesh is to be refined, we first transfer refinement markers of the submesh elements to the corresponding master elements. Control is then transferred to the mesh refinement routine applied to the master mesh. Once we reach the top-level master mesh we proceed as in the prior paragraph. The submesh refinement markers are reset during the refinement of the master meshes. The refinement algorithms are thus updated as follows:

Algorithm 3.3 (Recursive refinement of one simplex with submeshes)**.**

```
subroutine submesh_recursive_refine(T, 𝒯)
  do
    𝒜 := {T' ∈ 𝒯; T' is not compatibly divisible with T}
    for all T' ∈ 𝒜 do
      submesh_recursive_refine(T', 𝒯);
    end for
  until 𝒜 = ∅

  𝒜 := {T' ∈ 𝒯; T' is element at the refinement edge of T}
  for all T' ∈ 𝒜
    bisect T' into T'_0 and T'_1
    𝒯 := 𝒯\{T'} ∪ {T'_0, T'_1}
  end for
```

```
for all 𝒮 submesh of 𝒯 do
  S' := T ∩ ⋃_{S∈𝒮} S
  if S' ∈ 𝒮 then
    submesh_recursive_refine(S', 𝒮)
  end if
end for
```

The refinement of a given triangulation $\mathscr{T}$ where some or all elements are marked for refinement is then performed by

Algorithm 3.4 (General refinement with submeshes)**.**

```
subroutine submesh_refine(𝒯)
  if 𝒯 is submesh of a mesh 𝒰 then
    do
      𝒜 := {T ∈ 𝒰 | T has a face S ∈ 𝒯 marked for refinement}
      mark all elements of 𝒜 for refinement
      submesh_refine(𝒰)
    until 𝒜 = ∅
  end if
  for all T ∈ 𝒯 do
    if T is marked for refinement
      submesh_recursive_refine(T, 𝒯)
    end if
  end for
```

The second algorithm contains a `do` loop where we must guarantee that the exit criterion is fulfilled at some point. All $(d-1)$-dimensional subsimplices of a d-dimensional simplex are bisected after d successive element refinements. This guarantees that any submesh element is refined at least once after a d refinements of the assigned master element, whence the above loop terminates after a finite number of iterations.

4. Applications

We implemented the submesh concept within the finite element toolbox ALBERTA [48, 49]. We first describe some implementational aspects for linking bulk and surface grids, then give a very simple model problem to motivate how submeshes can be used and conclude the article by presenting numerical results for the applications described in §2. All applications are implemented within the current version ALBERTA 2.0, which, in addition to former versions, gives access to coupled bulk and surface grids. ALBERTA is freely available at `http://www.alberta-fem.de/`.

4.1. Implementational details

Submeshes are defined in ALBERTA on the initial triangulation by letting the user specify a callback routine. This routine implements the decision which sides, i.e., edges in 2d and faces in 3d, of elements in the master mesh are to become elements of the submesh. These sides may belong to the boundary of Ω as well as to its interior. Input to this callback routine are the current macro master element, the current side (numbered as $0, \cdots, d$, with d as the master mesh dimension), and user data that may be necessary for the decision. The function must return true if the side is part of the submesh, otherwise the return value is false. This function is called for each master macro element S and each of its $(d+1)$ sides to build a submesh. The remaining information, for instance neighbor and coordinate information, etc., is automatically calculated by ALBERTA.

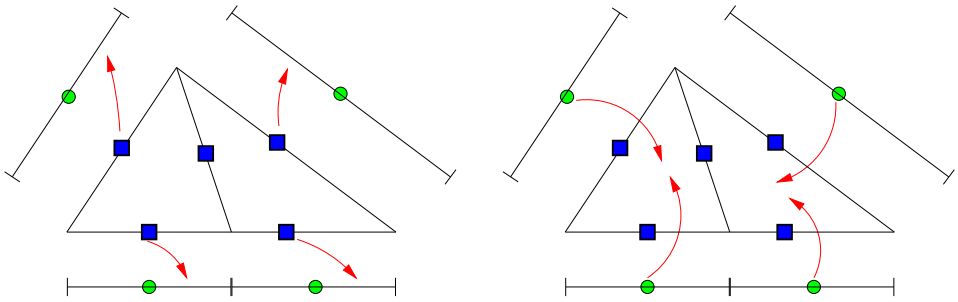

Figure 5: Used pointers from master element to submesh element (left) and vice versa (right).

Subsequent refinement of the master mesh or the submesh are naturally possible. During an adaptive simulation, where any mesh may be coarsened or refined, ALBERTA maintains two data structures for each submesh instance. The first is an array of pointers that links each submesh element to the corresponding master element. The second structure is an array of pointers essentially going in the opposite direction, with a pointer based on each master subsimplex and pointing either to nothing, or to a corresponding submesh element if present. Fig. 5 shows both pointer structures as red arrows on a simple mesh/submesh pair. These structures create some additional overhead in terms of memory usage and processing time during the refinement and coarsening process, but only if the submesh feature is actually used.

The pointer structures are used during the refinement process. As described above, refinement is based on the master mesh, with the submesh updated accordingly. Once an entire refinement patch on the master mesh is identified, the master-to-submesh pointers are used to build a corresponding submesh refinement patch. During the bisection of all patch elements the locally available information of both pointer structures is used to create new pointers between child master and submesh elements.

These pointer structures are also used for implementing an injective DOF mapping J from DOFs of the submesh to the corresponding DOFs of the master mesh. This mapping can be used to implement trace operators as well as prolongation operators, for instance the trivial prolongation by zero. The administration of this DOF mapping is currently

implemented for standard Lagrange finite element spaces of equal degree p defined on master and submesh. The use of this DOF mapping is illustrated in detail below.

