Journal of Computational Physics **163,** 489–528 (2000)
doi:10.1006/jcph.2000.6581, available online at http://www.idealibrary.com on IDEAL®

Structural Boundary Design via Level Set and Immersed Interface Methods[1]

J. A. Sethian and Andreas Wiegmann

Departments of Mathematics, University of California at Berkeley and Lawrence Berkeley National Laboratory, Berkeley, California 94720
E-mail: wiegmann@math.berkeley.edu, sethian@math.berkeley.edu

Received November 30, 1999; revised June 29, 2000

We develop and test an algorithmic approach to the boundary design of elastic structures. The goal of our approach is two-fold: first, to develop a method which allows one to rapidly solve the two-dimensional Lamé equations in arbitrary domains and compute, for example, the stresses, and second, to develop a systematic way of modifying the design to optimize chosen properties. At the core, our approach relies on two distinct steps. Given a design, we first apply an explicit jump immersed interface method to compute the stresses for a given design shape. We then use a narrow band level set method to perturb this shape and progress towards an improved design. The equations of 2D linear elastostatics in the displacement formulation on arbitrary domains are solved quickly by domain embedding and the use of fast elastostatic solvers. This effectively reduces the dimensionality of the problem by one. Once the stresses are found, the level set method, which represents the design structure through an embedded implicit function, is used in the second step to alter the shape, with velocities depending on the stresses in the current design. Criteria are provided for advancing the shape in an appropriate direction and for correcting the evolving shape when given constraints are violated. © 2000 Academic Press

Key Words: self design of elastic structures; level set method; explicit jump immersed interface method; immersed interface method; linear elastostatics.

1. INTRODUCTION

The goal of this paper is to advance methodology for computing linear elastostatics in complex geometries and to develop techniques for computing improved designs under user-supplied constraints. We present a combined level set and finite difference technique

[1] This work was supported by the Office of Science, Office of Advances Scientific Computing Research, Mathematical, Information and Computational Science Division, Applied Mathematical Sciences Subprogram of the U.S. Department of Energy under Contract DE-AC03-76SF00098.

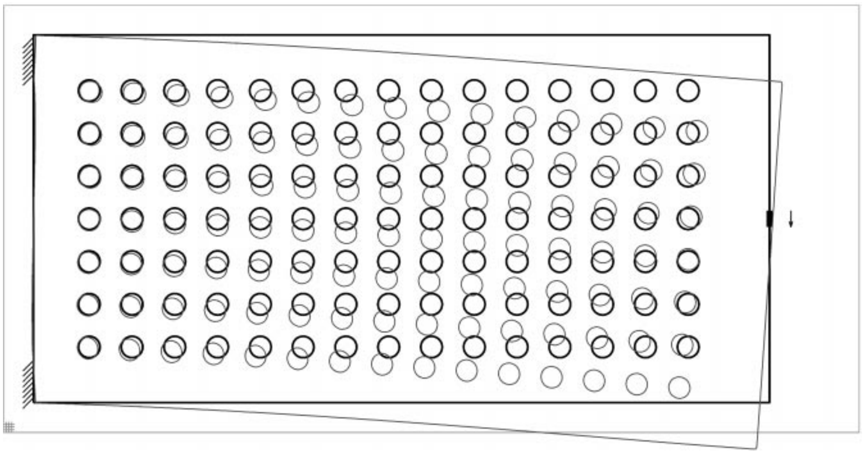


FIG. 1. Bending of the initial design of a cantilever with 105 circular holes. Parts of the left boundary are clamped; on the rest of the boundary, including all holes, the traction is specified with nonzero loading on a small portion about the center of the right boundary. The bending is beyond the regime of small displacement elastostatics and chosen only to illustrate the behavior. The larger rectangle is the computational domain with a 320×160 grid indicated in the lower left corner.

for constructing efficient designs which satisfy certain design criteria. The explicit jump immersed interface method is used to compute the solution of the elliptic problem in complex geometries, and the narrow band level set method is used to track the motion of the design boundaries under complex speed functions in the presence of topological changes. The application setting for these algorithms is the boundary design of a loaded elastic structure, with short structural boundary design. The design changes are based on the weight and stresses, and boundaries are moved, removed, or introduced based on these quantities.

By way of illustration, consider a clamped and loaded cantilever (see Fig. 1). Suppose our goal is to remove as much material as possible from the original shape, while still making sure that the compliance (defined as the yield under the load) or the maximal stress in the structure stays below a certain threshold value. We can start with the original perforated structure, compute the stress, and then try to add and remove material in order to reduce the weight in such a way that the compliance or stress does not rise above a given user-prescribed level (see Fig. 2). Different designs (that is, newly introduced, removed, or reshaped holes) will give different compliance and stresses in the design. Our approach is to devise a systematic way to add and remove material. This requires an accurate technique to compute the stresses for a given multiply connected domain and an accurate technique to remove or reshape existing boundaries and to introduce new ones. The level set method is instrumental in the addition and subtraction of material; the explicit jump immersed interface method is the key to computing the stress in arbitrary domains. The improved cantilever that is designed by this procedure looks like Fig. 35.

As a general outline, the algorithmic approach presented in this paper is as follows. In the first step, the explicit jump immersed interface method is applied to the equations of 2D linear elastostatics in the displacement formulation (from now on referred to as the Lamé equations), and problems on arbitrary domains are solved quickly and without mesh generation by domain embedding and the use of fast elastostatic solvers. This effectively reduces the dimensionality of the problem by one. In the second step, the given design is modified. The level set method, which represents the design structure through an embedded

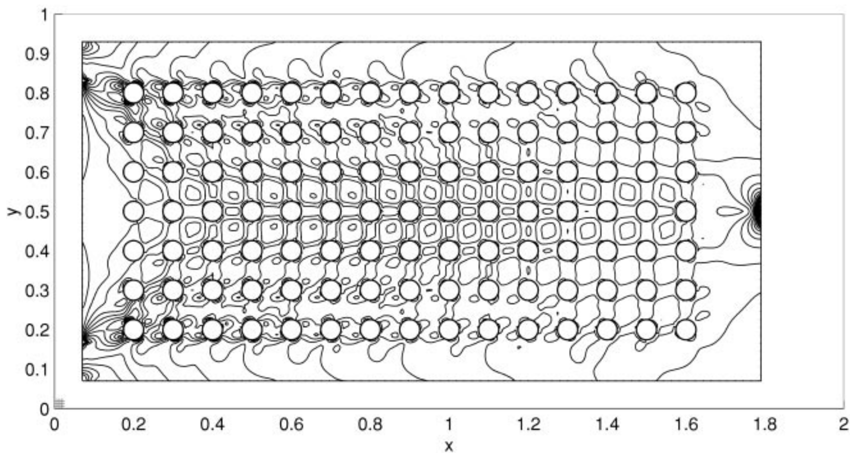


FIG. 2. Stress contours on the design from Fig. 1.

implicit function, is used to alter the shape, with velocities depending on the stresses in the current design. These stresses can be found from the displacements that were found in the first step. Criteria are provided for advancing the shape in an appropriate direction and for correcting the evolving shape when given constraints are violated. The two steps are iterated until no further improvement can be made.

The benefit of using the explicit jump immersed interface method is to avoid mesh generation and the use of fast elastostatic solvers; the benefit of using a level set method is the elegant handling of changes in topology such as the merging of holes or the filling in of the or holes with large stresses around them.

The application of the algorithmic methodology presented in this paper is by no means complete. Our techniques allow accurate and efficient computation of the linear elastic equations for arbitrary geometry and provide an approach to alter the design shape in a systematic way to satisfy user-prescribed constraints. As such, the application falls into the category of constrained minimization problems. As with many such problems, our technique allows us to improve the objective, but cannot be guaranteed to find the global minimum. In particular, our criteria for perturbing the given design and searching for nearby trial shapes which both satisfy the constraint and contain less material is somewhat ad hoc; we can neither prove that it converges nor guarantee that it extracts the global minimum. Nonetheless, it provides an appealing way to approach some of these problems.

2. BACKGROUND ON STRUCTURAL BOUNDARY DESIGN AND PREVIOUS WORK

One main approach to structural design for variable topologies is the method of *homogenization* [5, 6, 8]. The following summary is paraphrased from [3]: Homogenization extracts homogeneous effective parameters from heterogeneous media. The effective properties of a composite material are defined as the homogenized coefficients of a fine mixture of given phases. The difficult "layout" problem of material distribution is replaced by a much easier "sizing" problem for the density and effective properties of a perforated composite material obtained by cutting small holes in the original homogeneous material. Work in homogenization has shown that at low volume fractions the optimal solutions for perforated plates in plane stress and bending, respectively, tend to those of least-weight trusses and grillages (see [4, 7]).

From our point of view, the most important insight obtained by homogenization is that some objectives in the mathematical modeling of structural design are oversimplified and do not yield the intended results. The objective "to minimize a weighted sum of compliance and weight for a given load" results in designs with infinitesimal specially shaped pores in the material that make the structure not manufacturable and very sensitive to variations in the loading. In practical situations, one would like the material to be composed of macroscopic solid and void regions, to allow for variations in the loading, to account for cost of manufacturing, etc. Homogenization can deal with these issues by penalization of intermediate densities and postprocessing, but the attractive conceptual simplicity of the homogenization approach is lost for realistic requirements.

In the "evolutionary structural optimization" approach of Xie and Steven, [37], changes in the size, shape, and topology of the structure are achieved by removing elements (usually refered to as *hard kill*) from some fixed finite element discretization of a "ground structure," an initial large domain in which the final designed domain is *a priori* known to be contained. The weight reduction can be governed by the stress, stiffness, frequency, or buckling load. In the similar "reverse adaptivity" approach of Reynolds *et al.* [21], approximately fully stressed structures are found by removing a fixed percentage of relatively understressed material. Reverse adaptivity refines finite element meshes near the boundary during the design procedure to reduce computational cost or increase resolution. Essentially, both evolutionary structural optimization and reverse adaptivity are homotopy methods, the difference (apart from the adaptive mesh refinement in reverse adaptivity) is that the parameter is a percentage of stress or a percentage of weight.

Another related approach is the "bubble method" of Eschenauer *et al.* [12, 13], where the topology is modified in a prescribed fashion by placing holes of known shape at optimal positions in the structure, based on so-called characteristic functions of the stresses, strains, and displacements. From this method we have gathered the importance of letting the design for a given topology "settle into a good shape" before further changing the topology.

Sharp boundary-based methods for structural design are a more direct approach than homogenization. For example, they do not require the reformulation of the constrained problem via Lagrange multipliers, and in general they allow the modeler more explicitly to account for any features she wants to incorporate into the design. The possible complications make the notion of proving the optimality of the design hopeless, but it may be argued (as, for example, in [21]) that for many applications, optimality is not as useful a concept as improvement, and we will thus only make claims for our own method regarding the latter. As in [21], "we improve the design by making more efficient use of the material."

3. OVERVIEW OF WORK AND ALGORITHMIC APPROACH

Adopting the principal idea of redesigning the structure based on the distribution of stresses in the current design to find fully stressed structures, we focus on the resolution of the boundaries. The level set method [19], introduced by Osher and Sethian, is designed for moving boundaries with changes in topology, and relies in part on the theory and numerics of curve evolution developed by Sethian in [24–26]. In our case, the velocity of this boundary motion depends on the stress on the boundary. On arbitrary domains we obtain the stresses by solving the linear elastostatic equations in the displacement formulation and differencing the displacements using the explicit jump immersed interface method (see [36]), a finite difference technique on uniform grids after Li and LeVeque's immersed interface method (IIM, [14]) that is capable of dealing with non-grid-aligned boundaries with the same truncation error as interior differences. The biggest benefit of our approach compared to the earlier work [21, 37] is that it is easier to add material (with some subgrid resolution) at hole boundaries with high stress than on a triangulated finite element mesh. In particular, this allows us to start with designs that have holes cut "in the wrong place," and see these holes disappear.

In the elliptic portion of structural boundary design, mesh generation for the design domains can become the major cost. We avoid the mesh generation step by separating the representation of the boundary from the uniform computational grid. To keep the data structures simple and in order to use fast elastostatic solvers on rectangular domains [34], the problem is posed on a larger, rectangular domain R with zero normal boundary conditions. The boundary conditions on the original boundary are rewritten as jump conditions that introduce discontinuities in the displacements inside R. Our choice of jump and boundary conditions forces the extended solution to vanish on the extension, but to match the solution inside the structure. On the level of linear algebra, a Schur complement (as previously used, e.g., in [15, 16, 36]) reduces the number of variables from proportional to the grid points to proportional to the length of the boundary normalized by the mesh width.

