A finite element method for surface diffusion: the parametric case

Eberhard Bänsch* Pedro Morin† Ricardo H. Nochetto‡

January 23, 2004

Abstract

Surface diffusion is a (4th order highly nonlinear) geometric driven motion of a surface with normal velocity proportional to the surface Laplacian of mean curvature. We present a novel variational formulation for parametric surfaces with or without boundaries. The method is semi-implicit, requires no explicit parametrization, and yields a linear system of elliptic PDE to solve at each time step. We next develop a finite element method, propose a Schur complement approach to solve the resulting linear systems, and show several significant simulations, some with pinch-off in finite time. We introduce a mesh regularization algorithm, which helps prevent mesh distortion, and discuss the use of time and space adaptivity to increase accuracy while reducing complexity.

Keywords: Surface diffusion, fourth-order parabolic problem, finite elements, Schur complement, smoothing effect, pinch-off.

AMS subject classification: 35K55, 65M12, 65M15, 65M60, 65Z05.

1 Surface Diffusion and its Formulation

The overall goal of this project is to devise efficient numerical tools for simulating morphological changes in stressed epitaxial films and thereby study their complicated nonlinear dynamics. To model the misfit between the crystalline structure of the substrate and epitaxial film, the film may be thought of as subjected to mechanical stresses. This causes a plastic deformation of the free

*Weierstrass Institute for Applied Analysis and Stochastics, Mohrenstrasse 39, 10117 Berlin, and Freie Universität Berlin, GERMANY (`baensch@wias-berlin.de`).

†Instituto de Matemática Aplicada del Litoral (IMAL) and Departamento de Matemática, Facultad de Ingeniería Química, Universidad Nacional del Litoral, Güemes 3450, 3000 Santa Fe, Argentina (`pmorin@math.unl.edu.ar`). Partially supported by CONICET of Argentina and NSF Grant DMS-9971450. This work was partly developed while this author was visiting the University of Maryland.

‡Department of Mathematics and Institute for Physical Science and Technology, University of Maryland, College Park, MD 20742, USA (`rhn@math.umd.edu`). Partially supported by NSF Grants DMS-9971450 and DMS-0204670.

surface of the film. This morphological instability of the free surface may eventually lead to crack formation and fracture, an issue of paramount importance in Materials Science; see for instance [1, 9, 25] and the list of references in [7].

The dynamics of the free surface $\Gamma(t) \subseteq \mathbb{R}^d$ is governed by the highly nonlinear PDE

$$V = -\Delta_S(\kappa + \varepsilon), \tag{1.1}$$

where $d = 2, 3$, V and κ are the (scalar) normal velocity and mean curvature of Γ, respectively, $\Delta_S = \mathrm{div}_S \nabla_S$ is the Laplace-Beltrami operator and ε is the elastic energy density of the bulk $\Omega(t)$ enclosed by $\Gamma(t)$. In this paper we consider the reduced *purely geometric* model for which ε is a given forcing function. Our goal is to present a novel variational formulation for parametric surfaces based on a semi-implicit time discretization, which requires no explicit parametrization of the surface and yields a linear system of elliptic PDE to approximate at each time step. We then develop a finite element method (FEM) and discuss mesh distortion and adaptivity. This endeavor may be viewed as a building block towards solving the fully coupled system.

We recall now two fundamental properties of motion by surface diffusion. The first one is *conservation of volume* for closed surfaces:

$$\frac{d}{dt}|\Omega(t)| = \int_{\Gamma(t)} V = -\int_{\Gamma(t)} \Delta_S(\kappa + \varepsilon) = \int_{\Gamma(t)} \nabla_S(\kappa + \varepsilon) \cdot \nabla_S 1 = 0. \tag{1.2}$$

The second property is *area decrease* for $\varepsilon = 0$ and suitable boundary conditions:

$$\frac{d}{dt}|\Gamma(t)| = -\int_{\Gamma(t)} V\kappa = -\int_{\Gamma(t)} |\nabla_S \kappa|^2. \tag{1.3}$$

In fact motion by surface diffusion is formally the H^{-1} gradient flow for the area functional (see [9]). It is desirable to preserve these essential properties under discretization, as the proposed FEM below does. This method also handles two striking features which can occur for surface diffusion in finite time: a surface which starts as a graph may cease to be so [17] (see Figure 1.1), and a closed embedded hypersurface may selfintersect [19] (see Figure 1.2).

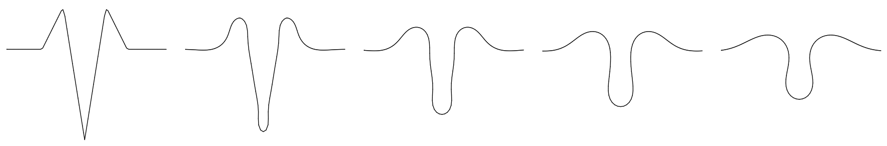
Figure 1.1: Evolution of a curve that ceases to be a graph in finite time.

A number of issues arise, from existence, well posedness and regularity to algorithm design for simulating (1.1), perhaps enforcing (1.2) and (1.3). In [18], Escher et. al. proved (local) existence, regularity, and uniqueness of solutions provided $\epsilon = 0$ and the initial surface is sufficiently smooth. They also proved that if the initial surface is embedded and close to a sphere, the solution exists

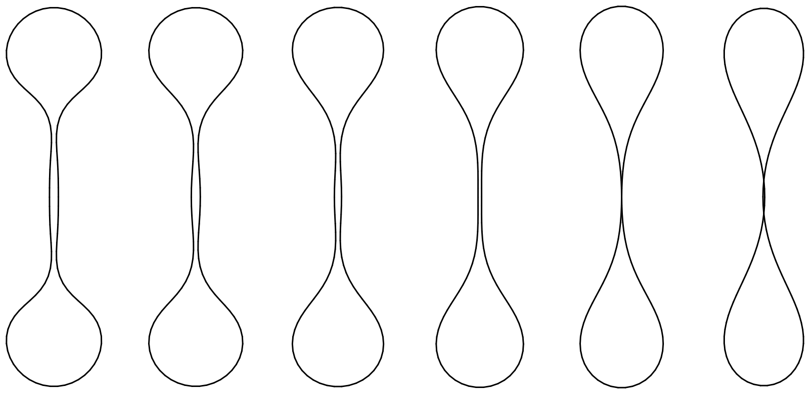

Figure 1.2: Evolution of an embedded curve which selfintersects in finite time.

globally and converges exponentially fast to a sphere. A fundamental mathematical obstruction to further progress arises from the 4th order nonlinear operator $\Delta_S \kappa$, which rules out maximum principle techniques.

A space-time finite element method for *axially symmetric* surfaces is presented by Coleman et al. in [11], along with several stability properties and very interesting dynamics, some not predicted by linearized stability. More recently, Deckelnick et al. provided an error analysis [15] for the axially symmetric case. The graph case was considered by Bänsch et al. [7] where an error analysis is derived for the space discretization, and this analysis was extended by Deckelnick et al. [14] to a fully discrete method for anisotropic surface diffusion of graphs.

In this article we present a novel finite element formulation for surface diffusion of more general surfaces, which requires no explicit parametrization. In contrast to finite difference approaches [10, 24], we exploit the underlying variational structure and derive an intrinsic formulation, which avoids writing (1.1) in local coordinates.

Basic differential geometry reveals that the surface Laplacian of the position vector $\vec{X}$ on a surface $\Gamma(t)$ is the vector curvature $\vec{\kappa}$, namely $\Delta_S \vec{X} = \vec{\kappa}$ and $\vec{\kappa}$ is a vector normal to $\Gamma(t)$ with magnitude equal to the sum of the principal curvatures. This identity is the chief idea of [16] for designing a finite element method for mean curvature flow of parametric surfaces. However, we also need to deal with the scalar curvature κ in the present context and cannot work directly with the curvature vector $\vec{\kappa}$. We propose instead to use four unknowns, namely scalar curvature κ, curvature vector $\vec{\kappa}$, normal velocity $\vec{V}$, and (scalar) normal velocity V. Multiplication by the unit normal vector $\vec{\nu}$ to $\Gamma(t)$, pointing outward of the bulk enclosed by $\Gamma(t)$ is further used to convert from scalar to vector quantities and vice versa, thereby leading to the following four equations:

$$\vec{\kappa} = \Delta_S \vec{X}, \quad \kappa = \vec{\kappa} \cdot \vec{\nu}, \quad V = -\Delta_S(\kappa + \varepsilon), \quad \vec{V} = V\vec{\nu}. \tag{1.4}$$

This conversion, trivial when $\Gamma(t)$ is smooth, cannot be enforced pointwise when

$\Gamma(t)$ is polyhedral because $\vec{\nu}$ is discontinuous and so would be κ according to (1.4). The relation between scalars and vectors will later be imposed weakly (or in average), which turns out to be essential. To relate position $\vec{X}$ and velocity $\vec{V}$, we resort to a *semi-implicit* time discretization: all the geometric quantities and the differential operator Δ_S are evaluated on the *current* boundary Γ^n, whereas the unknowns $\vec{\kappa}$, κ, V, and $\vec{V}$ are treated implicitly. If $\tau_n := t_{n+1} - t_n$ denotes the (variable) time-step from time t_n to t_{n+1}, then we could write

$$\vec{X}^{n+1} = \vec{X}^n + \tau_n \vec{V}^{n+1}. \tag{1.5}$$

Consequently, (1.4) becomes the following system of *linear* elliptic PDE on Γ^n:

$$\begin{aligned}
\vec{\kappa}^{\,n+1} - \tau_n \Delta_S \vec{V}^{\,n+1} &= \Delta_S \vec{X}^{\,n}, \\
\kappa^{\,n+1} - \vec{\kappa}^{\,n+1} \cdot \vec{\nu}^{\,n} &= 0, \\
V^{\,n+1} + \Delta_S \kappa^{\,n+1} &= -\Delta_S \varepsilon^{\,n}, \\
\vec{V}^{\,n+1} - V^{\,n+1} \vec{\nu}^n &= 0.
\end{aligned} \tag{1.6}$$

We now list several properties of and issues pertinent to this system.

- *Mixed method*: the operator splitting of (1.6) can be viewed as a mixed formulation involving only second and zero order operators.
- *Parametrization*: the formulation of (1.6), and thereby its space discretization, does not require an explicit parametrization of Γ^n; once $\vec{V}^{\,n+1}$ has been computed then (1.5) can be used to update the surface to Γ^{n+1}.
- *Avoiding C^1 elements*: since the operators involved are of either order 2 or 0, we can use C^0 piecewise polynomials of any degree to approximate (1.6); see §2. Therefore, we do not need C^1 elements even to approximate curvature κ. This simplifies the implementation without compromising accuracy
- *Boundary conditions*: in the present article we consider either closed surfaces or natural boundary conditions for which integration by parts yields no boundary terms. This restriction is for ease of presentation only, and helps highlight the novel variational formulation of the problem. But using the flexibility of finite elements, other boundary conditions can be considered as well, with slight changes in the implementation. Different, physically relevant boundary conditions will be addressed in a forthcoming article, where we will also tackle the coupling of surface diffusion with elasticity in the bulk.
- *Conservation*: testing the third equation in (1.6) with $\phi = 1$, and integrating by parts we realize that volume is preserved in the sense that $\int_{\Gamma^n} V^{\,n+1} = 0$, which mimics (1.2) (observe that also $\int_{\Gamma^n} \vec{\kappa}^{\,n+1} = 0$.) Multiplying the same equation by $\phi = \kappa^{\,n+1}$ we prove a discrete analog of (1.3); see Theorem 2.1.
- *Solvability*: we show in §4 that the *linear* algebraic system ensuing from (1.6) is uniquely solvable by examining a Schur complement approach for the single unknown V. This yields a symmetric and positive definite matrix, thus allowing for an efficient solution technique via preconditioned CG; see §5.