4.2. Robin boundary conditions

We present a simple example illustrating the use of submeshes. We consider Laplace's equation with Robin type boundary conditions, where volume and surface integrals have to be computed during the assemblage of the linear system for computing the discrete solution.

In a FEM code one normally has standard routines for the assemblage of load vectors and system matrices. The example of Robin boundary conditions highlights some of the advantages of using a finite element space Y_M defined on a submesh $\mathscr{S}$. On the upside, we can create simple and clear code by using standard assemblage routines on bulk and surface grid. Traversing the smaller submesh instead of the bulk mesh may even reduce computational effort. On the downside, we now need to store and manage a submesh $\mathscr{S}$ as well as the pointer structures described in Section 4.1 above.

Let $\Omega \subset \mathbb{R}^d$ triangulated by $\mathscr{T}_0$ and with boundary $\Gamma = \partial\Omega$. The strong version of the problem is

$$-\Delta u = 0 \quad \text{in } \Omega, \qquad \text{and} \qquad \alpha u + \nu \cdot \nabla u = g \quad \text{on } \Gamma, \tag{4.1}$$

with a given function $g : \Gamma \to \mathbb{R}$, outer unit normal vector $\nu : \Gamma \to \mathbb{R}^d$, and a constant $\alpha > 0$. For a variational formulation we will assume $g \in L_2(\Gamma)$. We then seek a solution $u \in H^1(\Omega)$ such that

$$\int_\Omega \nabla\varphi \cdot \nabla u + \alpha \int_\Gamma \gamma\varphi\, \gamma u = \int_\Gamma \gamma\varphi\, g$$

for all $\varphi \in H^1(\Omega)$. Hereafter, $\gamma\colon H^1(\Omega) \to L_2(\Gamma)$ denotes the trace operator. The inhomogeneous Robin boundary condition therefore leads to surface integrals on both sides of the weak formulation. Discretization of this variational problem using the finite element method involves approximating $H^1(\Omega)$ by a finite-dimensional subspace $X_N \subset H^1(\Omega)$ with basis $\{\varphi_1, \cdots, \varphi_N\}$. We choose X_N to be the space of standard globally continuous Lagrange elements of order p on the triangulation $\mathscr{T}$ of Ω.

This leads to a linear system in $\mathbb{R}^N$ of the form

$$\boldsymbol{A}\,\boldsymbol{u} = \boldsymbol{f},$$

where we have used the system matrix

$$A_{ij} = \int_\Omega \varphi_i\, \varphi_j + \int_\Gamma \gamma\varphi_i\, \gamma\varphi_j \in \mathbb{R}^{N\times N},$$

the coefficient vector

$$\boldsymbol{u} = (u_1, \cdots, u_N)^t \in \mathbb{R}^N,$$

and the load vector $\boldsymbol{f}$

$$\boldsymbol{f} = (f_1, \cdots, f_N)^t \in \mathbb{R}^N, \qquad f_i = \int_\Gamma \gamma\varphi_i\, g.$$

Assume that $\mathscr{S}$ is a submesh of $\mathscr{T}$ triangulating the boundary Γ. We define

$$Y_M = \text{span}\{\gamma\varphi \mid \varphi \in X_N\}.$$

Thanks to the submesh property we then have that Y_M is exactly the corresponding space of Lagrange elements of order p on the triangulation $\mathscr{S}$ of Γ. The spatial position of the DOFs of functions in Y_M coincides with the position of corresponding DOFs of functions in X_N.

We denote $\{\psi_1, \cdots, \psi_M\}$ as the basis of Y_M given by the collection of all nonzero $\gamma\varphi_j$. Define an injective mapping of indices J with the following properties:

$$\begin{aligned} J : \{1, \cdots, M\} &\to \{1, \cdots, N\}, \\ \psi_i &= \gamma\varphi_{J(i)} \qquad \text{for all } i = 1, \cdots, M. \end{aligned}$$

Making use of these properties, we can implement the assemblage of the load vector and system matrix as follows. Define temporary quantities $\boldsymbol{B}$ and $\boldsymbol{h}$ associated with Y_M as

$$\begin{aligned} \boldsymbol{B} &\in \mathbb{R}^{M\times M}, \qquad B_{ij} = \int_\Gamma \psi_i \psi_j, \\ \boldsymbol{h} &\in \mathbb{R}^{M}, \qquad h_i = \int_\Gamma \psi_i g. \end{aligned}$$

The point to note is that both temporary quantities are easily calculated using software routines for the standard FEM tasks of assembling system matrices and right hand sides on arbitrary meshes. Let $j \in \{1, \cdots, N\}$. We have

$$\int_\Gamma \gamma\varphi_j \, g = \int_\Gamma \gamma\varphi_{J(i)} \, g = \int_\Gamma \psi_i \, g = h_i$$

if $j = J(i)$ for some $i \in \{1, \cdots, N\}$ and

$$\int_\Gamma \gamma\varphi_j \, g = 0$$

otherwise. Similar properties hold for the mass matrix $\boldsymbol{B}$. We may therefore perform the assembling of the bulk linear system using the following simple algorithmic steps

$$\begin{aligned} f_i &:= 0 \qquad \text{for } i = 1, \cdots, N, \\ f_{J(k)} &:= f_{J(k)} + h_k \qquad \text{for } k = 1, \cdots, M, \end{aligned}$$

and

$$\begin{aligned} A_{ij} &:= \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j \qquad \text{for } i, j = 1, \cdots, N \\ A_{J(k)J(l)} &:= A_{J(k)J(l)} + B_{kl} \qquad \text{for } k, l = 1, \cdots, M. \end{aligned}$$

4.3. Flow with free capillary surfaces

We are interested in a sharp interface ALE-type approach where the domain essentially follows the motion of the fluid. This is most promising because of its simplicity, since the physical flows to be modeled are not expected to lead to domain topology changes, e.g. the formation of cusps. Examples of more general methods are Volume-of-Fluid [31] or the more recent Level Set methods [44, 50, 51, 56, 57]. For the treatment of the capillary stress boundary condition (2.1c) we opt for a variational formulation of the curvature term, [23, 24], thus extending standard finite element ideas to this context.

