A shape optimization approach for simulating contact of elastic membranes with rigid obstacles

A Thesis

Submitted For the Degree of

Master of Technology (Research)

in the Faculty of Engineering

by

Akriti Sharma

Mechanical Engineering

Indian Institute of Science

BANGALORE – 560 012

July 2019

To

My family

and

teachers

Acknowledgements

First and foremost I would like to thank my advisor Ramsharan Rangarajan for his enduring efforts, patient guidance and valuable suggestions throughout the journey. I would like to thank my parents and sister for believing in my decisions and supporting me in my low times. I would like to thank all my lab mates especially Karan and Anindya for always being there in the time of need and for providing me with a fresh perspective when I seemed to be stuck. I will definitely miss the company of my classmates and seniors without whom many adventures would not have happened.
Finally, I am grateful to the Ministry of Human Resource and Development for providing financial assistance, the Department of Mechanical Engineering for providing administrative support and Indian Institute of Science for providing such a peaceful atmosphere and taking care of our safe stay and health.

Publication

This thesis is based in its entirety on the below publication:

- Sharma, A, Rangarajan, R. A shape optimization approach for simulating contact of elastic membranes with rigid obstacles. Int J Numer Methods Eng. 2019; 117: 371 404. https://doi.org/10.1002/nme.5960

Abstract

The obstacle problem consists in computing equilibrium shapes of elastic membranes in contact with rigid obstacles. In addition to the displacement u of the membrane, the interface Γ on the membrane demarcating the region in contact with the obstacle is also an unknown and plays the role of a free boundary. Numerical methods that simulate obstacle problems as variational inequalities share the unifying feature of first computing membrane displacements and then deducing the location of the free boundary a posteriori. We present a shape optimization-based approach here that inverts this paradigm by considering the free boundary to be the primary unknown, and compute it as the minimizer of a certain shape functional using a gradient descent algorithm. In a nutshell, we compute Γ then u, and not u then Γ.

Our approach proffers clear algorithmic advantages. Unilateral contact constraints on displacements, which render traditional approaches into expensive quadratic programs, appear only as Dirichlet boundary conditions along the free boundary. Displacements of the membrane need to be approximated only over the noncoincidence set, thereby rendering smaller discrete problems to be resolved. The issue of suboptimal convergence of finite element solutions stemming from the reduced regularity of displacements across the free boundary is naturally circumvented. Most importantly perhaps, our numerical experiments reveal that the free boundary can be approximated to within distances that are two orders of magnitude smaller than the mesh size used for spatial discretization. The success of the proposed algorithm relies on a confluence of factors— choosing a suitable shape functional, representing free boundary iterates with smooth implicit functions, an ansatz for the velocity of the free boundary that helps realize a gradient descent scheme, and triangulating evolving domains with universal meshes. We discuss these aspects in detail and present numerous examples examining the performance of the algorithm.

Keywords: *obstacle problem; free boundary; variational inequalities; shape functional; gradient descent; universal meshes;*

Contents

List of Figures

Chapter 1

Introduction

The *obstacle problem* consists in computing the deformation of a thin elastic membrane subject to external loads and a frictionless unilateral contact constraint. Figure 1.1 provides a conceptual illustration of the problem, where the reference configuration of the membrane is the planar domain Ω. The membrane is clamped along its periphery and is loaded by an external force f. The presence of an obstacle is conveyed by the height function ψ. Denoting the deflection of the membrane by u, the obstacle problem seeks an equilibrium configuration of the membrane while satisfying the contact constraint that $u \leq \psi$ on Ω.

The obstacle problem represents one of the simplest unilateral contact problems in classical linear elasticity. The ansatz that the potential energy of a deformed membrane depends on the change in its area immediately reveals its close connection with problems of minimal surfaces. Its historical significance is also evidenced by the fact that it serves as a model for studying variational inequalities and free boundary problems [1, 2, 3]. In an engineering context, various applications related to casting, solidification, electrochemical shaping, phase transformations (Stefan problems), Hele-Shaw flows, the famous dam problem, and even certain pricing models in finance can be interpreted and resolved as obstacle problems following a suitable transformation of the unknowns [4, 5, 6].

The motivation behind the algorithm we propose here for numerically simulating the obstacle problem stems from the observation that in addition to the membrane deflection u, the set of points where the membrane makes contact with the obstacle, i.e., the *coincidence set* $\mathrm{C} \subset \Omega$, is not known a priori and must be determined as part of the solution. The coincidence set and its boundary Γ have clear physical interpretations in obstacle problems and it is therefore imperative to compute these sets accurately. In such a context, we highlight that an essential feature of the proposed algorithm is that it considers the location of the free boundary to be the primary unknown. In particular, the coincidence set is *not* inferred from the deflection of the membrane as is commonly done in existing algorithms. Instead, we directly compute Γ as the minimizer of a certain shape functional. Such an approach has clear algorithmic

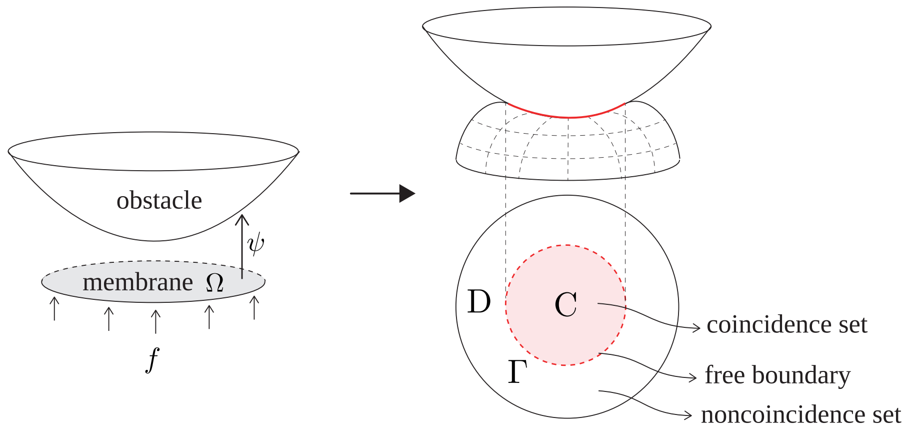


Figure 1.1: An illustration of the obstacle problem. A flat elastic membrane Ω is clamped along its periphery and subjected to a load distribution f. The membrane is constrained to remain below a rigid obstacle having a height function ψ. As indicated in the picture on the right, the coincidence set C is the collection of points where the membrane makes contact with the obstacle. Its boundary is denoted by Γ. The complement of C is the noncoincidence set D. Both the displacement of the membrane and the free boundary location are unknowns.

advantages, some of which we mention in the ensuing discussions and subsequently alongside numerical experiments presented in chapter 5.

Finite element methods for resolving obstacle problems predominantly rely on interpreting it as a variational inequality posed on a Hilbert space (the Sobolev space $\mathrm{H}^1_0(\Omega)$), or equivalently, as a problem of constrained energy minimization [7]. Throughout our presentation, we shall assume that the potential energy of an elastic membrane is given by the Dirichlet functional of its deflection. While this amounts to replacing the area measure of the deformed membrane by a linearized version, the problem remains nonlinear owing to the unilateral constraint. The set of admissible solutions $\mathrm{K} \subset \mathrm{H}^1_0(\Omega)$ is convex and the deflection u is endowed with a geometric characterization [1] as a projection onto K [2]. Well-posedness of finite element approximations of the obstacle problem also generally follow for the same reason, with the set of admissible solutions K_h being a convex subset of the finite element space $\mathrm{V}_h \subset \mathrm{H}^1_0(\Omega)$, where h denotes the mesh size parameter.

One of the main challenges in designing finite element methods for the obstacle problem lies in how the set of admissible functions is restricted to satisfy the contact constraint, i.e., in constructing the subset K_h of V_h. Invariably, this requires constraining nodal degrees of freedom of finite element functions by imposing constraints of the form $u_h(x_a) \leq \psi(x_a), 1 \leq a \leq n$, where $u_h \in \mathrm{V}_h$ and $\{x_a\}_{a=1}^n$ are the locations of the finite element nodes. The resulting discrete problem for computing u_h is an expensive quadratic program— minimize a quadratic energy functional while satisfying linear inequality constraints. We also recognize that such methods are in general nonconforming because functions in K_h

[1] In comparison, the solution of the analogous *variational equality* is interpreted as a projection onto a subspace of admissible solutions (Lax-Milgram theorem, [8]).

only satisfy the contact constraint at a prescribed set of points (nodes). More specifically, $\mathrm{K}_h \not\subset \mathrm{K}$, which can be interpreted as replacing the obstacle with a perturbed one in the discrete problem. Nevertheless, thanks to the stability of the solution u to the problem data (external forces, boundary conditions and obstacles), such variational crimes do not generally hinder convergence of the discrete solution u_h to u with mesh refinement [9].

Penalty and duality formulations of the obstacle problem are notable alternatives that simplify the inclusion of contact constraints. In the former, the set of admissible solutions is the unconstrained linear space V_h, but deviations of candidate solutions from the contact constraint are penalized by a suitable modification of the energy functional [10, 11, 12]. Convergence analyses of such methods reveal that the penalty parameter should scale as the inverse square of the mesh size. Large values of penalty parameters prevent penetration of the membrane into the obstacle, but also makes the conditioning of the resulting linear system that needs to be resolved progressively poorer. In dual formulations, on the other hand, Lagrange multipliers serve to impose the unilateral constraint and the resulting saddle point problem can be discretized and resolved with mixed finite element methods [13, 14]. The multiplier field has the useful interpretation of being the traction exerted by the obstacle on the membrane. In this context, we mention that augmented Lagrangian methods inherit the benefits of both penalty and dual formulations, with additionally permitting the penalty parameter to remain bounded [15, 16].

Irrespective of the specific formulation employed, a common feature of the methods mentioned above is that the deflection of the membrane serves as the principal unknown. This choice poses important difficulties in practice. First, the regularity of u is limited by the contact condition irrespective of the smoothness of the problem data [17, 18], which stands in clear contrast to the case of variational equalities, where the regularity of the solution is generally commensurate with that of the problem data. In particular, the curvature of the membrane is discontinuous across the free boundary regardless of the smoothness of the obstacle and $u \in \mathrm{H}^2(\Omega)$ at best. As a consequence, while linear finite elements yield optimal convergence for membrane displacements (in the energy norm for instance), the approximation is suboptimal even with quadratic elements [19, 20]. For this reason, primal finite element methods are invariably limited to piecewise linear polynomial basis functions. Similar remarks also apply to the case of approximation with mixed methods for dual formulations [13]. Consequently, the accuracy of discrete solutions $\{u_h\}_h$ to the obstacle problem relies on mesh refinement in an essential way, and adaptive refinement strategies based on a posteriori error estimators specifically designed for obstacle problems are crucial in practice [21, 22]. It is also possible to devise post-processing techniques with sub-grid scale calculations to improve the accuracy of the solution, see [23] for such ideas albeit in the setting of finite difference methods.

A second consequence of resolving the obstacle problem by considering the membrane deflection to be the primary unknown concerns identifying the coincidence set. The problem here is that it is possible to make small changes to the obstacle that results in large changes in the coincidence set and the free boundary. This observation is indicative of an issue of stability of the free boundary in the continuous problem. The well-known minimum speed criterion, which is a nondegeneracy condition, addresses this. It is, of course, our intention to only consider obstacle problems that are well-posed with the pairing (u, Γ) considered as the unknown. Nevertheless, subtle questions remain when computing the coincidence set and the free boundary a posteriori from the deflection of the membrane, starting with the fact that the nondegeneracy condition is in general not satisfied by finite element approximations u_h of the membrane deflection even if it holds for the exact solution u. Specifically, it is possible to have an arbitrarily close approximation u_h of u, and yet to not be able to infer much about the location of the free boundary Γ. A one-dimensional example illustrating this point can be found in [24]. One of the key points revealed by the analysis in [25, 26] is that it is not just convenient but in fact imperative to introduce tolerances when identifying the approximate coincidence set C_h and the approximate free boundary Γ_h from the finite element solution u_h. The choice of these tolerances is intimately related to error estimates for $u - u_h$ and to the minimum speed criterion satisfied by u. As astutely noted in [24] however, the error in the free boundary measured as a distance "*cannot beat the mesh size*" despite these estimates. Here again, adaptive mesh refinement is essential for improving the approximation of the free boundary, so that the error in Γ_h is limited by the local mesh size. Notably, such mesh refinement is required irrespective of the geometric features of the free boundary.

In light of the above discussions, the shape optimization based approach we propose here with Γ being the principal unknown provides an attractive alternative for resolving obstacle problems. This idea is reinforced by the qualitative observation that unlike the membrane deflection, the free boundary is expected to be as regular as the problem data [27, 28, 29]. It is also not surprising that the free boundary could be optimal in some sense. Roughly speaking, the deflection u satisfies a Poisson problem governing static equilibrium of the membrane over the noncoincidence set $\mathrm{D} = \Omega \setminus \mathrm{C}$, and the Cauchy data $u = \psi, \nabla u = \nabla \psi$ represent over specified boundary conditions along Γ. The location of the free boundary is such that this problem on D has a solution. This feature of having overspecified data along the free boundary is typical in free boundary problems in general, and especially of Bernoulli-type free boundary problems [30, 31].

Central to the success of a shape optimization approach for computing the free boundary is identifying a suitable domain/shape functional for which Γ serves as the minimizing element. This question is rather non-trivial. In chapter 2, we demonstrate using simple examples that seemingly natural choices for domain functionals may have multiple stationary points or an inflection at the correct solution and are therefore unsuitable in practice. Our choice for the domain functional, which we label as J_τ, is based on the work in [32]. There it has been shown that certain conditions on the problem data, which are not overly restrictive, suffice to guarantee that Γ is the unique minimizer of J_τ. The shape derivative of J_τ is straightforward to evaluate and furnishes an evolution "velocity" for the free boundary, which in turn serves as the basis for a gradient descent algorithm discussed in chapter 3. Hence we start with an initial guess for the free boundary, compute the membrane deflection and adjoint variables over the noncoincidence set using a standard finite element method, use these fields to evaluate the velocity of the free boundary, and advect the free boundary to the next guess while using a sufficiently small time step. This procedure, which constitutes one iteration, is repeated until the free boundary is deemed to have converged in some sense. Notice that at each iteration in this scheme, the deflection of the membrane is approximated over a domain defined by a guess for the free boundary. Such a paradigm is in stark contrast with the aforementioned algorithms that estimate the location of the free boundary from the computed deflection of the membrane. In chapters 3 and 5, we discuss in detail some of the algorithmic advantages of the proposed shape optimization based scheme. For now, we highlight that finite element solutions need only be computed over the noncoincidence set at each iteration, and in particular, not over the entire domain Ω. Besides the obvious benefit of having to resolve discrete problems with fewer degrees of freedom, we conspicuously avoid the detrimental effects of the lack of regularity of u along the free boundary. By virtue of the free boundary being the primary unknown, the contact condition that the deflection of the membrane equal the height of the obstacle over the coincidence set is automatically imposed without the need for any inequality constraints. Indeed, the unilateral constraint only manifests as a Dirichlet boundary condition for the linear problem being resolved over the noncoincidence set.

Evidently, the proposed algorithm entails computing finite element approximations of membrane displacement and its adjoint over noncoincidence sets which necessarily evolve with the free boundary iterates. Hence computing a sequence of approximations $\{\Gamma_k\}_k, k = 1, 2, \ldots$ for the free boundary requires resolving Poisson problems over a corresponding sequence of noncoincidence sets $\{D_k\}_k$. Such a scenario is naturally expected when simulating shape optimization problems, and we encounter the familiar challenge of how to discretize domains $\{D_k\}_k$ that may change drastically with each iteration. To this end, we adopt the idea of *universal meshes* introduced in [33] for such problems involving moving boundaries and evolving domains. In a nutshell, we consider a fixed triangulation over Ω as a universal mesh for the entire sequence of noncoincidence sets $\{D_k\}_k$ realized in the shape optimization scheme. At the k^{th} iteration, the current guess for the free boundary Γ_k is immersed in the universal mesh and a specific collection of its elements are mapped onto a conforming mesh over D_k. Details of this simple meshing algorithm are discussed and analyzed in [34, 35]. We include a terse outline of the steps involved in chapter 4. We note that unlike arbitrary Lagrangian-Eulerian methods, universal meshes permit large motions of the free boundary and by virtue preserving element connectivities enable retaining sparsity patterns of finite element data structures throughout the course of a simulation.

An important detail that arises when directly computing the free boundary concerns its representation. Since the free boundary can undergo dramatic shape changes during the course of a simulation, as will be evident in the numerical examples presented in chapter 5, adopting a parametric representation is ill-advised. Indeed, it becomes necessary to periodically reparameterize free boundary iterates to account for evolving curvatures and other geometric features. For this reason, we adopt an implicit representation instead, by defining the location of the free boundary as the zero level set of a function. While an implicit representation is not a requirement in the proposed algorithm, it is certainly convenient. Alternate boundary representations, including parametric representations as splines, can be used as well. We recognize that the application of level set methods to resolving obstacle problems [36], and more generally to problems of shape optimization [37, 38, 39] is not new. However, besides the use of an implicit representation, little else is common between our approach and level set methods.

We caution that the shape optimization problem we consider here is *not* a design problem involving contact constraints. Our goal is to compute the deflection and the coincidence set for a membrane in contact with an obstacle. It is not, for instance, to optimize the contact pressure or the area of contact [40, 41]. Similarly, it is not a problem of optimizing the domain shape/topology to extremize energy functionals arising in contact problems [42]. Nevertheless, a majority of the algorithmic ideas we introduce or adopt here apply verbatim to resolving such design-related problems as well.

We remark that it may be possible to exploit the ideas introduced here to benefit simulations of problems involving the mechanics of biological membranes [43, 44]. In particular, a broad range of applications related to targeted drug delivery require a detailed understanding of the mechanics of contact between nanoparticles and lipid vesicles or cell membranes [45, 46, 47]. After all, the region of contact is the location of chemical interactions between ligand molecules and receptors, and therefore plays a decisive role [48]. Compared to the obstacle problem we study here, the mechanics of contact interactions in biomembranes requires accounting for the curvature-dependent bending elasticity of the membrane [49] as well as adhesion between the membrane and the particle [50, 51]. While the idea of directly computing the contact region in such problems is appealing, the extension of our approach to these problems is likely to be nontrivial.

Organization: We begin in chapter 2 with a discussion of the choice of the shape functional for which the free boundary serves as a minimizer. With the aid of simple axisymmetric examples, we demonstrate that seemingly natural choices are ill-suited for our purposes. We then define the functional J_τ from [32] that we adopt in all our numerical experiments. With the shape functional at hand, we introduce an iterative gradient descent algorithm for computing the free boundary in chapter 3. Therein, we discuss the representation adopted for free boundary iterates using implicit functions, computing the sensitivity of the objective functional to identify descent directions and provide a backtracking algorithm that ensures a monotonically decreasing sequence of functional values. We briefly discuss meshing noncoincidence sets defined by the evolving free boundary using universal meshes in chapter 4, which enables us to compute finite element approximations of the membrane displacement and the adjoint variable. We present a series of numerical experiments in chapter 5 which are aimed at examining the performance of the proposed algorithm, demonstrating its convergence properties, and quantifying the accuracy of the computed free boundary and membrane displacements. In section 5.4.1, we also compare our algorithm with a variational inequality solution. Following a few concluding remarks in chapter 6, a detailed derivation of the sensitivity of the objective functional J_τ is provided in A.

Chapter 2

Choosing a shape functional

Descriptions of the obstacle problem as a variational inequality or as a problem of constrained energy minimization are well known [3, 7]. Our goal in this chapter is to motivate the choice of a domain functional of which the free boundary is a minimizer. Using examples of a circular membrane in contact with axisymmetric obstacles, we discount a couple of seemingly natural choices. We then introduce the functional adopted in the proposed algorithm. Throughout our discussions, we omit qualifying the requisite smoothness of the problem data, preferring instead to assume sufficient regularity.

2.1 The free boundary problem

Referring to Figure 1.1, we consider an elastic membrane having a reference configuration $\Omega \subset \mathbb{R}^2$ that is clamped along the boundary $\partial\Omega$. The membrane is loaded by a force $f : \Omega \to \mathbb{R}$ and is constrained to lie below an obstacle having a height function $\psi : \Omega \to \mathbb{R}$, resulting in a deflection $u : \Omega \to \mathbb{R}$ measured normal to the plane of Ω. The set of points in Ω where the membrane makes contact with the obstacle is the coincidence set

$$\mathrm{C} \triangleq \{x \in \Omega \,:\, u(x) = \psi(x)\}.$$

The complementary part of the membrane is the noncoincidence set $\mathrm{D} \triangleq \Omega \setminus \mathrm{C}$, and the free boundary is the set $\Gamma \triangleq \partial\mathrm{D} \cap \Omega$. The conditions for static equilibrium of the membrane are summarized as [3]

$$\begin{aligned}
-\Delta u &= f && \text{on D}, && \text{(2.1a)}\\
u &= 0 && \text{on } \partial\Omega, && \text{(2.1b)}\\
u &= \psi && \text{on C (and } \Gamma \text{ in particular)}, && \text{(2.1c)}\\
\partial u/\partial \mathbf{n} &= \partial\psi/\partial\mathbf{n} && \text{on } \Gamma, && \text{(2.1d)}
\end{aligned}$$

where $\mathbf{n}$ denotes the unit normal to Γ pointing away from D. Equation (2.1a) represents a balance between the tension in the membrane and the external loading over the region where the membrane does not make contact with the obstacle. Over the coincidence set C, the deflection simply satisfies $u = \psi$. A homogeneous Dirichlet boundary condition along $\partial\Omega$ is given by eq. (2.1b), while eqs. (2.1c) and (2.1d) represent transmission conditions along the free boundary. The overspecified boundary conditions along Γ serve as a reminder that it is also an unknown in eq. (2.1), i.e., that Γ is a free boundary. We shall assume that $\psi > 0$ over $\partial\Omega$, so that C and Γ are compactly contained in Ω.