In the design portion of structural boundary design, changes in topology provide the greatest challenge. The structure boundaries are viewed as moving; holes may merge with each other or the exterior or may have to be newly created. The level set method represents the boundaries implicitly as the zero level curve of a grid function that is essentially the distance from the boundary. Boundary motion and merging, as well as the introduction of new holes, are all performed using this grid function. This approach also allows the detection of regions that have become separated from the nontrivial boundary conditions and have to be dropped from the computations. For efficiency reasons, we use the narrow band level set method (see [1]).

We considered two options for the boundary velocity. The first is derived from the stress on the boundaries and inside the structure. The second is based on the stress on the boundaries only, and then extends the velocity onto the grid by constant values in the normal direction. We use the second option because the first does not allow for addition of material and because in our experience tended to give undesirable corners in the boundary geometry. In addition, the second option theoretically maintains a distance function from the zero contour.

In more detail, the elliptic aspects of structural boundary design are considered in Section 4 while the design aspects are treated in Section 5. To compute the stresses, we *derive jump conditions* from given displacement boundary conditions in Section 4.1, and from given traction boundary conditions in Section 4.2. The separate treatment of boundary conditions serves only to illustrate the concepts; in the numerical examples in Section 6, different types of boundary conditions are given on the same component of a boundary. We use a weighted least squares approach for derivative estimation (see Section 4.3) which is similar to ideas in Li [15] and improve and implement the corrections for cross-derivatives (previously described in [36]) in Section 4.4. Derivative estimation and corrections are for the first time carried out to third order for the purpose of achieving an $\mathcal{O}(h^2)$ truncation error at all points, including points neighboring the boundary. Section 4.5 briefly describes the Schur-complement approach for embedded irregular domains and Section 4.6 summarizes the ideas behind the fast elastostatic solver.

In Section 5, we review the narrow band level set method for front propagation in Section 5.1 before going into its extensions (e.g., determination of the boundary velocity) to structural boundary design in Section 5.2. Some remarks concerning the explicit jump immersed interface method for varying geometries follow in Section 5.3.

We demonstrate second order convergence of the explicit jump immersed interface method for elastostatics in Section 6.1. The third order approximations improve the solutions to be less sensitive to changes in the boundary geometry. Second order convergence of the displacements gives first order convergence of the stresses. These are again extrapolated to the boundaries using a weighted least squares fit in order to find the front velocities as required by the level set method.

In Section 6.2, we show that we can solve elastostatic problems in irregular regions with the same $\mathcal{O}(N \log N)$ efficiency that the fast solver exhibits for rectangular domains, and that it is feasible to solve medium-sized problems of up to 1.6 million variables on workstations.

In the examples of structural boundary design in Section 6.3, the idea is to reduce the weight of the structure while keeping the compliance below a certain bound and achieving as uniform a stress distribution as possible in the structure, i.e., to find so called *fully stressed structures*. We can start from a ground structure like [21, 37], but any existing design may be improved—or found to be unimprovable—with our approach. Three mechanisms allow changes in the size, shape and topology of the structure. We introduce new holes by cutting away material along contour lines of the von Mises stress. A homotopy parameter called the removal rate [37]) guides the choice of contour levels. The removal rate is a percentage of the maximum von Mises stress in the original design and is slowly increased according to another parameter, the evolutionary rate, to remove material from the structure. We then move the boundaries using a level set method with velocities given from the stress on the boundaries and by extending these velocities from the boundaries by a constant value normal to the boundaries, as suggested in [2]. The procedure stops when it cannot further decrease the weight of the structure without "constraint violation." Alternatively, an observer may want to choose the best design out of the sequence of designs.

4. FINITE DIFFERENCES FOR THE LAMÉ EQUATIONS IN GENERAL DOMAINS

Recall the two-dimensional Lamé equations: $\mathbf{u} = (u, v)$ are the displacements in x and y, respectively, and

$$-\frac{E}{2(1+\nu)}(\Delta u + u_{xx} + v_{xy}) - \frac{\nu E}{((1+\nu)(1-2\nu)}(u_{xx} + v_{xy}) = f^u \quad \text{in } \Omega,$$
$$-\frac{E}{2(1+\nu)}(\Delta v + u_{xy} + v_{yy}) - \frac{\nu E}{((1+\nu)(1-2\nu)}(u_{xy} + v_{yy}) = f^v \quad \text{in } \Omega.$$

Here E is the Young modulus, ν is the Poisson ratio, $\mu = E/(2+2\nu)$ and $\lambda = \nu E/((1+\nu)(1-2\nu)$ are the Lamé constants, $\mathbf{f} = (f^u, f^v)$ are body forces, and Ω is an open, connected, but not necessarily simply connected domain. We will also write (with $C = \mu/(\mu+\lambda)$)

$$C\Delta u + u_{xx} + v_{xy} = -\frac{f^u}{\mu+\lambda} \quad \text{in } \Omega, \tag{1}$$

$$C\Delta v + u_{xy} + v_{yy} = -\frac{f^v}{\mu+\lambda} \quad \text{in } \Omega. \tag{2}$$

Displacement boundary conditions are

$$\mathbf{u} = \bar{\mathbf{u}} \quad \text{on } \Gamma_1 \subset \partial\Omega. \tag{3}$$

Here $\bar{\mathbf{u}} = (\bar{u}, \bar{v})^T$ are given functions on Γ_1, the part of $\partial\Omega$, the boundary of Ω, where displacements are given. Traction boundary conditions are

$$\sigma(\mathbf{u})\mathbf{n} = \mathbf{g} \quad \text{on } \Gamma_2 \subset \partial\Omega. \tag{4}$$

We assume that the coefficients, geometry, and boundary values are such that the problem (1)–(4) has a unique solution.

Throughout, we consider second order centered finite difference discretizations for all occurring derivatives. The explicit jump immersed interface method is used to treat irregular boundaries. For details of this discretization of an elliptic equation we refer the reader to [36].

We embed Ω so that its closure is contained in the rectangle R and extend u and v by zero on the open complement of Ω in R, $R\backslash\text{cl}\{\Omega\}$. We will denote this open complement by Ω^+ and write Ω^- for the original domain Ω. The extended functions u and v are in general discontinuous across $\partial\Omega$, with nontrivial jumps in all derivatives that are determined by the boundary conditions and derived below. If q is a function on R and α is a point on $\partial\Omega^-$, the jump of q at α is

$$[q](\alpha) = \lim_{x^+ \to \alpha \text{ in } \Omega^+} q(x^+) - \lim_{x^- \to \alpha \text{ in } \Omega^-} q(x^-).$$

The explicit jump immersed interface method requires a complete set of independent jump conditions for all variables and their derivatives in the coordinate directions below the order of the discretization to achieve the same convergence rate that is obtained on rectangles.

4.1. Displacement Boundary Conditions

The jumps in displacements do not involve the differential geometry of the boundary, and can be written out immediately for Cartesian coordinates.

An extension of u and v by zero on $R\backslash\text{cl}\{\Omega\}$, where $R = [a, b] \times [c, d]$, and defining

$$\tilde{f}^u = \begin{cases} -f^u/(\mu + \lambda) & \text{in } \Omega \\ 0 & \text{in } R\backslash\text{cl}\{\Omega\} \end{cases}$$

$$\tilde{f}^v = \begin{cases} -f^v/(\mu + \lambda) & \text{in } \Omega \\ 0 & \text{in } R\backslash\text{cl}\{\Omega\} \end{cases}$$

yields the differential equation on the rectangle R that the explicit jump immersed interface method works on,

$$C\Delta u + u_{xx} + v_{xy} = \tilde{f}^u \quad \text{in } R\backslash\partial\Omega,$$
$$C\Delta v + u_{xy} + v_{yy} = \tilde{f}^v \quad \text{in } R\backslash\partial\Omega$$

with boundary conditions

$$v_x = u = 0 \quad \text{on } \Gamma_3 = \{x \in \{a, b\}, y \in [c, d]\} \subset \partial R, \tag{5}$$

$$u_x = v = 0 \quad \text{on } \Gamma_3 = \{y \in \{c, d\}, x \in [a, b]\} \subset \partial R. \tag{6}$$

The fact that u and v satisfy the same equation in Ω^+ as in Ω^- and the choice of these particular "normal" boundary conditions on ∂R is essential for the use of a fast solver on R.

Across Γ_1, the jump conditions are

$$[u] = -\bar{u} \quad \text{on } \Gamma_1, \tag{7}$$

$$[v] = -\bar{v} \quad \text{on } \Gamma_1, \tag{8}$$

$$\left[\frac{\partial^m u}{\partial x^i \partial y^{(m-i)}}\right] = -\frac{\partial^m u^-}{\partial x^i \partial y^{(m-i)}}, \quad i = 0, 1, \ldots, m;\ m = 1, 2, \ldots \tag{9}$$

$$\left[\frac{\partial^m v}{\partial x^i \partial y^{(m-i)}}\right] = -\frac{\partial^m v^-}{\partial x^i \partial y^{(m-i)}}, \quad i = 0, 1, \ldots, m;\ m = 1, 2, \ldots. \tag{10}$$

Alternatively, the equation and approximations of tangential derivatives may be used to derive jump conditions for second and higher derivatives as suggested in [36] and previously in [15].

THEOREM 1. *When $\Gamma_1 = \partial\Omega$, the extension of u and v by zero is the unique solution of the extended problem on R, where u and v satisfy the normal boundary conditions* (5) *and* (6) *on ∂R, the elastostatic equations* (1) *and* (2) *in Ω^- and in Ω^+, and the jump conditions* (7)–(10). *In other words, the restriction to Ω^- of the solution of the extended problem solves the original boundary value problem* (1)–(3).

Proof. By design, the extended functions u and v satisfy the boundary conditions on R, the jump conditions on $\partial\Omega$ and the differential equations in Ω^- and Ω^+. If any u and v satisfy (9) and (10), this implies $u_x^+ = u_y^+ = v_x^+ = v_y^+ = \cdots = v_{yy}^+ = 0$. This, together with the facts that u and v satisfy (5) and (6) on ∂R and satisfy the elastostatic equations in Ω^+ guarantee that u and v vanish on Ω^+. But then $u^+ = v^+ = 0$. Together with this, (7) and (8) imply that $u^- = \bar{u}$ and $v^- = \bar{v}$. But then the uniqueness of u and v on Ω^- is exactly the assumed uniqueness of the solution of the original boundary value problem. ■

4.2. Traction Boundary Conditions

Different from displacement boundary conditions, traction boundary conditions are more conveniently expressed in the local coordinates of the boundary, i.e., derivatives in the tangent and normal directions. The traction boundary condition couples the two Cartesian displacement variables.

4.2.1. *Jumps in Local Coordinates*

Again the goal in this section is to get a complete set of jump conditions from the traction boundary conditions upon extending the solution by zero.

For concreteness, we may think of (4) as realized in Cartesian coordinates. Then $\mathbf{u} = (u, v)$ is the vector of displacements in the x and y directions, σ is the stress tensor expressed in (x, y) coordinates, $\mathbf{n} = (n_1, n_2)$ is the inward normal to the boundary (given in (x, y) coordinates), and $\mathbf{g}$ is a vector of surface forces applied at that boundary, also given in (x, y) coordinates.