What makes this example interesting for this article is that the variational formulation requires the assembling of a stiffness term

$$\int_\Gamma \nabla_\Gamma(\gamma\boldsymbol{\varphi}) : \nabla_\Gamma(\gamma\boldsymbol{v}),$$

where $\nabla_\Gamma(\gamma\boldsymbol{v})$ is the surface gradient acting on the trace of the unknown velocity field on the boundary, $\boldsymbol{\varphi}$ a given basis function, and γ the trace operator. Note that this term results from the coupling of the volume flow equations with the capillary boundary condition. The contributions created by this term need to be added to the bulk system, compare the analogous mass term in the Robin boundary value problem above. In fact, we implemented this in the same way as for the Robin problem.

Let $t_0 = 0, t_1, t_2, \cdots$ be a division of the time interval $[0, T]$. The solution at time step t_n is given by $(\boldsymbol{v}^n, p^n)$ on the domain Ω^n. The time discretization scheme is to loop over the $n = 1, 2, \cdots$:

1. Solve for $(\boldsymbol{v}^n, p^n)$ on the domain Ω^n;
2. Update the free boundary Γ^n using $\boldsymbol{v}^n|_{\Gamma_n}$ to define Γ^{n+1} and Ω^{n+1};
3. Remesh Ω^{n+1} if necessary,

compare [34, 35] for a more precise description of the algorithm.

As an example we study the motion of initially deformed droplets and use data as specified in [7]. We define starting domains which approximate ellipses in 2D and ellipsoids in 3D with main axes along the coordinate axes. Since the mean curvature of $\Gamma = \partial\Omega$ is larger at the tips corresponding to smaller radii and vice versa one observes oscillatory motion with axial/planar symmetry, with conserved domain volume, see Fig. 9. The data used is

- All cases: initial velocity $\boldsymbol{v}_0 = 0$, time interval $[0, 7]$, constant of surface tension $\sigma = 1$.
- 2D case:
  - $\Omega_0 \subset \mathbb{R}^2$ as an ellipse with principal radii $r_1 = 1$, $r_2 = 1.2$.
  - Viscosity coefficient: $\eta = 1/300$.

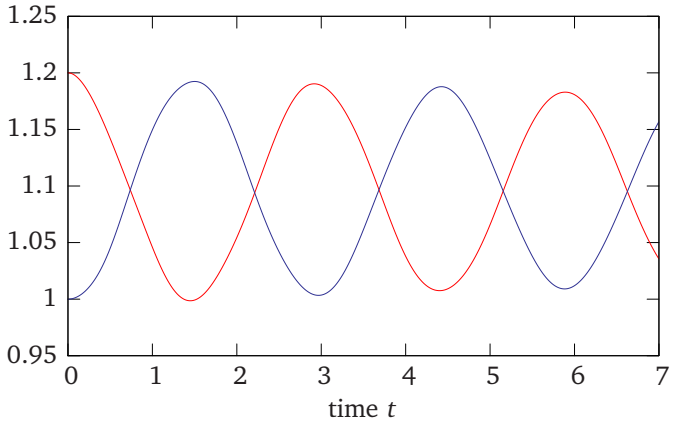

Figure 6: Trajectories of the 2D droplet tips.

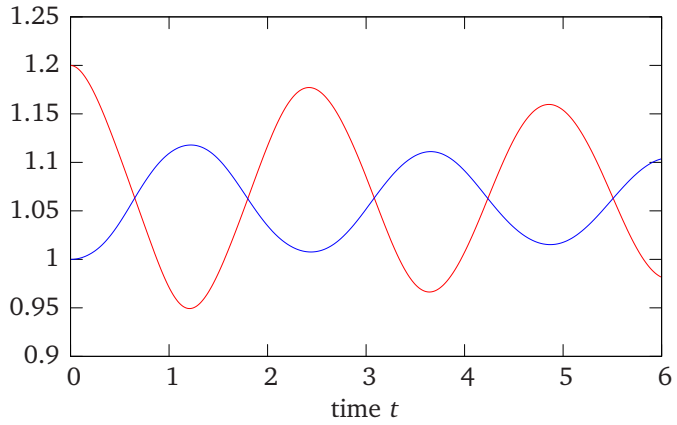

Figure 7: Trajectories of the 3D droplet tips, first case.

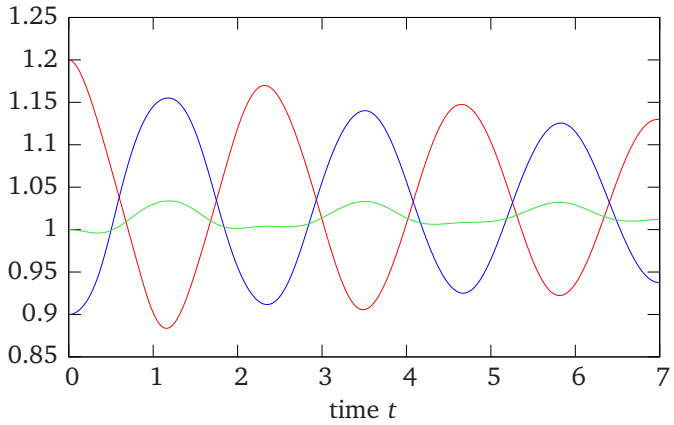

Figure 8: Trajectories of the 3D droplet tips, second case.

- First 3D case:
  - $\Omega_0 \subset \mathbb{R}^3$ as an ellipsoid. Principal radii were $r_i = 1, 1, 1.2$
  - Viscosity coefficient: $\eta = 1/300$.
- Second 3D case:
  - $\Omega_0 \subset \mathbb{R}^3$ as an ellipsoid. Principal radii were $r_i = 0.9, 1.0, 1.2$.
  - Viscosity coefficient: $\eta = 1/200$.