2.2 Axisymmetric problems: exact solutions

In the ensuing discussions, we will use the example of a circular membrane in contact with an axisymmetric obstacle to examine different choices for the shape functional. We set Ω to be a circular membrane of radius R centered at the origin of coordinates and assume that the obstacle is centered above the origin and has a height function $\psi(r)$. For simplicity, we set the forcing f to be a constant. It is then straightforward to verify that a solution to the obstacle problem satisfying all the conditions in eq. (2.1) is given by

$$u(r) = \begin{cases} \psi(r) & \text{for } r \le R_\Gamma, \\ -fr^2/4 + \alpha \log r + \beta & \text{for } R_\Gamma \le r \le R, \end{cases} \tag{2.2}$$

where the parameters $(\alpha, \beta, R_\Gamma)$ constitute a solution to the algebraic system of equations

$$\begin{aligned}
-fR^2/4 + \alpha \log R + \beta &= 0,\\
-fR_\Gamma^2/4 + \alpha \log R_\Gamma + \beta &= \psi(R_\Gamma),\\
-fR_\Gamma/2 + \alpha/R_\Gamma &= \psi'(R_\Gamma),
\end{aligned} \tag{2.3}$$

and Γ coincides with the boundary of the circle having radius R_Γ and centered at the origin.

For the specific example of a flat obstacle positioned parallel to the plane of Ω and at a height Z above the origin, the height function is $\psi(r) = Z$. Then, we find that

$$\left.\begin{aligned} R &= 1 \\ Z &= f\left(1 - 3\,e^{-2}\right)/4 \\ \psi(r) &= Z \end{aligned}\right\} \quad \Rightarrow \quad \alpha = fe^{-2}/2, \beta = f/4, R_\Gamma = e^{-1}. \tag{2.4}$$

Similarly, for a spherical obstacle having radius ρ and centered at a height Z above the origin, the height function is given by $\psi(r) = Z - \sqrt{\rho^2 - r^2}$, and we find that

$$\left.\begin{aligned} R &= 1 \\ \rho &= \sqrt{10}\,e^{-1} \\ Z &= 1/4 + 8\,e^{-1}/3 - 3\,e^{-2}/4 \\ \psi(r) &= Z - \sqrt{\rho^2 - r^2} \end{aligned}\right\} \quad \Rightarrow \quad \alpha = e^{-1}/3 + e^{-2}/2, \beta = 1/4, R_\Gamma = e^{-1}. \tag{2.5}$$

Moreover, the parameters $(\alpha, \beta, R_\Gamma)$ in eqs. (2.4) and (2.5) are unique solutions to the respective realizations of eq. (2.3). The solution $r \mapsto u(r)$, its first and second derivatives are plotted in Figures 2.1a to 2.1c for the case of the flat obstacle, and in Figures 2.2a to 2.2c for the case of the spherical obstacle. While continuity of u' confirms that eq. (2.1d) is satisfied at the free boundary $r = R_\Gamma$, the kink in the profile of u' and the jump in u'' at the free boundary indicates the discontinuity in the curvature of the membrane across the contact interface.

2.3 A functional with an inflection point

We use the examples in eqs. (2.4) and (2.5) to examine the behavior of a few different shape functionals for the free boundary. We exploit the symmetry in these examples to restrict the set of admissible shapes of free boundaries to be circular and concentric with Ω. In this way, each admissible candidate Γ_η for the free boundary is associated with the scalar parameter η denoting its radius. Consequently, shape functionals of the free boundary are transformed into functions of η.

A guiding principle for posing eq. (2.1) as a shape optimization problem for the free boundary lies in enforcing one of the two constraints imposed on Γ in eqs. (2.1c) and (2.1d) through a shape functional. A natural choice is furnished by the observation that

$$u = \arg\min_{v \in \mathrm{K}} \mathrm{E}[v], \quad \text{where} \quad \mathrm{E}[v] = \frac{1}{2}\int_\Omega \nabla v \cdot \nabla v - \int_\Omega f v \quad \text{and} \quad K = \{v \in \mathrm{H}^1_0(\Omega) \,:\, v \leq \psi\},$$

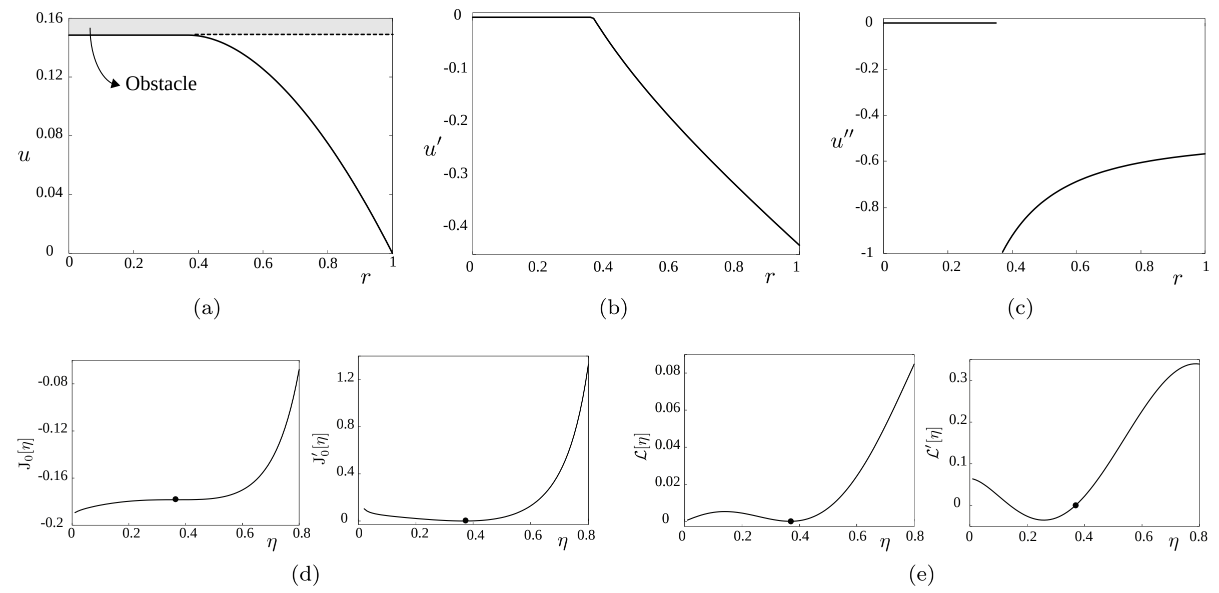


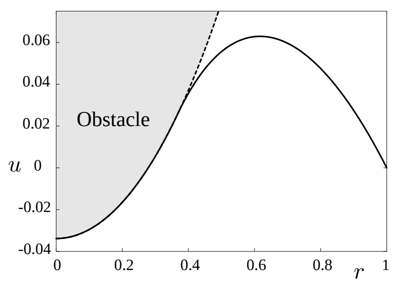
Figure 2.1: The axisymmetric solution $r \mapsto u(r)$ to the problem of a circular membrane loaded by a constant force and in contact with a flat obstacle is plotted in figure (a), and its derivatives are shown in (b) and (c). Details of the problem parameters and the solution are given in eq. (2.4). The continuity of u' at $\eta = R_\Gamma$ verifies that the condition $\partial u/\partial \mathbf{n} = \partial \psi/\partial \mathbf{n}$ is satisfied. The jump in u'' is indicative of the discontinuity in curvature across the contact interface. Figures (d) and (e) show the dependence of the functionals J_0 and $\mathcal{L}$ and their derivatives on the free boundary radius η. Both functionals have a stationary point at the correct free boundary radius, which is highlighted with a circular marker. However, the correct solution occurs at an inflection point of the functional J_0, while the functional $\mathcal{L}$ has a second stationary point. These features render J_0 and $\mathcal{L}$ unsuitable for simulating the obstacle problem.

leading us to consider the shape functional

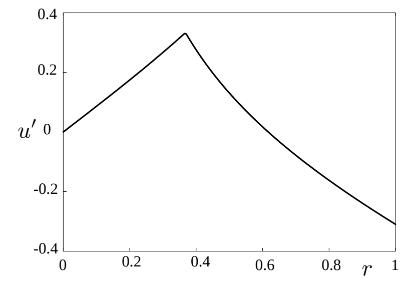
$$\mathrm{J}_0[\eta] \triangleq \mathrm{E}[u_\eta] \quad \text{where} \quad \begin{cases} -\Delta u_\eta = f & \text{on } \mathrm{D}_\eta = \{x \in \Omega \,:\, \eta < \|x\| < R\}, \\ u_\eta = 0 & \text{on } \partial\Omega, \\ u_\eta = \psi & \text{on } \mathrm{C}_\eta = \{x \in \Omega \,:\, \|x\| \leq \eta\}, \end{cases} \tag{2.6}$$

and $\|x\|$ denotes the Euclidean norm of $x \in \mathbb{R}^2$. In eq. (2.6), notice that we have omitted the condition $\partial u_\eta/\partial \mathbf{n} = \partial \psi/\partial \mathbf{n}$ along Γ_η; it will instead be enforced automatically at the optimal radius determined by extremizing J_0. For the example of the flat obstacle with parameters $R, Z, \psi(r)$ noted in eq. (2.4), the membrane deflection $u_\eta(r)$ parameterized by the free boundary radius $\eta \in (0,1)$ is given by

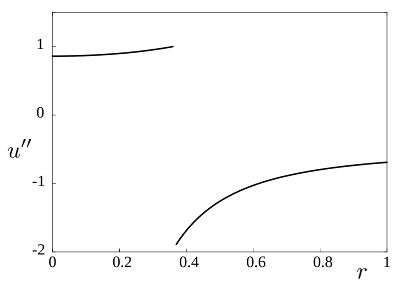
$$u_\eta(r) = \frac{1}{4}\left(\left(1 - r^2\right) + \frac{\left(\eta^2 - 3\,e^{-2}\right)\log r}{\log \eta}\right), \tag{2.7}$$



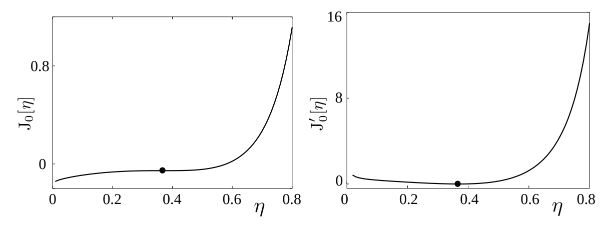
(a)



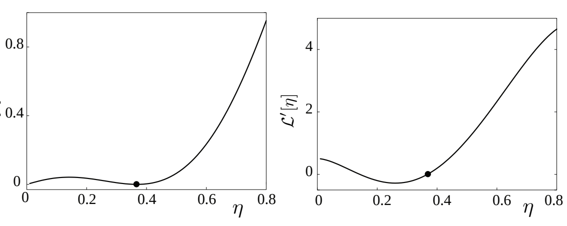
(b)



(c)



(d)



(e)

Figure 2.2: The solution $u(r)$ and its derivatives to the problem of a loaded circular membrane in contact with a spherical obstacle are shown in (a), (b) and (c), see eq. (2.5). The dependence of the functionals J_0 and $\mathcal{L}$ on the free boundary radius η are shown in (d) and (e). The observations made in Figure 2.1 for the case of the flat obstacle are reflected here as well. The correct solution $\eta = R_\Gamma$ is an inflection point of J_0, while $\mathcal{L}$ has multiple stationary points.

while for the example of the spherical obstacle in eq. (2.5), the solution u_η follows as

$$u_\eta(r) = \frac{1}{4}\left(1 - r^2\right) + \frac{\log r}{4\log\eta}\left(\eta^2 + \frac{32}{3}e^{-1} - 3e^{-2} - 4\sqrt{10e^{-2} - \eta^2}\right). \tag{2.8}$$

Inspecting the plots of $\eta \mapsto \mathrm{J}_0[\eta]$ in Figures 2.1d and 2.2d, we find that J_0 indeed has a stationary point at $\eta = R_\Gamma$, i.e., $\mathrm{J}_0'[R_\Gamma] = 0$. When computing the shape derivative of J_0 in section 3.3, we will see that this observation is not a coincidence— the exact solution for the free boundary is necessarily a stationary point of J_0 and the transmission condition $\partial u_\eta/\partial\mathbf{n} = \partial\psi/\partial\mathbf{n}$ is enforced at $\eta = R_\Gamma$. Unfortunately, the correct solution happens to be a point of inflection of J_0, which is evident from the fact that $\mathrm{J}_0'[\eta]$ is positive on either side of $\eta = R_\Gamma$. We conclude therefore that we cannot expect Γ to be a minimizer of J_0 in general.

2.4 A functional with multiple stationary points

An alternative to J_0 is the choice

$$\mathcal{L}[\eta] \triangleq \int_{\Gamma_\eta} (u_\eta - \psi)^2 \, d\Gamma \quad \text{where} \quad \begin{cases} -\Delta u_\eta = f & \text{on } \mathrm{D}_\eta, \\ u_\eta = 0 & \text{on } \partial\Omega, \\ \partial u_\eta / \partial \mathbf{n} = \partial \psi / \partial \mathbf{n} & \text{on } \Gamma_\eta. \end{cases} \tag{2.9}$$

In contrast to eq. (2.6), the state u_η in eq. (2.9) satisfies the flux condition eq. (2.1d) by definition while the constraint $u_\eta = \psi$ on Γ_η in eq. (2.1c) is enforced in a least-squares sense by the functional $\mathcal{L}$. Here again, the 1-parameter family of solutions $r \mapsto u_\eta(r)$ is straightforward to compute for each $\eta \in (0,1)$ for the examples in eqs. (2.4) and (2.5). The solution in the case of the flat obstacle with parameters (R, Z, ψ) given in eq. (2.4) is

$$u_\eta(r) = \frac{1}{4}\left(-r^2 + 2\eta^2 \log(r) + 1\right).$$

while for the case of the spherical obstacle with parameters (R, ρ, Z, ψ) in eq. (2.5) is

$$u_\eta(r) = \frac{1}{4}\left(1 - r^2\right) + \eta^2 \log r \left(\frac{1}{2} + \left(10e^{-2} - \eta^2\right)^{-1/2}\right).$$

Figures 2.1e and 2.2e inspect the dependence of $\mathcal{L}$ and its derivative on the free boundary radius η. As we did in the case of J_0, we find that $\mathcal{L}$ has a stationary point at the correct free boundary radius $\eta = R_\Gamma$. It can also be verified that $u_\eta(\eta) = \psi(\eta)$ at $\eta = R_\Gamma$, so that eq. (2.1c) is satisfied as well. However, we find that $\eta \mapsto \mathcal{L}[\eta]$ has multiple stationary points both in the case of the flat and the spherical obstacle.

2.5 Our choice: the functional J_τ with $\tau \geq 1$

The qualitative features of J_0 and $\mathcal{L}$ revealed with the aid of simple examples serves as a reminder that choosing a shape functional for computing the free boundary in the obstacle problem is a rather nontrivial task. The functional J_τ that we adopt is proposed in [32]. Denoting an admissible solution to the free boundary by γ and the corresponding membrane deflection by u_γ, we consider

$$\boxed{\mathrm{J}_\tau[\gamma] \triangleq \frac{1}{2}\int_\Omega \nabla u_\gamma \cdot \nabla u_\gamma + (\tau - 1)\int_\Omega f u_\gamma, \quad \text{where} \quad \begin{cases} -\Delta u_\gamma = f & \text{on } \mathrm{D}_\gamma, \\ u_\gamma = 0 & \text{on } \partial\Omega, \\ u_\gamma = \psi & \text{on } \mathrm{C}_\gamma, \end{cases}} \tag{2.10}$$

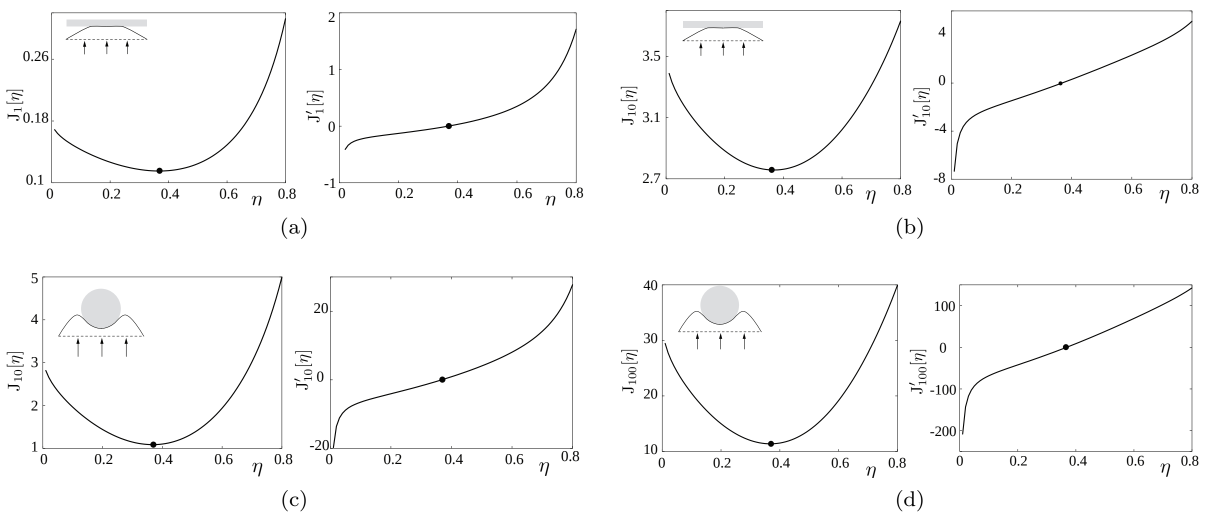


Figure 2.3: Plots showing the dependence of the shape functional $\mathrm{J}_\tau[\eta]$ on the free boundary radius η in the case of the flat and spherical obstacles for a few different values of the parameter $\tau \geq 1$. We find in each of the four examples considered in (a)–(d) that the correct free boundary radius $\eta = R_\Gamma$ is the unique minimizer of $\mathrm{J}_\tau[\eta]$. This observation is a manifestation of the results proved in [32] for general loadings and general shapes of domains, obstacles and free boundaries.

C_γ and D_γ are the coincidence and noncoincidence sets corresponding to γ. The rationale for choosing J_τ comes from the analysis in [32], where it is proven to have a unique minimizer at the exact solution provided that some sufficient conditions on the problem data hold. Evidently, J_0 coincides with J_τ for the choice $\tau = 0$. However, we shall expressly require that $\tau \geq 1$ based on the conditions identified in [32] and note that the claims on properties of J_τ for $\tau \geq 1$ do not generally extend to the choice $\tau = 0$.

Let us examine J_τ using the axisymmetric examples introduced in section 2.2. To this end, we identify γ with Γ_η, $\mathrm{J}_\tau[\gamma]$ with $\mathrm{J}_\tau[\Gamma_\eta]$, and u_γ with u_η in eq. (2.10). Then, the solutions $u_\eta(r)$ parameterized by the free boundary radius η are identical to the states computed in eqs. (2.7) and (2.8) for the examples of the flat and spherical obstacles with parameters defined in eqs. (2.4) and (2.5). Figure 2.3 shows plots of $\eta \mapsto \mathrm{J}_\tau[\eta]$ for a few representative values of $\tau \geq 1$. We find that irrespective of the value of τ chosen, J_τ has a unique stationary point at $\eta = R_\Gamma$ and that the stationary point is in fact a minimum. The shape sensitivity calculations in section 3.3 will reveal that the flux condition in eq. (2.1d) is automatically satisfied at $\eta = R_\Gamma$. These features of J_τ make it an ideal choice for our purpose of computing the free boundary for the obstacle problem.

The key assumptions considered in [32, Theorem 4.7] to guarantee a unique minimizer for J_τ at the free boundary are the conditions

$$-\Delta\psi + (2\tau - 1)f \geq 0 \quad \text{on } \Omega,\ \text{provided } f \geq 0. \tag{2.11a}$$

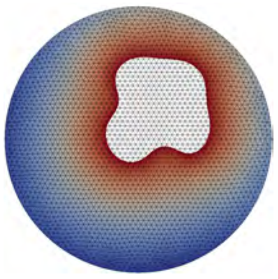
$$-\Delta\psi - f \neq 0 \quad \text{in any open set in } \Omega. \tag{2.11b}$$

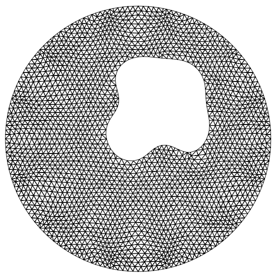
These restrictions on the problem data are only sufficient conditions, not necessary ones. Condition 2.11a is helpful in choosing the parameter τ in the shape functional. Condition 2.11b is invoked in [32, Theorem 5.7] to infer that a minimizing sequence of J_τ is itself convergent to the free boundary. The latter point is particularly relevant for the gradient descent algorithm discussed in the next chapter, where we compute free boundary iterates $\{\Gamma_k\}_k$ as a minimizing sequence of J_τ. The result mentioned is then crucial for deducing convergence of the computed sequence $\{\Gamma_k\}$ to Γ. We intentionally omit discussing further technical details from [32] related to the analysis of J_τ, but mention that the conditions in eq. (2.11) are satisfied in all the numerical examples presented in chapter 5.