- *Mesh smoothing*: the geometric flow by surface diffusion may lead to mesh distortions. We propose in §5.2 a procedure to maintain shape regularity which is volume preserving. This procedure has some independent interest.
- *Time adaptivity*: large timesteps may yield large changes of nodal positions with respect to neighboring nodes, and thus contribute to mesh distortion. On the other hand, large timesteps may be desirable when curvature changes slowly and the evolution is thus slow. We propose in §5.3 an effective timestep control mechanism.
- *Space adaptivity*: accurate description of a surface with minimal number of degrees of freedom fits quite naturally within the finite element framework. We propose in §5.4 a simple strategy to equidistribute pointwise errors in an intrinsic metric.
- *Topological changes*: the formulation (1.6) cannot handle topological changes without an a priori classification of possible singularities, which is not yet available for surface diffusion. The proposed method provides an efficient means for studying singularities as well as basic properties of the geometric flow, as explored in §5. We refer to [10, 24] for level set methods and to [8] for Cahn-Hilliard models with degenerate mobility, which are in general capable of handling topological changes. Efficient computation of surface diffusion is still under investigation for level set methods [24], and is much less developed for diffuse interface models. Both approaches are rather stiff, which justifies the search for suitable semi-implicit time discretizations [24].

The rest of this paper is organized as follows. We present a finite element discretization of (1.6) in § 2, together with discrete versions of (1.2) and (1.3). We discuss the ensuing *linear* algebraic problem in § 3 along with a Schur complement approach to its solution in § 4. We document the performance of our FEM in § 5 via several simulations, some exhibiting pinch-off, selfintersections, and mushroom formation in finite time. We discuss along key numerical issues such as mesh regularization to avoid mesh distortion, and time and space adaptivity to increase accuracy while reducing complexity. We finally draw conclusions in § 6.

2 Finite Element Discretization and Stability

We now discuss the finite element discretization of (1.6) along with a couple of properties. To simplify the notation we hereafter drop the scripts n and $n+1$.

Let $\mathcal{T}$ be a regular but possibly graded mesh of triangular finite elements over the surface Γ which, from now on, is assumed to be polyhedral. Let $T \in \mathcal{T}$ be a typical triangle and let $\vec{\nu}_T = (\nu_T^i)_{i=1}^d$ be the unit normal to T pointing outwards. We denote by $\vec{\nu}$ the outward unit normal to Γ, which satisfies $\vec{\nu}|_T = \vec{\nu}_T$ for all $T \in \mathcal{T}$, and is thus discontinuous across interelement boundaries. Let $\{\phi_i\}_{i=1}^I$ be the set of canonical basis functions of the finite element space $\mathcal{V}(\Gamma)$ of continuous piecewise polynomials $\mathcal{P}^k$ of degree $\leq k$ over $\mathcal{T}$ $(k \geq 1)$; we thus have a *conforming* approximation of $\mathcal{V}(\Gamma)$. We note that $\mathcal{V}(\Gamma) \subset H^1(\Gamma)$ and also set $\vec{\mathcal{V}}(\Gamma) := \mathcal{V}(\Gamma)^d$.

To derive a weak formulation, we multiply the equations (1.6) by test functions $\phi \in \mathcal{V}(\Gamma)$ and $\vec{\varphi} \in \vec{\mathcal{V}}(\Gamma)$ and use integration by parts for the second order operator Δ_S. Denoting by $\langle \cdot, \cdot \rangle$ the L^2-inner product over Γ, we arrive at the fully discrete problem: seek $\vec{V}, \vec{\kappa} \in \vec{\mathcal{V}}(\Gamma)$, $V, \kappa \in \mathcal{V}(\Gamma)$, such that

$$\langle \vec{\kappa}, \vec{\varphi} \rangle + \tau \left\langle \nabla_S \vec{V}, \nabla \vec{\varphi} \right\rangle = - \left\langle \nabla_S \vec{X}, \nabla_S \vec{\varphi} \right\rangle \qquad \forall\, \vec{\varphi} \in \vec{\mathcal{V}}(\Gamma), \tag{2.1}$$

$$\langle \kappa, \phi \rangle - \langle \vec{\kappa} \cdot \vec{\nu}, \phi \rangle = 0 \qquad \forall\, \phi \in \mathcal{V}(\Gamma), \tag{2.2}$$

$$\langle V, \phi \rangle - \langle \nabla_S \kappa, \nabla_S \phi \rangle = \langle \nabla_S \varepsilon, \nabla_S \phi \rangle \qquad \forall\, \phi \in \mathcal{V}(\Gamma), \tag{2.3}$$

$$\left\langle \vec{V}, \vec{\varphi} \right\rangle - \langle V, \vec{\varphi} \cdot \vec{\nu} \rangle = 0 \qquad \forall\, \vec{\varphi} \in \vec{\mathcal{V}}(\Gamma). \tag{2.4}$$

We first note that the relations (2.2) and (2.4) between scalars and vectors are imposed weakly and not pointwise; this allows for the 4 unknowns to be continuous whereas $\vec{\nu}$ is discontinuous. This is a distinctive aspect of our approach. Secondly, we see that taking $\phi = 1$ in (2.3) yields *volume conservation*:

$$\int_{\Gamma^n} V^{n+1} = 0 \qquad \forall\, 0 \le n \le N-1. \tag{2.5}$$

Since the integral in computed over Γ^n, and not Γ^{n+1}, the volume changes slightly due to truncation error. The change relative to the initial volume never exceeds 1.3% in our simulations, some rather singular (see Figure 5.11). We thirdly establish a result concerning the unconditional stability of the scheme, which mimics the area decrease expression (1.3) for $\varepsilon = 0$.

Theorem 2.1 (Unconditional Stability). *Let* $(V^n, \kappa^n, \vec{V}^n, \vec{\kappa}^n)_{n=1}^N$ *be the solution of either the semidiscrete equations* (1.6) *or of the fully discrete equations (2.1)–(2.4) and let* Γ^n *be the corresponding embedded surfaces. Then for all* $1 \le m \le N$ *we have*

$$|\Gamma^m| + \frac{1}{2} \sum_{n=0}^{m-1} \tau_n \int_{\Gamma^n} |\nabla_S \kappa^{n+1}|^2 \le |\Gamma^0| + \frac{1}{2} \sum_{n=0}^{m-1} \tau_n \int_{\Gamma^n} |\nabla_S \varepsilon(t_n)|^2. \tag{2.6}$$

Proof. We start by testing (2.3) with $\phi = \kappa^{n+1}$, thereby obtaining

$$\left\langle V^{n+1}, \kappa^{n+1} \right\rangle = \left\langle \nabla_S \kappa^{n+1}, \nabla_S \kappa^{n+1} \right\rangle + \left\langle \nabla_S \varepsilon(t_n), \nabla_S \kappa^{n+1} \right\rangle.$$

Combining (2.4) with $\vec{\varphi} = \vec{\kappa}^{n+1}$ and (2.2) with $\phi = V^{n+1}$, we easily arrive at

$$\left\langle \vec{V}^{n+1}, \vec{\kappa}^{n+1} \right\rangle = \left\langle V^{n+1}, \vec{\kappa}^{n+1} \cdot \vec{\nu}^n \right\rangle = \left\langle \kappa^{n+1}, V^{n+1} \right\rangle,$$

whence

$$\left\langle \vec{V}^{n+1}, \vec{\kappa}^{n+1} \right\rangle = \left\langle \nabla_S \kappa^{n+1}, \nabla_S \kappa^{n+1} \right\rangle + \left\langle \nabla_S \varepsilon(t_n), \nabla_S \kappa^{n+1} \right\rangle. \tag{2.7}$$

On the other hand, testing (2.1) with $\vec{\varphi} = \tau_n \vec{V}^{\,n+1}$ and observing that, according to (1.5), $\tau_n \vec{V}^{\,n+1} = \vec{X}^{n+1} - \vec{X}^n$ yields

$$\tau_n \left\langle \vec{V}^{\,n+1}, \vec{\kappa}^{n+1} \right\rangle + \left\langle \nabla_S \vec{X}^{n+1}, \nabla_S (\vec{X}^{n+1} - \vec{X}^n) \right\rangle = 0. \tag{2.8}$$

Multiplying (2.7) by τ_n and substituting into (2.8) we infer that

$$\left\langle \nabla_S \vec{X}^{n+1}, \nabla_S (\vec{X}^{n+1} - \vec{X}^n) \right\rangle + \tau_n \left\langle \nabla_S \kappa^{n+1}, \nabla_S \kappa^{n+1} \right\rangle = -\tau_n \left\langle \nabla_S \varepsilon(t_n), \nabla_S \kappa^{n+1} \right\rangle$$

Applying Lemma 2.2 below, we can further estimate

$$|\Gamma^{n+1}| - |\Gamma^n| + \tau_n \int_{\Gamma^n} |\nabla_S \kappa^{n+1}|^2 \leq \tau_n \int_{\Gamma^n} |\nabla_S \varepsilon(t_n)|^2.$$

Summing up over n, from 0 to $m-1$, yields the asserted result. $\square$

Lemma 2.2 (Area inequality [2]). *Let $d = 2, 3$ and Γ be a $d-1$–dimensional, closed, regular $C^{0,1}$–manifold embedded in $\mathbb{R}^k$, $k \in \mathbb{N}$. Moreover let $\vec{Y} : \Gamma \to \mathrm{rg}(\Gamma) \subseteq \mathbb{R}^k$ be a homeomorphism with $D\vec{Y}$, $(D\vec{Y})^{-1} \in L^\infty$. Then, if $\vec{X}$ denotes the position vector of the integration variable, the following inequality holds:*

$$\int_\Gamma \nabla_S \vec{Y} \cdot \nabla_S (\vec{Y} - \vec{X}) \;\geq\; |\vec{Y}(\Gamma)| \;-\; |\Gamma|.$$

The proof of the above lemma is rather technical and can be found in [2].

3 Matrix Formulation

We now turn our attention to an equivalent matrix formulation to the fully discrete problem (2.1)–(2.4). Given the matrix entries

$$M_{ij} := \langle \phi_i, \phi_j \rangle, \qquad \vec{M}_{ij} := M_{ij} \vec{Id}, \qquad \vec{N}_{ij} := \left\langle \phi_i, \phi_j \nu^k \right\rangle_{k=1}^d, \tag{3.1}$$

$$A_{ij} := \langle \nabla_S \phi_i, \nabla_S \phi_j \rangle, \qquad \vec{A}_{ij} := A_{ij} \vec{Id}, \tag{3.2}$$

with $\vec{Id} \in \mathbb{R}^{d \times d}$ being the identity matrix and $(\vec{e}_k)_{k=1}^d$ the canonical basis of $\mathbb{R}^d$, the mass and stiffness matrices are

$$M := (M_{ij})_{i.j=1}^I, \qquad \vec{M} := (\vec{M}_{ij})_{i.j=1}^I, \qquad \vec{N} := (\vec{N}_{ij})_{i.j=1}^I, \tag{3.3}$$

$$A := (A_{ij})_{i.j=1}^I, \qquad \vec{A} := (\vec{A}_{ij})_{i.j=1}^I. \tag{3.4}$$

We point out that $\vec{M}, \vec{A}$ and $\vec{N}$ possess matrix-valued entries and therefore the matrix-vector product is understood in the following sense

$$\vec{M}\vec{\boldsymbol{V}} = \Big(\sum_{j=1}^I \vec{M}_{ij} \vec{V}_j \Big)_{i=1}^I,$$

each component $\vec{V}_i$ of $\vec{\boldsymbol{V}}$, as well as each of $\vec{M}\vec{\boldsymbol{V}}$, is itself a vector in $\mathbb{R}^d$.