In the simulation we measured the motion of the tips with time. Fig. 6 shows a plot of the trajectories of the 2D droplet tips over time. Figs. 7 and 8 show corresponding plots for the two 3D cases. The results obtained coincide very well with those of [7], and we were able to confirm the conservation of droplet volume as well as the period of the oscillation.

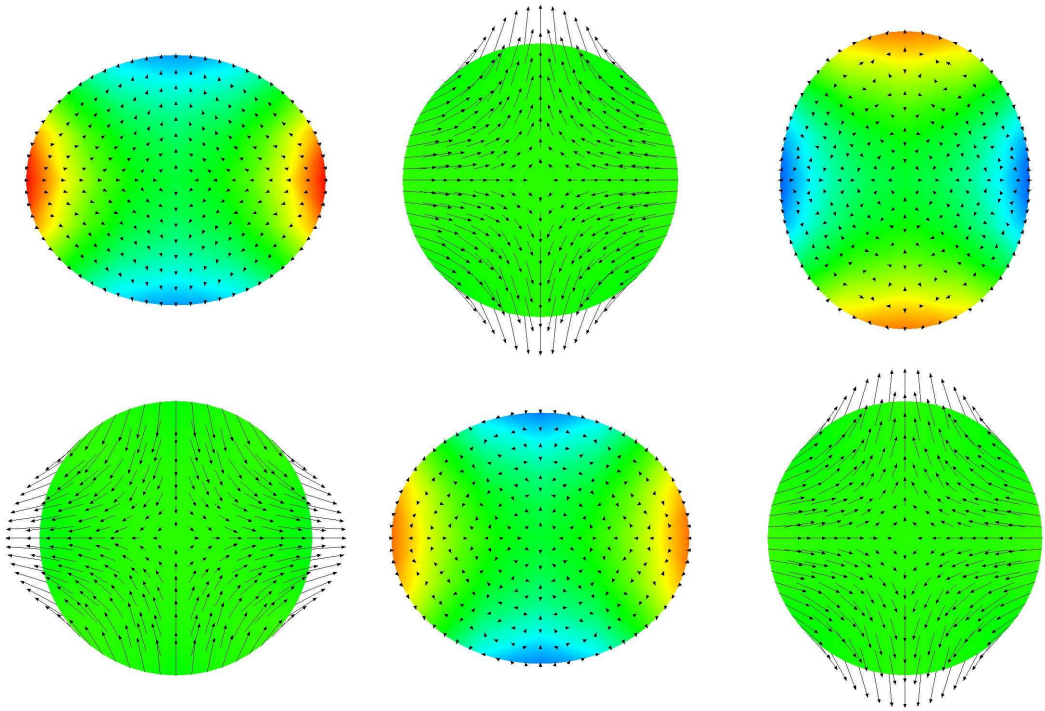

Figure 9: Solution of the 2D droplet at the times $t_1 = 0.025$, $t_2 = 0.725$, $t_3 = 1.475$, $t_4 = 2.2$, $t_5 = 2.95$, $t_6 = 3.675$. The color indicates pressure and the arrows velocity.

4.4. Liquid phase epitaxial growth

Also in this application we use a sharp interface model for the simulation of the liquid phase epitaxial growth which allows for a precise tracking of the thickness of the epitaxial layer. We follow the approach that the initial bulk mesh is aligned to the initial interface which in turn then defines the surface mesh. Similar to the previous section, the bulk and surface mesh may follow the motion of the discrete interface that always separates solid

and liquid phase. This means, that in each time step the new position of the interface is determined and coordinates of bulk and surface mesh are updated accordingly. On the other hand, the thickness of the layer is small compared to the size of the melting pot, and thus tracking the interface may be omitted. Both approaches are analyzed in [36] yielding similar results, and hereafter, we use the second variant with fixed liquid and solid phases Ω^ℓ, Ω^s and interface Γ.

As outlined in Section 2.2, we want to enrich continuous finite element space for the concentrations by suitable functions at the interface to account for jumps in between solid and liquid phase. Following the idea of [19] we multiply all continuous Lagrange basis functions with the Heaviside function

$$H(x) = \begin{cases} 1, & \text{if } x \in \bar{\Omega}^s, \\ 0, & \text{if } x \in \bar{\Omega}^\ell \setminus \Gamma. \end{cases}$$

The enriched finite element space exactly allows for jumps across Γ. A basis can easily be constructed by the standard nodal basis where one has to add and modify exactly basis functions with non-zero trace on Γ. The additional degrees of freedom can easily be stored via the submesh.

We formulate the system (2.2) in a variational way, and discretize the weak form using the classical Taylor-Hood element for velocity and pressure, continuous Lagrange finite elements for the temperature, and the enriched finite element space for the concentrations. The discrete nonlinear problem involves integrals of the form

$$-\int_\Gamma V\, \gamma\varphi_{J(j)} \quad \text{and} \quad -\int_\Gamma V\, \left[c_i\right]_s^\ell\, \gamma\varphi_{J(j)}, \quad j \in \{1, \cdots, M\}$$

with basis functions $\varphi_k \in X_N$ and γ, X_N defined as in Section 4.2. These integrals stem from integration by parts and the Stefan conditions (2.2g). Furthermore, the weak formulation of (2.2e) ((2.2f) analogously) yields

$$\int_\Gamma \gamma\theta\, \psi_j = \int_\Gamma \left(1.36 - 0.56c_2^\ell + 0.65c_1^\ell - \frac{1.03c_1^\ell}{1-c_2^\ell}\right) \psi_j, \quad j \in \{1, \cdots, M\},$$

with basis functions $\psi_j \in Y_M$ and γ, Y_M defined as in Section 4.2.