We indicate the tangent to the boundary by $\mathbf{t} = (n_2, -n_1)$ so that $(\mathbf{t}, \mathbf{n})$ form a right-hand system. The displacements in this coordinate system are $\xi = \mathbf{u} \cdot \mathbf{n}$ and $\eta = \mathbf{u} \cdot \mathbf{t}$. We think of gradients as row vectors, so $\nabla\mathbf{u} = \left(\begin{smallmatrix} u_x & u_y \\ v_x & v_y \end{smallmatrix}\right)$, and use the notation

$$\xi_n = \nabla(\mathbf{u}\cdot\mathbf{n})\mathbf{n} = \mathbf{n}^T(\nabla\mathbf{u})\mathbf{n}, \tag{11}$$

$$\eta_n = \nabla(\mathbf{u}\cdot\mathbf{t})\mathbf{n} = \mathbf{n}^T(\nabla\mathbf{u})\mathbf{t}, \tag{12}$$

$$\xi_t = \nabla(\mathbf{u}\cdot\mathbf{n})\mathbf{t} = \mathbf{t}^T(\nabla\mathbf{u})\mathbf{n}, \tag{13}$$

$$\eta_t = \nabla(\mathbf{u}\cdot\mathbf{t})\mathbf{t} = \mathbf{t}^T(\nabla\mathbf{u})\mathbf{t}. \tag{14}$$

We rewrite (4) in these *local coordinates* that are implied by the geometry of the boundary and change with the boundary

$$\left(\frac{E}{2(1+\nu)}\begin{pmatrix} 2\xi_n & \xi_t+\eta_n \\ \xi_t+\eta_n & 2\eta_t \end{pmatrix} + \frac{E\nu}{(1+\nu)(1-2\nu)}\begin{pmatrix} \xi_n+\eta_t & 0 \\ 0 & \xi_n+\eta_t \end{pmatrix}\right)\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}\mathbf{g}\cdot\mathbf{n}\\ \mathbf{g}\cdot\mathbf{t}\end{pmatrix}.$$

From this we derive formulas for the normal derivatives of the displacements in terms of tangential derivatives.

$$\xi_n = -\tilde{g}^{\xi} - \frac{\nu}{1-\nu}\eta_t, \tag{15}$$

$$\eta_n = -\tilde{g}^{\eta} - \xi_t, \tag{16}$$

where

$$\tilde{g}^{\xi} = -\frac{(1+\nu)(1-2\nu)\mathbf{g}\cdot\mathbf{n}}{E(1-\nu)},$$

$$\tilde{g}^{\eta} = -\frac{2(1+\nu)\mathbf{g}\cdot\mathbf{t}}{E}.$$

Now we have a complete set of jump conditions, albeit some not yet in Cartesian coordinates.

$$[u] = -u^- \quad \text{on } \Gamma_2, \tag{17}$$

$$[v] = -v^- \quad \text{on } \Gamma_2, \tag{18}$$

$$[\xi_n] = -\xi_n^- = \tilde{g}^{\xi} + \frac{\nu}{1-\nu}\eta_t^- \quad \text{on } \Gamma_2, \tag{19}$$

$$[\eta_n] = -\eta_n^- = \tilde{g}^{\eta} + \xi_t^- \quad \text{on } \Gamma_2, \tag{20}$$

$$[\xi_t] = -\xi_t^- \quad \text{on } \Gamma_2, \tag{21}$$

$$[\eta_t] = -\eta_t^- \quad \text{on } \Gamma_2. \tag{22}$$

The second- and higher-order jumps $[u_{xx}], [u_{xy}], \ldots, [v_{yyy}]$ are the same as in the displacement case, (9) and (10) for $m \geq 2$. The derivations in Section 4.2.2 bring (19)–(22) into the form required by the explicit jump immersed interface method.

4.2.2. *Jumps in Cartesian Coordinates*

Equations (11)–(14) can be used for coordinate transformations of jumps,

$$\begin{pmatrix}\mathbf{n}^T\\ \mathbf{t}^T\end{pmatrix}\begin{pmatrix}[u_x] & [u_y]\\ [v_x] & [v_y]\end{pmatrix}(\mathbf{n} \quad \mathbf{t}) = \begin{pmatrix}[\xi_n] & [\eta_n]\\ [\xi_t] & [\eta_t]\end{pmatrix},$$

or, using the orthonormality of $(\mathbf{n}, \mathbf{t})$,

$$\begin{pmatrix} [u_x] & [u_y] \\ [v_x] & [v_y] \end{pmatrix} = (\mathbf{n} \quad \mathbf{t}) \begin{pmatrix} [\xi_n] & [\eta_n] \\ [\xi_t] & [\eta_t] \end{pmatrix} \begin{pmatrix} \mathbf{n}^T \\ \mathbf{t}^T \end{pmatrix}.$$

Observe that

$$\begin{pmatrix} [\xi_n] & [\eta_n] \\ [\xi_t] & [\eta_t] \end{pmatrix} = \begin{pmatrix} \tilde{g}^\xi & \tilde{g}^\eta \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{\nu}{1-\nu}\eta_t^- & \xi_t^- \\ -\xi_t^- & -\eta_t^- \end{pmatrix}$$

and get

$$[u_x] = (n_1 \quad t_1)\left(\begin{pmatrix} \tilde{g}^\xi & \tilde{g}^\eta \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{\nu}{1-\nu}\eta_t^- & \xi_t^- \\ -\xi_t^- & -\eta_t^- \end{pmatrix}\right)\begin{pmatrix} n_1 \\ t_1 \end{pmatrix},$$

$$[u_y] = (n_1 \quad t_1)\left(\begin{pmatrix} \tilde{g}^\xi & \tilde{g}^\eta \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{\nu}{1-\nu}\eta_t^- & \xi_t^- \\ -\xi_t^- & -\eta_t^- \end{pmatrix}\right)\begin{pmatrix} n_2 \\ t_2 \end{pmatrix},$$

$$[v_x] = (n_2 \quad t_2)\left(\begin{pmatrix} \tilde{g}^\xi & \tilde{g}^\eta \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{\nu}{1-\nu}\eta_t^- & \xi_t^- \\ -\xi_t^- & -\eta_t^- \end{pmatrix}\right)\begin{pmatrix} n_1 \\ t_1 \end{pmatrix},$$

$$[v_y] = (n_2 \quad t_2)\left(\begin{pmatrix} \tilde{g}^\xi & \tilde{g}^\eta \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{\nu}{1-\nu}\eta_t^- & \xi_t^- \\ -\xi_t^- & -\eta_t^- \end{pmatrix}\right)\begin{pmatrix} n_2 \\ t_2 \end{pmatrix}.$$

Putting it all together,

$$[u_x] = \left(\frac{\nu}{1-\nu}n_1^2 - t_1^2\right)\eta_t^- + n_1^2\tilde{g}^\xi + n_1t_1\tilde{g}^\eta,$$

$$[u_y] = \left(\frac{\nu}{1-\nu}n_1n_2 - t_1t_2\right)\eta_t^- + (n_1t_2 - n_2t_1)\xi_t^- + n_1n_2\tilde{g}^\xi + n_1t_2\tilde{g}^\eta,$$

$$[v_x] = \left(\frac{\nu}{1-\nu}n_2n_1 - t_2t_1\right)\eta_t^- + (n_2t_1 - t_2n_1)\xi_t^- + n_2n_1\tilde{g}^\xi + n_2t_1\tilde{g}^\eta,$$

$$[v_y] = \left(\frac{\nu}{1-\nu}n_2^2 - t_2^2\right)\eta_t^- + n_2^2\tilde{g}^\xi + n_2t_2\tilde{g}^\eta,$$

where

$$\eta_t^- = (t_1, t_2)\begin{pmatrix} u_x^- & u_y^- \\ v_x^- & v_y^- \end{pmatrix}\begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = t_1^2u_x^- + t_1t_2u_y^- + t_2t_1v_x^- + t_2^2v_y^-,$$

$$\xi_t^- = (t_1, t_2)\begin{pmatrix} u_x^- & u_y^- \\ v_x^- & v_y^- \end{pmatrix}\begin{pmatrix} n_1 \\ n_2 \end{pmatrix} = t_1n_1u_x^- + t_1n_2u_y^- + t_2n_1v_x^- + t_2n_2v_y^-.$$

4.3. Weighted Least-Squares Grid-to-Interface Extrapolation of Function and Derivative Values

The least-squares fit idea was used to second order in [15] for interfaces in order to improve the stability to the immersed interface method in the presence of large jumps in the coefficient. Here, we describe it to third order and in the one-sided limit context of the explicit jump immersed interface method. It is a general method to impose boundary conditions and jump conditions on nongrid-aligned boundaries and interfaces that is easily programmed and easily extended to three space dimensions.

4.3.1. *Third-Order Weighted Least-Squares Fit*

Figure 3 shows an annulus embedded in a square. The annulus is not centered in the square in order to avoid symmetry in the grid effects in later considerations. The heavy dot marks an intersection of a boundary with the mesh and the circles mark grid points inside the domain within distance r from the boundary point that are used to extrapolate function and derivative values from the grid to the boundary. The arrow points in the outward normal direction to the boundary, denoted by $\mathbf{n} = (c, s)$. We denote the coordinates of the boundary–mesh intersection by (x_α, y_α) and a generic marked grid point by (x_i, y_j). Let $h_i = x_i - x_\alpha$ and $k_j = y_j - y_\alpha$. The restriction R selects the values U_R of a grid function U at the marked grid points. The bicubic polynomial p should ideally satisfy $u_{ij} = p(x_i, y_j)$, where $p(x_i, y_j) = p_0 + p_1 h_i + p_2 k_j + p_3 h_i^2 + p_4 h_i k_j + p5 k_j^2 + p_6 h_i^3 + p_7 h_i^2 k_j + p_8 h_i k_j^2 + p_9 k_j^3$, for each of the marked grid points. We will discuss this interpolation in more detail in Section 4.3.2. On $n_l > 10$ grid points, this linear system is overdetermined. We use a least-squares fit and introduce weights that allow better approximation closer to the extrapolation point. For example, one could use $w_{ij} = (1 + \cos(\pi d_{ij}/r))/2$ as in [15], where $0 \le d_{ij} = \sqrt{h_i^2 + k_j^2} \le r$, and r is the "radius of influence." For these weights, due to the use of one-sided stencils, we found that we need to use larger r than Li does in [15]. To be able to use a smaller radius of influence r, we adjusted the weights, $w_{ij} = 1/(1 + d_{ij}/h)$.

The weighted least squares problem for the coefficients of p is then

$$\min_p \sum_{l=1}^{n_l} w_{ij(l)}^2 \left(p\left(x_{i(l)}, y_{j(l)}\right) - u_{ij(l)}\right)^2,$$

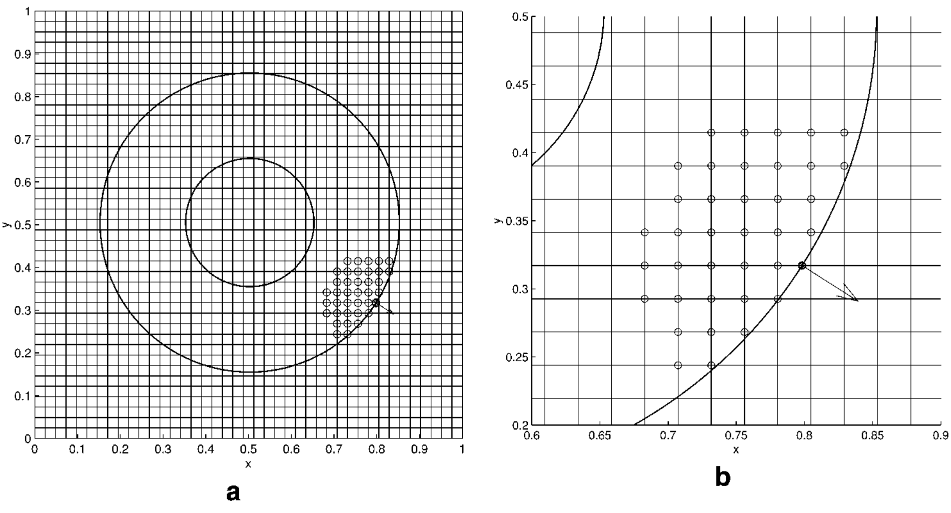


FIG. 3. In (a), we see an annulus embedded in a square and (b) shows a blowup of the same figure. The heavy dot marks an intersection of the annulus boundary with the mesh; the circles mark grid points inside the domain within distance r from the boundary point. The arrow points in the outward direction normal to the boundary.