We use the convention that a vector of nodal values of a finite element function is written in bold face: $\boldsymbol{V} = (V_i)_{i=1}^I \in \mathbb{V} := \mathbb{R}^I$ is equivalent to $V = \sum_{i=1}^I V_i \phi_i \in \mathcal{V}(\Gamma)$. We introduce the subspace $\mathcal{X}(\Gamma)$ of $\mathcal{V}(\Gamma)$ of functions with mean value zero, and the corresponding subspace $\mathbb{X}$ of $\mathbb{V}$ of vectors $\boldsymbol{V}$ satisfying $\boldsymbol{V} \cdot M\mathbf{1} = 0$ with $\mathbf{1} := (1)_{i=1}^I$. We then note that

$$V = \sum_{i=1}^I V_i \phi_i \in \mathcal{X}(\Gamma) \quad \Leftrightarrow \quad \boldsymbol{V} = (V_i)_{i=1}^I \in \mathbb{X}. \tag{3.5}$$

We are now in a position to write the matrix formulation of (2.1)–(2.4). Upon expanding the unknown scalar functions $V \in \mathcal{X}(\Gamma), \kappa \in \mathcal{V}(\Gamma)$ and vector functions $\vec{V} \in \vec{\mathcal{V}}(\Gamma), \vec{\kappa} \in \vec{\mathcal{X}}(\Gamma)$ in terms of the basis functions and setting $\phi = \phi_i$ and $\vec{\varphi} = \phi \vec{e}_k$, we easily arrive at

$$(2.1) \quad \rightsquigarrow \quad \tau \vec{A}\vec{\boldsymbol{V}} + \vec{M}\vec{\boldsymbol{K}} = -\vec{A}\vec{\boldsymbol{X}}, \tag{3.6}$$

$$(2.2) \quad \rightsquigarrow \quad M\boldsymbol{K} - \vec{N}^T\vec{\boldsymbol{K}} = \mathbf{0}, \tag{3.7}$$

$$(2.3) \quad \rightsquigarrow \quad -A\boldsymbol{K} + M\boldsymbol{V} = \boldsymbol{E}, \tag{3.8}$$

$$(2.4) \quad \rightsquigarrow \quad \vec{M}\vec{\boldsymbol{V}} - \vec{N}\boldsymbol{V} = \vec{\mathbf{0}}, \tag{3.9}$$

where $\boldsymbol{E} = (\langle \nabla_S \phi_i, \nabla_S \varepsilon \rangle)_{i=1}^I$. This system can be written equivalently in block-matrix form as follows: *find* $\vec{\boldsymbol{V}} \in \vec{\mathbb{V}}, \boldsymbol{K} \in \mathbb{V}, \vec{\boldsymbol{K}} \in \vec{\mathbb{X}}, \boldsymbol{V} \in \mathbb{X}$ *such that*

$$\begin{bmatrix} \tau\vec{A} & 0 & \vec{M} & 0 \\ 0 & -A & 0 & M \\ \vec{M} & 0 & 0 & -\vec{N} \\ 0 & M & -\vec{N}^T & 0 \end{bmatrix} \begin{bmatrix} \vec{\boldsymbol{V}} \\ \boldsymbol{K} \\ \vec{\boldsymbol{K}} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} -\vec{A}\vec{\boldsymbol{X}} \\ \boldsymbol{E} \\ \vec{\mathbf{0}} \\ \mathbf{0} \end{bmatrix}. \tag{3.10}$$

We discuss the solvability of (3.10) and propose an algorithm for its solution in §4. We point out that the mesh $\mathcal{T}$ can be suitably graded and the polynomial degree $k \geq 1$ is arbitrary, even though we restrict ourselves to piecewise linears in the simulations of §5. This flexibility is quite important to handle complicated geometries and possible pinch-off singularities. We also stress that $\vec{A}$, $\vec{M}$ need not be formed and stored in practice since they can be easily obtained from A, M.

4 Schur Complement Approach

Consider the following generic vector equation with a (possibly singular) square block A:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{Q} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{G} \end{bmatrix}.$$

Let A be symmetric with (nontrivial) kernel $\ker(A)$. Then the range $\mathbb{Y}$ of A is the orthogonal complement of $\ker(A)$. Let $S : \mathbb{Y} \to \mathbb{Y}$ be the inverse of A

restricted to $\mathbb{Y}$: $SA = AS = Id$ on $\mathbb{Y}$. If P denotes the orthogonal projection onto $\ker(A)$, we have

$$SA\boldsymbol{V} = \boldsymbol{V} - P\boldsymbol{V} = (Id - P)\boldsymbol{V} \qquad \forall \boldsymbol{V} \in \mathbb{R}^I = \mathbb{V}, \tag{4.1}$$

where $Id - P$ is the orthogonal projection onto $\mathbb{Y}$. The *Schur complement* equation for $\boldsymbol{Q}$ then reads

$$(-CSB + D)\boldsymbol{Q} + CP\boldsymbol{U} = \boldsymbol{G} - CS\boldsymbol{F}. \tag{4.2}$$

Solvability of this system depends on the structure of the two terms on the left hand-side of (4.2). We intend to apply this splitting to (3.10), which involves dealing with the upper left block containing $\vec{A}$ and A on the diagonal.

Since the kernel $\mathbb{Z}$ of A in (3.4) is the one dimensional subspace of $\mathbb{V} = \mathbb{R}^I$ spanned by $\boldsymbol{1} = (1)_{i=1}^I$, then the range $\mathbb{Y} = \mathbb{Z}^\perp$ of A is the orthogonal complement of $\mathbb{Z}$ with respect to the standard Euclidean inner product in $\mathbb{R}^I$. If $\mathbb{X}$ denotes the space defined in (3.5), $\mathbb{X}$ and $\mathbb{Y}$ are related as follows:

$$\boldsymbol{V} \in \mathbb{X} \qquad \Leftrightarrow \qquad M\boldsymbol{V} \in \mathbb{Y}. \tag{4.3}$$

Let $S : \mathbb{Y} \to \mathbb{Y}$ be the inverse of A restricted to $\mathbb{Y}$, and let $P : \mathbb{V} \to \mathbb{Z}$ be the orthogonal projection into $\mathbb{Z}$, thereby satisfying (4.1) with

$$P\boldsymbol{V} = \frac{1}{\boldsymbol{1}^T\boldsymbol{1}}\boldsymbol{1}^T\boldsymbol{V}\ \boldsymbol{1} = \frac{\boldsymbol{1}\otimes\boldsymbol{1}}{I}\ \boldsymbol{V} \qquad \forall\ \boldsymbol{V} \in \mathbb{V}. \tag{4.4}$$

We now would like to apply (4.2) to (3.10) with vectors $\boldsymbol{U} = [\vec{\boldsymbol{V}}, \boldsymbol{K}]^T$ and $\boldsymbol{Q} = [\vec{\boldsymbol{K}}, \boldsymbol{V}]^T$. Let us assume momentarily that there exists a solution $[\vec{\boldsymbol{V}}, \boldsymbol{K}, \vec{\boldsymbol{K}}, \boldsymbol{V}]^T$ to (3.10). Then from (4.2) $\vec{\boldsymbol{V}}$, $\boldsymbol{K}$, $\vec{\boldsymbol{K}}$, $\boldsymbol{V}$ satisfy

$$\begin{bmatrix} \frac{1}{\tau}\vec{M}\vec{S}\vec{M} & \vec{N} \\ \vec{N}^T & -MSM \end{bmatrix} \begin{bmatrix} \vec{\boldsymbol{K}} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} -\frac{1}{\tau}\vec{M}\vec{S}\vec{A}\vec{\boldsymbol{X}} + \vec{M}\vec{P}\vec{\boldsymbol{V}} \\ MP\boldsymbol{K} - MS\boldsymbol{E} \end{bmatrix}. \tag{4.5}$$

We observe that both $\vec{S}\vec{A}\vec{\boldsymbol{X}}$ and $S\boldsymbol{E}$ make sense because $\vec{A}\vec{\boldsymbol{X}} \in \vec{\mathbb{Y}}$ and $\boldsymbol{E} = (\langle\nabla_S\phi_i, \nabla_S\varepsilon\rangle)_{i=1}^I \in \mathbb{Y}$; this could be viewed as a compatibility condition. Multiplying (3.6) and (3.8) by $\vec{\boldsymbol{1}}$ and $\boldsymbol{1}$, respectively we see that both components of $\boldsymbol{Q}$ satisfy $\vec{\boldsymbol{K}} \in \vec{\mathbb{X}}$ and $\boldsymbol{V} \in \mathbb{X}$ or, in view of (4.3),

$$\vec{M}\vec{\boldsymbol{K}} \in \vec{\mathbb{Y}}, \qquad M\boldsymbol{V} \in \mathbb{Y}. \tag{4.6}$$

Since the upper left block of (4.5), $\vec{M}\vec{S}\vec{M} : \vec{\mathbb{X}} \to \vec{M}\vec{\mathbb{Y}}$, is nonsingular with inverse $\vec{M}^{-1}\vec{A}\vec{M}^{-1}$, we can apply (4.2) again to arrive at

$$\left(\tau\vec{N}^T\vec{M}^{-1}\vec{A}\vec{M}^{-1}\vec{N} + MSM\right)\boldsymbol{V} + MP\boldsymbol{K} = -\vec{N}^T\vec{M}^{-1}\vec{A}\vec{\boldsymbol{X}} + MS\boldsymbol{E}. \tag{4.7}$$

To decouple (4.7) we first eliminate the term $MP\boldsymbol{K}$ which acts like a Lagrange multiplier to the constraint $\boldsymbol{V} \in \mathbb{X}$. We achieve this by the orthogonal projection Π onto $\mathbb{X}$:

$$\Pi = Id - \frac{M\boldsymbol{1}\otimes M\boldsymbol{1}}{M\boldsymbol{1}^T\cdot M\boldsymbol{1}}. \tag{4.8}$$

Since $MP\boldsymbol{K} \in \text{span}\{M\mathbf{1}\} = \mathbb{X}^\perp$, upon multiplying (4.7) by Π we obtain the final form of the *Schur complement*, namely the reduced equation

$$\Pi\Big(\tau\vec{N}^T\vec{M}^{-1}\vec{A}\vec{M}^{-1}\vec{N} + MSM\Big)\Pi\boldsymbol{V} = \Pi\Big(-\vec{N}^T\vec{M}^{-1}\vec{A}\vec{\boldsymbol{X}} + MS\boldsymbol{E}\Big), \tag{4.9}$$

because $\Pi\boldsymbol{V} = \boldsymbol{V}$. This reasoning leads to the following solvability result.

Theorem 4.1 (Solvability). *There exists a unique solution* $[\vec{\boldsymbol{V}}, \boldsymbol{K}, \vec{\boldsymbol{K}}, \boldsymbol{V}]^T$ *of system* (3.10)*, the components of which can be obtained by sequentially solving the following (uniquely solvable) systems:*

$$\begin{aligned}
&\boldsymbol{V} \in \mathbb{X}: && \Pi\Big(\tau\vec{N}^T\vec{M}^{-1}\vec{A}\vec{M}^{-1}\vec{N} + MSM\Big)\Pi\boldsymbol{V} = \Pi\boldsymbol{F}, && (4.10)\\
&\vec{\boldsymbol{V}} \in \mathbb{V}: && \vec{M}\vec{\boldsymbol{V}} = \vec{N}\boldsymbol{V}, && (4.11)\\
&\vec{\boldsymbol{K}} \in \mathbb{V}: && \vec{M}\vec{\boldsymbol{K}} = -\vec{A}\vec{\boldsymbol{X}} - \tau\vec{A}\vec{\boldsymbol{V}}, && (4.12)\\
&\boldsymbol{K} \in \mathbb{V}: && M\boldsymbol{K} = \vec{N}^T\vec{\boldsymbol{K}}, && (4.13)
\end{aligned}$$

where $\boldsymbol{F} = -\vec{N}^T\vec{M}^{-1}\vec{A}\vec{\boldsymbol{X}} + MS\boldsymbol{E}$.