In a time-stepping scheme, the nonlinear discrete problem is solved by a Newton method. Considering the equation and terms above for the assemblage of the Jacobian several integrals of the form

$$\int_\Gamma w\, \psi_i\, \psi_j, \quad i, j \in \{1, \cdots, M\}$$

with different weighting functions w have to be calculated and added to the system matrix. This can easily and efficiently be realized with standard tools as presented in Section 4.2.

An efficient solution of the nonlinear discrete system is rather demanding. The dimension may become very large, already in two space dimensions. Furthermore, the Jacobian

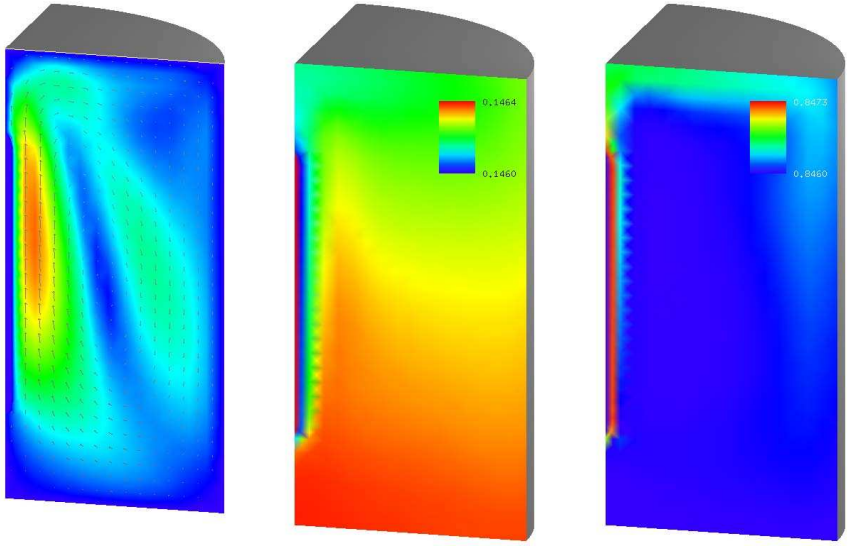

Figure 10: Results from 3d simulation in half of the melting pot, where the substrate is located on the left side. Displayed are (form left to right) the modulus of the flow velocity, concentrations c_1, and c_2.

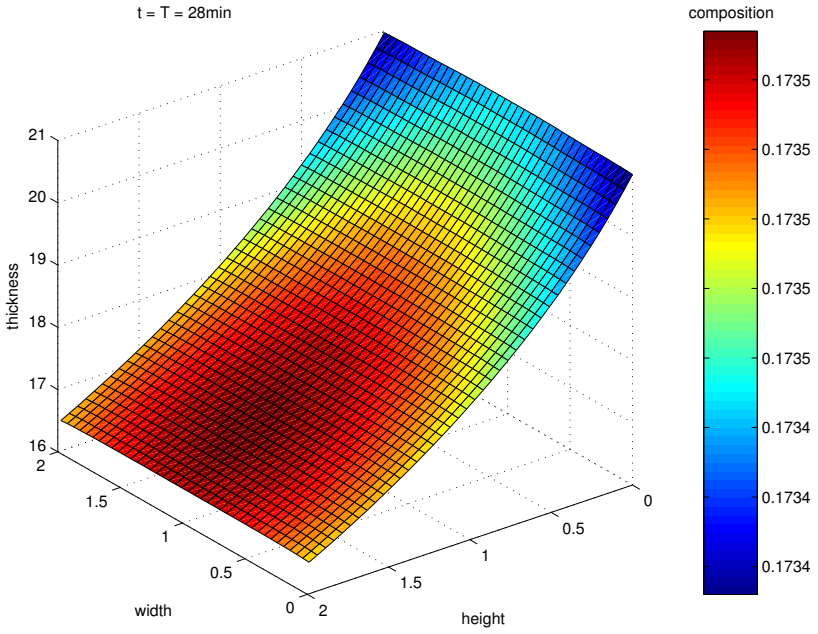

Figure 11: Homogeneity of the layer of the 3d simulation. The deviations from the desired homogeneity of the composition and layer thickness are acceptable.

used in the linear sub-problems of the Newton method is neither symmetric nor well conditioned. We tackle these challenges by applying robust Krylov space solvers in combination with powerful preconditioning techniques (cf. [46] for details). To be more precise, we

successfully combine transpose-free quasi-minimal residuals (TFQMR) with incomplete LU factorization (ILUT) which in combination also allows for 3-dimensional simulations. Finally, we have to face very undesirable oscillations of the discrete interface velocity. They are amplified through the nonlinear phase diagram equations (2.2e) and (2.2f). Combining a high grid resolution near the interface with a suitable penalizing term, similar to those in [13, 22], we were able to strongly damp these oscillations. The penalty term also involves integrals over the interface, which can be efficiently realized by the aforementioned tools.

Due to symmetry of the geometry, the solution only has to be computed within one half of the melting pot. Results from a 3d simulation are shown in Figs. 10 and 11. In Fig. 10 the flow field and the concentrations c_1, c_2 are displayed. The standards in industry for ensuring a good quality of the produced infrared detector are quite high. For the area of one square centimeter a 0.1% deviation from the perfect composition and a two micron deviation of the layer thickness should not be exceeded. Fig. 11 shows the thickness and the composition of the grown layer, which confirms that for the used parameters of the simulation the high standards are satisfied.

4.5. Minimal surfaces

We finally return to the minimal surface problem where we want to find a stationary point of the Dirichlet energy defined in (2.4). Since the Dirichlet integral is invariant under conformal reparameterizations one has to factor out the conformal group. This can be achieved by a three point condition, for instance fixing admissible parameterizations $s\colon \partial B \to S^1$ at three distinct points or removing the first three Fourier modes of s, compare [55] for details.