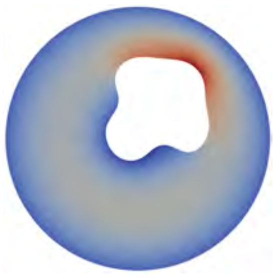
Chapter 3

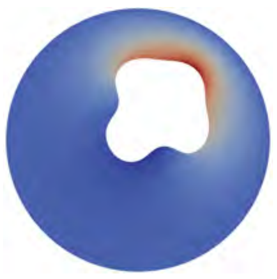
A gradient descent algorithm for computing the free boundary

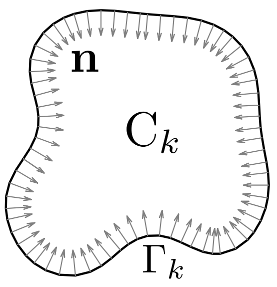
With the functional J_τ at hand, we propose a gradient descent algorithm for evolving an initial guess for the shape of the free boundary in an obstacle problem to its equilibrium configuration. Realizing such an algorithm requires making a few crucial choices, prominent among them being the representation adopted for free boundary iterates and an ansatz for the descent direction. The latter is often referred to as the free boundary "velocity" in the literature on shape optimization. We begin this chapter by providing a concise description of the algorithm in section 3.1. We discuss the implicit representation adopted for free boundary iterates using local maximum entropy basis functions in section 3.2. The velocity computed in the algorithm is based on the sensitivity of J_τ that is discussed in section 3.3, and is guaranteed to be a descent direction for the functional over sufficiently short time intervals. The post-processing technique used to improve the approximation of boundary fluxes for the state and its adjoint leading to more accurate calculations for the evolution velocity is discussed in section 3.4, which is followed by an assortment of remarks in section 3.5.


(a) Step 4: Nonpositive contours of ϕ_k superimposed on the universal mesh $\mathcal{U}_h$ over Ω. The boundary Γ_k is the zero level set of ϕ_k and D_k is the zero sublevel set.

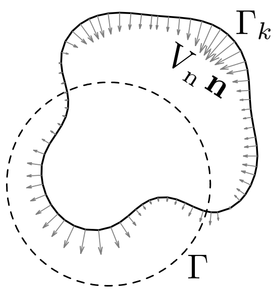

(b) Steps 5, 6: The routine `Compute_Conforming_Mesh` uses $\mathcal{U}_h$ to compute a mesh $\mathcal{T}_{h,k}$ conforming to D_k. The collection of edges in $\mathcal{T}_{h,k}$ defines the interpolant $\Gamma_{h,k}$ of Γ_k.


(c) Step 7: Contours of the finite element approximation of the membrane displacement $u_{h,k}$ over D_k computed with linear elements.

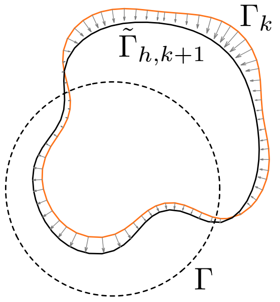

(d) Contours of the finite element approximation of the adjoint state $p_{h,k}$ over D_k computed with linear elements.



(e) Local normals to the free boundary iterate Γ_k at the nodes of $\Gamma_{h,k}$.



(f) Step 9: The iterate Γ_k is evolved along the local normal direction with the velocity V_n computed in step 9. The exact solution Γ is shown in dashed lines for reference.



(g) Step 18: Algorithm 2 advances the nodes of $\Gamma_{h,k}$ to $\tilde{\Gamma}_{h,k+1}$ over a time step Δt_{k+1} with the velocity $\mathrm{V}_n\mathbf{n}$.

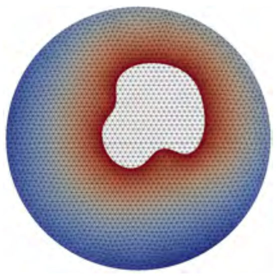

(h) Algorithm 2 computes the implicit function ϕ_{k+1} defining the next free boundary iterate Γ_{k+1} through a projection operation on $\tilde{\Gamma}_{h,k+1}$.

Figure 3.1: An illustrative description of the steps in Algorithm 1 during a generic iteration. Captions provided under each image explain the corresponding step of the algorithm. We have omitted the legends for the contour plots in (a), (c), (d) and (h) because they are irrelevant for the purpose of the explanations.

3.1 The algorithm

Algorithm 1 provides a step-by-step description of the gradient descent scheme for computing the free boundary in an obstacle problem. It is accompanied by a graphical illustration of the steps involved in Figure 3.1. The images shown in the figure are snapshots from a numerical experiment simulating the contact of a circular elastic membrane with a spherical obstacle. They serve as a visual aid for the following explanations.

The k^{th} iteration. The k^{th} iteration of the algorithm starts by identifying the location of the free boundary iterate Γ_k as the zero level set of the implicit function ϕ_k. While the initial guess ϕ_0 is user-defined, computing the sequence of functions $\phi_1, \phi_2, \ldots$ corresponding to subsequent time instants is the purpose of the algorithm. The sign convention adopted for the implicit function is such that the zero sublevel set included in Ω defines the noncoincidence set D_k, see step 4. Hence at each point along Γ_k, the gradient of ϕ_k points away from D_k.

To compute the membrane deflection u_k and its adjoint state p_k corresponding to the free boundary location Γ_k using a finite element method, it is necessary to mesh the domain D_k. This is accomplished using the universal mesh $\mathcal{U}_h$ over Ω and following the meshing algorithm discussed in chapter 4 and outlined in Algorithm 3. For now, it suffices to note that the routine `Compute_Conforming_Mesh` returns a mesh of triangles $\mathcal{T}_{h,k}$ that conforms to D_k in step 5. In particular, each connected component of Γ_k is interpolated by edges of the mesh $\mathcal{T}_{h,k}$ as illustrated in Figure 3.1b. We interpret $\Gamma_{h,k}$ to be a discretized version of Γ_k, since the former is composed of edges in $\mathcal{T}_{h,k}$ that interpolate Γ_k.

The (strong form) of the equations satisfied by the u_k and p_k are given in step 7. Notice that the Dirichlet boundary conditions along Γ_k is the height function of the obstacle. Computing the finite element approximations $(u_{h,k}, p_{h,k})$ of (u_k, p_k) are straightforward calculations. In all our numerical experiments, we adopt piecewise linear elements for both fields. There is, however, no inherent limitation imposed by the algorithm on adopting elements of higher order. Since the fluxes along $\Gamma_{h,k}$ of the computed finite element solutions are required to estimate the descent direction for the free boundary in subsequent steps, we post-process the computed fluxes $\partial u_{h,k}/\partial \mathbf{n}$ and $\partial p_{h,k}/\partial \mathbf{n}$ in step 8 following the well known technique of [52] that is discussed in section 3.4. The evolution velocity for Γ_k is computed in step 9. The rationale behind its definition is to ensure a monotonic decrease in the value of J_τ with each iteration. That the chosen velocity accomplishes this goal is justified by evaluating the sensitivity of J_τ, see section 3.3 and A.

We now arrive at the stage of the algorithm that evolves the free boundary to the next iterate by updating the implicit function ϕ_k to ϕ_{k+1} with the velocity along Γ_k prescribed to be V_n along the direction of the local normal. This is accomplished by the routine `Evolve_Free_Boundary` that is outlined in Algorithm 2 and discussed in section 3.2. The idea behind the update consists in evolving the nodes of $\Gamma_{h,k}$ along the local normal to Γ_k with the computed velocity over the given time step Δt_{k+1}, and then determining ϕ_{k+1} through a projection operation.

Time step and backtracking. The time step Δt_{k+1} used for advancing the boundary is computed based on an upper bound h_Γ for the distance by which the free boundary can be perturbed during the iteration. Specifically in step 10, we compute $\mathrm{V}_{\max}$ as the (maximum) norm of the nodal velocities along $\Gamma_{h,k}$. With the intention of limiting the perturbation of Γ_k during the k^{th} iteration to h_Γ, we set the time step for the iteration to be $\Delta t_k = h_\Gamma/\mathrm{V}_{\max}$ and advance the implicit function ϕ_k to ϕ_{k+1} with the routine `Evolve_Free_Boundary`. In particular, Δt_{k+1} defined by step 18 in this way is iteration-dependent. We highlight that such a choice is preferable to adopting a constant time step— the latter drastically slows down the convergence of free boundary iterates since the velocity of the free boundary progressively reduces as it approaches the exact solution. This observation is of course typical of gradient descent algorithms in general, and greedy time stepping choices coupled with backtracking techniques of the kind that we adopt in Algorithm 1 is a common way to overcome this bottleneck [53].

It is possible that the time step used to advance the free boundary iterate Γ_{k-1} to Γ_k is too large. To detect if this is indeed the case, we compare the values of $\mathrm{J}_\tau(\Gamma_k)$ and $\mathrm{J}_\tau(\Gamma_{k-1})$ in step 14. If $\mathrm{J}_\tau(\Gamma_k) < \mathrm{J}_\tau(\Gamma_{k-1})$ as desired, then we proceed to the next iteration. Otherwise, the estimated time step is deemed to be too large, i.e., the computed velocity is not a descent direction over the entire interval Δt_k. Hence we reverse course by reducing $h_\Gamma \to h_\Gamma/2$ and resetting ϕ_k to ϕ_{k-1} respectively and repeat the k^{th} iteration. In this way, backtracking in Algorithm 1 effectively reverts the current iteration to the previous one, while reducing the bound for free boundary perturbations. In our numerical experiments, we set $h_\Gamma = h$ and $h_{\text{tol}} = h/2^6$, so that free boundary perturbations are comparable to the mesh size during the initial few iterations.

Termination. Algorithm 1 terminates in one of two ways. The velocity-based criterion in step 12 terminates the algorithm if the velocity computed for evolving Γ_k is smaller than the prescribed tolerance V_{tol}. The specific norm of the velocity adopted for this criterion is generally insignificant and can be altered based on user preferences. The second criterion, namely $h_\Gamma > h_{\text{tol}}$, is distance-based and effectively terminates the algorithm after a fixed number of backtracking steps. The rationale behind it is that subsequent iterations of the algorithm would necessarily perturb the free boundary by distances smaller than h_{tol}. The simple bisection-based backtracking strategy employed in Algorithm 1 can be replaced by more general line search methods, cf. [53, Chapter 3]. Upon termination, the algorithm returns the pair $(\Gamma_{k_{\max}}, u_{h,k_{\max}})$ as the computed approximation to the solution of the obstacle problem.

3.2 Representation and update of free boundary iterates

We assume that the evolution of the free boundary admits an implicit representation as the zero level set of a function $\phi(x,t)$ defined on $\Omega \times [0,T]$, where $[0,T]$ represents the "time" interval of interest and $\phi(x,t=0) = \phi_0$ is a given initial guess. The sequence of implicit functions $\{\phi_k\}_k$ realized in Algorithm 1 is the result of specific discretizations adopted for ϕ in space and time. Permitting a minor abuse in notation, we do not distinguish between ϕ and its discretization in the following discussion.

Discretization in space: We require that at each instant $t > 0$, $\phi(x,t)$ belong to the finite dimensional space W_h that is spanned by the basis $\{\mathrm{N}_a\}_{a=1}^{\ell}$ consisting of local maximum entropy approximants defined using the collection of ℓ nodes in M_h provided as input in Algorithm 1. Hence we have

$$\phi(x,t) = \sum_{a=1}^{\ell} \phi^a(t)\, \mathrm{N}_a(x), \quad x \in \Omega \text{ and } t > 0, \tag{3.1}$$

where the coefficients $\{\phi^a(t)\}_a$ are understood to be the generalized coordinates or degrees of freedom of the free boundary at time t. Max-ent nodes in M_h are assumed to be distributed over a domain $\mathcal{D}$ that includes Ω, so that functions in W_h are well defined at each point in Ω. Specifically, $\mathcal{D} \supset \Omega$ automatically ensures that Ω is a subset of the convex hull of the nodes in M_h and hence that ϕ in eq. (3.1) is well defined over Ω. We omit details of the definition and evaluation of max-ent functions here and instead refer to [54, 55], which we closely follow in our implementation.

Discretization in time: A natural scheme for evolving the semi-discrete representation of ϕ in eq. (3.1) is using the advection equation

$$\frac{\partial \phi}{\partial t}(x,t) + \mathbf{V}(x,t) \cdot \nabla\phi(x,t) = 0 \text{ for } x \in \Omega \text{ and } t > 0, \tag{3.2}$$

where $\mathbf{V}(x,t)$ represents an extension of the velocity of the free boundary to the entire domain Ω. Equation (3.2) is indeed commonly used in the literature on level set methods— to evolve the geometry in structural shape optimization problems [37, 38], and more pertinently, in numerical methods for resolving variational inequalities [42] and obstacle problems [36]. Equation (3.2) can then be integrated in time, using the method of characteristics [56] for instance. In the context of our algorithm however, the choice of generalized coordinates $\{\phi^a(t)\}_a$ for ϕ makes such calculations quite tedious. Furthermore, we would like to avoid well known difficulties inherent in eq. (3.2)— extending the velocity defined over the free boundary to nearby neighborhoods or even over the entire domain Ω, and the need for periodic reinitialization of the level set function, to name just a couple.

Instead, we evolve ϕ in eq. (3.1) through a combination of collocation along the free boundary, a forward Euler scheme in time, and a projection operation onto W_h at each iteration. Algorithm 2 defining the routine Evolve_Free_Boundary provides details of the steps involved. Let us consider updating ϕ_k to ϕ_{k+1} with Algorithm 2. First, we exploit the fact that the mesh $\mathcal{T}_{h,k}$ conforms to Γ_k, i.e., that edges of $\mathcal{T}_{h,k}$ interpolate Γ_k. Then, we impose the computed velocity $\mathrm{V}_n\mathbf{n}$ of the free boundary at the nodes lying on Γ_k while assuming that the velocity remains a constant over the prescribed time interval. This effectively amounts to advecting the piecewise linear curve $\Gamma_{h,k}$ to $\tilde{\Gamma}_{h,k+1}$ using a forward Euler scheme, see steps 23 to 27 in Algorithm 2 and Figure 3.1g. Finally, a projection of the data $\mathcal{P}$ consisting of nodal locations and normals of $\tilde{\Gamma}_{h,k+1}$ onto W_h defines the implicit function ϕ_{k+1}. We label the projection operator mapping the oriented point cloud $\mathcal{P}$ (step 28) onto W_h as $\mathrm{SSD}_{\mathrm{W}_h}$, and discuss its details next. The label SSD is inspired by the nomenclature introduced in [57] for computing implicit representations for point cloud data in computer graphics applications.

Implicit functions from oriented point clouds: With the understanding that $\mathcal{P}$ is a collection of m pairs $\{(x_i, \mathbf{n}_i)\}_i$ of point locations and corresponding orientations realized at the end of step 33 in Algorithm 2, we define $\phi_{k+1} = \mathrm{SSD}_{\mathrm{W}_h}(\mathcal{P})$ to be the minimizer in W_h of the functional

$$\mathrm{E}_{\mathcal{P}}[v] \triangleq \frac{1}{2}\sum_{i=1}^{m} v^2(x_i) + \frac{1}{2}\sum_{i=1}^{m} \|\nabla v(x_i) - \mathbf{n}_i\|^2 + \frac{\alpha}{2}\int_{\mathcal{D}} \mathbf{H}(v) : \mathbf{H}(v)\, d\mathcal{D}, \tag{3.3}$$

where $\mathbf{H}(v)$ denotes the Hessian $\nabla\nabla v$ and $\alpha > 0$ is a user-defined parameter presumed to be small. The first and second terms in the definition of $\mathrm{E}_{\mathcal{P}}$ in eq. (3.3) constrain the zero level set of ϕ_{k+1} to lie close to the given collection of points $\{x_i\}_i$ and the gradients $\{\nabla\phi_{k+1}(x_i)\}_i$ to approximate the given orientations $\{\mathbf{n}_i\}_i$. The third term, which can be interpreted as a regularization, suppresses the appearance of large curvatures in ϕ_{k+1} and more crucially, renders the functional coercive. Hence we seek

$$\phi_{k+1} = \mathrm{SSD}_{\mathrm{W}_h}(\mathcal{P}) \triangleq \arg\min_{v\in \mathrm{W}_h} \mathrm{E}_{\mathcal{P}}[v]. \tag{3.4}$$

Noticing that $\mathrm{E}_{\mathcal{P}}$ is a quadratic functional, it is straightforward to demonstrate that the problem in eq. (3.4) is well defined for any choice of $\alpha > 0$ and hence, that ϕ_{k+1} exists and is unique.

Steps 37 to 39 in Algorithm 2 define the linear system of equations to be resolved for computing the coefficients of ϕ_{k+1} in the basis $\{\mathrm{N}_a\}_a$ of W_h. Therein, we evaluate integrals over $\mathcal{D}$ using standard quadrature rules and a triangular mesh over $\mathcal{D}$. Note that unlike D_k, the domain $\mathcal{D}$ is fixed and hence needs to be meshed just once. In fact, since the contribution $\hat{\mathbf{K}}_{\mathrm{ab}} = \int_{\mathcal{D}} \mathbf{H}(\mathrm{N}_a) : \mathbf{H}(\mathrm{N}_b)\, d\mathcal{D}$ to the stiffness matrix $\mathbf{K}$ that depends on $\mathcal{D}$ in step 39 is independent of the data $\mathcal{P}$, the matrix $\hat{\mathbf{K}}$ needs to be evaluated just once at the beginning of a simulation and can be reused at each iteration when assembling $\mathbf{K}$.

3.3 Sensitivity of the functional J_τ and the descent direction

The idea behind the choice of the descent direction in Algorithm 1 stems from the shape derivative (i.e., the sensitivity) of the functional J_τ. The shape derivative of $\mathrm{J}_\tau(\gamma)$ at the free boundary location γ in the direction of the velocity field $\mathbf{V}$ is denoted by $d\mathrm{J}_\tau(\gamma;\mathbf{V})$ and defined to be the limit [58]

$$d\mathrm{J}_\tau(\gamma;\mathbf{V}) \triangleq \lim_{\varepsilon\to 0} \frac{\mathrm{J}_\tau(\gamma_\varepsilon) - \mathrm{J}_\tau(\gamma)}{\varepsilon}, \tag{3.5}$$

where $\mathbf{V} : \Omega \to \mathbb{R}^2$ is a sufficiently smooth vector field that vanishes on $\partial\Omega$ and $\gamma_\varepsilon \triangleq \{x+\varepsilon\,\mathbf{V}(x) \,:\, x \in \gamma\}$. Underlying eq. (3.5) is the fact that the limit exists in the first place, i.e., that J_τ is shape differentiable

[32, Theorem 6.2]. Following the detailed calculations provided in A, we find the shape derivative of J_τ to be

$$d\mathrm{J}_\tau(\gamma;\mathbf{V}) = \int_\gamma \left(\frac{\partial u_\gamma}{\partial \mathbf{n}} - \frac{\partial \psi}{\partial \mathbf{n}}\right)\left(\frac{1}{2}\left(\frac{\partial u_\gamma}{\partial \mathbf{n}} + \frac{\partial \psi}{\partial \mathbf{n}}\right) - \tau\frac{\partial p_\gamma}{\partial \mathbf{n}} + (\tau-1)\frac{\partial u_\gamma}{\partial \mathbf{n}}\right)(\mathbf{V}\cdot\mathbf{n})\, d\gamma, \tag{3.6}$$

where the state u_γ and the adjoint p_γ are solutions of

$$\begin{cases} -\Delta u_\gamma = f & \text{on} \quad \mathrm{D}_\gamma, \\ u_\gamma = 0 & \text{on} \quad \partial\Omega, \\ u_\gamma = \psi & \text{on} \quad \gamma \end{cases} \qquad \text{and} \qquad \begin{cases} -\Delta p_\gamma = 0 & \text{on} \quad \mathrm{D}_\gamma, \\ p_\gamma = 0 & \text{on} \quad \partial\Omega, \\ p_\gamma = \psi & \text{on} \quad \gamma. \end{cases} \tag{3.7}$$

We highlight a few of aspects of eq. (3.6). First, it is not surprising that $d\mathrm{J}_\tau(\gamma;\mathbf{V})$ is an integral over just the free boundary. This is to be expected from the Hadamard structure theorem [58], which assures that the shape derivative admits a structure of the form $\langle G, (\mathbf{V}\cdot\mathbf{n})\rangle_\gamma$ for some kernel G and duality pair $\langle\cdot,\cdot\rangle_\gamma$ on γ. In our case, $\langle\cdot,\cdot\rangle_\gamma$ is the inner product in $\mathrm{L}^2(\gamma)$ and the kernel G is the term multiplying $(\mathbf{V}\cdot\mathbf{n})$ in eq. (3.6). Second, eq. (3.6) reveals as predicted by the structure theorem, that the shape derivative depends only on the component of the velocity $\mathbf{V}$ normal to γ. Intuitively, the component of $\mathbf{V}$ tangential to γ does not contribute to changing the shape of the free boundary and therefore leaves the functional J_τ unaffected. Finally, the mismatch in fluxes along the free boundary $(\partial u_\gamma/\partial\mathbf{n} - \partial\psi/\partial\mathbf{n})$ that is omitted in the problem for u_γ in eq. (3.7), appears naturally in the shape derivative. *This guarantees that the exact solution for the free boundary is necessarily a stationary point of* J_τ.

With the intention of rendering the shape derivative to be negative, we set

$$\mathbf{V} = -\left(\frac{\partial u_\gamma}{\partial \mathbf{n}} - \frac{\partial \psi}{\partial \mathbf{n}}\right)\left(\frac{1}{2}\left(\frac{\partial u_\gamma}{\partial \mathbf{n}} + \frac{\partial \psi}{\partial \mathbf{n}}\right) - \tau\frac{\partial p_\gamma}{\partial \mathbf{n}} + (\tau-1)\frac{\partial u_\gamma}{\partial \mathbf{n}}\right)\mathbf{n}, \tag{3.8}$$

so that

$$d\mathrm{J}_\tau(\gamma;\mathbf{V}) = -\int_\gamma \left(\frac{\partial u_\gamma}{\partial \mathbf{n}} - \frac{\partial \psi}{\partial \mathbf{n}}\right)^2\left(\frac{1}{2}\left(\frac{\partial u_\gamma}{\partial \mathbf{n}} + \frac{\partial \psi}{\partial \mathbf{n}}\right) - \tau\frac{\partial p_\gamma}{\partial \mathbf{n}} + (\tau-1)\frac{\partial u_\gamma}{\partial \mathbf{n}}\right)^2 d\gamma. \tag{3.9}$$

Notice that the integrand in eq. (3.9) is strictly nonnegative and therefore, $d\mathrm{J}_\tau(\gamma;\mathbf{V}) \leq 0$ as desired. Consequently, $\mathrm{J}_\tau(\gamma_{\Delta t}) \leq \mathrm{J}_\tau(\gamma)$ for $\Delta t > 0$ that is sufficiently small. In fact, we are generally assured of a strict descent for J_τ with eq. (3.8) because a vanishing shape derivative in eq. (3.9) implies that $\mathbf{V}$ vanishes on γ, which corresponds to γ being a stationary point of J_τ and hence to a converged solution in the algorithm.