Proof. By the argument preceding the statement of the theorem we conclude that if $[\vec{\boldsymbol{V}}, \boldsymbol{K}, \vec{\boldsymbol{K}}, \boldsymbol{V}]^T$ is a solution to (3.10) then $\vec{\boldsymbol{V}}$, $\boldsymbol{K}$, $\vec{\boldsymbol{K}}$, $\boldsymbol{V}$ are solutions to (4.10)–(4.13), respectively.

The reciprocal part of the proof consists of proving that systems (4.10)–(4.13) have unique solutions and they constitute a solution of (3.10).

Let us first check the solvability of systems (4.10)–(4.13). It is easy to verify that the operator $\Pi MSM\Pi : \mathbb{X} \to \mathbb{X}$ is symmetric and positive definite, and $\Pi\vec{N}^T\vec{M}^{-1}\vec{A}\vec{M}^{-1}\vec{N}\Pi : \mathbb{X} \to \mathbb{X}$ is symmetric and positive semidefinite. Therefore the matrix ensuing from (4.10) is positive definite and since the right hand side of the equation belongs to $\mathbb{X}$, this symmetric system has a unique solution $\boldsymbol{V} \in \mathbb{X}$. Systems (4.11)–(4.13) involve mass matrices, which are positive definite in $\mathbb{V} = \mathbb{R}^I$, existence and uniqueness are thus ensured.

Let us now verify that the solutions to (4.10)–(4.13) constitute a solution to (3.10). Since $\Pi\boldsymbol{V} = \boldsymbol{V}$, by (4.10) and (4.11) we have that

$$\tau\Pi\vec{N}^T\vec{M}^{-1}\vec{A}\vec{\boldsymbol{V}} + \Pi MSM\boldsymbol{V} = \Pi\Big(-\vec{N}^T\vec{M}^{-1}\vec{A}\vec{\boldsymbol{X}} + MS\boldsymbol{E}\Big),$$

or

$$\Pi MSM\boldsymbol{V} = \Pi\Big(-\vec{N}^T\vec{M}^{-1}(\vec{A}\vec{\boldsymbol{X}} + \tau\vec{A}\vec{\boldsymbol{V}}) + MS\boldsymbol{E}\Big).$$

Hence (4.12) implies $\Pi MSM\boldsymbol{V} = \Pi\Big(\vec{N}^T\vec{\boldsymbol{K}} + MS\boldsymbol{E}\Big)$, and (4.13) yields

$$\Pi MSM\boldsymbol{V} = \Pi M\big(\boldsymbol{K} + S\boldsymbol{E}\big).$$

Since Π is the projection onto $\mathbb{X}$, $MSM\boldsymbol{V} - M\big(\boldsymbol{K} + S\boldsymbol{E}\big) \in \mathbb{X}^\perp = \text{span}\{M\mathbf{1}\}$, we infer that

$$M^{-1}(MSM\boldsymbol{V} - M\big(\boldsymbol{K} + S\boldsymbol{E}\big)) = SM\boldsymbol{V} - \big(\boldsymbol{K} + S\boldsymbol{E}\big) \in \mathbb{Y}^\perp = \text{span}\{\mathbf{1}\}.$$

Therefore, since $\mathbb{Y}^{\perp} = \ker(A)$,

$$A\Big(SM\boldsymbol{V} - \big(\boldsymbol{K} + S\boldsymbol{E}\big)\Big) = M\boldsymbol{V} - A\boldsymbol{K} - \boldsymbol{E} = 0,$$

which coincides with the second equation in (3.10). The rest of the equations in (3.10) are immediately deduced from (4.11)–(4.13). □

The method actually implemented in ALBERT consists of first solving for $\boldsymbol{V}$ using (4.10), next solving (4.11) for $\vec{\boldsymbol{V}}$ and finally updating $\vec{\boldsymbol{X}}$ via $\vec{\boldsymbol{X}} + \tau\vec{\boldsymbol{V}}$.

5 Implementation and Simulations

In this section we describe the implementation of (2.1)–(2.4) together with several enhancements. The latter are mesh regularization, space-time adaptivity and control of element angles. They are motivated through examples showing the necessity of tackling such issues, and the beneficial effect of our approach to solving them. Throughout this section, we take $\varepsilon \equiv 0$ in the simulations, because for the time being we are mainly interested in the effect of plain surface diffusion. Computations with given ε, as well as the coupling with elasticity in the bulk Ω, will be the subject of future work.

5.1 Implementation

The implementation was performed within the finite element toolbox ALBERT [22, 23], after adding suitable data structures to handle surfaces in $\mathbb{R}^3$ and curves in $\mathbb{R}^2$. The basic algorithm consists of the following steps:

Algorithm 5.1 (Basic Algorithm).

```
1. Take a mesh representing the initial surface
2. Choose a timestep $\tau$
3. Build the matrices $A$, $M$ and $\vec{N}$ ($\vec{A}$, and $\vec{M}$ are not
   really necessary)
4. Solve (4.10) and (4.11)
5. Update $\vec{\boldsymbol{X}} \leftarrow \vec{\boldsymbol{X}} + \tau\vec{\boldsymbol{V}}$.
6. Go to step 3
```

Notice that the matrices need to be re-built in each timestep because they depend on the current surface. In step 4 we solve the following linear systems:

$$\begin{aligned}
\boldsymbol{V} \in \mathbb{X}: &\qquad \Pi\Big(\tau\vec{N}^T\vec{M}^{-1}\vec{A}\vec{M}^{-1}\vec{N} + MSM\Big)\Pi\boldsymbol{V} = -\Pi\vec{N}^T\vec{M}^{-1}\vec{A}\vec{\boldsymbol{X}}, \\
\vec{\boldsymbol{V}} \in \mathbb{R}^I: &\qquad \vec{M}\vec{\boldsymbol{V}} = \vec{N}\boldsymbol{V}.
\end{aligned}$$

We solve both of them by a conjugate gradient (CG) method. Solving the second one is trivial since we only have to invert a mass matrix which has bounded condition number. To solve the first one, in each iteration of CG we have to compute a matrix-vector product for the matrix ensuing from this system, namely $\Pi\Big(\tau\vec{N}^T\vec{M}^{-1}\vec{A}\vec{M}^{-1}\vec{N} + MSM\Big)\Pi$, where the matrix S is the inverse of A restricted to $\ker(A)^{\perp}$. We do not compute this inverse explicitly, but we solve a system of the form $A\xi = b$ using another CG iteration (inner loop). Since A is a discretization of a Laplace operator, we use a hierarchical basis preconditioner which greatly improves the performance of the inner loop. The design and study of effective preconditioners for the full system is still open and we leave it for a forthcoming article. This issue is crucial to speed up the computations.

As a first example we show in Figure 5.1 the evolution of a unit cube toward a ball with the same volume. As can be seen in Figure 5.1 the geometric flow by

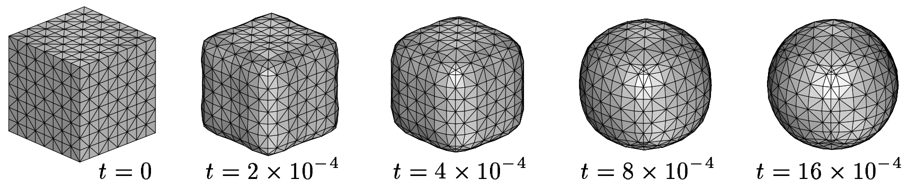

Figure 5.1: Evolution of a unit cube by surface diffusion. All the surfaces are represented by 768 triangles and 386 vertices. The (uniform) timestep used in the computations is $\tau = 1 \times 10^{-4}$.

surface diffusion is not as gentle as the corresponding mean curvature flow [16], and leads to severe mesh distortions. Even if our formulation of §2 allows corners and edges, which are rather singular for surface diffusion, they give rise to fast node motion and mesh distortion. This is illustrated by the creation of *ears*

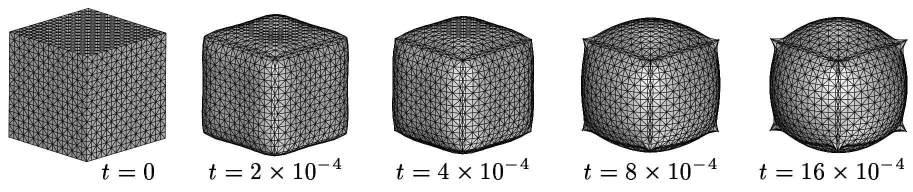

Figure 5.2: Pathological *ear* formation in the evolution of a unit cube by surface diffusion. All the surfaces are represented by 3072 triangles and 1538 vertices. Ear formation is the fatal manifestation of mesh distortion and is caused by clustering of nodes, crossing of element sides and folding, and is due to an inadequate tangential motion. It is cured with mesh regularization and timestep control. The (uniform) timestep $\tau = 1 \times 10^{-4}$ used in the computations is too large for the underlying mesh.

during the evolution of the same cube when represented with a finer mesh; see Figures 5.2 and 5.3. This is clearly a numerical artifact and cannot be cured by mesh refinement and/or coarsening.

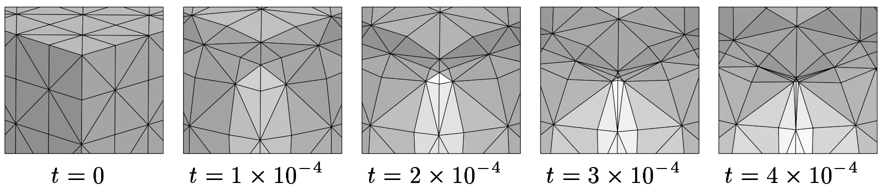

Figure 5.3: Steps toward the pathological formation of ears. Zoom into a vertex of the initial cube. After 6 timesteps some triangles collapse into points and others into segments, thereby making the mesh degenerate and producing numerical artifacts. The surfaces are represented by 3072 triangles and 1538 vertices. The (uniform) timestep $\tau = 1 \times 10^{-4}$ used in the computations is too large for the underlying mesh resolution.

There are two reasons that contribute to mesh distortion: clustering of nodes in regions of high velocity (along with crossing of elements sides and folding), and large timesteps. The first issue is due to the absence, in our formulation of § 2, of a geometric law for tangential flow to maintain mesh quality; the cure is thus *mesh regularization* and is discussed in §5.2. On the other hand, large timesteps yield changes of nodal positions tangential to the surface which may exceed the local meshsize and also lead to mesh distortion; a cure is timestep control and is discussed in §5.3.

5.2 Mesh Regularization

Mesh regularization is a procedure to maintain mesh quality, namely to keep all angles on element stars approximately of the same size; a *star* ω_z is the support of a basis function corresponding to node z. It is known that good approximability of the surface and the PDE on it hinges on avoiding mesh distortion. Mesh regularization is thus a redistribution of nodes on the surface, which entails a tangential flow and does not affect the normal motion.