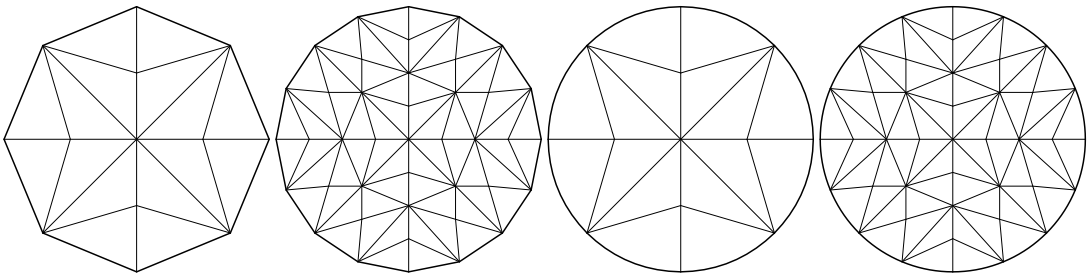

Figure 12: Comparison of affine meshes for the $\mathbb{P}_1$ discretization (left two pictures) and the iso-parametric meshes for the $\mathbb{P}_2$ discretization (right two pictures) after one respectively two global refinements of the initial grid. The higher quality of boundary approximation for the parametric grid is obvious.

Given a bulk triangulation $\mathscr{T}$ of the unit disc B, the induced surface grid $\mathscr{S}$ is a triangulation of ∂B. On the surface grid $\mathscr{S}$ we employ continuous finite elements for the discretization of s. A discrete stationary point of the Dirichlet integral is computed by a Newton method, where each Newton iteration requires the computation of several discrete harmonic extensions that are computed in a finite element space over the bulk grid $\mathscr{T}$. We refer to [21] for the precise algorithm and details. Defining the surface space as the trace of

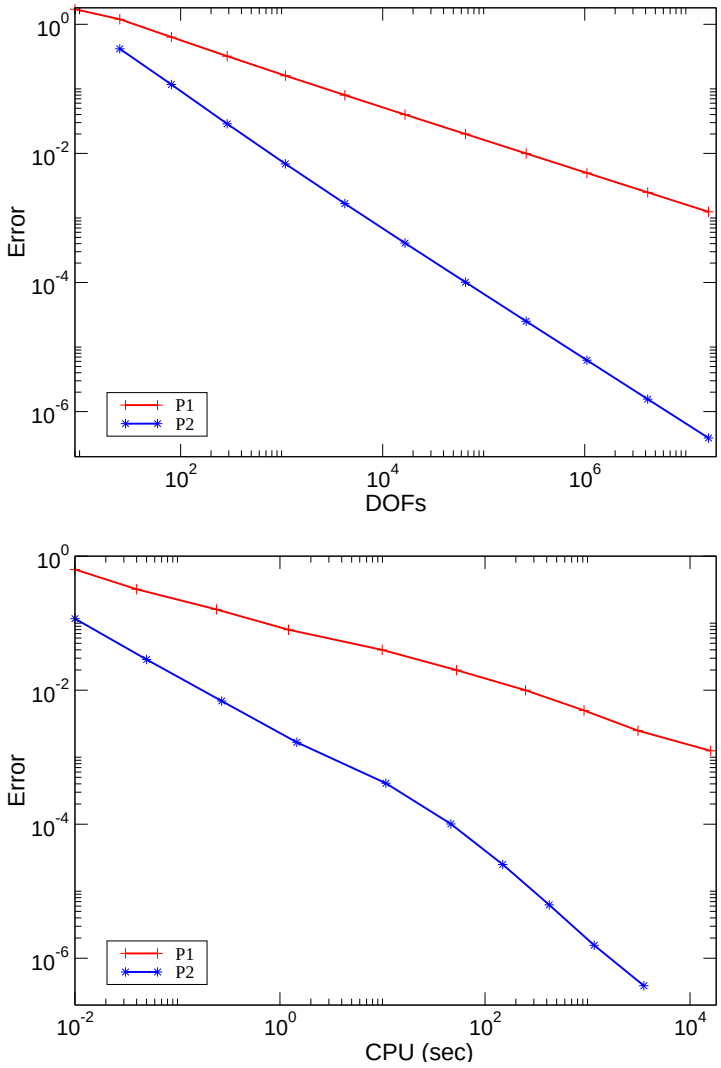

Figure 13: Stable Enneper surface: H^1 error of the parameterization $\boldsymbol{u}$ versus DOFs (top) and CPU time (bottom) for the $\mathbb{P}_1$ and $\mathbb{P}_2$ discretizations. The enormous profit from using a higher order discretization is apparent.

the bulk space makes exchange of data between bulk and surface spaces straight forward.

Since minimal surfaces are very regular objects, a discretization employing higher order elements deems to be a very promising idea and we are going to compare a $\mathbb{P}_1$ with a $\mathbb{P}_2$ discretization, i.e., we compare piecewise linear with piecewise quadratic finite elements for both the boundary and the bulk space. Relying on higher order approximations to s and $\boldsymbol{u}$ one also has to account for a higher order approximation of the curved boundary of ∂B by using iso-parametric elements for surface and bulk grids. Affine and iso-parametric grids for both discretizations are compared in Fig. 12. In the context of coupling bulk and surface grids, one has to use the same parameterization for both meshes. To our best