where n_l is the number of marked points and the minimum is taken over all bicubic polynomials p. Letting[2] $W = \mathrm{diag}((w_{ij(l)})_{l=1}^{n_l})$ and $P = [p_0, p_1, \ldots, p_9]^T$ and using the $n_l \times 10$ matrix M with rows corresponding to grid points, i.e., the lth row

$$M_l = \left[1, h_{i(l)}, k_{j(l)}, h_{i(l)}^2, h_{i(l)}k_{j(l)}, k_{j(l)}^2, h_{i(l)}^3, h_{i(l)}^2 k_{j(l)}, h_{i(l)}k_{j(l)}^2, k_{j(l)}^3\right],$$

we find that for a given grid function U, the coefficients of the weighted least squares fit polynomial are given by $P = (M^T W^2 M)^{-1} M^T W^2 RU$. In the explicit jump immersed interface framework, the grid function U is not known but has to satisfy jump conditions. These can be expressed conveniently using rows of the matrix $D = S(M^T W^2 M)^{-1} M^T W^2 R$, where $S = \mathrm{diag}(1, 1, 1, 2, 1, 2, 6, 2, 2, 6)$. This is true because p was derived with origin (x_α, y_α), so function values and derivative values of U at the boundary point are approximated as follows:

$$\begin{aligned}
u(x_\alpha, y_\alpha) &= p_0 + \mathcal{O}(h^4),\\
u_x(x_\alpha, y_\alpha) &= p_1 + \mathcal{O}(h^3),\\
u_y(x_\alpha, y_\alpha) &= p_2 + \mathcal{O}(h^3),\\
u_{xx}(x_\alpha, y_\alpha) &= 2p_3 + \mathcal{O}(h^2),\\
u_{xy}(x_\alpha, y_\alpha) &= p_4 + \mathcal{O}(h^2),\\
u_{yy}(x_\alpha, y_\alpha) &= 2p_5 + \mathcal{O}(h^2),\\
u_{xxx}(x_\alpha, y_\alpha) &= 6p_6 + \mathcal{O}(h),\\
u_{xxy}(x_\alpha, y_\alpha) &= 2p_7 + \mathcal{O}(h),\\
u_{xyy}(x_\alpha, y_\alpha) &= 2p_8 + \mathcal{O}(h),\\
u_{yyy}(x_\alpha, y_\alpha) &= 6p_9 + \mathcal{O}(h).
\end{aligned}$$

For example, a condition on the directional derivative normal to the boundary can be written in terms of $cD_2 + sD_3$, where (c, s) is the unit normal and D_2 and D_3 are the second and third rows of D, respectively, which correspond to taking the x and y derivatives of U at (x_α, y_α).

4.3.2. *Remarks on Interpolation*

Lorentz [17] discusses analytic conditions under which quadratic interpolation on six points in 2D is possible; this is used in [36]. Similar limitations hold for third-order interpolation on 10 points. We found it easiest to deal with this issue numerically, independently of the number of points in the stencil. Whenever $M^T W^2 M$ is singular or close to singular, we know that the geometry results in a bad stencil and revert to a second-order weighted least-squares fit. If that fails also, we revert to first order. Since derivatives are needed to enforce traction boundary conditions, first order is the minimum that we need to be able to use. If $M^T W^2 M$ is singular even in that case, then a finer mesh is needed to resolve the geometry.

4.4. Corrections for Laplacian and Cross Derivatives to $\mathcal{O}(h^2)$

Consider the situation of an interface Γ in the neighborhood of the point (x_i, y_j) as seen in Fig. 4. Similar to Peskin's (first order) immersed boundary method, we consider the discontinuity along an interface as being "spread onto a grid." The difference is that we are guided

[2] The notation diag(vector) indicates a square diagonal matrix with the vector on the diagonal.

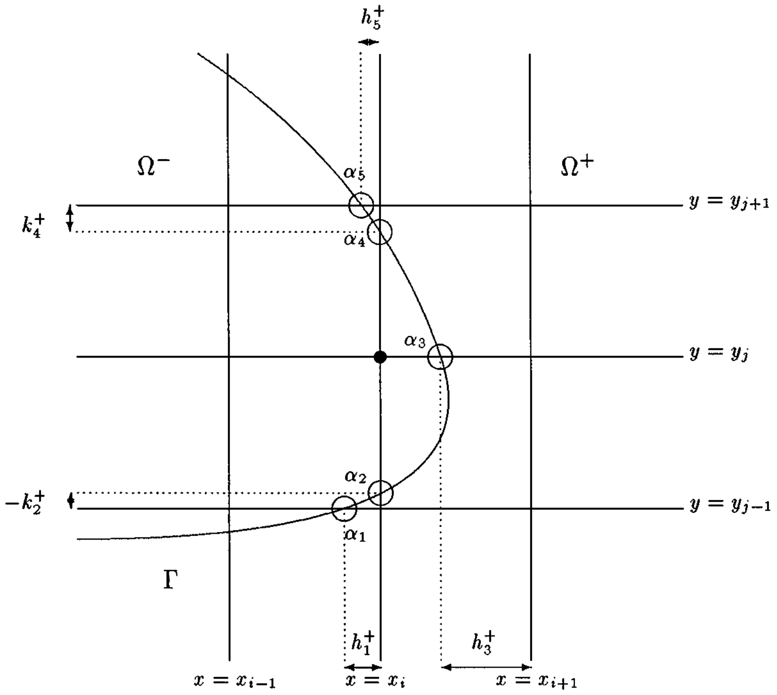


FIG. 4. Interface and mesh geometry near a lattice point (x_i, y_j). Five intersections of the interface and the mesh affect the discretization of linear elastostatics for the point (x_i, y_j); they are labeled $\alpha_1, \alpha_2, \ldots, \alpha_5$.

by "sharpness," i.e., to achieve the smallest possible influence stencil, and we want to achieve the highest possible order of the truncation error—here $\mathcal{O}(h^2)$ for all terms that need to be discretized in 2D linear elastostatics, extending the derivations in [36], which were $\mathcal{O}(h)$.

4.4.1. *The Laplacian*

As in [36], discontinuities along Γ, imposed at $\alpha_1, \alpha_2, \alpha_3, \alpha_4$, and α_5, lead to the following corrections for the Laplacian at (x_i, y_j):

$$\begin{aligned}\Delta u(x_i, y_j) = {} & \frac{u(x_{i+1}, y_j) + u(x_{i-1}, y_j) + u(x_i, y_{j+1}) + u(x_i, y_{j-1}) - 4u(x_i, y_j)}{h^2} \\ & - \frac{1}{h^2}\sum_{m=0}^{3} \frac{(h_3^+)^m}{m!}\left[\frac{\partial^m u}{\partial_x^m}\right]_{\alpha_3} - \frac{1}{h^2}\sum_{m=0}^{3} \frac{(k_4^+)^m}{m!}\left[\frac{\partial^m u}{\partial_y^m}\right]_{\alpha_4} \\ & - \frac{1}{h^2}\sum_{m=0}^{3} \frac{(k_2^+)^m}{m!}\left[\frac{\partial^m u}{\partial_y^m}\right]_{\alpha_2} + \mathcal{O}(h^2).\end{aligned}$$

The only other grid point where the jumps at α_3 enter the discretization of the Laplacian is the second neighboring grid point, (x_{i+1}, y_j):

$$\begin{aligned}\Delta u(x_{i+1}, y_j) = {} & \frac{u(x_{i+2}, y_j) + u(x_i, y_j) + u(x_{i+1}, y_{j+1}) + u(x_{i+1}, y_{j-1}) - 4u(x_{i+1}, y_j)}{h^2} \\ & - \frac{1}{h^2}\sum_{m=0}^{3} \frac{(h_3^-)^m}{m!}\left[\frac{\partial^m u}{\partial_x^m}\right]_{\alpha_3} + \mathcal{O}(h^2).\end{aligned}$$

This is true in general: For the discretization of the Laplacian, every intersection of interface and mesh only affects its nearest two grid neighbors.

4.4.2. *Cross Derivatives*

For the discretization of $u_{xy} = \partial^2/\partial x \partial y$, think of

$$u_{xy}(x_i, y_j) = \frac{1}{2} \frac{\frac{u(x_{i+1},y_{j+1}) - u(x_{i-1},y_{j+1})}{2h} - \frac{u(x_{i+1},y_{j-1}) - u(x_{i-1},y_{j-1})}{2h}}{2h}$$
$$+ \frac{1}{2} \frac{\frac{u(x_{i+1},y_{j+1}) - u(x_{i+1},y_{j-1})}{2h} - \frac{u(x_{i-1},y_{j+1}) - u(x_{i-1},y_{j-1})}{2h}}{2h} + \mathcal{O}(h^2). \qquad (23)$$

In the case of smooth functions, we simply add the same terms, but in the presence of an interface, the difference approximations require different corrections. In the case illustrated in Fig. 4 we break the corrections into pieces:

$$u_x(x_i, y_{j+1}) = \frac{u(x_{i+1}, y_{j+1}) - u(x_{i-1}, y_{j+1})}{2h} - \frac{1}{2h} \sum_{m=0}^{3} \frac{(h_5^-)^m}{m!} \left[\frac{\partial^m u}{\partial x^m}\right]_{\alpha_5}$$
$$+ \frac{2}{2h} \frac{h^3}{6} u_{xxx}(x_i, y_{j+1}) + \mathcal{O}(h^3)$$

and

$$u_x(x_i, y_{j-1}) = \frac{u(x_{i+1}, y_{j-1}) - u(x_{i-1}, y_{j-1})}{2h} - \frac{1}{2h} \sum_{m=0}^{3} \frac{(h_1^-)^m}{m!} \left[\frac{\partial^m u}{\partial x^m}\right]_{\alpha_1}$$
$$+ \frac{2}{2h} \frac{h^3}{6} u_{xxx}(x_i, y_{j-1}) + \mathcal{O}(h^3).$$

Also,

$$u_{xy}(x_i, y_j) = \frac{u_x(x_i, y_{j+1}) - u_x(x_i, y_{j-1})}{2h} - \frac{1}{2h} \sum_{m=0}^{2} \frac{(k_4^+)^m}{m!} \left[\frac{\partial^{m+1} u}{\partial x\, \partial y^m}\right]_{\alpha_4}$$
$$- \frac{1}{2h} \sum_{m=0}^{2} \frac{(k_2^+)^m}{m!} \left[\frac{\partial^{m+1} u}{\partial x\, \partial y^m}\right]_{\alpha_2} + \mathcal{O}(h^2).$$

Hence

$$u_{xy}(x_i, y_j) = \frac{u(x_{i+1}, y_{j+1}) - u(x_{i-1}, y_{j+1})}{4h^2} - \frac{u(x_{i-1}, y_{j+1}) - u(x_{i-1}, y_{j-1})}{4h^2}$$
$$- \frac{1}{2h} \sum_{m=0}^{2} \frac{(k_4^+)^m}{m!} \left[\frac{\partial^{m+1} u}{\partial x\, \partial y^m}\right]_{\alpha_4} - \frac{1}{2h} \sum_{m=0}^{2} \frac{(k_2^+)^m}{m!} \left[\frac{\partial^{m+1} u}{\partial x\, \partial y^m}\right]_{\alpha_2}$$
$$- \frac{1}{4h^2} \sum_{m=0}^{3} \frac{(h_1^-)^m}{m!} \left[\frac{\partial^m u}{\partial x^m}\right]_{\alpha_1} - \frac{1}{4h^2} \sum_{m=0}^{3} \frac{(h_5^-)^m}{m!} \left[\frac{\partial^m u}{\partial x^m}\right]_{\alpha_5}$$
$$+ \frac{h}{12} u_{xxx}(x_i, y_{j+1}) - \frac{h}{12} u_{xxx}(x_i, y_{j-1}) + \mathcal{O}(h^2).$$

Now use $u_{xxx}(x_i, y_{j+1}) = u_{xxx}(x_i, y_j) + [u_{xxx}]_{\alpha_4} + \mathcal{O}(h)$ and $u_{xxx}(x_i, y_{j-1}) = u_{xxx}(x_i, y_j) + [u_{xxx}]_{\alpha_2} + \mathcal{O}(h)$ and get

$$\begin{aligned} u_{xy}(x_i, y_j) = {} & \frac{u(x_{i+1}, y_{j+1}) - u(x_{i-1}, y_{j+1})}{4h^2} - \frac{u(x_{i+1}, y_{j-1}) - u(x_{i-1}, y_{j-1})}{4h^2} \\ & - \frac{1}{2h}\sum_{m=0}^{2}\frac{(k_4^+)^m}{m!}\left[\frac{\partial^{m+1}u}{\partial x\,\partial y^m}\right]_{\alpha_4} - \frac{1}{2h}\sum_{m=0}^{2}\frac{(k_2^+)^m}{m!}\left[\frac{\partial^{m+1}u}{\partial x\,\partial y^m}\right]_{\alpha_2} \\ & - \frac{1}{4h^2}\sum_{m=0}^{3}\frac{(h_1^-)^m}{m!}\left[\frac{\partial^m u}{\partial x^m}\right]_{\alpha_1} - \frac{1}{4h^2}\sum_{m=3}^{2}\frac{(h_5^-)^m}{m!}\left[\frac{\partial^m u}{\partial x^m}\right]_{\alpha_5} \\ & + \frac{h}{12}\left[\frac{\partial^3 u}{\partial x^3}\right]_{\alpha_4} - \frac{h}{12}\left[\frac{\partial^3 u}{\partial x^3}\right]_{\alpha_2} + \mathcal{O}(h^2). \end{aligned}$$

For the second term in (23), we find in a similar fashion

$$\begin{aligned} u_{xy}(x_i, y_j) = {} & \frac{u(x_{i+1}, y_{j+1}) - u(x_{i+1}, y_{j-1})}{4h^2} - \frac{u(x_{i-1}, y_{j+1}) - u(x_{i-1}, y_{j-1})}{4h^2} \\ & - \frac{1}{2h}\sum_{m=0}^{2}\frac{(h_3^+)^m}{m!}\left[\frac{\partial^{m+1}u}{\partial y\,\partial x^m}\right]_{\alpha_3} + \frac{h}{12}\left[\frac{\partial^3 u}{\partial x^3}\right]_{\alpha_3} + \mathcal{O}(h^2). \end{aligned}$$

Comparing with (23), intersection α_1 affects the second inner difference in the first term, α_2 affects the outer difference in the first term, α_3 affects the outer difference in the second term, α_4 affects the outer difference in the first term, and α_5 affects the first inner difference in the first term.