Since surface diffusion is a geometric evolution that preserves the volume of the bulk $\Omega(t)$ enclosed by $\Gamma(t)$, we present a *volume preserving* mesh regularization algorithm which consists of a Gauss-Seidel type iteration:

Algorithm 5.2 (Regularization sweep).

```
For each node $z$ of the mesh do the following:

  1. Compute a normal $\vec{\nu}_z$ to the node $z$.
  2. Compute a weighted average $\hat{z}$ of all the vertices that
     belong to the star centered at $z$.
  3. Consider the line that passes through $\hat{z}$ in the
     direction of the normal $\vec{\nu}_z$.  Replace the node $z$ by the
     only point belonging to this line that keeps unchanged
     the volume of the bulk.
```

We now describe each step of this procedure in detail. In the first step, we take the normal to the node to be the weighted average of the normals of the elements sharing that node. The weight is given by the size $|T|$ of the element over the size of the star. That is, for each node $\boldsymbol{z}$, the normal $\vec{\nu}_{\boldsymbol{z}}$ is defined by

$$\vec{\nu}_z = \frac{1}{\sum_{T\in\mathcal{T}_{\boldsymbol{z}}}|T|}\sum_{T\in\mathcal{T}_{\boldsymbol{z}}}\vec{\nu}_T|T|,$$

where $\mathcal{T}_{\boldsymbol{z}}$ denotes the set of all the elements of the mesh that contain $\boldsymbol{z}$, and thus form the star ω_z, and $\vec{\nu}_T$ is the outer normal of the element T.

In the second step, we take $\hat{\boldsymbol{z}}$ to be the average of the barycenters of all the elements in the star ω_z:

$$\hat{\boldsymbol{z}} = \frac{1}{\#(\mathcal{T}_{\boldsymbol{z}})}\sum_{T\in\mathcal{T}_{\boldsymbol{z}}}\frac{\sum_{i=1}^{d}\boldsymbol{z}_T^i}{d}$$

where $\boldsymbol{z}_T^i$ denotes the i-th node of the element T. The result thus coincides with a weighted average of all the nodes in ω_z.

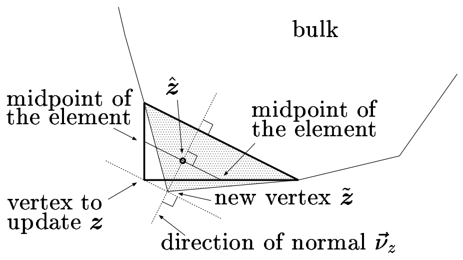

Figure 5.4: Volume preserving mesh regularization in 2d. The area of the shaded triangle coincides with that of the triangle marked with thick lines. Then the area of the whole bulk remains unchanged

The implementation of the third step depends on the dimension. In 2d the situation is simple. Given that the bulk is the interior of a closed polygonal curve (mesh), consider a node $\boldsymbol{z}$ and its two adjacent nodes as depicted in Figure 5.4. The direction $\vec{\nu}_{\boldsymbol{z}}$ turns out to be perpendicular to the segment joining the adjacent nodes. The idea is then to compute the new vertex $\tilde{\boldsymbol{z}} = \hat{\boldsymbol{z}} + t\vec{\nu}_z$, that will replace $\boldsymbol{z}$, in such a way that the area of the triangles with vertices $\boldsymbol{z}$ (triangle with thick lines) and $\tilde{\boldsymbol{z}}$ (shaded triangle) is the same (see Figure 5.4).

To perform the third step in 3d we first observe the fact that, given a fixed point $\bar{\boldsymbol{z}}$, the volume of the enclosed region is proportional to the sum of element contributions $\boldsymbol{v}_T$ defined as follows:

$$\boldsymbol{v}_T = (\boldsymbol{z}_T^1 - \bar{\boldsymbol{z}}) \times (\boldsymbol{z}_T^2 - \bar{\boldsymbol{z}}) \cdot (\boldsymbol{z}_T^3 - \bar{\boldsymbol{z}}),$$

where $\boldsymbol{z}_T^i$, $i = 1, 2, 3$ denote the vertices of the (surface) element T following a positive orientation with respect to the outer normal. The idea is now to compute the new vertex $\tilde{\boldsymbol{z}} = \hat{\boldsymbol{z}} + t\vec{\nu}_z$, that will replace $\boldsymbol{z}$, in such a way that the contribution to the volume of the modified star is the same as that of the original star. We take $\bar{\boldsymbol{z}} := \hat{\boldsymbol{z}}$ in the definition of $\boldsymbol{v}_T$ above, and number the vertices of each element in such a way that $\boldsymbol{z} = \boldsymbol{z}_T^1$. Then the volume contributions of the old and the new star will be equal if

$$\sum_{T\in\mathcal{T}_{\boldsymbol{z}}}(\boldsymbol{z} - \hat{\boldsymbol{z}}) \times (\boldsymbol{z}_T^2 - \hat{\boldsymbol{z}}) \cdot (\boldsymbol{z}_T^3 - \hat{\boldsymbol{z}}) = \sum_{T\in\mathcal{T}_{\boldsymbol{z}}}(\tilde{\boldsymbol{z}} - \hat{\boldsymbol{z}}) \times (\boldsymbol{z}_T^2 - \hat{\boldsymbol{z}}) \cdot (\boldsymbol{z}_T^3 - \hat{\boldsymbol{z}}).$$

Since $\tilde{z} - \hat{z} = t\vec{\nu}_z$, this equation will hold for

$$t = \frac{\sum_{T\in\mathcal{T}_z}(z - \hat{z}) \times (z_T^2 - \hat{z}) \cdot (z_T^3 - \hat{z})}{\sum_{T\in\mathcal{T}_z} \vec{\nu}_z \times (z_T^2 - \hat{z}) \cdot (z_T^3 - \hat{z})}.$$

The beneficial effect of this mesh regularization is reflected in the simulation depicted in Figure 5.5, which displays the evolution of the unit cube represented initially by the same fine mesh of Figure 5.2. No ear formation is now observed.

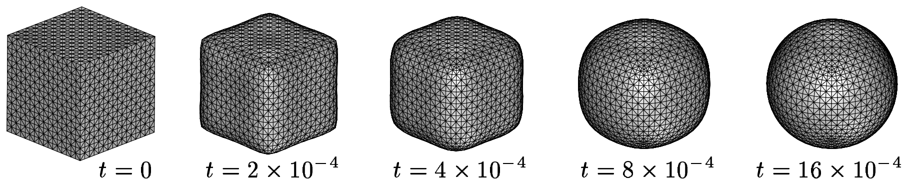

Figure 5.5: Evolution of a unit cube by surface diffusion using *mesh regularization.* After each timestep, the mesh regularization sweep is applied twice to the surface to cure mesh distortions. All the surfaces are represented by 3072 triangles and 1538 vertices. The timestep used in the computations is $\tau = 1 \times 10^{-4}$, as in Figure 5.2.

This simple minded mesh smoothing algorithm has some intrinsic merits which, in particular, make it instrumental for mesh improvement and update even in dealing with the volume enclosed by Γ^n (the bulk).

5.3 Timestep Control

The timestep control is twofold. First it is meant to prevent large timesteps for which the position change of a node, tangential to the surface and relative to that of neighboring nodes, is larger than the element size. This may be responsible for mesh distortion and even node crossing. The second objective is to allow large timesteps when the normal velocity does not exhibit large variations, and to force small timesteps otherwise. The very disparate time scales that can be observed in all the evolutions presented in this section, which are typical of fourth order problems, suggest that timestep control represents an important improvement in accuracy while maintaining a moderate number of timesteps.

To determine a criterion for timestep control, we argue as follows. Let z_0 be a generic node and let z be an adjacent node, both belonging to an element T. In view of (1.5), their relative position change is $\tau(\vec{V}(z_0) - \vec{V}(z))$. If $\vec{\tau}_T$ is any unit tangent vector to T, then the relative position change tangential to Γ is given by

$$\tau\big|(\vec{V}(z_0) - \vec{V}(z)) \cdot \vec{\tau}_T\big| \leq C\tau h_T|\nabla_S\vec{V}_T|,$$

with $C > 0$ a mesh independent constant. We would like this quantity not to exceed a fraction of the local meshsize h_T, which thereby leads to

$$\tau|\nabla_S\vec{V}_T| \leq \epsilon_t \qquad \forall\, T \in \mathcal{T}.$$

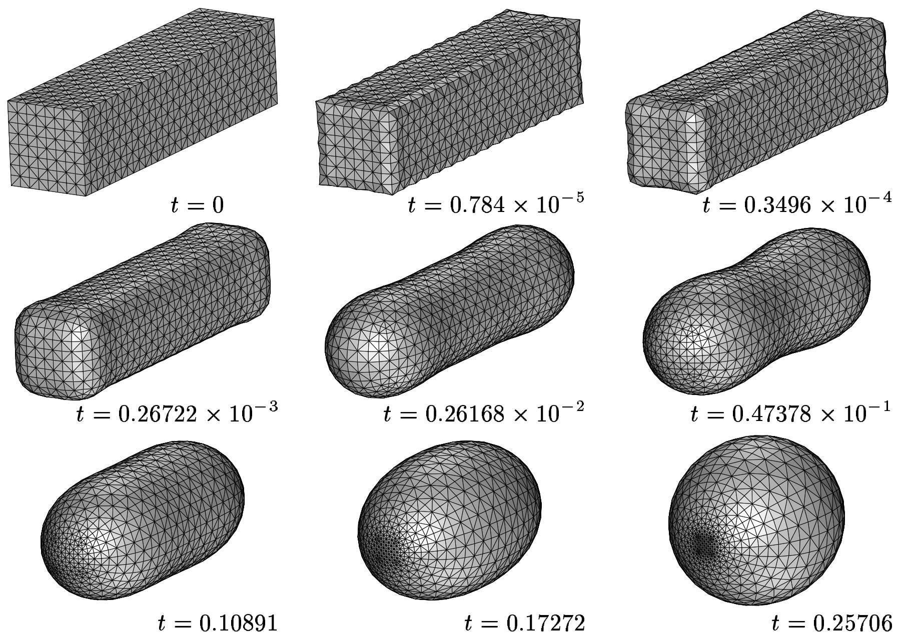
$t = 0$
$t = 0.784 \times 10^{-5}$
$t = 0.3496 \times 10^{-4}$
$t = 0.26722 \times 10^{-3}$
$t = 0.26168 \times 10^{-2}$
$t = 0.47378 \times 10^{-1}$
$t = 0.10891$
$t = 0.17272$
$t = 0.25706$

Figure 5.6: Evolution of a $4 \times 1 \times 1$ prism toward a ball with equal volume using *mesh regularization* and *timestep control*. Solutions obtained every 9 adaptive timesteps. All the surfaces are represented by 2304 triangles and 1154 vertices. The mesh regularization sweep was run two times after each timestep, and the parameters of the timestep control routine were $\epsilon_t = 0.1$, $\tau_{\min} = 1 \times 10^{-7}$, $\tau_{\max} = 5 \times 10^{-3}$.

This gives rise to the following algorithm, which uses input parameters $\epsilon_t, \tau_{\min}$ and $\tau_{\max} > 0$ (in all our simulations $\epsilon_t = 10^{-1}, \tau_{\min} = 10^{-7}, \tau_{\max} = 5 \times 10^{-3}$).