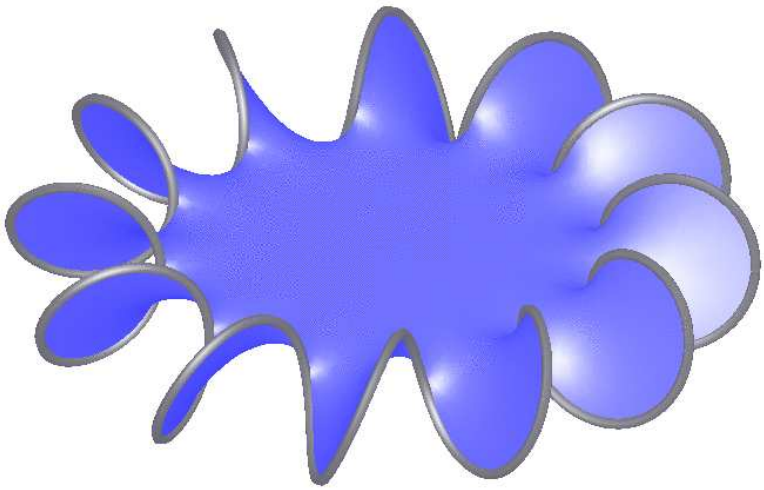

Figure 14: A minimal surface spanning a curve winding around a torus.

knowledge, this is the first higher order finite element approximation to this problem.

One minimal surface with an explicitly known parameterization is the famous Enneper surface. This surface we have used in order to benchmark the $\mathbb{P}_1$ and $\mathbb{P}_2$ discretizations. The results are shown in Fig. 13, where we have plotted the decay of the H^1 error for the parameterization $\boldsymbol{u}$ versus DOFs (top) and CPU time (bottom). In the left picture we can observe that the error decay is optimal, i.e., $N^{-1/2}$ for the $\mathbb{P}_1$ and N^{-1} for the $\mathbb{P}_2$ discretization, where N denotes the number of DOFs. The $\mathbb{P}_1$ discretization is clearly outmatched by the higher order method. This effect is even more prominent, when relating error to CPU time. Although any higher order is in general more costly, the $\mathbb{P}_2$ discretization benefits in this example dramatically from less Newton iterations. The beauty of minimal surfaces is apparent from Fig. 14, which presents a minimal surface spanning a curve winding around a torus.

References

[1] M. Ainsworth and J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis*, Wiley-Interscience, New York, 2000.

[2] M. H. Aliabadi, C. A. Brebbia, and J. Mackerle, eds., *The Boundary Element Reference Book. 2nd Edition*, Southampton: Computational Mechanics Publications, 1994.

[3] D. N. Arnold, A. Mukherjee, and L. Pouly, *Locally adapted tetrahedral meshes using bisection*, SIAM J. Sci. Comput., 22 (2000), pp. 431–448.

[4] R. J. Asaro and W. A. Tiller, *Surface morphology development during stress corrosion cracking: Part I: Via surface diffusion*, Metall. Trans., 3 (1972).

[5] I. Babuška and T. Strouboulis, *The Finite Element Method and its Reliability*, Clarendon Press, Oxford, 2001.

[6] E. Bänsch, *Local mesh refinement in 2 and 3 dimensions*, IMPACT Comput. Sci. Engrg., 3 (1991), pp. 181–191.

[7] E. Bänsch, *Finite element discretization of the Navier–Stokes equations with a free capillary surface*, Numer. Math., 88 (2001), pp. 203–235.
[8] E. Bänsch and A. Schmidt, *Simulation of dendritic crystal growth with thermal convection*, Interfaces and Free Boundaries, 2 (2000), pp. 95–115.
[9] P. Binev, W. Dahmen, and R. DeVore, *Adaptive finite element methods with convergence rates*, Numer. Math, 97 (2004), pp. 219–268.
[10] D. Braess, *Finite Elemente*, Springer Verlag, 1992.
[11] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer Verlag, 1994.
[12] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer Verlag, 1991.
[13] E. Burman and P. Hansbo, *Edge stabilization for Galerkin approximations of convection-diffusion-reaction problems*, Comp. Methods Appl. Mech. Engrg., 193 (2004), pp. 1437–1453.
[14] J. M. Cascón, C. Kreuzer, R. H. Nochetto, and K. G. Siebert, *Quasi-optimal convergence rate for an adaptive finite element method*. Preprint 009/2007, Universität Augsburg, 2007.
[15] P. G. Ciarlet, *The finite element methods for elliptic problems*, Classics in Applied Mathematics, 40, SIAM, 2002.
[16] D. Coutand and S. Shkoller, *Motion of an elastic solid inside an incompressible viscous fluid*, Arch. Rational Mech. Anal, 176 (2005), pp. 25–102.
[17] U. Dierkes, S. Hildebrandt, A. Küster, and O. Wohlrab, *Minimal surfaces I. Boundary value problems*, Springer, Berlin, 1992.
[18] ———, *Minimal surfaces II. Boundary regularity*, Springer, Berlin, 1992.
[19] J. Dolbow and R. Merle, *Solving thermal and phase change problems with the eXtended finite element method*, Comput. Mech., 28 (2002), pp. 339–350.
[20] W. Dörfler, *A convergent adaptive algorithm for Poisson's equation*, SIAM J. Numer. Anal., 33 (1996), pp. 1106–1124.
[21] W. Dörfler and K. G. Siebert, *An adaptive finite element method for minimal surfaces*, in Geometric analysis and nonlinear partial differential equations, S. Hildebrandt and H. Karcher, eds., Springer Berlin, 2003, pp. 147–175.
[22] J. Douglas and T. Dupont, *Interior penalty procedures for elliptic and parabolic Galerkin methods*, in Computing Methods in Applied Sciences, R. Glowinski and J. Lions, eds., Berlin, 1976, Springer.
[23] G. Dziuk, *Finite elements for the Beltrami operator on arbitrary surfaces*, in Partial Differential Equations and Calculus of Variations, S. Hildebrandt and R. Leis, eds., Springer Verlag, 1988, pp. 142–155.
[24] ———, *An algorithm for evolutionary surfaces*, Numer. Math., 58 (1991), pp. 603–611.
[25] G. Dziuk and J. E. Hutchinson, *The discrete Plateau problem: Algorithm and numerics*, Math. Comp., 68 (1999), pp. 1–23.
[26] ———, *The discrete Plateau problem: Convergence results*, Math. Comp, 68 (1999), pp. 519–546.
[27] ———, *Finite element approximations to surfaces of prescribed variable mean curvature*, Numer. Math., 102 (2006), pp. 611–648.
[28] W. Hackbusch, *Integral Equations: Theory and Numerical Treatment*, Birkhäuser, 1995.
[29] T. Harman, *Liquidus isotherms, solidus lines and LPE growth in the Te-rich corner of Hg-Cd-Te systems*, Journal of Electronic Materials, 9 (1980), pp. 945–961.
[30] J. J. Heys, T. A. Manteuffel, and J. W. Ruge, *First-order system leas square (FOSLS) for coupled fluid-elastic problems*, J. Comp. Phys., 195 (2004), pp. 560–575.
[31] C. W. Hirt and B. D. Nichols, *Volume of fluid (VOF) method for the dynamics of free boundaries*, J. Comp. Phys., 39 (1981), pp. 201–225.