The case $\tau = 0$: Let us revisit the functional J_0 introduced in chapter 2. Setting $\tau = 0$ in eq. (3.6), we find that that the shape derivative of J_0 is

$$d\mathrm{J}_0(\gamma;\mathbf{V}) = -\int_\gamma \left(\frac{\partial u_\gamma}{\partial \mathbf{n}} - \frac{\partial \psi}{\partial \mathbf{n}}\right)^2 (\mathbf{V}\cdot\mathbf{n})\, d\gamma. \tag{3.10}$$

It is clear from eq. (3.10) that J_0 has a stationary point at the exact free boundary where $\partial u_\gamma/\partial\mathbf{n} = \partial\psi/\partial\mathbf{n}$. However, the choice $\mathrm{V}_n = (\partial u_\gamma/\partial\mathbf{n} - \partial\psi/\partial\mathbf{n})^2$ as the velocity of the free boundary is futile in practice because even though it vanishes at the exact solution, it is always positive away from it and therefore results in a continually shrinking coincidence set in numerical simulations.

The case $f = 0$: Another interesting scenario that the shape derivative in eq. (3.6) helps to inspect is the case when $f = 0$. Noting from eq. (3.7) that the state and the adjoint solutions coincide over D_γ when $f = 0$, eq. (3.6) yields

$$d\mathrm{J}_\tau^{f=0}(\gamma;\mathbf{V}) = -\int_\gamma \left(\frac{\partial u_\gamma}{\partial \mathbf{n}} - \frac{\partial \psi}{\partial \mathbf{n}}\right)^2 (\mathbf{V}\cdot\mathbf{n})\, d\gamma \tag{3.11}$$

which is identical to eq. (3.10) and independent of the value of τ. For the same reasons mentioned previously, setting the free boundary velocity V_n to $(\partial u_\gamma/\partial\mathbf{n} - \partial\psi/\partial\mathbf{n})^2$ causes the coincidence set to shrink at all times. We argue that such a scenario is in fact not a discrepancy. Recall from eq. (2.11a) that we require $-\Delta\psi + (2\tau-1)f \geq 0$ on Ω for J_τ to have a minimum at the free boundary. In the present context with $f = 0$, this condition reduces to the requirement that $\Delta\psi \leq 0$, implying that ψ is super harmonic on Ω. By virtue of the maximum principle [59], we infer that if ψ has a minimum in Ω, then ψ is a constant. Then the assumption that $\psi > u$ on $\partial\Omega$ implies that the membrane does not, in fact, make contact with the obstacle. In such a case, the coincidence set should indeed be empty. On the other hand, if ψ does not have a minimum in Ω, it is intuitively clear that the coincidence set is again empty.

3.4 Flux recovery along the free boundary

From eq. (3.8), we infer that the accuracy of the descent direction V_n in Algorithm 1 depends directly on the approximation of the boundary fluxes $\partial u_k/\partial\mathbf{n}$ and $\partial p_k/\partial\mathbf{n}$, where k denotes the iteration count. It is also well known that direct approximation of these fluxes along $\Gamma_{h,k}$, say as $\nabla u_{h,k}\cdot\mathbf{n}$ and $\nabla p_{h,k}\cdot\mathbf{n}$, is ill advised— the rate of convergence of fluxes is notably inferior to that of the primal variables themselves. Practical issues invariably arise from oscillatory behavior of solution derivatives, especially with linear finite element approximations.

While it is certainly possible to use sophisticated flux recovery strategies, we adopt the simple technique introduced in [52]. Hence the post-processed fluxes at node i of the mesh $\mathcal{T}_{h,k}$ lying on Γ_k are computed as

$$\partial_{\mathbf{n}}\hat{u}_{h,k}(x_i) = \frac{1}{h_i}\sum_{j=1}^{n} \mathrm{A}_{ij}\mathrm{U}_j - \mathrm{b}_i \qquad \text{and} \qquad \partial_{\mathbf{n}}\hat{p}_{h,k}(x_i) = \frac{1}{h_i}\sum_{j=1}^{n} \mathrm{A}_{ij}\mathrm{P}_j, \tag{3.12}$$

where $\mathbf{A}$ and $\mathbf{b}$ are the finite element matrix and force vector with components

$$\mathrm{A}_{ij} = \int_{\mathrm{D}_k} \nabla\chi_i \cdot \nabla\chi_j \, d\mathrm{D}_k \quad \text{and} \quad \mathrm{b}_i = \int_{\mathrm{D}_k} f\chi_i \, d\mathrm{D}_k,$$

$\{\chi_i\}_{i=1}^n$ are the piecewise linear shape functions defined over the mesh $\mathcal{T}_{h,k}$ and spanning the finite element space, $\mathbf{U}_{n\times 1}$ and $\mathbf{P}_{n\times 1}$ are solutions of the linear systems $\mathbf{AU} = \mathbf{b}$ and $\mathbf{AP} = 0$ respectively, while subject to the Dirichlet boundary conditions along $\Gamma_{h,k}$ and $\partial\Omega$ noted in eq. (3.7), and h_i is the average of the lengths of the edges of $\Gamma_{h,k}$ intersecting at the node x_i. With this notation at hand, we note that the finite element solutions computed in step 7 of Algorithm 1 are simply

$$u_{h,k} = \sum_{a=1}^{n} \mathrm{U}_a\chi_a \quad \text{and} \quad p_{h,k} = \sum_{a=1}^{n} \mathrm{P}_a\chi_a.$$

In particular, post-processing the fluxes along $\Gamma_{h,k}$ in eq. (3.12) only requires reassembling components of the residual vectors $\mathbf{r}_u = \mathbf{AU} - \mathbf{F}$ and $\mathbf{r}_p = \mathbf{AP}$ corresponding to indices of nodes lying on Γ_k, which are ignored while imposing Dirichlet boundary conditions. We refer to [52, 60] for discussions on interpreting eq. (3.12) as a projection of the fluxes $\nabla u_{h,k} \cdot \mathbf{n}$ and $\nabla p_{h,k} \cdot \mathbf{n}$ on the finite element space while employing a lumped mass matrix, and to [61] for results on the super convergent behavior of fluxes computed in this way.

3.5 A few remarks

We conclude this chapter with a few remarks that were deferred until details of Algorithm 1 were fully discussed.

1. The implicit representation for the free boundary that we have adopted is especially convenient when the free boundary can undergo large shape changes during the course of a simulation. The numerical experiments in chapter 5 provide categorical evidence of this fact. In contrast, choosing a parametric representation (e.g., splines) quickly results in a nonuniform distribution of tracking points when the motion of the free boundary is nonuniform.

2. The appearance of point cloud representations for free boundaries in Algorithm 2 is an artifact of the scheme we have chosen to update the implicit functions $\{\phi_k\}_k$. Algorithm 1 always maintains a smooth representation for free boundary iterates.

3. The rationale behind choosing local maximum entropy approximants for computing implicit functions is their smoothness. Hence at each iteration k, ϕ_k is a smooth function and consequently, its zero level set Γ_k is also a smooth curve provided that $\nabla\phi_k$ is nonsingular along Γ_k. Maintaining a C^2-regular free boundary Γ_k is crucial for the robustness of the triangulation algorithm with universal meshes discussed in chapter 4. Additionally, maintaining a differentiable representation for the free boundary ensures unambiguous definitions of local normals, and hence of the direction of the evolution velocities. It is also possible to adopt, for instance, spaces spanned by spline functions to represent level set functions [62]. We favor max-ent functions for their meshfree-nature, which makes them convenient for the purpose of adaptivity.

4. For a given evolution $\{\Gamma_k\}_k$ of the free boundary, the choice of the corresponding implicit functions $\{\phi_k\}_k$ is not unique. The function $\phi_k \in \mathrm{W}_h$ is but one representation for Γ_k that is identified using the operator $\mathrm{SSD}_{\mathrm{W}_h}$.

5. We observe in our numerical experiments that computing implicit functions as projections of oriented point clouds with the operator $\mathrm{SSD}_{\mathrm{W}_h}$ suppresses the possibility of topological changes in the free boundary. Specifically, this happens because of the opposing orientations of its connected components. For this reason, we assume in all our experiments that the initial guess Γ_0 has the same topology as the desired solution. Nevertheless, free boundaries with multiple connected components can certainly be computed. It is worth noting in this context that Algorithm 1 provides complete control over prescribing the topology of the region of contact. Such control is lost when computing the free boundary a posteriori from membrane deflections.

6. An important practical consideration in shape optimization problems in general, and our problem in particular, concerns controlling sizes of geometric features along the free boundary. Even if the exact solution is smooth, it is possible that the computed free boundary has small features that vary over length scales comparable to the mesh size h of $\mathcal{U}_h$. In design optimization problems for example, it is common to add perimeter-based constraints to suppress oscillatory behavior of the free boundary. Here we adopt a simple rule of thumb wherein the spacing between nodes in M_h is set to be larger than $2h$, so that the projection operation $\mathrm{SSD}_{\mathrm{W}_h}(\mathcal{P})$ suppresses small scale features present in the point cloud $\mathcal{P}$ and in turn in the free boundary. This strategy works well in practice and is used in all the numerical examples presented in chapter 5.

7. In Algorithm 1, we have used a finite element method to essentially map Dirichlet boundary conditions in eq. (2.1c) to the boundary fluxes of the state and its adjoint. Such a calculation can also be performed using a boundary element method as well. Our choice of the finite element method is motivated by the goal of adopting techniques proposed here to more general classes of

shape optimization problems. Moreover, the proposed algorithm seems suitable for implementation within commercial and open source finite element codes.

8. A notable feature of Algorithm 1 is that it restricts calculations of the state and its adjoint to just the noncoincidence sets. This has the obvious benefit of avoiding computing solutions over the larger domain Ω, considering the fact that we know $u_k = \psi$ over the coincidence set at each iteration k. Furthermore, we avoid issues of suboptimal accuracy of finite element solutions stemming from the lack of regularity of u_k along the free boundary.

9. A minor detail in Algorithm 2 involves estimating normals to $\tilde{\Gamma}_{h,k+1}$ at its nodes. We exploit the fact that $\Gamma_{h,k}$ interpolates the zero level set of ϕ_k to evaluate nodal normals to $\Gamma_{h,k}$ in step 26. In contrast, $\tilde{\Gamma}_{h,k+1}$ is a piecewise linear curve and there is necessarily some ambiguity in defining its nodal normals in step 31. In our implementation, we set the normal at a node of $\tilde{\Gamma}_{h,k+1}$ by normalizing averages of edgewise normals.

10. It may seem that because the mesh $\mathcal{T}_{h,k}$ is updated after each iteration k, so must the finite element data structures required to compute $u_{h,k}$ and $p_{h,k}$. This would indeed be the case if, for instance, the domain D_k is meshed with a Delaunay triangulation algorithm [63]. However, Algorithm 3 discussed in the following chapter ensures that the connectivity of the mesh $\mathcal{T}_{h,k}$ is a subset of the connectivity of the universal mesh $\mathcal{U}_h$. In particular, the routine `Compute_Conforming_Mesh` does not introduce any new vertices when computing $\mathcal{T}_{h,k}$ from $\mathcal{U}_h$; it merely perturbs the locations of a few vertices in $\mathcal{U}_h$. For this reason, it is possible to use data structures whose sizes and sparsity are based solely on the universal mesh $\mathcal{U}_h$. While the stiffness matrices and force vectors required to compute $u_{h,k}$ and $p_{h,k}$ have to be reassembled at each iteration k, they have to be allocated just once at the beginning of a simulation.

11. The parameter τ is practically invisible in Algorithm 1 and appears only while evaluating the free boundary velocities. Yet, it plays a crucial in identifying a descent direction for the shape functional.

12. A minor detail in Algorithm 1 lies in evaluating the shape functional J_τ in step 14. Computing $\mathrm{J}_\tau(\Gamma_k)$ requires evaluating an integral over the coincidence set C_k, which we do by triangulating C_k and resorting to numerical quadrature. The routine `Compute_Conforming_Mesh` and the same universal mesh $\mathcal{U}_h$ used to triangulate D_k serves to mesh C_k as well. In fact, in our implementation, we simply switch the sign of the implicit function ϕ_k when invoking the routine `Compute_Conforming_Mesh` to triangulate C_k instead of D_k.

Algorithm 1: Gradient descent algorithm to compute the free boundary in an obstacle problem

```
Input:
$(\Omega, f, \psi)$ : Problem data                                        ▷ See eq. (2.1).
  $\phi_0$ : Implicit function defining the initial guess for the free boundary
  $\mathcal{U}_h$ : Universal mesh over $\Omega$
  $\mathrm{M}_h$ : Nodes defining local maximum entropy approximants over $\mathcal{D} \supset \Omega$
  $\mathrm{V}_{\mathrm{tol}}$ : Velocity-based tolerance for identifying convergence of the free boundary
  $h_\Gamma$ : Upper bound for free boundary perturbations over an iteration, e.g.: $h_\Gamma = h$.
  $h_{\mathrm{tol}}$ : Distance-based tolerance for identifying convergence of the free boundary, e.g., $h_{\mathrm{tol}} = h_\Gamma/2^6$
1  Set $k \leftarrow 0$
2  while $h_\Gamma > h_{\mathrm{tol}}$        ▷ Distance-based criterion for convergence of the free boundary
3   do
4     Set $\mathrm{D}_k = \{x : \phi_k(x) < 0\} \cap \Omega$ and $\Gamma_k = \{x : \phi_k = 0\}$   ▷ Identify the noncoincidence set
       and the free boundary iterate
5     $\mathcal{T}_{h,k} \leftarrow$ Compute_Conforming_Mesh($\phi_k$,$\mathcal{U}_h$)   ▷ $\mathcal{T}_{h,k}$ is a triangle mesh conforming to $\mathrm{D}_k$,
       see chapter 4
6     $\Gamma_{h,k} \leftarrow$ union of edges in $\mathcal{T}_{h,k}$ with both nodes on $\Gamma_k$   ▷ $\Gamma_{h,k}$ is a piecewise linear
       interpolant of $\Gamma_k$
7     Use the mesh $\mathcal{T}_{h,k}$ to compute finite element approximations of the state $u_{h,k}$ and its adjoint $p_{h,k}$: :
       $$\begin{cases} -\Delta u_k = f & \text{on} \quad \mathrm{D}_k, \\ u_k = 0 & \text{on} \quad \partial\Omega, \\ u_k = \psi & \text{on} \quad \Gamma_k \end{cases} \qquad \text{and} \qquad \begin{cases} -\Delta p_k = 0 & \text{on} \quad \mathrm{D}_k, \\ p_k = 0 & \text{on} \quad \partial\Omega, \\ p_k = \psi & \text{on} \quad \Gamma_k \end{cases}$$
8     $\partial_{\mathbf{n}}\hat{u}_{h,k}, \partial_{\mathbf{n}}\hat{p}_{h,k} \leftarrow$ post-processed fluxes $\partial u_{h,k}/\partial\mathbf{n}, \partial p_{h,k}/\partial\mathbf{n}$ along $\Gamma_{h,k}$   ▷ See section 3.4
9     $\mathrm{V}_n \leftarrow -\left(\partial_{\mathbf{n}}\hat{u}_{h,k} - \partial_{\mathbf{n}}\psi\right)\left(\frac{1}{2}\left(\partial_{\mathbf{n}}\hat{u}_{h,k} + \partial_{\mathbf{n}}\psi\right) - \tau\,\partial_{\mathbf{n}}\hat{p}_{h,k} + (\tau-1)\,\partial_{\mathbf{n}}\hat{u}_{h,k}\right)$   ▷ Evolution velocity
       along $\Gamma_{h,k}$, see section 3.3
10    $\mathrm{V}_{\max} \leftarrow \max_{\text{node } x_a \in \Gamma_{h,k}} |\mathrm{V}_n(x_a)|$   ▷ Any norm of the vector of nodal velocities.
11    if $\mathrm{V}_{\max} < \mathrm{V}_{\mathrm{tol}}$ then
12       return $(\Gamma_k, u_{h,k})$                     ▷ Velocity-based convergence criterion
13    end
14    Compute $\mathrm{J}_\tau(\Gamma_k) = \int_{\mathrm{D}_{h,k}} \left(\frac{1}{2}\nabla u_{h,k} \cdot \nabla u_{h,k} + (\tau-1) f\, u_{h,k}\right) + \int_{\mathrm{C}_k} \left(\frac{1}{2}\nabla\psi \cdot \nabla\psi + (\tau-1) f\,\psi\right)$
15    if $\mathrm{J}_\tau(\Gamma_k) > \mathrm{J}_\tau(\Gamma_{k-1})$ then
          ▷ Time step $\Delta t_k$ was too large. Backtrack and repeat the $k^{\mathrm{th}}$ iteration with a
         reduced time step.
       $h_\Gamma \leftarrow h_\Gamma/2$
16     Set $\phi_k = \phi_{k-1}$
17    else
18     $\Delta t_{k+1} \leftarrow h_\Gamma/\mathrm{V}_{\max}$               ▷ Greedy time step for this iteration
        ▷ Evolve the implicit function $\phi_k$ to $\phi_{k+1}$
       $\phi_{k+1} \leftarrow$ Evolve_Free_Boundary($\phi_k, \Gamma_{h,k}, \mathrm{V}_n, \Delta t_{k+1}, \mathrm{M}_h$)
        ▷ Proceed to the next iteration
       $k \leftarrow k + 1$
19    end
20 end
21 return $(\Gamma_{k_{\max}}, u_{h,k_{\max}})$
```

Algorithm 2: Evolve_Free_Boundary: Evolve the free boundary over a short time interval with a prescribed normal velocity.

Input:
ϕ_k : Implicit function representing the free boundary iterate Γ_k
$\Gamma_{h,k}$: Piecewise linear interpolant of Γ_k
V_n : Evolution velocity along the normal to Γ_k at the nodes of $\Gamma_{h,k}$
Δt_{k+1} : Time step
M_h : Nodes defining local maximum entropy approximants

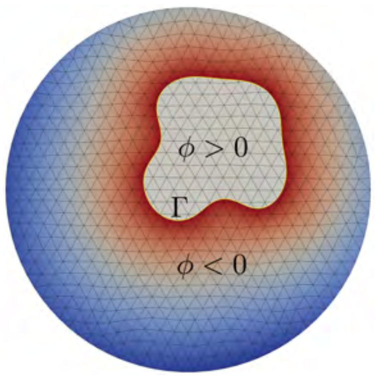
```
22 $m \leftarrow$ number of nodes in $\Gamma_{h,k}$
   ▷ Advance $\Gamma_{h,k}$ to the piecewise linear curve $\tilde{\Gamma}_{h,k+1}$
23 Initialize $\tilde{\Gamma}_{h,k+1} \leftarrow \Gamma_{h,k}$
24 foreach node $a$ of $\tilde{\Gamma}_{h,k+1}$ do
25     $x_a \leftarrow$ coordinates of node $a$
26     Update: $x_a \leftarrow x_a + \Delta t_{k+1}\, \mathrm{V}_n(x_a)\, \nabla\phi_k(x_a)/\|\nabla\phi_k(x_a)\|$    ▷ No sum on $a$ implied. $x_a$ is
        advected along the normal to $\Gamma_k$
27 end
28 Initialize $\mathcal{P} \leftarrow \emptyset$
29 foreach node $a$ of $\tilde{\Gamma}_{h,k+1}$ do
30     $x_a \leftarrow$ coordinates of node $a$
31     $\mathbf{n}_a \leftarrow$ normal to $\tilde{\Gamma}_{h,k+1}$ at $x_a$
32     Append $(x_a, \mathbf{n}_a)$ to $\mathcal{P}$
33 end
   ▷ Compute $\phi_{k+1} = \mathrm{SSD}_{\mathrm{W}_h}(\mathcal{P})$, see section 3.2
34 $\{\mathrm{N}_a\}_{a=1}^{\ell} \leftarrow$ local max-ent basis functions    ▷ Basis functions spanning $\mathrm{W}_h$
35 $\mathbf{K}_{\ell\times\ell}, \mathbf{F}_{\ell\times 1} \leftarrow$ stiffness matrix and force vector
36 for $a = 1$ to $\ell$ do
37     $\mathbf{F}_a \leftarrow \sum_{i=1}^{m} \mathbf{n}_i \cdot \nabla\mathrm{N}_a(x_i)$
38     for $b = 1$ to $\ell$ do
39         $\mathbf{K}_{ab} \leftarrow \sum_{i=1}^{m} (\mathrm{N}_a(x_i)\mathrm{N}_b(x_i) + \nabla\mathrm{N}_a(x_i)\cdot\nabla\mathrm{N}_b(x_i)) + \alpha \int_{\mathcal{D}} \mathbf{H}(\mathrm{N}_a) : \mathbf{H}(\mathrm{N}_b)\, d\mathcal{D}$
40     end
41 end
42 Resolve the linear system of $\ell$ equations $\mathbf{K}\,\boldsymbol{\phi} = \mathbf{F}$    ▷ $\mathbf{K}$ is a sparse and symmetric matrix
43 $\phi_{k+1} \leftarrow \sum_{a=1}^{\ell} \boldsymbol{\phi}_a \mathrm{N}_a$    ▷ $\phi_{\mathrm{k+1}} = \arg\min_{v\in\mathrm{W}_h} \mathrm{E}_{\mathcal{P}}[v]$
44 return $\phi_{k+1}$
```

Chapter 4

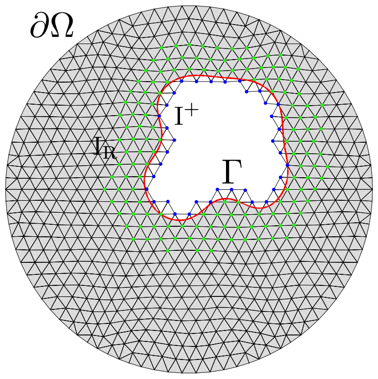
Discretizing evolving domains with universal meshes

Algorithm 3 outlines the routine `Compute_Conforming_Mesh` that computes triangulations conforming to C^2-regular domains immersed in a universal mesh $\mathcal{U}_h$. It is accompanied by supporting graphical illustrations in Figures 4.1 and 4.2. We invoke the routine in Algorithm 1 to triangulate the sequence of noncoincidence sets $\{\mathrm{D}_k\}_k$, as well as to triangulate coincidence sets $\{\mathrm{C}_k\}_k$ when evaluating the shape functional J_τ at each iteration.