Algorithm 5.3 (Timestep Control).

```
1. Compute the quantity $\rho = \dfrac{\epsilon_t}{\max|\nabla_S \vec{V}|}$
2. If $\tau \leq \rho$ update $\vec{\boldsymbol{X}} \leftarrow \vec{\boldsymbol{X}} + \tau\vec{\boldsymbol{V}}$
3. Otherwise neglect the computation and keep $\vec{\boldsymbol{X}}$ as is.
4. In any case let the candidate for $\tau$ be
       $\tau_* = \begin{cases} \tau & \text{if } 0.9\rho \leq \tau \leq \rho \\ 0.9\rho & \text{otherwise} \end{cases}$
5. Set $\tau = \begin{cases} \tau_{\min} & \text{if } \tau_* < \tau_{\min} \\ \tau_* & \text{if } \tau_{\min} \leq \tau_* \leq \tau_{\max} \\ \tau_{\max} & \text{if } \tau_{\max} < \tau_*. \end{cases}$
```

In Figure 5.6 we show the combined effect of mesh regularization together with timestep control in the evolution of a $4 \times 1 \times 1$ prism. The pictures correspond to the solution obtained every 9 adaptive timesteps. It is apparent from the pictures that the timestep control not only prevented mesh distortion, but also allowed for big timesteps where the evolution was slow, and forced small timesteps at the beginning, when the surface was too rough and the timescale very fast. Since the pictures correspond to the solution obtained every 9 timesteps, we observe that the timestep control mechanism was able to capture the very disparate timescales present due to the fourth order nature of this problem.

On the other hand, Figure 5.6 reveals unnecessary clustering of nodes in smooth regions and lack of resolution in other regions. This is tackled by space adaptivity and is discussed next.

5.4 Space Adaptivity

In this section we present a method for refining/coarsening meshes that define a surface Γ, with the purpose of having an accurate representation of Γ in the sense that the density of nodes should correlate with the local variation (regularity) of Γ. We cannot rely on parametrizations to quantify regularity of Γ because this concept would not be invariant under reparametrization. Therefore, we need an intrinsic measure of regularity such as the second fundamental form $\nabla_S \vec{\nu}$ and not just its trace, namely the mean curvature κ which is at our disposal.

We thus argue as follows. Let $T_1, T_2 \in \mathcal{T}$ be two adjacent elements with unit normals $\vec{\nu}_1, \vec{\nu}_2$, which share the side (node in 2d) S. We could compute $\nabla_S \vec{\nu}$ as

$$\left|\nabla_S \vec{\nu}\right| \approx \frac{\left|\vec{\nu}_1 - \vec{\nu}_2\right|}{h_S} \approx \frac{\alpha_S}{h_S},$$

where h_S stands for the local meshsize at S and α_S for the angle between $\vec{\nu}_1$ and $\vec{\nu}_2$. Since the pointwise accuracy of the mesh in representing Γ is proportional to $h_S^2 \left|\nabla_S \vec{\nu}\right|$, we end up with the following test for mesh quality

$$h_S \alpha_S \leq \epsilon_s,$$

where ϵ_s is a given parameter. If we add refinement and coarsening parameters $\gamma_R, \gamma_C > 0$, we end up with the following algorithm.

Algorithm 5.4 (Mesh Adaptation).

```
1. Compute all $\alpha_S$ and let $\mathcal{A}_T := \sum_{S \subset T} h_S \alpha_S$, $\forall T \in \mathcal{T}$.
2. Let $\mathcal{A}_{\max}$ be the maximum $\mathcal{A}_T$.
3. If $\mathcal{A}_{\max} > \epsilon_s$, mark for refinement all the elements $T$
   having $\mathcal{A}_T > \gamma_R \mathcal{A}_{\max}$.
4. Perform $d-1$ bisections to every marked element.
5. Mark for coarsening all the elements $T$ having
   $\mathcal{A}_T < \gamma_C \mathcal{A}_{\max}$.
6. Coarsen the marked elements.
7. If the mesh was modified go to step 1.
```

The effect of mesh adaptation is twofold: first, it helps us get a better resolution close to edges and angles, and secondly, it reduces the computing time by decreasing the number of degrees of freedom in smooth regions. In Figure 5.7 we show the evolution of the $4 \times 1 \times 1$ prism presented before using now this adaptation routine; we took $\epsilon_s = 0.1$, $\gamma_C = 0.3$, $\gamma_R = 0.7$. The initial mesh is that of Figure 5.6 after applying Algorithm 5.4. We used the same mesh regularization and timestep control as before. Additionally, after each timestep, we ran the adaptation algorithm followed by two mesh regularizations. The saving in spatial degrees of freedom is apparent by comparing Figure 5.7 with Figure 5.6, for which 1154 vertices were employed throughout.

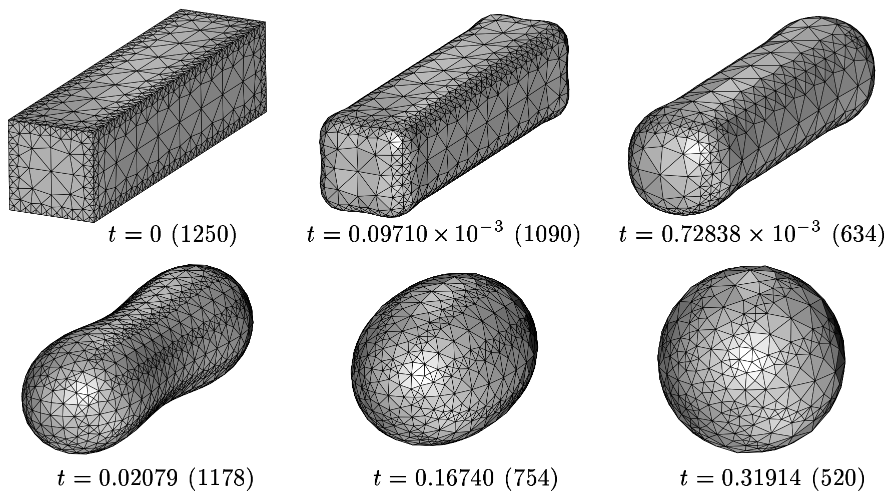

Figure 5.7: Evolution of a $4 \times 1 \times 1$ using timestep control, mesh regularization and *mesh refinement/coarsening*. Between parentheses we indicate the number of degrees of freedom (vertices) used to represent the surface and should be compared with 1154 for Figure 5.6 without space adaptivity. The parameters for the mesh refinement/coarsening routine were $\epsilon_s = 0.1$, $\gamma_C = 0.3$, $\gamma_R = 0.7$.

To further investigate the nonlinear dynamics of surface diffusion we compute the evolution of a longer prism, and we verify numerically that surface diffusion can lead to *pinch-off* depending on the aspect ratio of the initial surface, see Figures 5.8–5.10. During the evolution toward this topology change of the surface, some elements degenerate, especially those close to the pinch-off, producing in turn some loss of accuracy. Since it is known that wide angles are responsible for loss of accuracy, we introduce in §5.5 a procedure to control wide angles.

5.5 Angle Width Control

The routine for controlling the size of the widest angles is very simple, and it consists of a single splitting of those elements with angles wider than a certain threshold $\alpha_{\max}$, followed by n_{MR} mesh regularization sweeps.

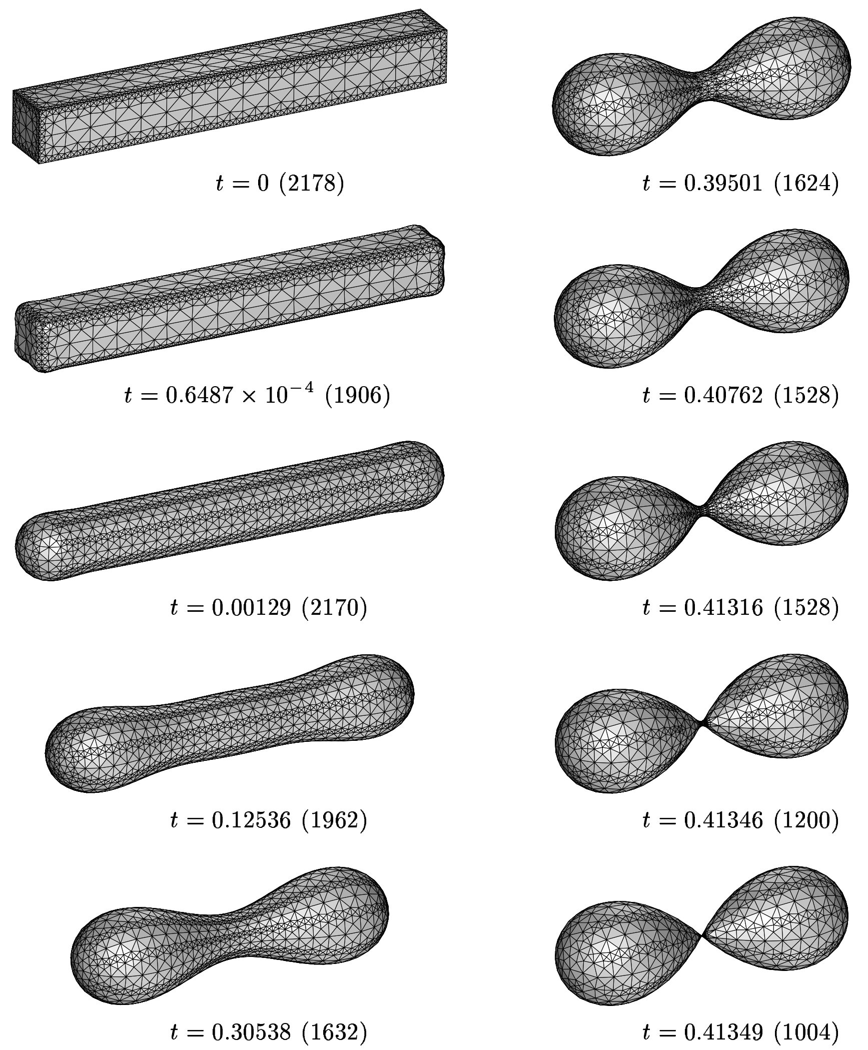

Figure 5.8: Pinch-off in finite time. Evolution of an $8 \times 1 \times 1$ prism at various time instants leading to a dumbbell and cusp formation (between parentheses we indicate the number of vertices used to represent the surface.) The evolution was computed using timestep control, mesh regularization, mesh refinement/coarsening, and a routine for *controlling wide angles.*

Algorithm 5.5 (Angle Width Control).

```
1. Mark all the elements having at least one angle bigger
   than $\alpha_{\max}$.
2. If there are elements marked,
   (a) Halve (one bisection) all the marked elements.
   (b) Perform $n_{\mathrm{MR}}$ regularization sweeps.
   (c) Go to 1.
3. If there are no elements marked, continue.
```

Here, $\alpha_{\max}$, n_{MR} are fixed parameters. The element halving is done following the newest-vertex bisection rule, which keeps the number of elements in a star uniformly bounded, but may not necessarily split the widest angle. The subsequent mesh regularization takes care of this issue. It is important to point out that only *one bisection* is done to the elements at this stage: two bisections would lead to elements having the same angles as the original! Figure 5.9 shows a detailed view of the evolution of the $8 \times 1 \times 1$-prism when approaching the pinch-off. The control of wide angles, coupled with mesh regularization, refinement and coarsening produce very good meshes, even very close to the pinch-off.

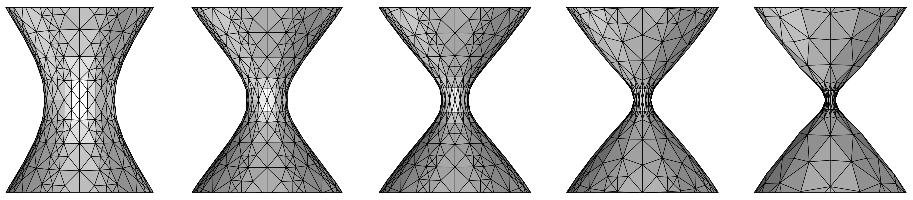

Figure 5.9: Detailed view of the pinch-off for the $8 \times 1 \times 1$ prism. The control of wide angles, coupled with mesh regularization, refinement and coarsening cure mesh distortion until the very moment of pinch-off, when the elements are rather elongated but not degenerate. An angle is considered to be wide when bigger than $120°$.