Averaging terms in (23) has the effect of not giving preference to either the x- or y-direction, and allows a uniform treatment for all intersections. Every intersection always affects six grid points. The relative geometry of the six points depends only on whether the intersection occurs for a horizontal or vertical mesh line. The corrections are needed in four "inner differences" and two "outer differences." The horizontal and vertical case each split up into two cases, depending on which of the nearest grid neighbors of the intersection lies in Ω^+. For example, interface–grid intersection α_3 affects the discretization of $\partial^2/\partial x \partial y$ at the points (x_{i+1}, y_{j-1}), (x_{i+1}, y_j), (x_{i+1}, y_{j+1}), $(x_i, y_{j-1})(x_i, y_j)$, and (x_i, y_{j+1}), and intersection α_4 affects the discretization of $\partial^2/\partial x \partial y$ at the points (x_{i-1}, y_j), (x_{i-1}, y_{j+1}), (x_i, y_j), (x_i, y_{j+1}), (x_{i+1}, y_j), and (x_{i+1}, y_{j+1}).

Remark 4.1. In the case that an intersection coincides with a grid point, this point should be thought of as belonging to Ω^- or Ω^+, with intersections chosen accordingly.

Remark 4.2. Through our implementation we have found that the solutions may depend more smoothly on the geometry if we change the equations for exterior grid neighbors of the boundary from "corrected linear elastostatics" to "fix the value to zero." To be able to use the fast solver from [34], this has to be implemented as a perturbation of the standard stencil as described in [36].

4.5. Finding the Displacements Using an Elastostatic Solver

Recall that boundary conditions on $\partial\Omega$ are realized through jump conditions inside R. If the jumps were known, only the right-hand side of the system of linear elastostatics on

R would need to be modified, according to the formulas in Section 4.4. To take advantage of this fact the unknown jumps at the intersections of boundaries and mesh are introduced as auxiliary variables. The jump conditions (c.f. Sections 4.1 and 4.2) are discretized based on grid values of the displacements. For known grid functions $\mathbf{U} = (U^T, V^T)^T$, the jumps can be found simply by appropriately differencing the grid functions (c.f. Section 4.3). Symbolically, the system is thus

$$\begin{pmatrix} A & \Psi \\ D & I \end{pmatrix} \begin{pmatrix} \mathbf{U} \\ \mathbf{J} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{pmatrix}.$$

Here $\mathbf{U}$ denotes the vector of stacked grid functions U and V, $\mathbf{J}$ is the vector of jumps, A is the matrix for the discretization of linear elastostatics by centered differences with normal boundary conditions, (5) and (6) on R, Ψ is the matrix that distributes the jumps to the equations with appropriate coefficients (Section 4.4), D takes appropriate finite differences on the grid functions (Sections 4.3, 4.1, 4.2), and I is an appropriately sized identity matrix. The first row in the above system discretizes the differential equation on R, the second row is the discretization of the jump conditions. Given jumps correspond to zero rows in D and nonzero entries on the right-hand side $\mathbf{F}_2$.

Eliminating $\mathbf{U}$ from the system using

$$\mathbf{U} = A^{-1}(\mathbf{F}_1 - \Psi\mathbf{J}), \tag{24}$$

we find a system for the jumps,

$$DA^{-1}(\mathbf{F}_1 - \Psi\mathbf{J}) + \mathbf{J} = \mathbf{F}_2;$$

that is,

$$(I - DA^{-1}\Psi)\mathbf{J} = \mathbf{F}_2 - DA^{-1}\mathbf{F}_1. \tag{25}$$

This Schur complement for the jumps is solved iteratively with GMRES [23] or BiCGSTAB [32], conjugate gradient methods for nonsymmetric matrices. An iterative method is needed because A^{-1} is not known, but it can be applied rapidly in $\mathcal{O}(n \log n)$ using a fast elastostatic solver [34]. To make efficient use of the solver, the number of mesh points in each direction (after reflection, see [34]) should be a power of 2. Once the jumps are known, the displacement vector $\mathbf{U}$ is found applying the fast elastostatic solver one more time to evaluate (24).

Remark 4.3. This is the application of Schur-complement methods for Poisson problems on irregular domains from [36] (based on earlier work by Buzbee *et al.* [9] and the capacitance matrix method by Proskurowski and Widlund [20], also used by Yang [38]) to the elastostatic equations.

4.6. The Idea of the Fast Elastostatic Solver

The fast solver that applies A^{-1} in (25) requires $\mathcal{O}(N \log N)$ operations to solve the elastostatic equations (1), (2) on a rectangle with normal boundary conditions, (5)–(6). It is based on the formulas for periodic boundary conditions. For $c > 0$, $1 \leq k \leq N_1$ and $1 \leq m \leq N_2$, but $km > 1$,

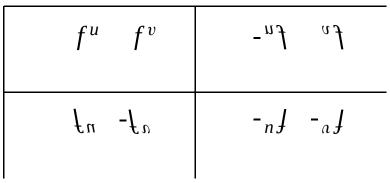
$$\bar{U}_{k,m} = \frac{d_{k,m}\bar{F}^u_{k,m} - b_{k,m}\bar{F}^v_{k,m}}{a_{k,m}d_{k,m} - b^2_{k,m}} h^2, \tag{26}$$

$$\bar{V}_{k,m} = \frac{a_{k,m}\bar{F}^v_{k,m} - b_{k,m}\bar{F}^u_{k,m}}{a_{k,m}d_{k,m} - b^2_{k,m}} h^2, \tag{27}$$

where N_1 and N_2 are the numbers of grid points in the horizontal and vertical directions, bars on capitalized variables indicate the two-dimensional Fourier transform and

$$\begin{aligned}
a_{k,m} &= -4c - 2 + 2(c+1)\cos\frac{2\pi(k-1)}{N_1} + 2c\cos\frac{2\pi(m-1)}{N_2},\\
b_{k,m} &= -\sin\frac{2\pi(k-1)}{N_1}\sin\frac{2\pi(m-1)}{N_2},\\
d_{k,m} &= -4c - 2 + 2c\cos\frac{2\pi(k-1)}{N_1} + 2(c+1)\cos\frac{2\pi(m-1)}{N_2}.
\end{aligned}$$

Due to the periodic boundary conditions we have to require that each component of the right-hand side sum to zero, and after that the solution is unique only up to two constants. The problem with normal boundary conditions is solved using the solution for periodic boundary conditions on a larger problem that results from reflecting the right-hand side as follows:



This reflection naturally satisfies the condition that each component of the right-hand side has to sum to zero. The undetermined constants arising from the solution of the periodic problem are both set to zero in order to match the displacement boundary conditions. Finally, the solution of the problem with normal boundary conditions is just the upper left block of the solution of the problem with periodic boundary conditions. A careful implementation [35] requires a sequence of one-dimensional FFTs on appropriately once-reflected (doubled) data structures and thus avoids the quadrupling described in [34].

For further details of the fast solver we refer to [34].

5. STRUCTURAL BOUNDARY DESIGN

So far, we have discussed techniques for computing the stresses given a particular design configuration. Our goal is to find the design that minimizes the total amount of material and keeps the compliance below a certain value, subject to loading and clamping boundary conditions. This is a constrained minimization problem; the constraint is the compliance, and the goal is to minimize the amount of material used in the design.

In many situations the solution of such a problem is not unique. An additional problem is that there may be many local minima, designs for which any small perturbation that satisfies the constraint require more material, yet the given shape is not the global minimum.

Our approach is an evolutionary one; see, for example, [37]. The principal idea is to remove material in regions of low stress and to add material in regions of high stress. We establish a removal rate RR which determines a percentage of the maximal initial stress below which material may be eliminated, and above which material should be added. The removal rate determines the closed stress contours along which new holes are cut and also the velocity of the boundary motion. It is increased only after no new holes are cut, and the design boundaries have stabilized. When the constraint is violated, the removal rate is decreased in order to add more material in regions of high stress and remove less material in regions of low stress. The lowered stresses are empirically seen to result also in lower values for the compliance. We terminate when this procedure cannot improve the weight while satisfying the compliance any more. Formally, we proceed as follows:

MAIN ALGORITHM.

```
1: Initialize; find stresses in initial design.
2: While termination criteria are not satisfied do
3:     Cut new holes.
4:     Move boundaries.
5:     Find displacements, stresses, etc.
6:     If the constraints are violated reduce $RR$ and revert to previous iteration.
7:     Update $RR$.
```

In order to execute this technique, we must accurately move the boundaries based on the stresses computed for the current design. It is important that this reconfiguration allow holes to merge, new holes to be cut, and the shape to continuously change topology under the trial motions. These requirements are handled well by level set methods, which we now briefly discuss. For more details, see [28, 29].

5.1. Brief Review of Level Set Methods

Level set methods were introduced by Osher and Sethian [19] and offer highly robust and accurate methods for tracking interfaces moving under complex motions. They grew out of the theory of curve and surface evolution developed by Sethian in [24–26], which constructs the notion of weak solutions and entropy limits for evolving interfaces, and links upwind numerical methodology for hyperbolic conservation laws to front propagation problems. The resulting level set approach works in any number of space dimensions, handles topological merging and breaking naturally, and is easy to program.

The level set method works by embedding the interface as the zero level set of a higher dimensional function. More precisely, given a moving closed hypersurface $\Gamma(t)$, that is, $\Gamma(t=0) : [0, \infty) \rightarrow R^N$, propagating with a speed F in its normal direction, we wish to produce an Eulerian formulation for the motion of the hypersurface propagating along its normal direction with speed F, where F can be a function of various arguments, including the curvature, normal direction, etc. Let $\pm d$ be the signed distance to the interface. If we embed this propagating interface as the zero level set of a higher dimensional function ϕ, that is, let $\phi(x, t=0)$, where $x \in R^N$ is defined by

$$\phi(x, t=0) = \pm d, \tag{28}$$

then an initial value partial differential equation can be obtained for the evolution of ϕ, namely

$$\phi_t + F|\nabla\phi| = 0 \tag{29}$$

$$\phi(x, t = 0) \quad \text{given.} \tag{30}$$

This is known as the level set equation, introduced in [19] by Osher and Sethian. As discussed in [24–26], propagating fronts can develop shocks and rarefactions in the slope, corresponding to corners and fans in the evolving interface, and numerical techniques designed for hyperbolic conservation laws can be exploited to construct upwind schemes which produce the physically correct entropy solution.

The above formulation in fact reveals *two* central embeddings, each of which can be handled by computationally efficient algorithms.

5.1.1. *The Level Set Embedding*

First, in the initialization step (28), the signed distance function is used to build a function ϕ which corresponds to the interface at the level set $\phi = 0$. This step is known as "initialization;" when performed at some later point in the calculation beyond $t = 0$, it is referred to as "reinitialization." The need for reinitialization in level set methods was first discussed by Chopp in his work on minimal surfaces; see [11].