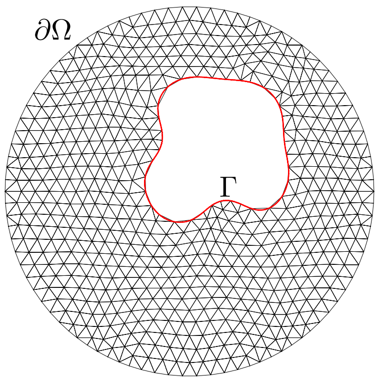
Related literature: Algorithm 3 is based on two key ideas— parameterizing immersed boundaries over a specific collection of edges in $\mathcal{U}_h$ with the closest point projection onto the immersed boundary, and an optimization based directional vertex relaxation (`dvr`) algorithm. Detailed discussions, examples and analysis of parameterizing immersed boundaries with the closest point projection map can be found in [64, 34]. The dvr algorithm is introduced and analyzed in [35], and efficient methods for implementing it are discussed in [65]. We mention that Algorithm 3 differs from the meshing algorithm described in [33] only in the choice of the vertex relaxation algorithm— in place of the `dvr` algorithm employed here, vertices are relaxed by explicitly constructing a continuous extension of the closest point projection map in [33]. A three-dimensional analog of Algorithm 3 for triangulating domains immersed in tetrahedral meshes is described and tested in [66]. Cognizant of the fact that details and analyses supporting Algorithm 3 are well documented in the literature, in the following, we only provide a terse description of the meshing algorithm with sufficient detail to facilitate a correct implementation. We mention conditions on the mesh size and on angles in the universal mesh, and on the regularity of the immersed domain assumed in the algorithm. We do not, however, delve into the rationale behind these assumptions and direct interested readers to the references mentioned above.



(a) Nonpositive contours of an implicit function ϕ superimposed on a universal mesh. The colored region is the set ω to be triangulated. The function ϕ is identical to that in Figure 3.1a.



(b) The mesh shown consists of triangles having at least one vertex where $\phi < 0$. Positive vertices (I^+) are highlighted in blue, while vertices to be relaxed (I_R) are highlighted in green.



(c) Projecting vertices in I^+ to their closest point in Γ while relaxing those in I_R with dvr transforms the mesh in (b) to a mesh conforming to ω.

Figure 4.1: An illustration of the steps in Algorithm 3. Additional details of nodal perturbations that map the nonconforming mesh in (b) to the conforming one in (c) are shown in Figure 4.2.

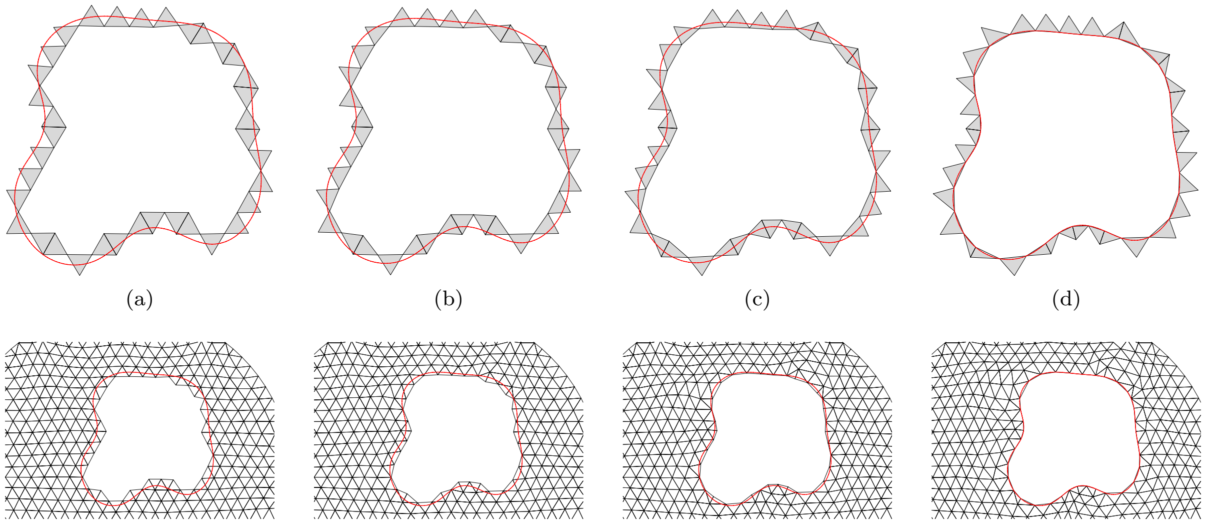


Figure 4.2: For the example in Figure 4.1, the first row of images visualize the projection of positive vertices towards the immersed boundary Γ over $N_P = 4$ passes. The second row of images show the relaxation the vertices in I_R computed after each projection pass.

Description of the meshing algorithm: The domain to be triangulated, say $\omega \subset \mathbb{R}^2$, is specified implicitly in Algorithm 3 as the zero sublevel set of a function ϕ. The boundary of ω is the zero level set of ϕ and is denoted by Γ. In the context of Algorithm 1, ϕ, ω, and Γ correspond to the iterates ϕ_k, D_k, and Γ_k, respectively. In addition to the universal mesh $\mathcal{U}_h$, the algorithm requires two integer-valued parameters $\mathrm{N_P}$ and $\mathrm{N_R}$. The values for these parameters that we have used in our numerical experiments are indicated in the algorithm and their purpose will be explained shortly.

The algorithm starts in step 46 by initializing a mesh $\mathcal{T}_h$ to be the collection of all triangles in $\mathcal{U}_h$ having $\phi \leq 0$ at one or more of its vertices. In steps 47 and 48, we identify two subsets of vertices in $\mathcal{T}_h$. The first set consists of vertices where $\phi \geq 0$ and is labeled as I^+, while the second set $\mathrm{I_R}$ consists of vertices in the vicinity of Γ having $\phi < 0$. In keeping with the terminology introduced for universal meshes in previous work, we call I^+ the collection of *positive vertices*. In subsequent steps, $\mathcal{T}_h$ will be transformed into a mesh that conforms to ω by mapping each vertex in I^+ to their respective closest points in Γ, while simultaneously relaxing vertices in $\mathrm{I_R}$ with the dvr algorithm. Figure 4.1 shows an illustrative example to this effect, where the function ϕ and the domain ω are identical to those appearing in Figure 3.1a. Figure 4.1a shows the nonpositive contours of ϕ superimposed on the universal mesh $\mathcal{U}_h$ over a circular domain. The submesh $\mathcal{T}_h$ of $\mathcal{U}_h$ consisting of triangles that have at least one vertex lying in ω are shown in Figure 4.1b, where the collections of vertices in I^+ and $\mathrm{I_R}$ are also highlighted. Algorithm 3 maps the mesh shown in Figure 4.1b to the conforming mesh in Figure 4.1c by projecting vertices in blue to their closest point in Γ and by relaxing vertices in green. In particular, the meshes appearing in Figures 4.1b and 4.1c differ only in the locations of their vertices in $\mathrm{I}^+ \cup \mathrm{I_R}$.

Vertices in I^+ are projected onto their respective closest point in Γ incrementally through the updates defined in step 51. At the end of $\mathrm{N_P}$ projection passes, each vertex in I^+ lies on Γ. These intermediate perturbations are visualized in Figures 4.2a to 4.2d. To accommodate perturbations of positive vertices towards Γ, we iteratively relax vertices in $\mathrm{I_R}$ along prescribed directions with the dvr algorithm in step 55. In particular, a vertex $i \in \mathrm{I_R}$ is perturbed along a prescribed direction $\mathbf{d}$ by a coordinate that maximizes the minimum among the qualities of the triangles in its 1-ring. Denoting the collection of triangles in the 1-ring of vertex i by $\{K_{i_j}\}_{j=1}^{n_i}$, and with the understanding that perturbing vertex i from its location x_i to $x_i + \lambda\,\mathbf{d}$ alters triangle K_{i_j} to $K_{i_j}^{\lambda}$, the coordinate λ^{opt} defining the update for i is therefore computed in step 55 as

$$\lambda^{\mathrm{opt}} \triangleq \arg\max_{\lambda \in \mathbb{R}} \min_{1 \leq j \leq n_i} Q(K_{i_j}^{\lambda}). \tag{4.1}$$

In all our simulations, we set Q to be the mean ratio triangle quality metric so that vertex relaxations seek to render triangles that are as close to equilateral as possible. Figures 4.2e to 4.2h visualize these relaxations realized after successive projection passes shown in the first row of images in Figure 4.2.

Remarks. The apparent simplicity of Algorithm 3 is deceptive. Numerous examples shown in

[64, 33] reveal that without restrictions on the mesh size h, on angles in $\mathcal{U}_h$, and on the regularity of Γ, the mesh $\mathcal{T}_h$ computed by the algorithm can have inverted, overlapping or degenerate elements. Analyzing projections of positive vertices onto Γ reveals two main restrictions on $\mathcal{U}_h$— (i) the mesh size h should be sufficiently small compared to the geometric features of Γ (e.g., curvatures and feature sizes), and (ii) certain interior angles of its triangles that are intersected by Γ should be strictly acute. The latter is referred to as the *acute conditioning angle* assumption in the algorithm. In practice, a simple way to satisfy this requirement is to ensure that all triangles in the vicinity of the immersed boundary are acute angled. In the context simulating problems with evolving boundaries, as is the case in Algorithm 1, the location of the immersed boundary changes with each solution iteration. Consequently, we ensure if possible, that all triangles in $\mathcal{U}_h$ are strictly acute angled, see section 5.1. The condition that Γ be a C^2-regular boundary stems from the choice of the closest point projection to map positive vertices onto Γ. It is for the purpose of maintaining a C^2-regular description for the free boundary that we choose max-ent basis functions to define the implicit representations $\{\phi_k\}_k$ in Algorithm 1. Precise statements of the assumptions mentioned above and of their consequences can be found in [34]

We refer to [35] for assumptions on the element quality metric required in dvr, and the guarantees on mesh improvement it provides. There, the rationale behind iteratively relaxing vertices is discussed at length. In general, choosing a larger number relaxation iterations ($\mathrm{N_R}$) improves the final mesh quality further. The reason for projecting positive vertices onto Γ over multiple steps (i.e., $\mathrm{N_P} > 1$) is to ensure that no element in $\mathcal{T}_h$ is inverted in the process, which is a prerequisite for dvr to be successful in improving element qualities. Implementing vertex updates by computing optimal perturbations in dvr requires resolving nonsmooth max-min problems of the form mentioned in eq. (4.1). To this end, we use the algorithms provided in [65].

We end this chapter mentioning that we have not attempted to describe the most general algorithm for triangulating domains immersed in universal meshes. For instance, the relaxation direction for vertices in dvr need not be restricted to the cardinal directions. Nor have we diligently reviewed results related to the robustness of the meshing algorithm. Instead, we have only described how evolving domains realized in Algorithm 1 are meshed with the same universal mesh. The references provided in this chapter document various details involved in Algorithm 3 that we have intentionally omitted here.

Algorithm 3: `Compute_Conforming_Mesh`: Triangulate smooth domains immersed in universal meshes

```
Input:
$\phi$ : Implicit function whose zero sublevel set is to be triangulated   ▷ Implicit representation of the $C^2$-regular domain
  $\mathcal{U}_h$ : Universal mesh   ▷ Sufficient refinement, acute conditioning angles
  $\mathrm{N_P}$ : Number of projection passes, e.g., $\mathrm{N_P} = 4$
  $\mathrm{N_R}$ : Number of relaxation passes, e.g., $\mathrm{N_R} = 10$
45 $\Gamma \leftarrow$ zero level set of $\phi$
46 Initialize $\mathcal{T}_h \leftarrow$ collection of triangles in $\mathcal{U}_h$ having $\phi < 0$ at at least one vertex
47 $\mathrm{I}^+ \leftarrow$ indices of vertices in $\mathcal{T}_h$ where $\phi \geq 0$   ▷ Positive vertices
48 $\mathrm{I_R} \leftarrow$ indices of vertices in $\mathcal{T}_h$ lying close to $\Gamma$ and where $\phi < 0$   ▷ Notice that $\mathrm{I}^+ \cap \mathrm{I_R} = \emptyset$
   ▷ Multi-pass boundary projection with interior relaxation
49 for $p = 1$ to $\mathrm{N_P}$ do
       ▷ Project vertices in $\mathrm{I}^+$ towards $\Gamma$, see [64, 34]
       $\lambda \leftarrow p/\mathrm{N_P}$
50     foreach $i \in \mathrm{I}^+$ do
51         $x_i \leftarrow \lambda\, \pi(x_i) + (1-\lambda)\, x_i$   ▷ $\pi$ is the closest point projection onto $\Gamma$.
52     end
       ▷ DVR: Iteratively relax vertices in $\mathrm{I_R}$ along cardinal directions, see [35, 65]
       for $r = 1$ to $\mathrm{N_R}$ do
53         Set $\mathbf{d} \leftarrow \mathbf{e}_x$ if $r$ is odd and $\mathbf{d} \leftarrow \mathbf{e}_y$ otherwise
54         foreach $i \in \mathrm{I_R}$ do
               ▷ Relax vertex $i$ along direction $\mathbf{d}$ to maximize the minimum among the qualities of triangles in its 1-ring
               $\{K_{i_j}\}_{j=1}^{n_i} \leftarrow$ triangles in the 1-ring of vertex $i$ in the mesh $\mathcal{T}_h$
55             $\lambda^{\mathrm{opt}} \leftarrow \arg\max_{\lambda \in \mathbb{R}} \min_{1 \leq j \leq n_i} Q(K_{i_j}^{\lambda})$   ▷ See the text for an explanation of the notation.
               ▷ $Q$ is the mean ratio triangle quality metric
56             $x_i \leftarrow$ Cartesian coordinates of vertex $i$
57             Update $x_i \leftarrow x_i + \lambda^{\mathrm{opt}}\, \mathbf{d}$
58         end
59     end
60 end
61 return $\mathcal{T}_h$   ▷ Conforming mesh over the zero sublevel set of $\phi$
```

Chapter 5

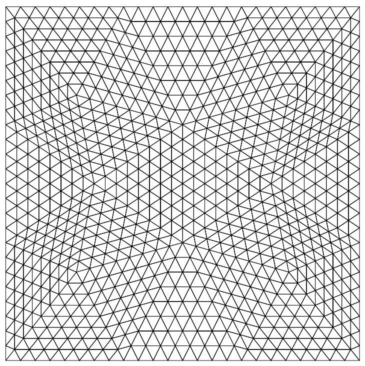
Numerical experiments

We devote this chapter to examining the performance of Algorithm 1 while paying special attention to demonstrating:

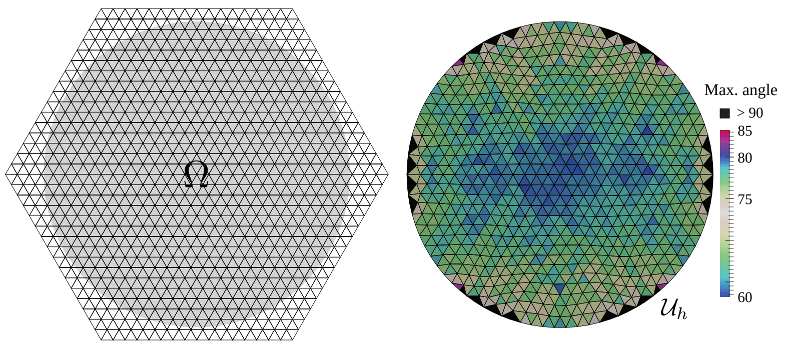
- the insensitivity of computed solutions to the choice of the initial guess Γ_0,
- the insensitivity of computed solutions to the choice of the parameter τ, provided that it satisfies $\tau \geq 1$ and eq. (2.11a). In particular, we reinforce the point that τ is *not* a penalty-type parameter. Algorithm 1 is applicable for *any* choice of $\tau \geq 1$ that satisfies eq. (2.11a).
- that the functional J_τ decays monotonically with increasing number of iterations of the algorithm,
- that the velocity of the free boundary decays to zero close to the correct solution, which in the context of the proposed algorithm is equivalent to a vanishing shape derivative,
- the convergence of the computed free boundary to the exact solution in cases when the latter is known (see sections 5.4 and 5.5),
- and to inspecting the magnitude of the error in the free boundary when compared to the mesh size.

5.1 Choice of universal meshes

First we discuss the choice of universal meshes in our examples. Since the evolution of the free boundary starting from an initial guess is not known a priori, we adopt a universal mesh $\mathcal{U}_h$ over Ω that is uniformly refined. Similarly, to guarantee that the acute conditioning angle assumption required in Algorithm 3 is satisfied at each iteration of Algorithm 1, we ensure that all triangles in $\mathcal{U}_h$ are strictly acute, except possibly in a small vicinity of $\partial\Omega$. The exception along $\partial\Omega$ is inconsequential in simulations because the



(a) Universal mesh for the example in section 5.5. All triangles are strictly acute.



(b) The examples in sections 5.3, 5.4 and 5.6 require a universal mesh over a circular domain. To this end, we immerse the circular domain in a background mesh of equilateral triangles and use Algorithm 3 to compute a mesh conforming to the circular geometry. As indicated on the right, all triangles in the mesh computed this way, besides a few along the boundary, are strictly acute angled.

Figure 5.1: Choice of universal meshes for the examples in chapter 5. Since the evolution of the free boundary is not known a priori, we choose meshes with uniform refinement. In order to satisfy the acute conditioning angle requirement in Algorithm 3, we ensure that all triangles in the mesh, except possibly along the boundary of the domain, are acute angled.

free boundary necessarily remains away from $\partial\Omega$. It is however useful in practice, especially when Ω is a curved domain, because constructing acute triangle meshes over such domains is a non trivial task.

Figure 5.1a shows the universal mesh over a square-shaped domain that is used for simulating the example in section 5.5. All angles in the mesh are smaller than 80°. In simulations requiring more mesh refinement, we subdivide each triangle into four self-similar ones so that the mesh size is halved with each refinement while the angle bounds remain unchanged. Figure 5.1b illustrates the procedure for constructing universal meshes over circular domains required in the examples in sections 5.3, 5.4 and 5.6. Therein, the idea is to consider a background mesh of equilateral triangles as a universal mesh for triangulating the circular domain. Since $\partial\Omega$ is fixed, the mesh $\mathcal{U}_h$ over Ω needs to be computed just once at the start of a simulation. We ensure that the angles in the mesh $\mathcal{U}_h$ computed this way are strictly acute, except possibly along the boundary $\partial\Omega$. For more refined universal meshes over Ω, we uniformly refine the background equilateral mesh and recompute the mesh $\mathcal{U}_h$ over Ω.

It is certainly possible to construct adaptively refined universal meshes, by tiling adaptively refined quadtrees with stencils of acute triangulations for instance [67, 33]. Such meshes are particularly beneficial when free boundary iterates are deemed to be close to a converged solution, so that mesh refinement can be restricted to a vicinity of the correct free boundary and tailored to match its geometric features. We have not considered such refinement in our experiments. Nevertheless, we highlight that the universal mesh in Algorithm 1 can be chosen in an iteration-dependent manner with no alterations in any of its steps. After all, the $(k+1)^{\text{th}}$ iteration depends on the k^{th} one only through the free boundary iterate— there is no requirement or restriction that the universal meshes over Ω at successive iterations be identical. However, we caution against frequent revisions in the choice of the universal mesh because of the computational overheads involved in reallocating mesh and finite element data structures.

5.2 Finite element approximation

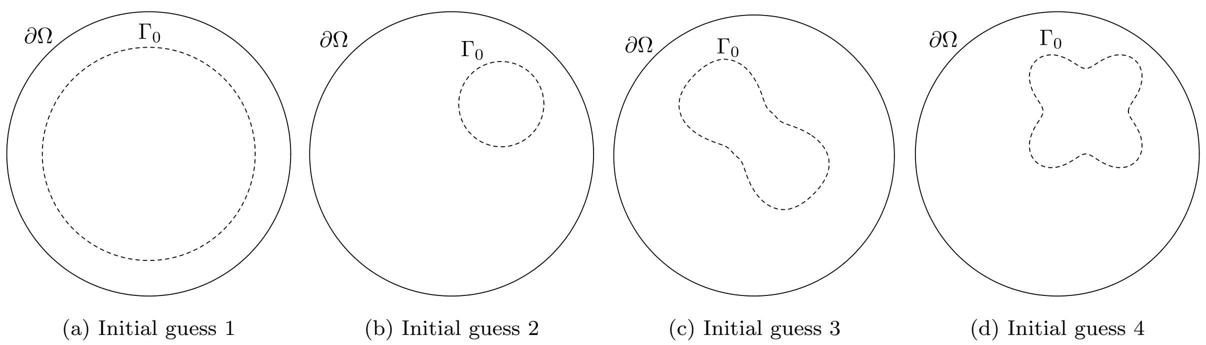
At the k^{th} iteration of Algorithm 1, we approximate the restriction of the state and adjoint fields over D_k in the finite element space $\mathrm{V}_{h,k}$ of continuous piecewise affine functions defined over the mesh $\mathcal{T}_{h,k}$, with Dirichlet boundary conditions imposed over Γ_k and $\partial\Omega$. Hence

$$u_{h,k} = \begin{cases} \arg\min_{v_h \in \mathrm{V}_{h,k}} \int_\Omega \left(\frac{1}{2}\nabla v_h \cdot \nabla v_h - f v_h\right) d\Omega \quad \text{over} \quad \mathrm{D}_k, \\ \psi \quad \text{over} \quad \mathrm{C}_k \end{cases} \tag{5.1a}$$

$$\text{and} \quad p_{h,k} = \arg\min_{q_h \in \mathrm{V}_{h,k}} \int_\Omega \frac{1}{2}\nabla q_h \cdot \nabla q_h \, d\Omega \quad \text{over} \quad \mathrm{D}_k. \tag{5.1b}$$

The choice of linear elements for approximating u_k and p_k is made solely for convenience. There is no inherent limitation in the algorithm to first order elements as we mentioned previously.