5.6 Full Adaptive Algorithm

We start this section by describing the final version of our adaptive algorithm for surface diffusion.

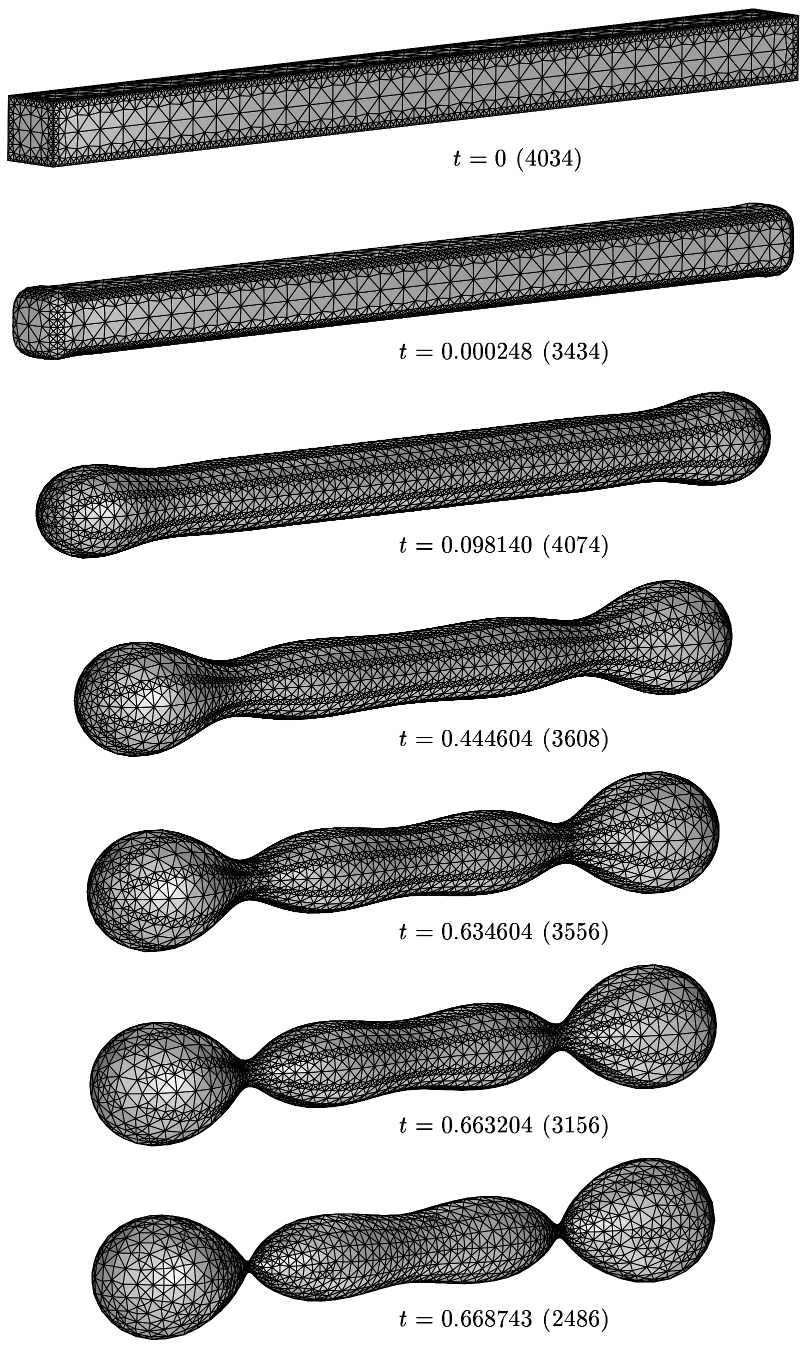

Figure 5.10: Evolution of a $16 \times 1 \times 1$ prism toward two simultaneous cusps revealing that the number of singularities depends on the aspect ratio of the initial prism. All the parameters used for this simulation are the same as those for the $8 \times 1 \times 1$ prism.

Algorithm 5.6 (Final Version of Surface Diffusion).

```
1. Start with an initial mesh, and let $\vec{\boldsymbol{X}}$ be the vector
   of coordinates. Let $\tau$ be the initial timestep.
2. Set the values for the following parameters:
   Mesh regularization: $n_{\mathtt{MR}} \in \mathbb{Z}_+$ (number of sweeps)
   Timestep control: $0 < \tau_{\mathtt{min}} < \tau_{\mathtt{max}}$, $\epsilon_t > 0$
   Space adaptivity: $\epsilon_s > 0$, $0 < \gamma_C < \gamma_R < 1$
   Control of angles width: $60^o < \alpha_{\mathtt{max}} < 180^o$.
3. Perform $n_{\mathtt{MR}}$ regularization sweeps (Algorithm 5.2).
4. Run the mesh adaptation routine (Algorithm 5.4).
5. If $d = 3$, run the routine for controlling wide angles
   (Algorithm 5.5).
6. Solve (4.10) for $\boldsymbol{V}$ and (4.11) for $\vec{\boldsymbol{V}}$.
7. Apply timestep control and update $\vec{\boldsymbol{X}}$ (Algorithm 5.3).
8. Go to 3
```

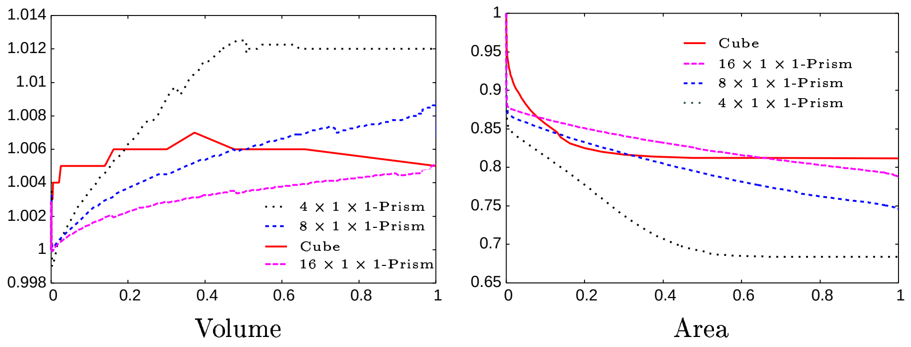

Figure 5.11: Relative volume and surface area with respect to the initial values vs. normalized time (t/T_{final}). The computations were performed with the full adaptive algorithm (Algorithm 5.6).

In order to obtain quantitative information of our algorithm we compared the behavior using the full adaptive algorithm in four test cases: a cube, a $4 \times 1 \times 1$-prism, an $8 \times 1 \times 1$-prism, and a $16 \times 1 \times 1$-prism. In all of the experiments we used the same parameters:

- Mesh regularization: $n_{\text{MR}} = 2$ (number of sweeps).
- Timestep control: $\epsilon_t = 0.1$ (tolerance), $\tau_{\text{min}} = 1 \times 10^{-7}$ (minimum timestep), $\tau_{\text{max}} = 5 \times 10^{-3}$ (maximum timestep).
- Space adaptivity: $\epsilon_s = 0.1$ (tolerance), $\gamma_R = 0.7$ (refinement threshold), $\gamma_C = 0.3$ (coarsening threshold).
- Control of angles width: $\alpha_{\text{max}} = 120^o$ (widest angle allowed).

Figure 5.11 shows volume and surface area vs. time. Volume change is minimal (less than 1.3%), and thus consistent with (2.5). Surface areas are always decreasing with t as predicted by (2.6).

Figure 5.12 provides information about the behavior of the timesteps due to the timestep control routine: it shows histograms with the number of timesteps used in every tenth of the whole time interval. In all the experiments, and due to the sharp sides of the initial prisms, which imply a fast motion of points, the timestep size was $\tau_{\min}$ at the beginning. This situation changes due to the smoothing effect of surface diffusion. For the cases where singularities occur ($8 \times 1 \times 1$- and $16 \times 1 \times 1$-prism) the timesteps are again very small at the end due to the infinite velocity of those points of the surface which are close to the pinch-off.

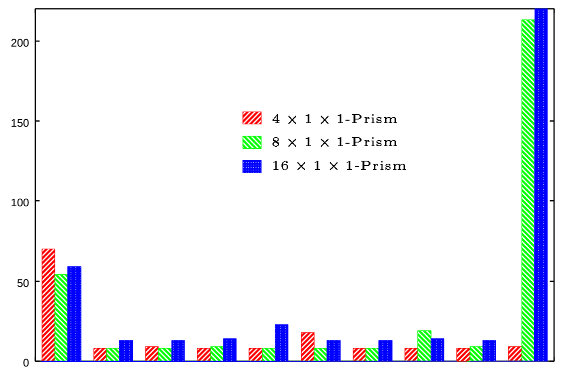

Figure 5.12: Timestep control: Number of timesteps used in each tenth of the whole time interval of computation. In all the experiments, and due to the sharp sides of the initial prisms, the timestep size was $\tau_{\min}$ at the beginning. For the cases where singularities occur ($8 \times 1 \times 1$- and $16 \times 1 \times 1$-prism) the timesteps are again very small at the end due to the infinite velocity of the points of the surface close to the pinch-off.

To end this section we present in Figure 5.13 the evolution of the corner of a cube using natural boundary conditions. Here we can observe in detail the evolution of sharp edges that, being rather singular for surface diffusion are handled transparently by our method.

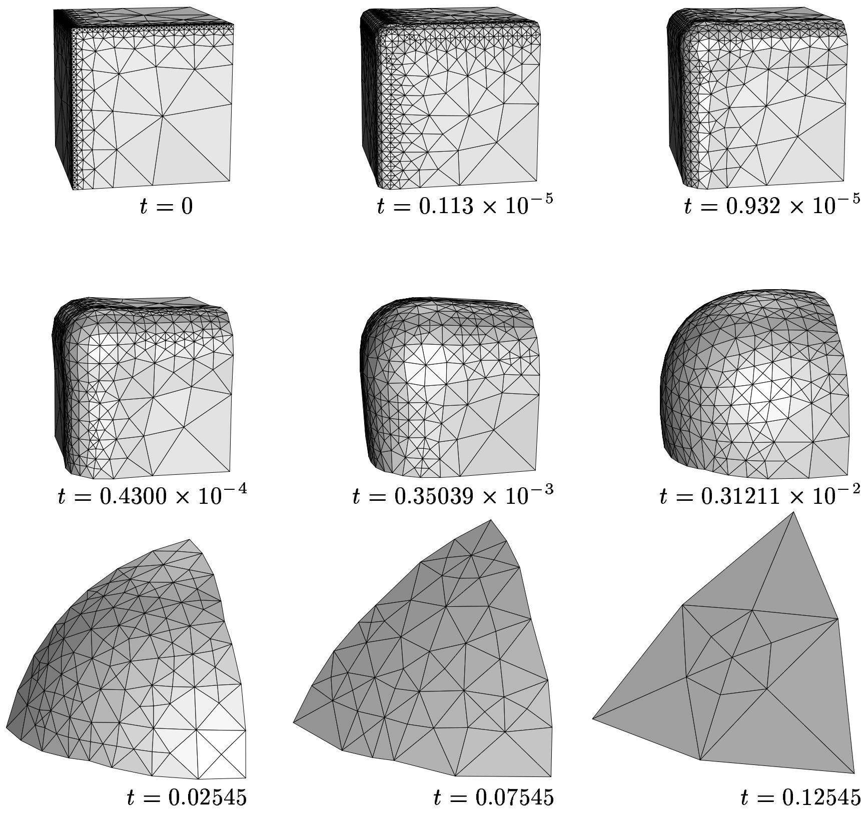
$t = 0$
$t = 0.113 \times 10^{-5}$
$t = 0.932 \times 10^{-5}$
$t = 0.4300 \times 10^{-4}$
$t = 0.35039 \times 10^{-3}$
$t = 0.31211 \times 10^{-2}$
$t = 0.02545$
$t = 0.07545$
$t = 0.12545$