The algorithm for the level set equation (29) given in [19] uses upwind schemes based on an ENO-construction. As discussed in that paper, the level set equation is solved throughout the entire computational domain. This is impractical and inefficient. The algorithm becomes computationally efficient through the use of a *narrow band level set method*, introduced by Adalsteinsson and Sethian [1], which confines computation to a narrow band around the interface of interest. The narrow band is of user-specified size. As the front moves and reaches the edge of the narrow band, the calculation is stopped, and a new initial level set function corresponding to the signed distance function is rebuilt. A very large narrow band means that one is essentially computing everywhere, and this reinitialization is never performed. A very thin narrow band means that one is computing only very close to the front, and hence reinitializing every time step. The numerical tests reported in [1] indicated that a narrow band of a particular size (around 6–10 grid points on each side of the front) seems to be the correct balance between work spent updating points in the band and work spent doing reinitialization.

5.1.2. *The Velocity Embedding*

Second, the construction of the initial value PDE given in (29) means that the velocity F is now defined for **all** the level sets, not just the zero level set corresponding to the interface itself. We can be more precise by rewriting the level set equation as

$$\phi_t + F_{\text{ext}}|\nabla\phi| = 0 \tag{31}$$

where F_{ext} is some velocity field which, at the zero level set, equals the given speed F. In other words,

$$F_{\text{ext}} = F \text{ on } \phi = 0$$

This new velocity field F_{ext} is known as the "extension velocity."

In [2], Adalsteinsson and Sethian introduce a very fast technique for constructing this extension velocity, through the use of fast marching methods [27], which are computational techniques for solving the Eikonal and related equations and rely on upwinding, causality principles, and heap sort techniques. The idea is as follows. Desirable properties of an extension velocity are that it should match the given velocity on the front itself, and that it should move the neighboring level sets in such a way that the signed distance function is preserved. Consider for a moment an initial signed distance function $\phi(x, t = 0)$, and suppose one builds an extension velocity of the form (see [10])

$$\nabla F_{\text{ext}} \cdot \nabla\phi = 0. \tag{32}$$

It can be shown that the level set function ϕ remains the signed distance function for all time, assuming that both F and ϕ are smooth. Adalsteinsson and Sethian solve (32) in $O(N \log N)$ time, where N is the total number of points where one wants to create this extension velocity. As such, it is a very efficient technique, and is warranted when a velocity is given only on an interface and must be extended throughout the narrow band in order to apply the level set update.

5.2. Using the Narrow Band Level Set/Fast Marching Methods in Structural Boundary Design

By convention, the interior of the structure is labeled “negative,” the outside “positive.” In the narrow band, we maintain the distance from the structure boundaries; on the rest of the computational domain, we keep track only of the sign of the distance function (in addition to the fact that we are outside of the narrow band). Several extensions of the basic narrow band level set method with extension velocities are required in structural boundary design.

5.2.1. *Cutting New Holes*

A new hole is cut in the interior of the structure by computing its distance function in its own narrow band (with positive label inside the hole) and then taking the maximum of the two distance functions on the intersection of the narrow bands for the structure and the hole. On the nonintersecting parts of the narrow bands the distances are maintained. The union of the two narrow bands is the narrow band for the resulting structure.

5.2.2. *Hanging Nodes*

Hanging nodes occur when the boundary motion severs part of the structure from the nontrivial boundary conditions. Fig. 5a) shows a domain with three holes that merge into one hole, Fig. 5b). After the motion, two small islands of material are left inside the resulting hole.

In structural boundary design, subdomains that become disconnected from the nontrivial boundary conditions should be removed from the computations. Practically, they are not relevant and computationally, they become independent underdetermined subproblems. We detect and remove the disconnected component as follows. After the motion, we find the components of the zero contour of the distance function “from the outside toward the inside,” starting with the exterior boundary. From these, we rebuild a distance function. Hanging

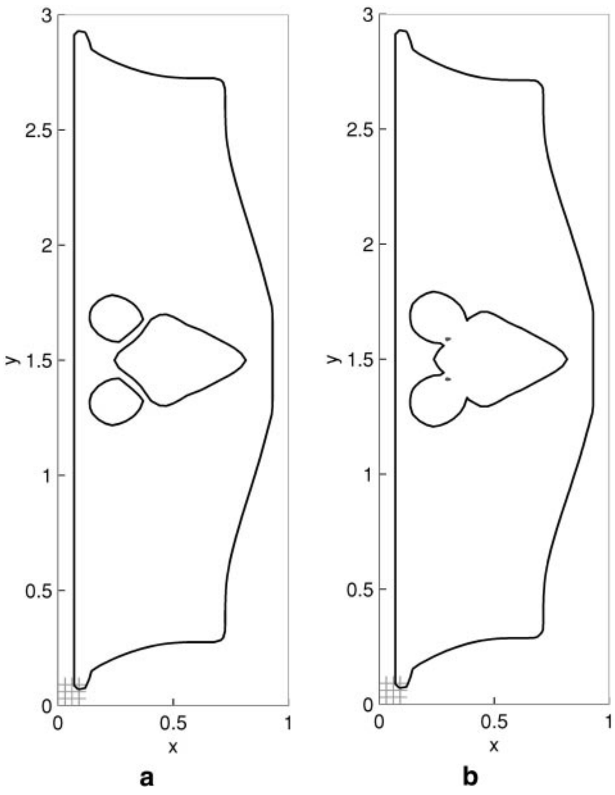


FIG. 5. A domain with three holes (a) that merge into one (b). After the motion, two small islands of material are left inside the resulting hole. They have to be removed from the computations.

nodes are recognized as being cut into the "outside" portion of the structure based on the sign of the partially rebuilt distance function and ignored in the distance rebuilding phase.

5.2.3. *Stabilization of Boundaries*

We do not cut new holes or change the removal rate for a few design steps after cutting one or more new holes, merging holes or after a change in the removal rate, because we observed that stress values immediately after these operations were not as reliable as under a stable boundary motion.

Another form of stabilization is symmetrization. For many of the problems in this paper, the structure geometry should remain symmetric due to the symmetry of the clamping and loading conditions. We found that unless we enforce this symmetry, ultimately roundoff errors lead to the departure from this symmetry. Since in our cases the symmetry axis is usually aligned with a coordinate axis, we can simply symmetrize the geometry by averaging distance values on the sides of the symmetry axis.

5.2.4. *Velocities from Stresses*

Recall that we solve for the displacements and difference them to find the (symmetric) stress tensor, with Lamé constants μ and λ,

$$\sigma = \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T) + \lambda \text{ trace } (\nabla \mathbf{u}) I.$$

From the stress tensor, we calculate the von Mises stress as

$$S = \sqrt{\sigma_{11}^2 + \sigma_{22}^2 - \sigma_{11}\sigma_{22} + 3\sigma_{12}^2}.$$

We extend the von Mises stress from the grid to the boundaries by the least squares extrapolation method from Section 4.3. In principle, for every component of the boundary the parameters of motion may be chosen separately, based for example on the maximal and minimal values of the stress on that component. In the examples in this paper, we distinguish only between the exterior boundary and hole boundaries. Five parameters describe the boundary motion.

1. s, a lower bound below which the boundary moves to reduce the structure with maximal speed.
2. S, an upper bound below which the boundary moves to grow the structure with maximal speed.
3. $\bar{s}$ and $\bar{S}$ with $s < \bar{s} < \bar{S} < S$ create an interval of stress values $[\bar{s}, \bar{S}]$ in which the boundary does not move—an interval is needed to avoid shearing the boundary. By rescaling the intervals $[s, \bar{s}]$ and $[\bar{S}, S]$ to the interval $[0, \pi]$ and using $\pm(1 + \cos)/2$ on this interval we arrive at the C^1 velocity profile depicted in Fig. 6. This profile is scaled to have maximum absolute value $2h/5$ to automatically satisfy a CFL condition.
4. The number of steps taken to advance the front. The velocity is chosen so that a single step is always allowable. To achieve similar motion on a different grid the number of steps has to be scaled inversely proportional to the change in mesh width.

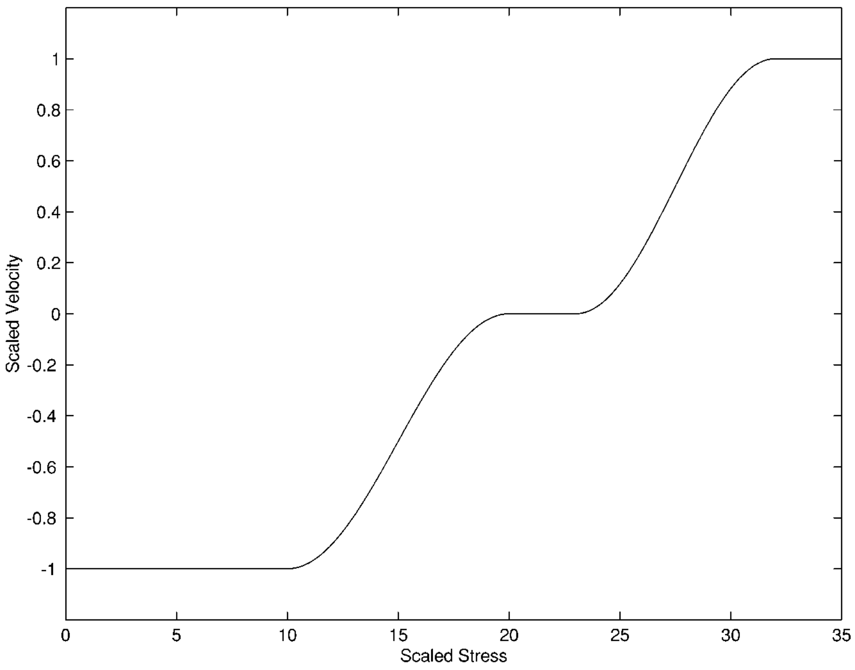


FIG. 6. The velocity profile for $s = 10$, $RR \approx \bar{s} = 20$, $\bar{S} = 23$, and $S = 32$. The speed is at most one to allow mesh-dependent scaling and automatic handling of a CFL condition in the level set motion.

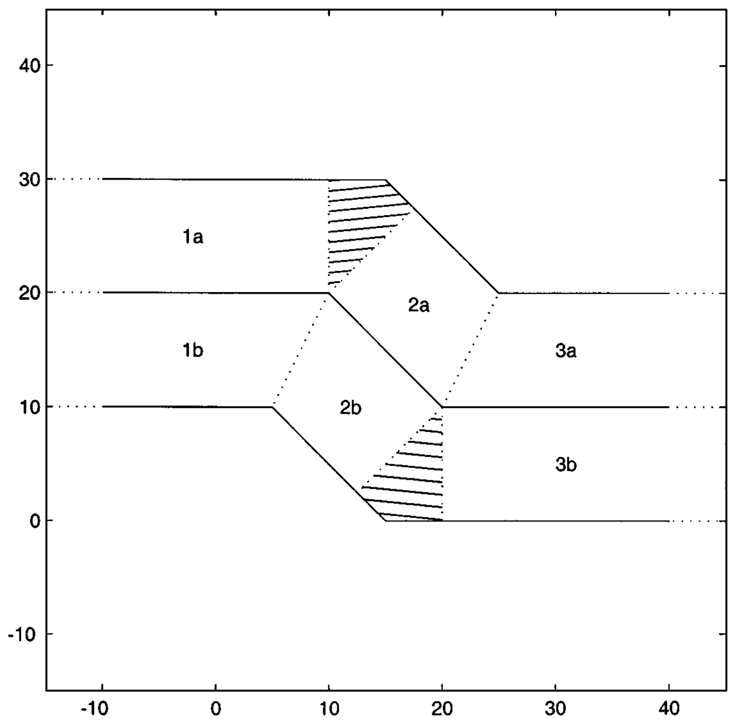


FIG. 7. Values of velocity extension are only assigned inside the narrow band. In the shaded regions, the velocity is constant and equal to the value at the corner. In the other regions, the values are the same as on the nearest point in the polygon, where in turn the values are obtained by linear interpolation between the values at the corners.

5.2.5. *Velocity Extension*

In the implementation, we extrapolate from grid points inside the structure to get stress values at the corners of a polygon whose corners results from calculating the intersections of the zero level set with the mesh lines of the level set mesh. From the stress values, we calculate the velocities at these points. All grid points inside the narrow band inherit the velocity from the nearest polygon point, with linear interpolation of velocities in the polygon segments. Figure 7 illustrates this procedure. At the corners, where the normal does not exist, the extension is different on the two sides of the polygon. On the side with the acute angle, the extension arrives from the interior of the polygon segments. On the side with the obtuse angle, there exists a region where the velocity is extended by the constant value from the corner point.