An important point we highlight here is that as the free boundary iterates evolve, the mesh $\mathcal{T}_{h,k}$ changes and so does the finite element space $\mathrm{V}_{h,k}$. However, because the connectivity of $\mathcal{T}_{h,k}$ is necessarily a subset of $\mathcal{U}_h$ irrespective of the iteration count k, it is possible to devise ways to avoid expensive reallocation of data structures at each iteration. Indeed, computing $u_{h,k}$ and $p_{h,k}$ can be interpreted as inverting a sub-block of a stiffness matrix assembled over $\mathcal{U}_h$. While the specific entries in these matrices need to be updated (reassembled) at each iteration because of vertex perturbations, their sparsity structure remains unchanged.



(a) Initial guess 1 (b) Initial guess 2 (c) Initial guess 3 (d) Initial guess 4

Figure 5.2: Initial guesses for the free boundary used in the examples in sections 5.3, 5.4 and 5.6.

5.3 Circular membrane in contact with a flat obstacle

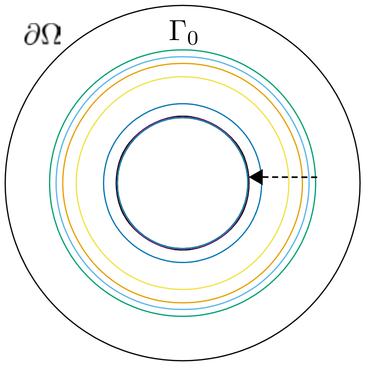
As a first example, we revisit the axisymmetric problem of a circular membrane in contact with a flat obstacle introduced in section 2.2. Recall that with parameters chosen according to eq. (2.4), the exact solution for the free boundary is a circle of radius $\exp(-1)$ concentric with the membrane boundary. Figures 5.3a and 5.3b show the sequence of free boundary iterates computed by Algorithm 1 when starting from the initial guesses shown in Figures 5.2a and 5.2c respectively, with the parameter $\tau = 10$ and while using the same universal mesh over the unit circle Ω. The images only show a representative few iterates and not the entire sequence $\{\Gamma_k\}_k$. In Figure 5.3a, notice that the algorithm computes a sequence of approximately circular free boundary iterates when starting from a circular initial guess. This is to be expected, thanks to the symmetries in the problem and the initial data, and it is reassuring to see this feature being preserved in the solution iterates as well. Additional details of the simulation corresponding to Figure 5.3b are provided in figs. 5.3d to 5.3k. These images show snapshots of the computed meshes $\mathcal{T}_{h,k}$ and the velocity of free boundary estimated at corresponding time instants. A three-dimensional rendering of the final solution is shown in Figure 5.3c.

Evolution of the shape functional and its derivative.

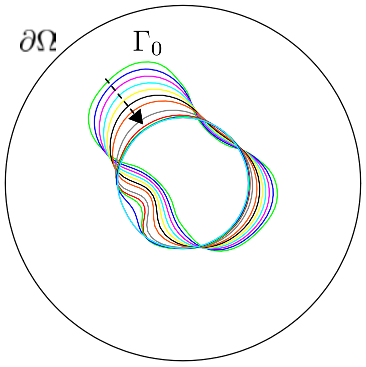
Figure 5.4a shows the evolution of the functional $\mathrm{J}_{10}(\Gamma_k)$ with the number of iterations k when starting from each one of the four initial guesses in Figure 5.2. In each case, the shape functional values converge to the correct one $\mathrm{J}_{10}(\Gamma) = 2.75763$ within about 20 iterations. Correspondingly, the shape derivative along the direction of the velocity computed in the algorithm approaches zero as seen in Figure 5.4b, indicating convergence of the free boundary iterates. It is not a coincidence that the shape derivative values plotted in the figure are all nonpositive— the choice of the free boundary velocity discussed in section 3.3 guarantees this. The monotonic decrease in the shape functional observed in Figure 5.4a is in turn a direct consequence of maintaining a nonpositive shape derivative and a sufficiently small time step. **Error in discrete solutions and convergence with mesh size.** Next we inspect the error in computed free boundaries and their dependence on the initial guess Γ_0, the parameter τ, and on the mesh size h. We measure the error in the solution iterate Γ_k with the metrics denoted by $\mathrm{E}_{\mathrm{rms}}(\Gamma_k)$ and $\mathrm{E}_{\mathrm{L}_2}(\Gamma_k)$. The first error metric is the root mean square of the distances of vertices lying on Γ_k from the exact solution Γ:

$$\mathrm{E}_{\mathrm{rms}}(\Gamma_k) \triangleq \left(\frac{1}{\#\,\text{vertices in } \Gamma_{h,k}} \sum_{\text{vertex } i \in\ \Gamma_{h,k}} \|x_i - \Pi_\Gamma(x_i)\|^2 \right)^2, \tag{5.2}$$

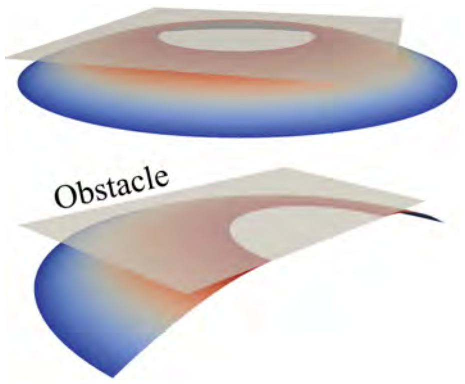
where x_i is the Cartesian coordinates of vertex i lying on Γ_k, and $\Pi_\Gamma(x_i)$ is the closest point projection



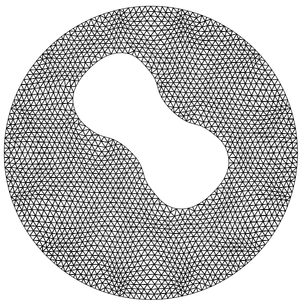
(a) Evolution of free boundary iterates when starting from the initial guess in Figure 5.2a. Notice the circular symmetry of solution iterates.



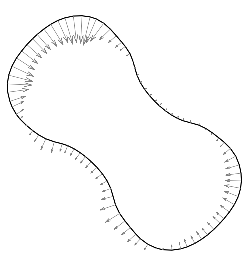
(b) Evolution of free boundary iterates when starting from initial guess in Figure 5.2c. The final solution is essentially identical to that in (a).



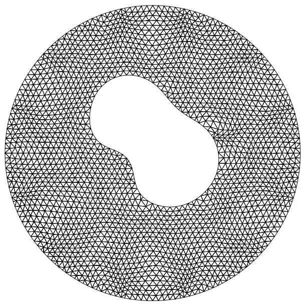
(c) A 3D rendering of the deformed membrane in contact with a flat obstacle. Only the noncoincidence set is depicted. Colors indicate contours of the displacement (legend omitted).



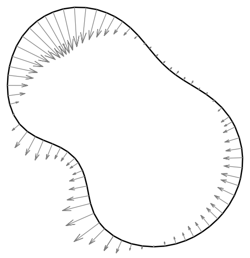
(d) The mesh $\mathcal{T}_{h,0}$.



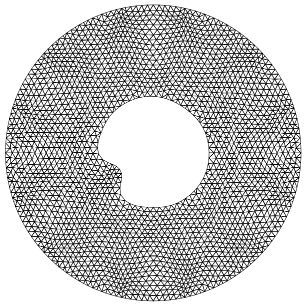
(e) Free boundary velocity at the 0th iteration.



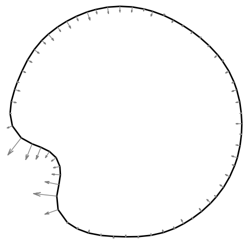
(f) The mesh $\mathcal{T}_{h,8}$



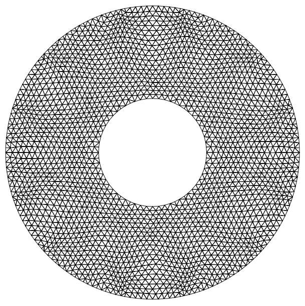
(g) Free boundary velocity at the 8th iteration.



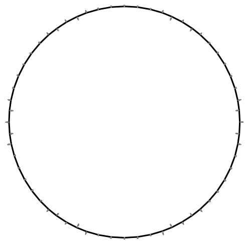
(h) The mesh $\mathcal{T}_{h,20}$



(i) Free boundary velocity at the 20th iteration.



(j) The mesh $\mathcal{T}_{h,44}$

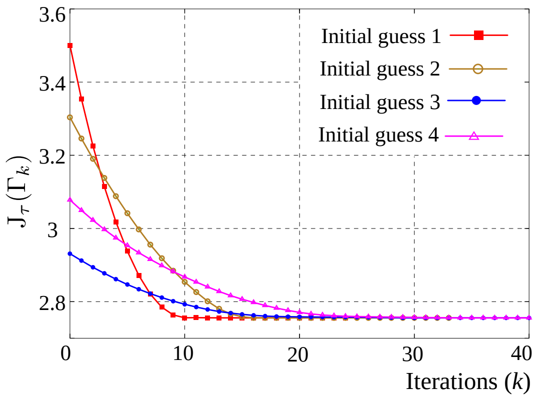


(k) Free boundary velocity at the 44th iteration.

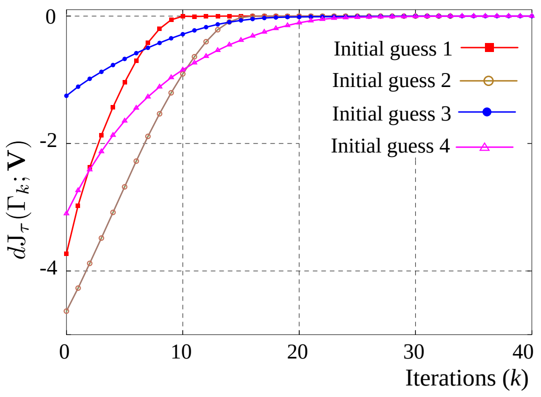
Figure 5.3: Details of the numerical example in section 5.3 simulating contact between a circular membrane and a flat obstacle. The first row of images show the evolution of free boundary iterates when starting from different initial guesses for Γ_0, and a 3D rendering of the computed deformation of the membrane. Images (d)–(k) show details of the meshes and the free boundary velocities realized at different iterations corresponding to the simulation in (b). Note that the magnitudes of the velocity vectors shown have been rescaled (uniformly for visualization purposes, and that the velocities approach zero in (k) indicating convergence of the free boundary.

of x_i onto Γ. The second error metric measures the deviation of Γ_k from Γ in an L^2 sense instead:

$$\mathrm{E}_{\mathrm{L}_2}(\Gamma_k) \triangleq \left(\int_{x\in\Gamma_{h,k}} \|x - \Pi_\Gamma(x)\|^2 \, ds \right)^{1/2} . \tag{5.3}$$



(a)



(b)

Figure 5.4: Convergence of shape functional values (left) and shape derivatives (right) with the number of iterations in Algorithm 1 for the example in section 5.3 when starting from each one of the four initial guesses in Figure 5.2. The monotonic decrease in J_τ and the strictly nonpositive values of its shape derivative evident from the plots are direct consequences of the ansatz eq. (3.8) for the free boundary velocity.

In both eqs. (5.2) and (5.3), the discretized version $\Gamma_{h,k}$ serves as a proxy for Γ_k. In eq. (5.2), vertices on $\Gamma_{h,k}$ sample Γ_k when computing the RMS error, while in eq. (5.3), edges in $\Gamma_{h,k}$ facilitate straightforward evaluation of the integral error measure.

Table 5.5a lists the errors in the free boundary computed by Algorithm 1 when starting from each one of the initial guesses in Figure 5.2, with all other parameters ($\tau = 10$ and $\mathcal{U}_h$) being common. Table 5.5b reports the errors for a combination of initial guesses and τ values while using the same universal mesh. *We observe that the location of the free boundary is computed within a small fraction of the mesh size— the accuracy of the free boundary is not limited to h, but is roughly two orders of magnitude smaller than it.* In Table 5.5c, we inspect the convergence of the error in the free boundary with the mesh size. For this purpose, we use a fixed time step without backtracking in Algorithm 1 so that free boundaries computed with progressively refined meshes can be compared at the same final time instant. We scale the time step linearly with the mesh size while maintaining the end time fixed. Hence the number of descent iterations scales inversely with the mesh size. From the data, it appears that the error converges at least linearly with the mesh size.

Γ_0	$\mathrm{E_{rms}}$	$\mathrm{E_{L_2}}$
Figure 5.2a	4.69×10^{-4}	$4.73 \times 10^{-4}\,(0.012\,h)$
Figure 5.2b	5.22×10^{-4}	$5.32 \times 10^{-4}\,(0.014\,h)$
Figure 5.2c	5.50×10^{-4}	$8.24 \times 10^{-4}\,(0.021\,h)$
Figure 5.2d	5.43×10^{-4}	$5.67 \times 10^{-4}\,(0.015\,h)$

(a) Insensitivity to the initial guess

τ	Γ_0	$\mathrm{E_{rms}}$	$\mathrm{E_{L_2}}$
2	Figure 5.2a	3.21×10^{-4}	$3.93 \times 10^{-4}\,(0.010\,h)$
5	Figure 5.2b	4.88×10^{-4}	$5.44 \times 10^{-4}\,(0.014\,h)$
10	Figure 5.2c	6.96×10^{-4}	$1.03 \times 10^{-3}\,(0.026\,h)$
100	Figure 5.2d	5.39×10^{-4}	$5.52 \times 10^{-4}\,(0.014\,h)$

(b) Insensitivity to the choice of parameter τ

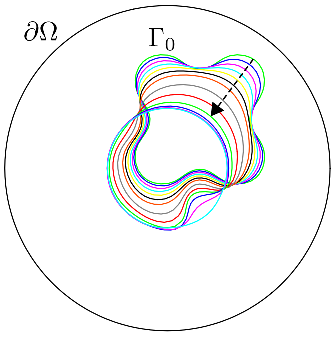
h	$\mathrm{E_{rms}}$	$\mathrm{E_{L_2}}$
5/64	1.81308×10^{-3}	1.6×10^{-3}
5/128	8.36×10^{-4}	9.74×10^{-4}
5/128	3.91×10^{-4}	4.86×10^{-4}

(c) Convergence of the free boundary with mesh refinement in Algorithm 1.

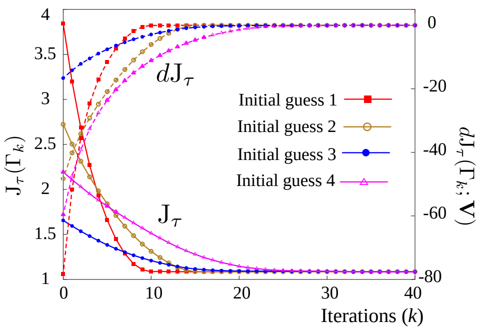
Figure 5.5: Tables (a) and (b) demonstrate insensitivity of the free boundary computed by Algorithm 1 to the choice of the initial guess and to the parameter τ, while (c) demonstrates convergence of the free boundary with mesh refinement.

5.4 Circular membrane in contact with a spherical obstacle

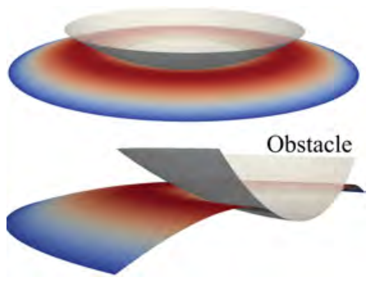
Next we consider the example of a loaded circular membrane in contact with a spherical obstacle introduced in section 2.2. With parameters chosen according to eq. (2.5), the exact solution for the free boundary is a circle of radius $\exp(-1)$ concentric with the membrane. We repeat the investigations performed in the previous example and summarize the results in Figure 5.6. Figure 5.6a shows the evolution of free boundary iterates when starting from the initial guess depicted in Figure 5.2d. The computed three-dimensional solution is rendered in Figure 5.2c and details of intermediate iterations from the simulation are shown in figs. 5.6d to 5.6k. Figure 5.6b shows the convergence of the shape functional and its derivative with the number of iterations in Algorithm 1 when starting from each one of the four initial guesses in Figure 5.2. As we observed previously in Figure 5.4, the value of the shape



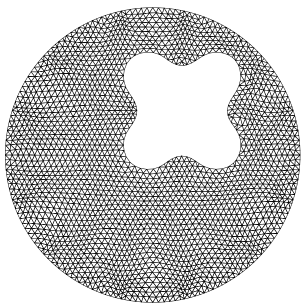
(a) Evolution of free boundary iterates when starting from the initial guess in Figure 5.2d



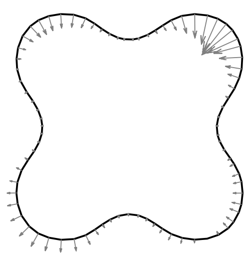
(b) Convergence of the functional J_τ and its shape derivative with number of the iterations in Algorithm 1 when starting from each of the initial guesses in Figure 5.2.



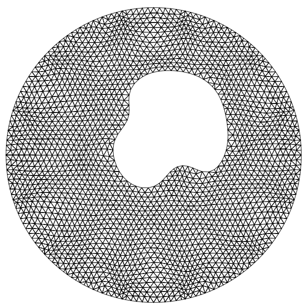
(c) 3D rendering of the computed deformation of the membrane.



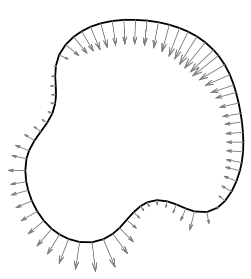
(d) The mesh $\mathcal{T}_{h,0}$.



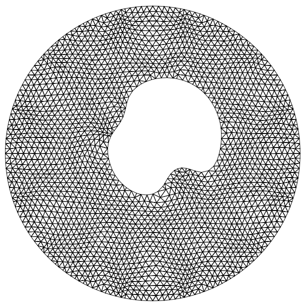
(e) Free boundary velocity at the 0th iteration.



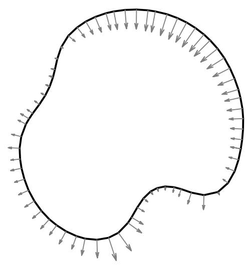
(f) The mesh $\mathcal{T}_{h,12}$.



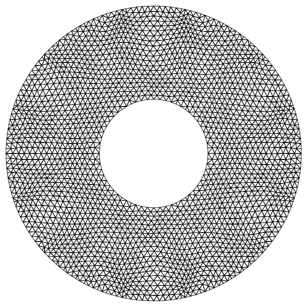
(g) Free boundary velocity at the 12th iteration.



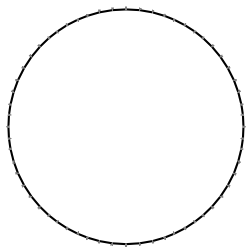
(h) The mesh $\mathcal{T}_{h,16}$.



(i) Free boundary velocity at the 16th iteration.



(j) The mesh $\mathcal{T}_{h,50}$.



(k) Free boundary velocity at the 50th iteration.

Γ_0	$\mathrm{E}_{\mathrm{L}_2}$
Figure 5.2a	$6.91 \times 10^{-4}\,(0.018\,h)$
Figure 5.2b	$4.13 \times 10^{-4}\,(0.011\,h)$
Figure 5.2c	$3.51 \times 10^{-4}\,(0.009\,h)$
Figure 5.2d	$3.55 \times 10^{-4}\,(0.009\,h)$

(l) Insensitivity of the computed free boundary to the choice of initial guess.

τ	Γ_0	$\mathrm{E}_{\mathrm{rms}}$
2	Figure 5.2a	$3.97 \times 10^{-3}\,(0.102\,h)$
5	Figure 5.2b	$3.89 \times 10^{-4}\,(0.01\,h)$
50	Figure 5.2c	$3.62 \times 10^{-4}\,(0.009\,h)$
100	Figure 5.2d	$3.63 \times 10^{-4}\,(0.009\,h)$

(m) Insensitivity of the computed free boundary to the parameter τ.

h	$\mathrm{E}_{\mathrm{rms}}$	$\mathrm{E}_{\mathrm{L}_2}$
5/64	8.23×10^{-4}	1.71×10^{-3}
5/128	4.23×10^{-4}	7.23×10^{-4}
5/256	0.96×10^{-4}	1.4×10^{-4}

(n) Convergence of error in the free boundary with mesh refinement in Algorithm 1.

Figure 5.6: Details of the numerical experiment discussed in section 5.4 simulating contact between a circular membrane and a spherical obstacle. Images (d)–(k) show details of intermediate iterations from the simulation in (a).

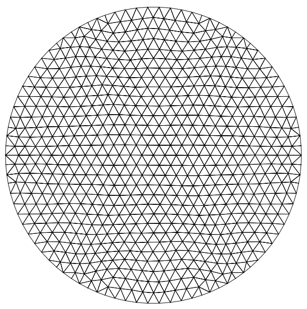
functional decreases monotonically and converges within about 20 iterations, while its shape derivative remains nonpositive and converges to zero. Tables 5.6l and 5.6m demonstrate the insensitivity of the algorithm to the initial guess and to the parameter τ. In Table 5.6n, we examine the convergence of the computed free boundary to the exact one with mesh refinement. The specific simulation parameters used in the convergence study are identical to those used in the previous example in section 5.3. We draw attention to the fact that the error $\mathrm{E}_{\mathrm{L}^2}$ at $h = 5/256$ is roughly four times smaller than that at $h = 5/128$, which suggests that the error in the free boundary converges quadratically with the mesh size.