[32] M. KIMURA, Z. QIN, AND S. DOST, *A solid-liquid diffusion model for growth and dissolution of ternary alloys by liquid phase epitaxy*, J. Crystal Growth, 158 (1996), pp. 231–240.
[33] I. KOSSACZKÝ, *A recursive approach to local mesh refinement in two and three dimensions*, J. Comput. Appl. Math., 55 (1994), pp. 275–288.
[34] D. KÖSTER, *Numerical Simulation of Acoustic Streaming on SAW-driven biochips*, Ph.D. thesis, University of Augsburg, 2006.
[35] D. KÖSTER, *Numerical simulation of acoustic streaming on saw-driven biochips*, SIAM J. Sci. Comput., 29 (2007), pp. 2352–2380.
[36] O. KRIESSL, *Efficient numerical simulation of the liquid phase epitaxy*, Ph.D. thesis, University of Augsburg, 2006.
[37] L. D. LANDAU AND E. M. LIFSCHITZ, *Lehrbuch der Theoretischen Physik VI: Hydrodynamik*, Akademie Verlag, 1991.
[38] A. LIU AND B. JOE, *Quality local refinement of tetrahedral meshes based on bisection*, SIAM J. Sci. Comput., 16 (1995), pp. 1269–1291.
[39] J. M. MAUBACH, *Local bisection refinement for n-simplicial grids generated by reflection*, SIAM J. Sci. Comput., 16 (1995), pp. 210–227.
[40] W. F. MITCHELL, *A comparison of adaptive refinement techniques for elliptic problems*, ACM Trans. Math. Softw., 15 (1989), pp. 326–347.
[41] P. MORIN, R. H. NOCHETTO, AND K. G. SIEBERT, *Convergence of adaptive finite element methods*, SIAM Review, 44 (2002), pp. 631–658.
[42] P. MORIN, K. G. SIEBERT, AND A. VEESER, *A basic convergence result for conforming adaptive finite elements*. Preprint 007/2007, Universität Augsburg, 2007. To appear in Math. Models Methods Appl. 18 (2008).
[43] J. C. C. NITSCHE, *Lectures on Minimal Surfaces*, vol. 1, Cambridge University Press, 1989.
[44] S. OSHER AND J. SETHIAN, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi-formulations*, J. Comp. Phys., 79 (1988), pp. 12–49.
[45] M.-C. RIVARA, *Mesh refinement processes based on the generalized bisection of simplices*, SIAM J. Numer. Anal., 21 (1984), pp. 604–613.
[46] Y. SAAD, *Iterative methods for sparse linear systems. 2nd ed*, Society for Industrial and Applied Mathematics, Philadelphia, 2003.
[47] A. SCHMIDT, *Computation of three dimensional dendrites with finite elements*, J. Comp. Phys., 125 (1996), pp. 293–312.
[48] A. SCHMIDT AND K. G. SIEBERT, *ALBERT - Software for scientific computations and applications*, Acta Math. Univ. Comenianae, 70 (2000), pp. 105–122.
[49] A. SCHMIDT AND K. G. SIEBERT, *Design of Adaptive Finite Element Software: The Finite Element Toolbox ALBERTA*, Springer Verlag, 2005.
[50] J. A. SETHIAN, *Level Set Methods and Fast Marching Methods*, Cambridge Monographs on Applied and Computational Mathematics, 1999.
[51] P. SMEREKA, *Level set methods for two-fluid flows*. Lecture notes from a short course given at INRIA, 1996.
[52] B. J. SPENCER, S. H. DAVIS, AND P. W. VOORHEES, *Morphological instability in epitaxially-strained dislocation-free solid films: nonlinear evolution*, Phys. Rev. B, 47 (1993), pp. 9760–9777.
[53] R. STEVENSON, *Optimality of a standard adaptive finite element method*, Found. Comput. Math., 7 (2007), pp. 245–269.
[54] ———, *The completion of locally refined simplicial partitions created by bisection*, Math. Comp., 77 (2008), pp. 227–241.
[55] M. STRUWE, *Plateau's Problem and the Calculus of Variations*, Princeton University Press, Princeton, 1988.

[56] M. Sussman, E. Fatemi, P. Smereka, and S. Osher, *An improved level set method for incompressible two-phase flows*, Computers and Fluids, 27 (1997).
[57] M. Sussman, P. Smereka, and S. J. Osher, *A level set approach for computing solutions to incompressible two-phase flow*, J. Comp. Phys., 114 (1994), pp. 146–159.
[58] C. T. Traxler, *An algorithm for adaptive mesh refinement in n dimensions*, Computing, 59 (1997), pp. 115–137.
[59] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*, Adv. Numer. Math., John Wiley, Chichester, UK, 1996.