5.2.6. *Fixing Boundaries*

The level set representation also allows inhibiting the motion of parts of the boundary. Following a suggestion by Adalsteinsson, we simply copy values prior to the motion in regions that should not move. Similarly, it it possible to select only inward motion or outward motion on part of the domain by simple operations (e.g., $\max(F, 0)$, etc.) on the extended velocity on the grid.

5.2.7. *Choosing Parameter Values*

The final geometry found by our procedure depends on the choices of many parameters. This is typical for nonconvex optimization problems and is not unique to our approach. For example, it was observed also for the reverse adaptivity approach [21]. A typical value for the intial removal rate is between 0.01 and 0.1, the update is usually 0.01 or 0.025. The lower and upper bounds for the velocity typically are $s = 0$, $\overline{s} = RR$, $\overline{S} = 1.1\overline{s}$ and $S = \min(5.5\overline{s}, S_{\max})$, where $S_{\max}$ is the maximal stress in the initial design. The constraint on the compliance and on the maximal stress can be chosen arbitrarily. We terminate, for example, when a new design achieved under motion of $RR = 0.01$ does not satisfy the constraint anymore.

5.3. Using the Immersed Interface Method in Structural Boundary Design

5.3.1. *Solving the PDE When the Structure Becomes "Narrow"*

Geometries may arise in which the discretization to second order is not possible because there are not enough grid points inside the structure. This is the case especially for truss-like structures, where hole boundaries are close together and parallel over a length of three times the mesh size or more. To deal with these cases correctly, one could use a finer grid that resolves the geometry with enough grid points in the structure. But this is costly, especially when large parts of the domain are not yet truss-like. Alternatively, we can avoid the troublesome geometries using morphological operations suggested by Sarti. By eroding (shrinking) the structure by H and then dilating (growing) it by the same amount, we make sure that all trusses after this operation have width at least $2H$. To see this, note that in general, the structure changes very little under these operations. However, where a long, narrow truss of width less than $2H$ was originally present, the erosion has removed it, and dilation cannot recreate it.

Figure 8 shows a rectangular cantilever with three holes on the computational grid. Erosion leads to the light gray gray boundaries, and dilation returns the boundaries to the

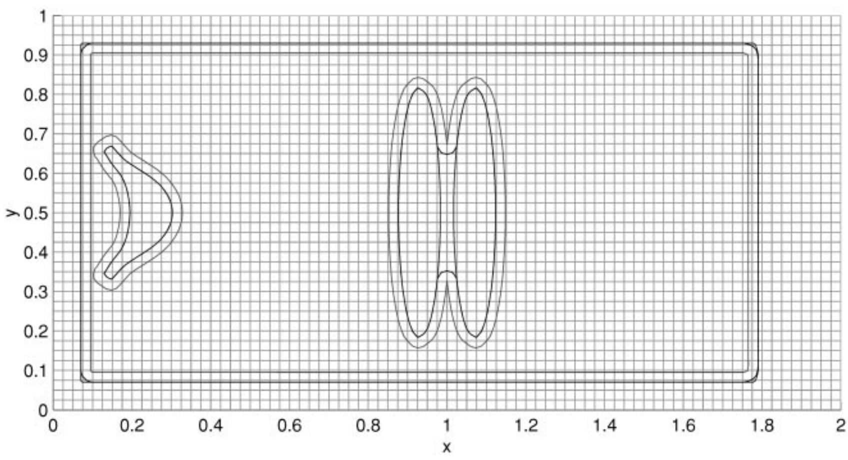


FIG. 8. Erosion and dilation remove the narrow truss between the two elongated ellipses, while all other boundary portions remain almost unchanged. The boundary motion is carried out on a different grid of half the shown meshwidth.

original locations except for a little rounding of the corners and in the narrow region between the two ellipses that is too narrow for the discretization.

5.3.2. *Grid Crossing*

For any fixed geometry, the explicit jump immersed interface method converges with $O(h^2)$, but this says nothing about the quality of the solution when the grid is kept fixed and the boundary geometry is modified. In structural boundary design, whenever a boundary crosses a grid point, this point "enters" or "leaves" the structure and is newly used in the computation of the stresses or dropped from this computation. In this situation, the computed displacements and stresses may change discontinuously, even though the boundary motion is perfectly smooth. To improve this behavior, we abandon the six-point stencil approach from [33, 36] and use the weighted least squares from Section 4.3. This dramatically improves the "continuous" dependence of the computed stresses on the boundary shape.

6. NUMERICAL EXAMPLES

The examples in this section treat three different issues.

1. Our implementation of the explicit jump immersed interface method for the Lamé equations finds displacements with second order accuracy on arbitrary domains with displacement and traction boundary conditions.
2. With a Schur-complement technique, the fast solver for rectangles from [34] can be used for problems on irregular domains as well. Our implementation scales like $\mathcal{O}(N \log N)$, just as the solver for rectangles.
3. The narrow band level set method can be used to alter the shape and topology of the structure, with velocities depending on the stresses in the current design. We recover the solution of a well-studied short cantilever example and show that our rules of stress and weight reducing design lead to truss-like structures.

6.1. Second-Order Convergence for the Lamé Equations

The following radially symmetric example was first studied in the pure loading case by Lamé which can also be found in the books of Murnaghan [18] and Timoshenko [31].

Consider (1) and (2) on the annulus $r_1 < r < r_2$; see Fig. 9. Clearly

$$u = \frac{sr_1\left(C(x^2+y^2)+r_2^2\right)}{\left(r_2^2+Cr_1^2\right)(x^2+y^2)}x,$$

$$v = \frac{sr_1\left(C(x^2+y^2)+r_2^2\right)}{\left(r_2^2+Cr_1^2\right)(x^2+y^2)}y$$

satisfy the elastostatic equations (1), (2) with $f^u = f^v = 0$. The displacements u and v also satisfy the displacement boundary conditions (3)

$$u = \frac{s}{r_1}x,$$

$$v = \frac{s}{r_1}y$$

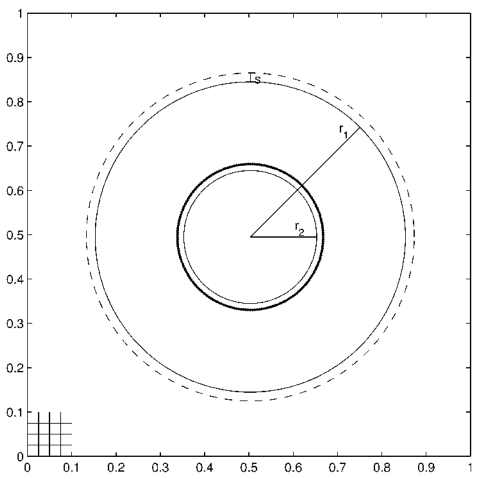


FIG. 9. The annulus $0.15 = r_2 < r < r_1 = 0.35$, centered at $(0.503, 0.495)$. The choice of center avoids symmetry of the domain with respect to the partially shown computational grid ($h = 0.025$). On the outer boundary, a displacement in the normal direction of length s is indicated by the dashed circle. The displacement on the traction-free inner boundary is indicated by the heavy circle.

on the inner boundary $\{x^2 + y^2 = r_1^2\}$ and a traction-free boundary condition ((4) with $\mathbf{g} = 0$) on the outer boundary $\{x^2 + y^2 = r_2^2\}$.

Figure 10 shows the analytic solution (for $s = 0.02$, $r_1 = 0.35$, $r_2 = 0.15$ and $C = 1/3$; on a polar grid for visualization only). To test how well we can recover the symmetry of the solution on a Cartesian grid, we have shifted the origin to the point (0.503, 0.495). The

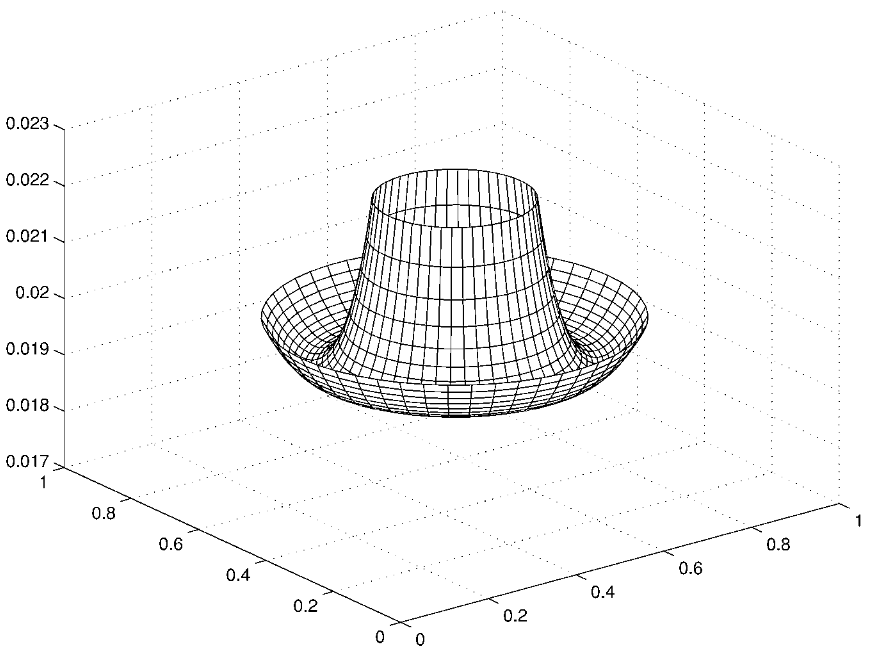


FIG. 10. Wire frame of the length of displacement of the exact solution (for $C = 1/3$, $r_1 = 0.35$, $r_2 = 0.15$, $s = 0.02$ centered at (0.503, 0.495)) evaluated on a polar grid.

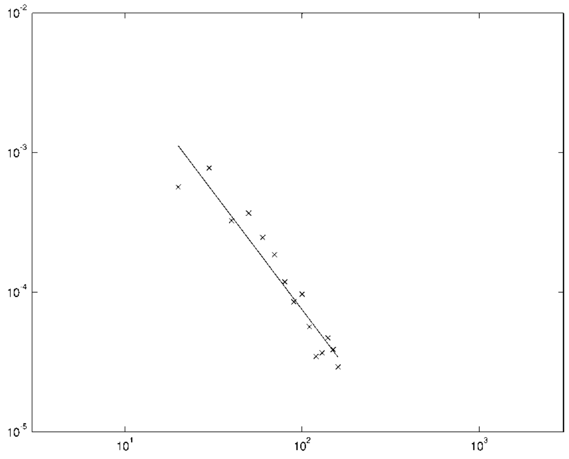


FIG. 11. The maximum error over the displacements for $n = 20, 30, 40, \ldots, 160$. The affine least squares fit indicates convergence with $\mathcal{O}(h^{1.7})$. Here the original 6-point stencil selection (see [36]) was used. The scatter shows significant influence of the mesh–boundary geometry.

computational grids are uniformly spaced with $n - 1$ interior grid points and mesh width $h = 1/n$ in the unit square $[0, 1] \times [0, 1]$.

Figures 11–13 display values for the maximum error of the displacement over all grid points inside the annulus as the mesh is refined, with $n = 20, 30, 40, \ldots, 160$ points in each direction. Figure 11 shows results for the original 6-point stencil selection scheme, Figure 12 shows results for biquadratic weighted least squares extrapolation, and Fig. 13 shows the results for bicubic weighted least squares extrapolation. We observe second-order reduction of the error in the maximum norm in all figures, but much more uniform behavior for the least squares fit, with slightly better results for the cubic fit. Figure 14 contains scatter plots that show the length of displacement as a function of the distance of a grid point from the point (0.503, 0.495). As the grid is refined for $n = 20, 40, 80, 160$, the radial symmetry of the solution is recovered.