5.4.1 Comparison with the variational inequality approach

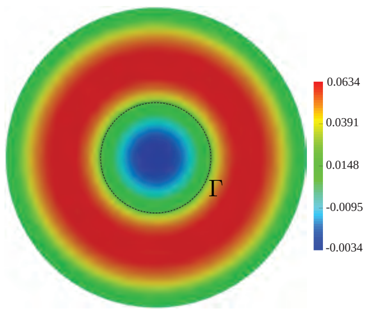
We use this example to compare the accuracy of Algorithm 1 with the conventional approach of first computing the displacement of the membrane by resolving a variational inequality and identifying the coincidence set a posteriori. Specifically, we first compute

$$u_h \triangleq \arg\min_{w \in \mathrm{K}_h} \mathrm{E}[v_h], \quad \text{where} \quad \begin{cases} \mathrm{E}[v_h] & = \int_\Omega \left(\frac{1}{2}\nabla v_h \cdot \nabla v_h - f v_h\right), \\ \mathrm{K}_h & = \{v_h \in \mathrm{V}_h \,:\, v(x_a) \leq \psi(x_a)\}, \end{cases} \tag{5.4}$$

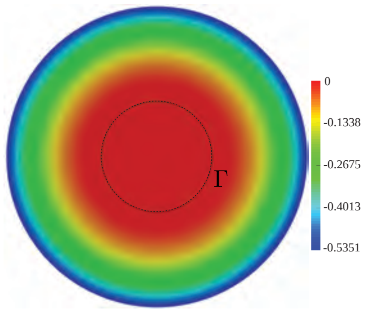
V_h is the finite element space of continuous piecewise linear functions defined over the triangulation $\mathcal{U}_h$ of Ω that vanish on $\partial\Omega$, and $\{x_a\}_a$ are the vertices (nodes) in the mesh $\mathcal{U}_h$. The mesh $\mathcal{U}_h$ used for this calculation is shown in Figure 5.7a, where the mesh size h is approximately 5/64. Notice that the set of admissible solutions K_h is convex but not linear. As we highlighted in the discussions in chapter 1, constructing K_h requires imposing as many constraints as the number of degrees of freedom



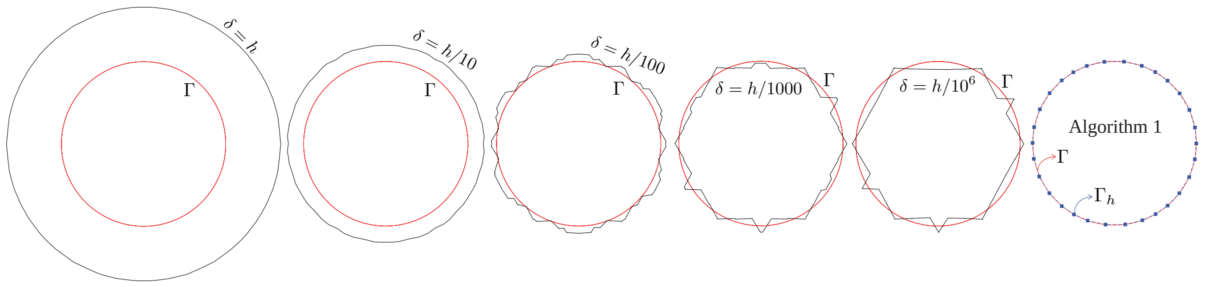
(a) Mesh used for computing membrane displacements using the variational inequality approach in eq. (5.4).



(b) Contours of the solution u_h of eq. (5.4).



(c) Contours of the function $u_h - \psi$, whose zero level set defines the coincidence set.



(d) Recovering the free boundary location from the computed membrane displacement u_h. The first five images show the predictions resulting from a range of choices for the tolerance δ, while the sixth image shows the free boundary computed by Algorithm 1.

Figure 5.7: Comparison of the shape optimization-based approach for computing the free boundary in Algorithm 1 with the conventional approach of first computing the membrane displacement by resolving a variational inequality and then determining the free boundary a posteriori for the example in section 5.4. The second row of images show the free boundaries predicted with specific choices of the tolerance parameter δ in eq. (5.5). These predictions should be compared with the last image showing the free boundary computed at the end of 20 iterations of Algorithm 1 while using the triangulation in (a) as the universal mesh.

in the problem.

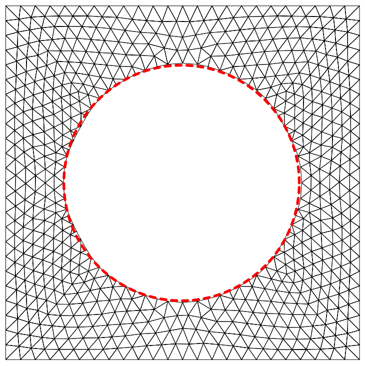
To resolve eq. (5.4), we use the active set semi-smooth method discussed in [68] and implemented in the PETSc library [69]. The algorithm is iterative, with each iteration consisting in resolving a linear system of equations, followed by an update of the active set of constraints. In this particular example, we find that the solution converges in 11 iterations. Figure 5.7b shows the contours of the computed solution u_h. Contours of the function $(u_h - \psi)$ are plotted in Figure 5.7c. Notice that the set of points where $(u_h - \psi) = 0$ is the predicted coincidence set C_h. Evidently, it is necessary to choose a tolerance for identifying C_h, at least to account for calculations being performed with finite precision. We do not delve into the details of how to choose such tolerances here; discussions and analysis to this end can be found in [25, 26, 70]. Instead, we simply examine the consequences of choosing a broad range of tolerances. Setting

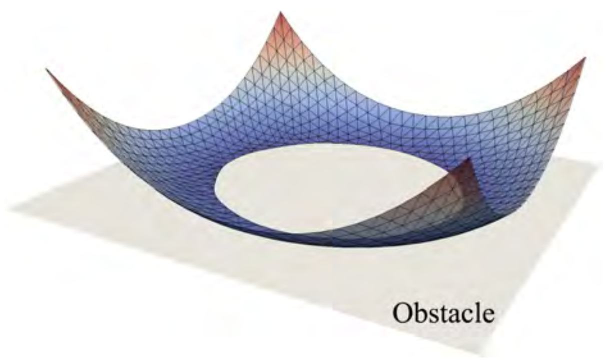
$$\mathrm{C}_h^\delta \triangleq \{x \in \Omega : u_h(x) > \psi(x) - \delta\}, \tag{5.5}$$

Figure 5.7c shows the free boundary $\partial\mathrm{C}_h^\delta$ for the choices $\delta = h, h/10^1, h/10^2, h/10^3, h/10^6$. The last image in Figure 5.7c shows the free boundary computed at the end of 20 iterations of Algorithm 1 when using $\mathcal{U}_h$ shown in Figure 5.7a as the universal mesh. Comparing it with the previous five images in Figure 5.7c unequivocally reveals the benefit of computing the free boundary directly as we have proposed here, rather than computing it a posteriori from the membrane deflection.

Before proceeding, we mention that the comparison shown in Figure 5.7 should be interpreted with some caution. It is well known that due to the limited regularity of membrane deflections, it is necessary to adaptively refine the mesh in the vicinity of the free boundary when resolving obstacle problems as variational inequalities. We have not considered any such refinement here. Nevertheless, the comparison is also fair in the sense that the same mesh used for resolving eq. (5.4) also serves as the universal mesh for Algorithm 1. Moreover, the computational cost of the semi-smooth active set method is comparable with that of Algorithm 1— iterations over the active set of constraints in the former are replaced by iterations for the free boundary location in the latter. Finally, we mention that problematic issues in identifying the free boundary from membrane displacements that we have highlighted through this example is not specific to calculations with piecewise linear finite elements but is generally symptomatic of methods that identify the free boundary a posteriori from approximate membrane displacements.

5.5 Square-shaped membrane in contact with a flat obstacle


(a)



(b)

Figure 5.8: Simulating the contact of a square-shaped membrane with a flat obstacle. The left image shows the triangulated noncoincidence set computed by Algorithm 1 and the exact location of the free boundary indicated by a dashed red line. The image on the right depicts a 3D rendering of the computed deformation of the membrane.

Next we consider an example from [71] that consists of a square-shaped membrane $\Omega = (-1.5, 1.5)^2$ loaded by a constant force $f = 2$ and subject to Dirichlet boundary conditions

$$u(x) = \|x\|^2/2 - \log\|x\| - 1/2 \quad \text{along } \partial\Omega.$$

The membrane is in contact with a flat obstacle with height function $\psi = 0$. Unlike the examples in sections 5.3 and 5.4, the Dirichlet boundary condition here is nonhomogeneous. The exact solution to this problem is

$$u(x) = \begin{cases} \|x\|^2/2 - \log\|x\| - 1/2 & \text{if } \|x\| \geq 1, \\ 0 & \text{otherwise,} \end{cases}$$

and the free boundary Γ is the unit circle centered at the origin of coordinates. For the sake of conformity with [71], we consider the contact constraint $u \geq \psi$ in this example rather than $u \leq \psi$ that we have assumed thus far. In the simulation, we use the triangulation shown in Figure 5.2a as the universal mesh and set $\tau = 10$. Figure 5.8a shows the exact free boundary in red and the converged noncoincidence set (triangulated domain). We find the errors in the location of the free boundary to be $\mathrm{E}_{\mathrm{rms}} = 0.0078$ and $\mathrm{E}_{\mathrm{L}^2} = 0.019$, which are much smaller than the mesh size $h \approx 1/16$. We highlight that such accuracy is despite the coarseness of the universal mesh used and the lack of any mesh adaptivity in the vicinity of the free boundary. These observations stand out when compared to variational inequality-based approaches that resolve this problem by computing u as the primal unknown while using extensive mesh refinement despite the lack of any special features in either u or Γ [71].

5.6 Circular membrane in contact with a cylindrical obstacle

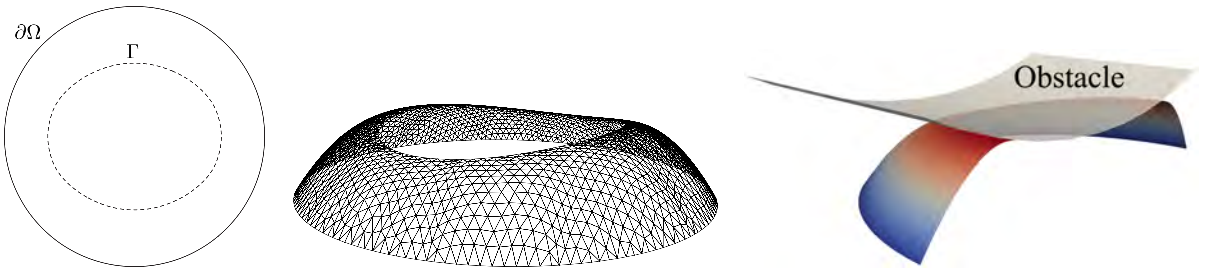


Figure 5.9: Simulating contact of a circular membrane with a cylindrical obstacle. The converged free boundary is shown on the left. 3D renderings of the deformed membrane are shown in (b) and (c). Notice that the image of the free boundary in the deformed membrane is non planar.

In this final example, we consider the circular membrane from the examples in sections 5.4 and 5.5

to be loaded by a constant force $f = 5$ and to be in contact with a cylindrical obstacle. The height function of the obstacle is

$$\psi(x, y) = Z - \sqrt{\rho^2 - y^2}, \tag{5.6}$$

where $\rho = 2.5$ is the radius of the cylinder, $Z = 2.75$ is the height of the axis of the cylinder above the plane of the membrane and (x, y) denotes the Cartesian coordinates of a point in Ω. As evident from eq. (5.6), the axis of the cylinder is parallel to the plane of Ω.

Figure 5.9a shows the converged solution for the free boundary, while Figures 5.9b and 5.9c show three-dimensional renderings of the deformation of the membrane over the noncoincidence set. In all our previous examples, the interface between the obstacle and the deformed membrane was planar. In contrast, here we find that the image of the free boundary on the deformed membrane is non planar. Curiously, we find that the computed free boundary is very well approximated by an ellipse. We have verified this observation for various combinations of the parameters f, Z and ρ. For instance, the free boundary depicted in Figure 5.9a has semi-major and semi-minor axes approximately equal to 0.668 and 0.562 respectively. We are not aware of any analytical solutions to this problem and hope that this observation fuels further investigations.

Chapter 6

Concluding remarks

We have described an algorithm for simulating the obstacle problem by considering the free boundary to be the primary unknown. Such an approach has clear algorithmic benefits over conventional methods based on resolving variational inequalities, with the most prominent ones being the lack of any inequality constraints and the far superior accuracy possible in computing the region of contact. We discussed various aspects of the algorithm in detail and demonstrated its performance with numerous examples. It is certainly possible to conceive of ways to combine our shape optimization approach with other algorithms. For instance, the solution computed using a penalty formulation can be "polished" with the algorithm proposed here.

A crucial aspect of the proposed algorithm that is left open in this article concerns the analysis of the error in the computed free boundary and its convergence to the exact solution with refinement of the universal mesh. While the examples in sections 5.3 and 5.4 clearly confirm convergence, it is premature to speculate any further. Presumably, the error in the computed free boundary depends on a combination of factors— the mesh size, the time step, the accuracy of fluxes of the displacement and its adjoint computed along the free boundary, and on the representation adopted for the free boundary. A detailed examination of each of these factors is beyond the scope of this article but is important nonetheless.

It remains to be seen whether the approach adopted here can be extended to problems of contact between plates/shells with rigid obstacles or even to the Signorini problem. While many of the techniques used here will presumably be applicable verbatim, we expect the question of identifying a suitable shape functional to be the key challenge. In this context, we highlight an interesting perspective provided by [72] that recommends computing the free boundary velocity based on certain unbalanced residuals. Although there may not exist a shape functional underlying such a choice, the idea appears to work well in practice.

Appendix A

Sensitivity of the functional J_τ

We provide a concise derivation for the the sensitivity of the functional J_τ. We closely follow the proof of Theorem 7.1 in [32] and provide intermediate details to keep the explanations self-contained.

We start with a given location for the free boundary Γ and a prescribed velocity $\mathbf{V}$ for its evolution. Assuming that $\mathbf{V}$ is extended in a sufficiently smooth manner to $\overline{\Omega}$ such that $\mathbf{V} = 0$ on $\partial\Omega$, consider the 1-parameter family of mappings defined over Ω $\varepsilon \mapsto \varphi_\varepsilon$ as

$$x \mapsto \varphi_\varepsilon(x) \triangleq x + \varepsilon\,\mathbf{V}(x), \tag{A.1}$$

and set $\Gamma_\varepsilon = \varphi_\varepsilon(\Gamma)$. Formally, we are interested in computing

$$d\mathrm{J}_\tau(\Gamma;\mathbf{V}) \triangleq \lim_{\varepsilon\to 0} \frac{d}{d\varepsilon}\mathrm{J}_\tau(\Gamma_\varepsilon). \tag{A.2}$$

That $d\mathrm{J}_\tau$ is well defined in eq. (A.2) is proved in [32, Theorem 6.2], see also [73, Chapter 4] for arguments starting from first principles.

As usual, we denote the coincidence and noncoincidence sets corresponding to the free boundary Γ by C and D, respectively. The analogous sets corresponding to Γ_ε shall be denoted by C_ε and D_ε. Hence

$$\mathrm{J}_\tau(\Gamma_\varepsilon) = \int_\Omega \left(\frac{1}{2}\nabla u_\varepsilon \cdot \nabla u_\varepsilon + (\tau - 1) f u_\varepsilon\right), \tag{A.3}$$

where the states u and u_ε satisfy

$$\begin{cases} -\Delta u = f & \text{on D}, \\ u = 0 & \text{on } \partial\Omega, \\ u = \psi & \text{on C}, \end{cases} \qquad \text{and} \qquad \begin{cases} -\Delta u_\varepsilon = f & \text{on } \mathrm{D}_\varepsilon, \\ u_\varepsilon = 0 & \text{on } \partial\Omega, \\ u_\varepsilon = \psi & \text{on } \mathrm{C}_\varepsilon. \end{cases}$$

We have assumed homogeneous Dirichlet boundary conditions along $\partial\Omega$ for the sake of simplicity.

Material and shape derivatives of u**.** Evidently, we will need the shape derivative of u to evaluate eq. (A.2). To this end, we set $U(x,\varepsilon) \triangleq u_\varepsilon(x+\varepsilon\,\mathbf{V}(x))$, so that a direct application of the chain rule yields

$$\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0} U(x,\varepsilon) = \frac{\partial U(x,0)}{\partial \varepsilon} + \nabla U(x,0)\cdot \mathbf{V}(x). \tag{A.4}$$

Permitting a minor abuse of notation, eq. (A.4) is more succinctly expressed as

$$\dot{u} = u' + \nabla u \cdot \mathbf{V}, \tag{A.5}$$

where we identify $\dot{u}$ to be the material derivative and u' to be the shape derivative of u. Since the forcing f and the height function ψ are prescribed, their shape derivatives f' and ψ' are zero, and hence

$$\dot{f} = \nabla f \cdot \mathbf{V} \quad \text{and} \quad \dot{\psi} = \nabla\psi\cdot\mathbf{V}.$$

We use the definition of u' in eq. (A.5) to deduce the conditions satisfied by it. Exploiting the fact that the shape derivative and spatial gradients commute, and using $f'=0$, we get

$$-\Delta u_\varepsilon = f \;\Rightarrow\; \Delta u' = 0 \quad \text{on } \Omega, \tag{A.6}$$

while on Γ, we have $u=\psi \;\Rightarrow\; \dot{u}=\dot{\psi} \;\Rightarrow\; u'+\nabla u\cdot\mathbf{V} = \nabla\psi\cdot\mathbf{V} \;\Rightarrow\; u' = (\nabla\psi - \nabla u)\cdot\mathbf{V}$. Since $u=\psi$ along Γ, the tangential derivatives of u and ψ along Γ coincide. Therefore,

$$u' = \left(\frac{\partial\psi}{\partial n} - \frac{\partial u}{\partial \mathbf{n}}\right)\mathrm{V}_n \quad \text{on } \Gamma,$$

where $\mathrm{V}_n \triangleq \mathbf{V}\cdot\mathbf{n}$, and $\mathbf{n}$ is the unit normal to Γ oriented such that it points away from D. Finally, we note that the boundary condition $u=0$ on $\partial\Omega$ implies that u' also vanishes in $\partial\Omega$. In this way, we find that $u' : \mathrm{D}\to\mathbb{R}$ is the solution to the problem

$$\begin{cases} -\Delta u' = 0 & \text{on } \Omega, \\ u' = (\partial\psi/\partial\mathbf{n} - \partial u/\partial\mathbf{n})\mathrm{V}_n & \text{on } \Gamma, \\ u' = 0 & \text{on } \partial\Omega. \end{cases} \tag{A.7}$$

To proceed, we rewrite eq. (A.3) as a sum of integrals over the domains C_ε and D_ε, i.e.,

$$\begin{aligned}\mathrm{J}_\tau(\Gamma_\varepsilon) &= \int_{\mathrm{C}_\varepsilon} \left(\frac{1}{2}\nabla u_\varepsilon \cdot \nabla u_\varepsilon + (\tau-1) f u_\varepsilon\right) + \int_{\mathrm{D}_\varepsilon} \left(\frac{1}{2}\nabla u_\varepsilon \cdot \nabla u_\varepsilon + (\tau-1) f u_\varepsilon\right) \\ &\triangleq \mathrm{J}_\tau^{\mathrm{C}}(\Gamma_\varepsilon) + \mathrm{J}_\tau^{\mathrm{D}}(\Gamma_\varepsilon). \end{aligned} \tag{A.8}$$

In the following, we compute the sensitivities of $\mathrm{J}_\tau^{\mathrm{C}}$ and $\mathrm{J}_\tau^{\mathrm{D}}$ individually.