Figure 5.13: Evolution of the corner of a cube using the full adaptive algorithm and *natural boundary conditions.*

5.7 Simulations of Curves in $\mathbb{R}^2$

We finally illustrate the behavior of curves in $\mathbb{R}^2$. Figure 5.14 shows the evolu-

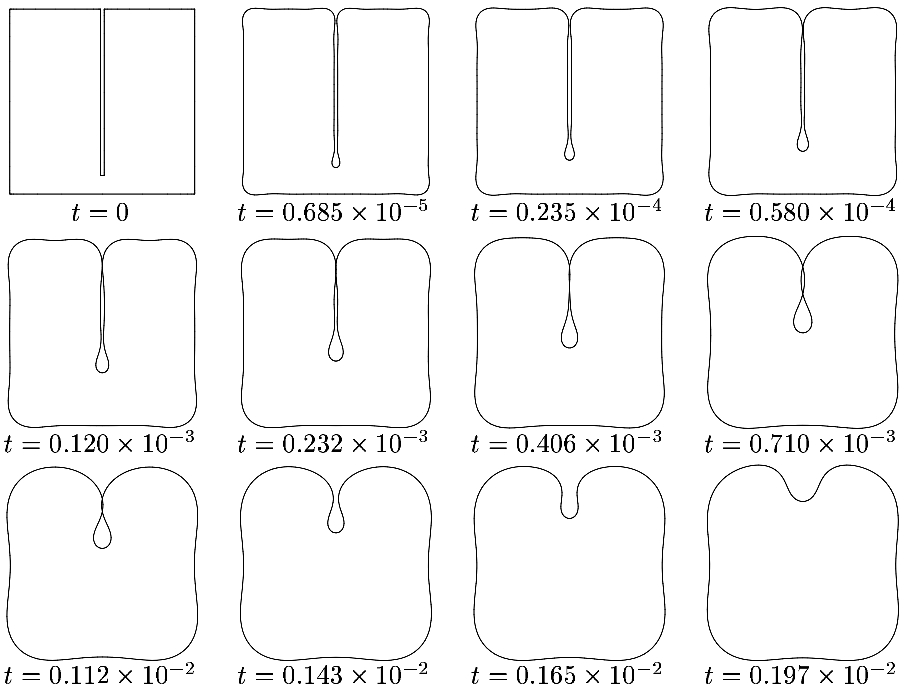
$t = 0$
$t = 0.685 \times 10^{-5}$
$t = 0.235 \times 10^{-4}$
$t = 0.580 \times 10^{-4}$
$t = 0.120 \times 10^{-3}$
$t = 0.232 \times 10^{-3}$
$t = 0.406 \times 10^{-3}$
$t = 0.710 \times 10^{-3}$
$t = 0.112 \times 10^{-2}$
$t = 0.143 \times 10^{-2}$
$t = 0.165 \times 10^{-2}$
$t = 0.197 \times 10^{-2}$

Figure 5.14: Bubble formation during the evolution of a curve by surface diffusion. Solution obtained every 60 adaptive timestesps. The curve defines initially an *almost slit* domain, next develops a mushroom shape before selfintersecting and crossing, and finally opens up. It is important to observe the very disparate time scales of this evolution. This purely geometric motion might be a mechanism for the creation of inclusions (or islands).

tion of a 2×2-square from which a very thin rectangle (0.02×1.8) is missing; we call it an *almost slit* domain. We observe here a pinch-off, followed by a curve crossing, which in contrast to $3d$ does not create a problem because both parts of the curve are evolving separately and do not see each other. The figure finally evolves to a circle, the stable asymptotic configuration in $2d$.

In Figure 5.15 we show the evolution of a four-leafed rose, which was computed previously by Escher et. al. using a finite difference scheme [18]. We plot the solutions obtained with our full adaptive algorithm for our values of t closest to those shown in [18]. The qualitative agreement of both computations is excellent.

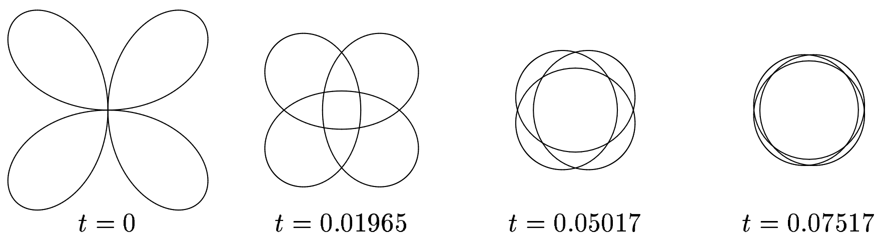

Figure 5.15: Evolution of the rose given in polar coordinates by $r(\theta) = \sin(2\theta)$. The stable asymptotic limit is a circle on which the curve winds three times. The qualitative agreement with the results presented in [18] is excellent.

6 Conclusions

We have devised and implemented a new FEM for the purely geometric motion of parametric surfaces (or curves) by surface diffusion. The scheme hinges on

- an operator splitting into second and zero order equations;
- dealing with both continuous scalar and vector velocities and curvatures, which relate weakly with the discontinuous unit normals;
- a semi-implicit time discretization, which leads to linear PDE to be solved at each time step, allows for relatively large time steps, and requires no explicit parametrization of the surface;
- an effective Schur complement approach for the solution of the ensuing linear systems;
- mesh smoothing to avoid mesh distortions, as well as space adaptivity and timestep control to optimize the computational effort.

We documented the performance of the new FEM with an extensive list of simulations, some exhibiting pinch-off, crossing, and mushroom formation in finite time. The algorithm is well suited for the study of surface diffusion as well as the coupling of it with other physical processes such as elasticity. In the present paper we restricted ourselves to considering closed surfaces or natural boundary conditions. The flexibility of finite elements, however, allows for other boundary conditions via slight changes in the implementation. Animations of the computational results presented above can be found in

`http://www.math.umd.edu/~rhn/SurfDiff/Movies`

We mention [3, 4] which uses the 2d version of our scheme for island dynamics with adatom diffusion and adsorption-desorption, where the dynamics of the island boundaries is governed by a two-sided flux together with surface diffusion.

Acknowledgments

We would like to thank K. G. Siebert for his assistance with new data structures to handle surfaces in $\mathbb{R}^3$ and curves in $\mathbb{R}^2$ within ALBERT [22], [23]. We also thank J. Colin for bringing this research area to our attention and F. Haußer for pointing out (and fixing!) a bug in our 2d code.
We were all partially supported by the NSF-DAAD international cooperation grants INT-9910086 and INT-0129243. We are also grateful to the Isaac Newton Institute, Cambridge UK, where part of this work was done. The programme *Computational Challenges in PDEs 2003* at the Isaac Newton Institute provided an excellent support as well as a stimulating and friendly scientific environment.

References

[1] R. J. Asaro, W. A. Tiller, *Surface morphology development during stress corrosion cracking: Part I: via surface diffusion*, Metall. Trans., 3 (1972), 1789-1796.

[2] E. Bänsch, *Finite element discretization of the Navier–Stokes equations with a free capillary surface*, Numer. Math. 88, 203-235 (2001).

[3] E. Bänsch, F. Hausser, O. Lakkis, B. Li, A. Voigt, *Finite element method for epitaxial growth with attachment-detachment kinetics*, J. Comput. Physics (to appear.)

[4] E. Bänsch, F. Hausser, A. Voigt, *Finite element method for epitaxial growth with thermodynamic boundary conditions*, Preprint no. 873, WIAS–Berlin (2003). Submitted.

[5] A.J. Bernoff, A.L. Bertozzi, T.P. Witelski, *Axisymmetric surface diffusion: Dynamics and stability of self-similar pinchoff*, J. Stat. Phys. 93, 725–776 (1998).

[6] E. Bänsch, P. Morin, R.H. Nochetto, *Finite element methods for surface diffusion.* In: P. Colli, C. Verdi, A. Visintin (eds.), *Free Boundary Problems.* International Series of Num. Math., vol. 147, 53–63, Birkhäuser (2003)

[7] E. Bänsch, P. Morin, R.H. Nochetto, *Surface diffusion of graphs: variational formulation, error analysis and simulation*, SIAM J. Numer. Anal. (to appear.)

[8] J. Cahn, C.M. Elliott, and A. Novik-Cohen, *The Cahn-Hilliard equation with a concentration dependent mobility: motion by minus the Laplacian of the mean curvature*, Euro. J. Appl. Math. **7**, 287-301 (1996)

[9] J. Cahn, J. Taylor, *Surface motion by surface diffusion*, Acta Metall. Mater., 42 (1994), pp. 1045–1063.

[10] D.L. Chopp, J. Sethian, *Motion by intrinsic Laplacian of curvature*, Interfaces and Free Boundaries 1, 107–123 (1999).

[11] B.D. Coleman, R.S. Falk, M. Moakher, *Space-time finite element methods for surface diffusion with applications to the theory of stability of cylinders*, SIAM J. Sci. Comp., 17 (1996), 1434-1448.

[12] K. Deckelnick, G. Dziuk, *Convergence of a finite element method for the non-parametric mean curvature flow*, Numer. Math. 72 (1995), 197-222.

[13] K. Deckelnick, G. Dziuk, *Error estimates for a semi-implicit fully discrete finite element scheme for the mean curvature flow of graphs*, Interfaces and Free Boundaries 2, (2000), 341-359.

[14] K. Deckelnick, G. Dziuk, C. Elliott, *Fully discrete semi-implict second order splitting for anisotropic surface diffusion of graphs*, preprint.

[15] K. Deckelnick, G. Dziuk, C. Elliott, *Semidiscrete finite elements for axially symmetric surface diffusion*, SIAM J. Num. Anal., to appear.

[16] G. Dziuk, *An algorithm for evolutionary surfaces*, Numer. Math., 58, (1991) 603–611.

[17] C.M. Elliott, S. Maier-Paape, *Losing a graph with surface diffusion*, Hokkaido Math. J., 30 (2001), 297-305.

[18] J. Escher, U.F. Mayer, G. Simonett, *The surface diffusion flow for immersed hypersurfaces*, SIAM J. Math. Anal., 29 (1998), 1419–1433.

[19] Y. Giga, K. Ito, *On pinching of curves moved by surface diffusion*, Comm. Appl. Anal., 2 (1998), 393–405.

[20] U.F. Mayer, *Numerical solutions for the surface diffusion flow in three space dimensions*, Comp. Appl. Math. (to appear).

[21] A. Schmidt, *Computation of three dimensional dendrites with finite elements*, J. Comput. Physics, 125, 293-312 (1996).

[22] A. Schmidt, K.G. Siebert, *ALBERT: An adaptive hierarchical finite element toolbox*, Documentation, Preprint 06/2000 Universität Freiburg, 244 p.

[23] A. Schmidt, K.G. Siebert, *Concepts of the finite element toolbox ALBERT*, Preprint 17/1998 Universität Freiburg, 12 p., to appear in Notes on Numerical Fluid Mechanics.

[24] P. Smereka, *Semi-Implicit Level Set Methods for Curvature and Surface Diffusion Motion*, J. Sci. Comp. 19, nos. 1–3, 439–456 (2003).

[25] B.J. Spencer, S.H. Davis, P.W. Voorhees, *Morphological instability in epitaxially-strained dislocation-free solid films: nonlinear evolution*, Phys. Rev. B, 47 (1993), 9760-9777.