Sensitivity of $\mathrm{J}_\tau^{\mathrm{C}}$. Since $u_\varepsilon = \psi$ on C_ε, we have

$$\mathrm{J}_\tau^{\mathrm{C}}(\Gamma_\varepsilon) = \int_{\mathrm{C}_\varepsilon} \left(\frac{1}{2}\nabla \psi \cdot \nabla \psi + (\tau-1) f \psi\right).$$

By a direct application of the Reynolds transport theorem (see [74, Chapter 3]) and noting the convention that $\mathbf{n}$ is the inward normal to $\partial\mathrm{C}$, we get

$$\left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} \mathrm{J}_\tau^{\mathrm{C}}(\Gamma_\varepsilon) = -\int_{\Gamma} \left(\frac{1}{2}\|\nabla\psi\|^2 + (\tau-1) f\psi\right) \mathrm{V}_n. \tag{A.9}$$

Sensitivity of $\mathrm{J}_\tau^{\mathrm{D}}$. A similar application of the transport theorem for $\mathrm{J}_\tau^{\mathrm{D}}$ yields

$$\left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} \mathrm{J}_\tau^{\mathrm{D}}(\Gamma_\varepsilon) = \int_{\mathrm{D}} \left(\nabla u' \cdot \nabla u + (\tau-1) f u'\right) + \int_{\partial\mathrm{D}} \left(\frac{1}{2}\|\nabla u\|^2 + (\tau-1) f u\right) \mathrm{V}_n. \tag{A.10}$$

- Recognizing that $\partial\mathrm{D} = \partial\Omega \cup \Gamma$, $\mathbf{V} = 0$ on $\partial\Omega$ and $u = \psi$ in Γ, we get

$$\int_{\partial\mathrm{D}} \left(\frac{1}{2}\|\nabla u\|^2 + (\tau-1) f u\right) \mathrm{V}_n = \int_{\Gamma} \left(\frac{1}{2}\|\nabla u\|^2 + (\tau-1) f \psi\right) \mathrm{V}_n. \tag{A.11}$$

- To simplify the term $\int_{\mathrm{D}} \nabla u' \cdot \nabla u$, we invoke $\Delta u' = 0$ on D and $u = 0$ on $\partial\Omega$ to get

$$\int_{\mathrm{D}} \nabla u' \cdot \nabla u = \int_{\mathrm{D}} \left(\nabla \cdot (u \nabla u') - u \Delta u'\right) = \int_{\partial\Omega} u \frac{\partial u'}{\partial \mathbf{n}} + \int_{\Gamma} u \frac{\partial u'}{\partial \mathbf{n}} = \int_{\Gamma} u \frac{\partial u'}{\partial \mathbf{n}}. \tag{A.12}$$

- Next, we simplify the term $\int_{\mathrm{D}} (\tau-1) f u'$. Since $-\Delta u = f$ and $\Delta u' = 0$ in D, we have

$$f u' = (-\Delta u) u' = (u \Delta u' - u' \Delta u) = \nabla \cdot (u \nabla u' - u' \nabla u) \quad \text{in D.}$$

Then, an application of the divergence theorem, while noting $u = u' = 0$ in $\partial\Omega$ yields

$$\begin{aligned}\int_{\mathrm{D}} fu' &= \int_{\partial\mathrm{D}} \left(u\frac{\partial u'}{\partial \mathbf{n}} - u'\frac{\partial u}{\partial \mathbf{n}} \right) \\ &= \int_{\Gamma} \left('\frac{\partial u'}{\partial \mathbf{n}} - u'\frac{\partial u}{\partial \mathbf{n}} \right) \\ &= \int_{\Gamma} u\frac{\partial u'}{\partial \mathbf{n}} - \int_{\Gamma} \frac{\partial u}{\partial \mathbf{n}} \left(\frac{\partial \psi}{\partial n} - \frac{\partial u}{\partial \mathbf{n}} \right) \mathrm{V}_n. \end{aligned} \tag{A.13}$$

Summarizing the calculations from eqs. (A.11) to (A.13) in eq. (A.10), we find that

$$\left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} \mathrm{J}_\tau^{\mathrm{D}}(\Gamma_\varepsilon) = \int_\Gamma \left(\tau u\frac{\partial u'}{\partial \mathbf{n}} - (\tau-1)\frac{\partial u}{\partial \mathbf{n}} \left(\frac{\partial \psi}{\partial \mathbf{n}} - \frac{\partial u}{\partial n} \right) \mathrm{V}_n + \left(\frac{1}{2}\|\nabla u\|^2 + (\tau-1) f\psi \right) \mathrm{V}_n \right). \tag{A.14}$$

Sensitivity of J_τ in terms of u and u'. Adding eq. (A.9) and eq. (A.14), we get

$$d\mathrm{J}_\tau(\Gamma; \mathbf{V}) = \int_\Gamma \left(\tau u\frac{\partial u'}{\partial \mathbf{n}} + (\tau-1)\frac{\partial u}{\partial \mathbf{n}} \left(\frac{\partial u}{\partial \mathbf{n}} - \frac{\partial \psi}{\partial n} \right) \mathrm{V}_n + \frac{1}{2} \left(\|\nabla u\|^2 - \|\nabla \psi\|^2 \right) \mathrm{V}_n \right). \tag{A.15}$$

As we argued previously,

$$u = \psi \text{ on } \Gamma \quad \Rightarrow \quad \|\nabla u\|^2 - \|\nabla \psi\|^2 = \left(\frac{\partial u}{\partial \mathbf{n}} \right)^2 - \left(\frac{\partial \psi}{\partial \mathbf{n}} \right)^2 \quad \text{on } \Gamma.$$

Therefore eq. (A.15) becomes

$$d\mathrm{J}_\tau(\Gamma; \mathbf{V}) = \int_\Gamma \left(\tau u\frac{\partial u'}{\partial \mathbf{n}} + (\tau-1)\frac{\partial u}{\partial \mathbf{n}} \left(\frac{\partial u}{\partial \mathbf{n}} - \frac{\partial \psi}{\partial n} \right) \mathrm{V}_n + \frac{1}{2} \left(\left(\frac{\partial u}{\partial \mathbf{n}} \right)^2 - \left(\frac{\partial \psi}{\partial \mathbf{n}} \right)^2 \right) \mathrm{V}_n \right). \tag{A.16}$$

As predicted by the Hadamard structure theorem, the sensitivity $d\mathrm{J}_\tau(\Gamma; \mathbf{V})$ in eq. (A.16) depends only on information along the free boundary Γ. We also expect $d\mathrm{J}_\tau(\Gamma; \mathbf{V})$ to depend linearly on V_n. While this is apparent for the second and third terms on the righthand side of eq. (A.16), we introduce the adjoint state p to elucidate the dependence of first term ($u\,\partial u'/\partial \mathbf{n}$) on V_n.

The adjoint state. Let $p : \mathrm{D} \to \mathbb{R}$ be the solution to the problem

$$\begin{cases} -\Delta p = 0 & \text{on D}, \\ p = \psi & \text{on } \Gamma, \\ p = 0 & \text{on } \partial\Omega. \end{cases} \tag{A.17}$$

Since $u = \psi = p$ on Γ, we have

$$\int_\Gamma u\frac{\partial u'}{\partial \mathbf{n}} = \int_\Gamma p\frac{\partial u'}{\partial \mathbf{n}}.$$

Then, exploiting the facts that $\Delta u' = \Delta p = 0$ on D and $u' = p = 0$ on $\partial\Omega$, we get

$$\begin{aligned}
\int_\Gamma u\frac{\partial u'}{\partial \mathbf{n}} = \int_\Gamma p\frac{\partial u'}{\partial \mathbf{n}} = \int_\Gamma p\frac{\partial u'}{\partial \mathbf{n}} + \int_{\partial\Omega} p\frac{\partial u'}{\partial \mathbf{n}} = \int_{\mathrm{D}} \nabla\cdot(p\nabla u') &= \int_{\mathrm{D}} (\nabla p\cdot\nabla u' + p\Delta u') \\
&= \int_{\mathrm{D}} (\nabla p\cdot\nabla u' + u'\Delta p) \\
&= \int_{\mathrm{D}} \nabla\cdot(u'\nabla p) = \int_{\partial\mathrm{D}} u'\frac{\partial p}{\partial \mathbf{n}} \qquad (\mathrm{A.18}) \\
&= \int_\Gamma u'\frac{\partial p}{\partial \mathbf{n}} + \int_{\partial\Omega} u'\frac{\partial p}{\partial \mathbf{n}} \qquad (\mathrm{A.19}) \\
&= \int_\Gamma \frac{\partial p}{\partial \mathbf{n}} u'. \qquad (\mathrm{A.20})
\end{aligned}$$

Since $u' = (\partial\psi/\partial\mathbf{n} - \partial u/\partial\mathbf{n})\mathrm{V}_n$ on Γ by eq. (A.7), eq. (A.20) becomes

$$\int_\Gamma u\frac{\partial u'}{\partial \mathbf{n}} = \int_\Gamma \frac{\partial p}{\partial \mathbf{n}}\left(\frac{\partial \psi}{\partial \mathbf{n}} - \frac{\partial u}{\partial \mathbf{n}}\right)\mathrm{V}_n. \tag{A.21}$$

Finally, substituting eq. (A.21) in eq. (A.16), we get

$$d\mathrm{J}_\tau(\Gamma;\mathbf{V}) = \int_\Gamma \left(\frac{\partial u}{\partial \mathbf{n}} - \frac{\partial \psi}{\partial \mathbf{n}}\right)\left(\frac{1}{2}\left(\frac{\partial \psi}{\partial \mathbf{n}} + \frac{\partial u}{\partial \mathbf{n}}\right) - \tau\frac{\partial p}{\partial \mathbf{n}} + (\tau-1)\frac{\partial u}{\partial \mathbf{n}}\right)\mathrm{V}_n,$$

which is precisely the expression provided in eq. (3.6) in section 3.3.

Remarks.

1. The extension of the free boundary velocity $\mathbf{V}$ to the entire domain Ω is required only for the purpose of this derivation. In particular, Algorithm 1 does not require any extension of $\mathbf{V}$ away from Γ. Shape optimization algorithms based on advecting level set functions sometimes explicitly require constructing such extensions, cf. [38].

2. Equation (A.1) is identical to the evolution of the free boundary in Algorithm 1, with the parameter ε identified with the time step Δt.

3. Equation (A.16) shows how to compute the sensitivity of J_τ using the shape derivative u' rather than using the adjoint p as we have done in Algorithm 1. The decision to compute p rather than u' to evaluate the free boundary velocity is motivated by the close resemblance of the adjoint and the state problems, and the simplicity of the boundary conditions for p along Γ compared to that for u'. In particular, the state and adjoint problems can be solved simultaneously, while u' can be computed only after u.

Bibliography

[1] J. Lions and G. Stampacchia, "Variational inequalities," *Comm. Pure and Appl. Math.*, vol. 20, no. 3, pp. 493–519, 1967.

[2] D. Kinderlehrer and G. Stampacchia, *An introduction to variational inequalities and their applications*, vol. 31. SIAM, 1980.

[3] J. Rodrigues, *Obstacle problems in mathematical physics*, vol. 134. Elsevier, 1987.

[4] C. Elliott and J. Ockendon, *Weak and variational methods for moving boundary problems*, vol. 59. Pitman Publishing, 1982.

[5] C. Baiocchi and A. Capelo, "Variational and quasivariational inequalities: Applications to free boundary problems.," *John Wiley & Sons, New York*, 1984.

[6] J. Crank, "Free and moving boundary problems (oxford science publications)," 1987.

[7] R. Tremolieres, J. Lions, and R. Glowinski, *Numerical analysis of variational inequalities*, vol. 8. Elsevier, 2011.

[8] P. Ciarlet, "The finite element method for elliptic problems," *Classics in applied mathematics*, vol. 40, pp. 1–511, 2002.

[9] U. Mosco and G. Strang, "One-sided approximation and variational inequalities," *Bulletin of the American Mathematical Society*, vol. 80, no. 2, pp. 308–312, 1974.

[10] R. Scholz, "Numerical solution of the obstacle problem by the penalty method," *Computing*, vol. 32, no. 4, pp. 297–306, 1984.

[11] C. Johnson, "Adaptive finite element methods for the obstacle problem," *Math. Model. Methods Appl. Sci.*, vol. 2, no. 04, pp. 483–487, 1992.

[12] G. Tran, H. Schaeffer, W. Feldman, and S. Osher, "An l^1 penalty method for general obstacle problems," *SIAM J. Appl. Math.*, vol. 75, no. 4, pp. 1424–1444, 2015.

[13] F. Brezzi, W. Hager, and P. Raviart, “Error estimates for the finite element solution of variational inequalities,” *Numer. Math.*, vol. 31, no. 1, pp. 1–16, 1978.

[14] T. Gustafsson, R. Stenberg, and J. Videman, “Mixed and stabilized finite element methods for the obstacle problem,” *SIAM J. Numer. Anal.*, vol. 55, no. 6, pp. 2718–2744, 2017.

[15] K. Ito and K. Kunisch, “An augmented lagrangian technique for variational inequalities,” *Appl. Math. Optim.*, vol. 21, no. 1, pp. 223–241, 1990.

[16] J. Simo and T. Laursen, “An augmented lagrangian treatment of contact problems involving friction,” *Comput. & Structures*, vol. 42, no. 1, pp. 97–116, 1992.

[17] H. Lewy and G. Stampacchia, “On the regularity of the solution of a variational inequality,” *Comm. Pure and Appl. Math.*, vol. 22, no. 2, pp. 153–188, 1969.

[18] H. Brézis, D. Kinderlehrer, and H. Lewy, “The smoothness of solutions to nonlinear variational inequalities,” *Indiana Univ. Math. J.*, vol. 23, no. 9, pp. 831–844, 1974.

[19] F. Brezzi, W. Hager, and P. Raviart, “Error estimates for the finite element solution of variational inequalities,” *Numer. Math.*, vol. 28, no. 4, pp. 431–443, 1977.

[20] L. Wang, “On the quadratic finite element approximation to the obstacle problem,” *Numer. Math.*, vol. 92, no. 4, pp. 771–778, 2002.

[21] Z. Chen and R. Nochetto, “Residual type a posteriori error estimates for elliptic obstacle problems,” *Numer. Math.*, vol. 84, no. 4, pp. 527–548, 2000.

[22] R. Nochetto, K. Siebert, and A. Veeser, “Pointwise a posteriori error control for elliptic obstacle problems,” *Numer. Math.*, vol. 95, no. 1, pp. 163–195, 2003.

[23] P. Lee, T. Kim, and S. Kim, “Accurate and efficient numerical solutions for elliptic obstacle problems,” *Journal of inequalities and applications*, vol. 2017, no. 1, p. 34, 2017.

[24] F. Brezzi, “Recent results in the approximation of free boundaries,” in *Equadiff 6*, pp. 285–289, Springer, 1986.

[25] F. Brezzi and L. Caffarelli, “Convergence of the discrete free boundaries for finite element approximations,” *RAIRO. Anal. Numér.*, vol. 17, no. 4, pp. 385–395, 1983.

[26] R. Nochetto, “A note on the approximation of free boundaries by finite element methods,” *ESAIM: Math. Model. and Numer. Anal.*, vol. 20, no. 2, pp. 355–368, 1986.

[27] D. Kinderlehrer and L. Nirenberg, “Regularity in free boundary problems,” *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, vol. 4, no. 2, pp. 373–391, 1977.

[28] L. Caffarelli, “The obstacle problem revisited,” *Journal of Fourier Analysis and Applications*, vol. 4, no. 4-5, pp. 383–402, 1998.

[29] A. Petrosyan, H. Shahgholian, and N. Uraltseva, *Regularity of Free Boundaries in Obstacle-type Problems.* Graduate studies in mathematics, American Mathematical Society, 2012.

[30] J. Haslinger, K. Kunisch, and G. Peichl, “Shape optimization and fictitious domain approach for solving free boundary problems of bernoulli type,” *Comput. Optim. Appl.*, vol. 26, no. 3, pp. 231–251, 2003.

[31] K. Eppler and H. Harbrecht, “Efficient treatment of stationary free boundary problems,” *Appl. Numer. Math.*, vol. 56, no. 10-11, pp. 1326–1339, 2006.

[32] A. Bogomolny and J. Hou, “Shape optimization approach to numerical solution of the obstacle problem,” *Appl. Math. Optim.*, vol. 12, no. 1, pp. 45–72, 1984.

[33] R. Rangarajan and A. Lew, “Universal meshes: A method for triangulating planar curved domains immersed in nonconforming meshes,” *Internat. J. Numer. Methods Engrg.*, vol. 98, no. 4, pp. 236–264, 2014.

[34] R. Rangarajan and A. Lew, “Analysis of a method to parameterize planar curves immersed in triangulations,” *SIAM J. Numer. Anal.*, vol. 51, no. 3, pp. 1392–1420, 2013.

[35] R. Rangarajan and A. Lew, “Provably robust directional vertex relaxation for geometric mesh optimization,” *SIAM J. Sci. Comput.*, vol. 39, no. 6, pp. A2438–A2471, 2017.

[36] K. Majava and X. Tai, “A level set method for solving free boundary problems associated with obstacles,” *Int. J. Numer. Anal. Model*, vol. 1, no. 2, pp. 157–171, 2004.

[37] M. Wang, X. Wang, and D. Guo, “A level set method for structural topology optimization,” *Comput. Methods Appl. Mech. Engrg.*, vol. 192, no. 1-2, pp. 227–246, 2003.

[38] G. Allaire, C. Dapogny, and P. Frey, “Shape optimization with a level set based mesh evolution method,” *Comput. Methods Appl. Mech. Engrg.*, vol. 282, pp. 22–53, 2014.

[39] O. Pironneau, *Optimal shape design for elliptic systems.* Springer Science & Business Media, 2012.

[40] J. Haslinger, I. Hlaváček, and J. Nečas, “Numerical methods for unilateral problems in solid mechanics,” *Handbook of numerical analysis*, vol. 4, pp. 313–485, 1996.

[41] E. Fancello, J. Haslinger, and R. Feijoo, “Numerical comparison between two cost functions in contact shape optimization,” *Structural Optimization*, vol. 9, no. 1, pp. 57–68, 1995.

[42] P. Fulmański, A. Laurain, J. Scheid, and J. Sokołowski, “A level set method in shape and topology optimization for variational inequalities,” *Int. J. Appl. Math. Comput. Sci.*, vol. 17, no. 3, pp. 413–430, 2007.

[43] U. Seifert, “Configurations of fluid membranes and vesicles,” *Advances in physics*, vol. 46, no. 1, pp. 13–137, 1997.

[44] S. Dasgupta, T. Auth, and G. Gompper, “Nano-and microparticles at fluid and biological interfaces,” *Journal of Physics: Condensed Matter*, vol. 29, no. 37, p. 373003, 2017.

[45] S. Zhang, J. Li, G. Lykotrafitis, G. Bao, and S. Suresh, “Size-dependent endocytosis of nanoparticles,” *Advanced Materials*, vol. 21, no. 4, pp. 419–424, 2009.

[46] C. Huang, Y. Zhang, H. Yuan, H. Gao, and S. Zhang, “Role of nanoparticle geometry in endocytosis: laying down to stand up,” *Nano letters*, vol. 13, no. 9, pp. 4546–4550, 2013.

[47] S. Dasgupta, T. Auth, and G. Gompper, “Wrapping of ellipsoidal nano-particles by fluid membranes,” *Soft Matter*, vol. 9, no. 22, pp. 5473–5482, 2013.

[48] H. Gao, W. Shi, and L. Freund, “Mechanics of receptor-mediated endocytosis,” *Proc. Natl. Acad. Sci.*, vol. 102, no. 27, pp. 9469–9474, 2005.

[49] W. Helfrich, “Elastic properties of lipid bilayers: theory and possible experiments,” *Zeitschrift für Naturforschung C*, vol. 28, no. 11-12, pp. 693–703, 1973.

[50] W. Shi, X. Feng, and H. Gao, “Two-dimensional model of vesicle adhesion on curved substrates,” *Acta Mechanica Sinica*, vol. 22, no. 6, pp. 529–535, 2006.

[51] R. Rangarajan and H. Gao, “A finite element method to compute three-dimensional equilibrium configurations of fluid membranes: Optimal parameterization, variational formulation and applications,” *J. Comput. Phy.*, vol. 297, pp. 266–294, 2015.

[52] G. Carey, S. Chow, and M. Seager, “Approximate boundary-flux calculations,” *Comput. Methods Appl. Mech. Engrg.*, vol. 50, no. 2, pp. 107–120, 1985.

[53] J. Nocedal and S. Wright, *Numerical Optimization.* Springer Series in Operations Research and Financial Engineering, Springer New York, 2006.

[54] M. Arroyo and M. Ortiz, “Local maximum-entropy approximation schemes: a seamless bridge between finite elements and meshfree methods,” *Internat. J. Numer. Methods Engrg.*, vol. 65, no. 13, pp. 2167–2202, 2006.

[55] M. Arroyo and M. Ortiz, "Local maximum-entropy approximation schemes," in *Meshfree methods for partial differential equations III*, pp. 1–16, Springer, 2007.

[56] C. Dapogny and P. Frey, "Computation of the signed distance function to a discrete contour on adapted triangulation," *Calcolo*, vol. 49, no. 3, pp. 193–219, 2012.

[57] F. Calakli and G. Taubin, "Ssd: Smooth signed distance surface reconstruction," in *Computer Graphics Forum*, vol. 30, pp. 1993–2002, Wiley Online Library, 2011.

[58] J. Sokolowski and J. Zolesio, "Introduction to shape optimization," in *Introduction to Shape Optimization*, pp. 5–12, Springer, 1992.

[59] J. Conway, *Functions of One Complex Variable I.* Springer, 1978.

[60] G. Carey, "Some further properties of the superconvergent flux projection," *Internat. J. Numer. Methods Biomedical Engrg.*, vol. 18, no. 4, pp. 241–250, 2002.

[61] A. Pehlivanov, R. Lazarov, G. Carey, and S. Chow, "Superconvergence analysis of approximate boundary-flux calculations," *Numer. Math.*, vol. 63, no. 1, pp. 483–501, 1992.

[62] K. Hollig, *Finite element methods with B-splines*, vol. 26. SIAM, 2003.

[63] J. Shewchuk, "Triangle: Engineering a 2d quality mesh generator and delaunay triangulator," in *Applied computational geometry towards geometric engineering*, pp. 203–222, Springer, 1996.

[64] R. Rangarajan and A. Lew, "Parameterization of planar curves immersed in triangulations with application to finite elements," *Internat. J. Numer. Methods Engrg.*, vol. 88, no. 6, pp. 556–585, 2011.

[65] R. Rangarajan, "On the resolution of certain discrete univariate max–min problems," *Comput. Optim. Appl.*, vol. 68, no. 1, pp. 163–192, 2017.

[66] R. Rangarajan, H. Kabaria, and A. Lew, "An algorithm for triangulating smooth three-dimensional domains immersed in universal meshes," *Internat. J. Numer. Methods Engrg.*, vol. in press, 2018.

[67] M. Bern, D. Eppstein, and J. Gilbert, "Provably good mesh generation," *Journal of computer and system sciences*, vol. 48, no. 3, pp. 384–409, 1994.

[68] S. Benson and T. Munson, "Flexible complementarity solvers for large-scale applications," *Optim. Methods Softw.*, vol. 21, no. 1, pp. 155–168, 2006.

[69] B. S and et al, "PETSc Web page." `http://www.mcs.anl.gov/petsc`, 2018.

[70] Y. Zhang, “Convergence of free boundaries in discrete obstacle problems,” *Numer. Math.*, vol. 106, no. 1, pp. 157–164, 2007.

[71] D. Braess, C. Carstensen, and R. Hoppe, “Convergence analysis of a conforming adaptive finite element method for an obstacle problem,” *Numer. Math.*, vol. 107, no. 3, pp. 455–471, 2007.

[72] R. Donaldson and B. Wetton, “Solving steady interface problems using residual velocities,” *IMA J. Appl. Math.*, vol. 71, no. 6, pp. 877–897, 2006.

[73] M. Delfour and J. Zolesio, *Shapes and geometries: metrics, analysis, differential calculus, and optimization*, vol. 22. SIAM, 2011.

[74] J. Haslinger, *Introduction to shape optimization: theory, approximation, and computation*, vol. 7. SIAM, 2003.