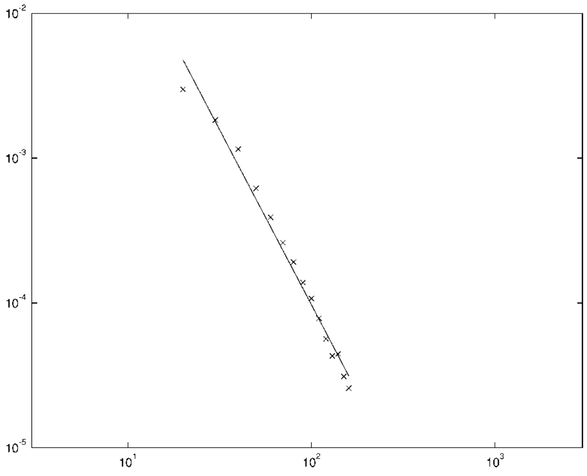


FIG. 12. Maximum error as in Fig. 11, but for biquadratic least squares interpolation. The affine least squares fit indicates convergence with $\mathcal{O}(h^{2.4})$. The scatter of the errors is significantly reduced.

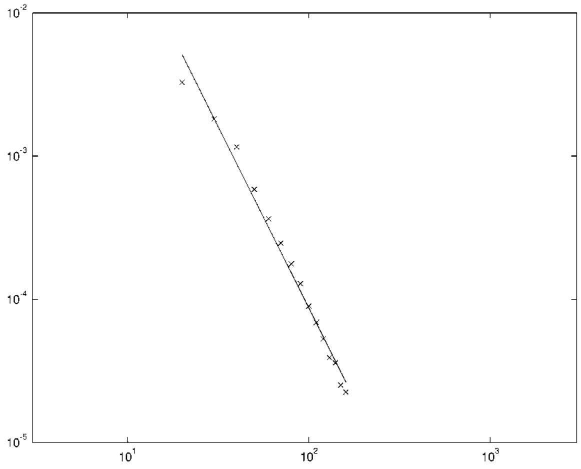


FIG. 13. Maximum error as in Fig. 11, but for bicubic least squares interpolation. The affine least squares fit indicates convergence with $\mathcal{O}(h^{2.5})$, slightly better than for the biquadratic fit.

6.2. The $\mathcal{O}(N \log N)$ Solution Method

The fast solver that is outlined in Section 4.6 is used to apply A^{-1} in the Schur-complement inside GMRES or BiCGSTAB iterations. For large problems and/or high required accuracy, we find that BiCGSTAB is preferable because it requires less memory and the convergence of restarted GMRES is not very fast. However, because A^{-1} is applied four times per

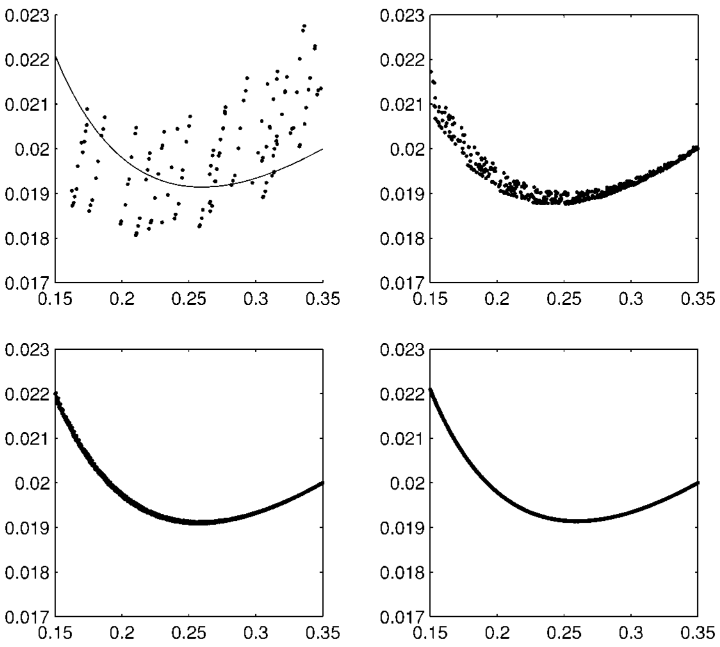


FIG. 14. Scattering of length of displacement values for $n = 20$, $n = 40$, $n = 80$, and $n = 160$. The first plot also shows the exact solution as a function of the radius. As the grid is refined, the symmetry of the exact solution is fully recovered.

TABLE I
Performance of BiCGSTAB for the Structure in Fig. 2

N	Iterations	Flops	Flop ratio	Mflops	Seconds	Time ratio
$39 \times 79 \times 2$	36	2.7e09	—	77	3.5e1	—
$79 \times 159 \times 2$	49	3.4e10	12.6	148	2.3e2	6.6
$159 \times 319 \times 2$	48	1.8e11	5.3	164	1.1e3	4.8
$319 \times 639 \times 2$	60	6.8e11	3.8	145	4.7e3	4.3
$639 \times 1279 \times 2$	81	4.4e12	6.5	126	3.5e4	7.4

BiCGSTAB iteration and only once per GMRES iteration, for problems that we can solve with GMRES without restarting, GMRES is the preferred method.

Figure 2 shows the stress contour lines of an elastic structure that is loaded and clamped as indicated in Fig. 1. The perforated rectangle is embedded in a larger rectangle, the computational domain, which is indicated in light gray in both figures. The same problem was solved on grids with 40×80, 80×160, 160×320, 320×640 and 640×1280 points. The stresses in Fig. 2 were found on the 160×320 grid, but on the other grids the results were similar, with convergence under refinement. Because the largest problem requires BiCGSTAB in order to reside in the memory of our SUN workstation without swapping, we have used it even for the smaller problems where GMRES would apply in order to be able to compare the results. Table I establishes that also on irregular domains the operations and time required to solve the problem scale like $\mathcal{O}(N \log N)$ and that we can use this method to solve medium-sized problems on workstations. For each problem size, we have reduced the residue by six orders of magnitude. This ensured that the number of iterations was similar for all problems sizes and that we could compare the amount of work performed. In order to really benefit from the higher resolution, one should reduce the residue even further for the larger problems. The odd numbers of interior grid points were chosen to be able to compare point values with the finest grid. This means that the prime factorizations and hence the efficiency of the FFT vary, but growth like $\mathcal{O}(N \log N)$ is observable nontheless.

6.3. Structural Boundary Design via Level Set and Immersed Interface Method

6.3.1. *Stress, Weight, and Compliance Reduction*

The cantilever in Fig. 15 is clamped and loaded in the same fashion as indicated in Fig. 1. High stresses occur at the inner endpoints of the clamped boundary portions and at the inward corners. This behavior is typical in elasticity, and the rounding of such corners is known as filleting.

Here, we show that our approach is able to find good fillets while at the same time reducing the compliance and the weight of the structure. We do not claim to reach an optimum, but believe that the reduction of the maximal stress at the fillet by 60%, reduction of compliance by 20% and reduction of weight by 10% together with the simplicity of the design rules are practically relevant.

As the homotopy parameter, we have arbitrarily chosen $\bar{s}$ as 20% of the maximal von Mises stress in the initial design in Fig. 15. The corresponding contour is fattened in Figs. 15–17. We allow outward motion (i.e., for stress above $\bar{s}$) in a neighborhood of the initial corners and cut one hole along a contour of low stress. This hole is then allowed to grow on portions of its boundary that are stressed below $\bar{s}$. In principle, the hole could

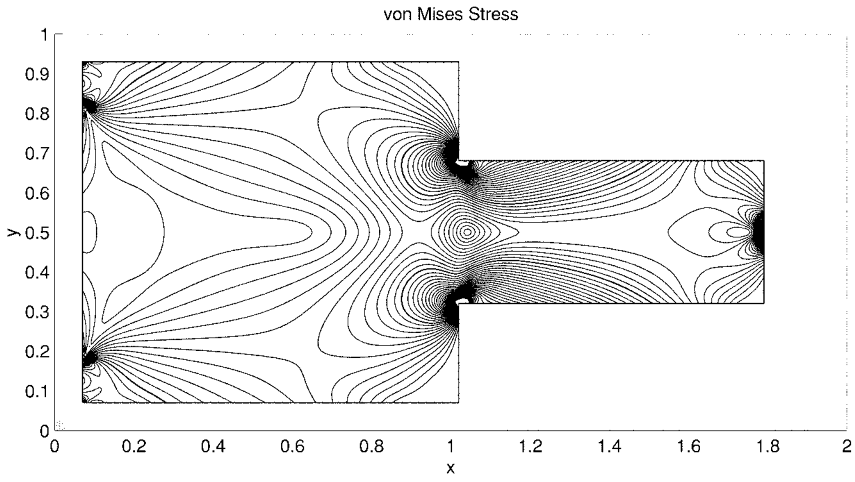


FIG. 15. Stress contours in the initial design. The dark curves represent the 20% contour of the maximal initial stress.

shrink as well, but the stresses on its boundary do not reach the point where shrinking would occur. Cutting the hole allows us to design a cantilever with less weight than the original one even though we add material in the corners.

After 10 design steps, with shape seen in Fig. 16, the stress in the fillets has been reduced by 50%. Now the growth of the hole drives further changes in the fillet shape until that also stabilizes at the shape in Fig. 17. The fillet shape that develops is similar to that found by Xie and Stevens [37], but note that we achieve it by the opposite procedure. Rather than removing material from a ground structure, we add material where the stresses are high. Also, the stress contours are much smoother in the final design than in the initial design. On the down side, at the inner endpoints of the clamped boundaries, the stress has increased by 25%.

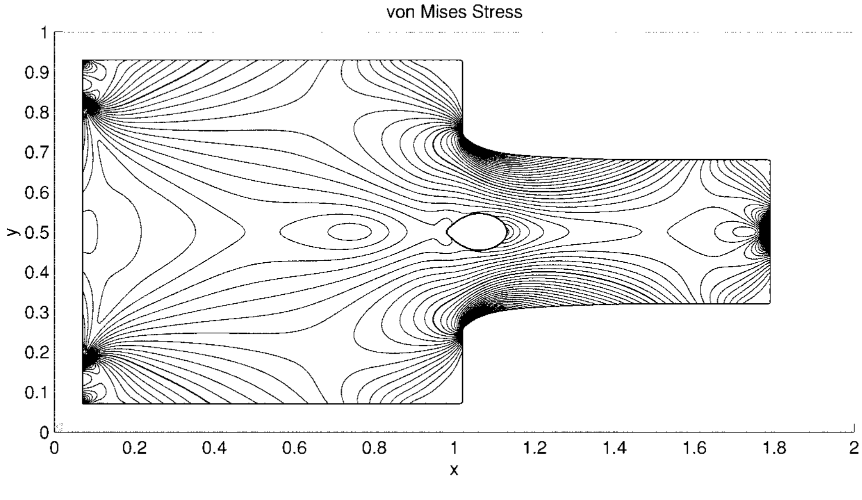


FIG. 16. The formation of the fillet for the design in Fig. 15 is essentially finished, the hole is still growing. This growth drives further changes of the fillet. The dark curves represent the 20% contour of the maximal initial stress.

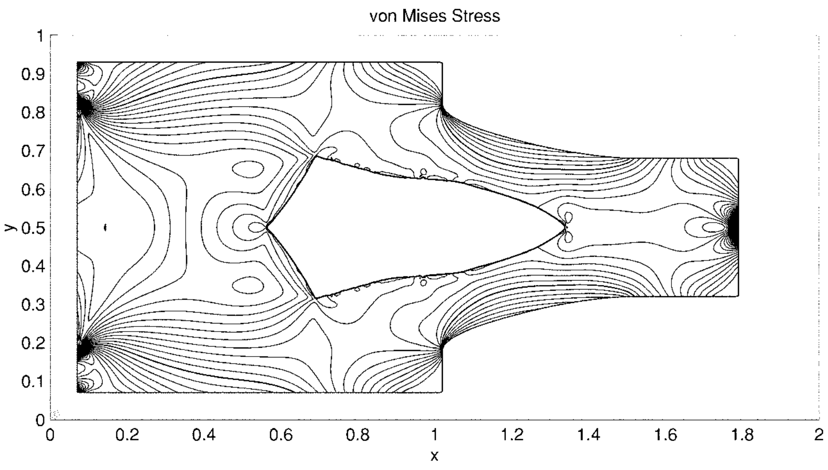


FIG. 17. Formation of fillet and hole are finished. The dark curves represent the 20% contour of the maximal initial stress.

Figure 18 shows the behavior of the maximal stress (maximum taken away from the clamped and loaded boundary portions) weight, and compliance. They are all scaled to one at the initial design. The oscillations in the compliance and stress for the later designs are due to a “ripple motion” as parts of the hole boundary are stressed close to the limit of outward motion.

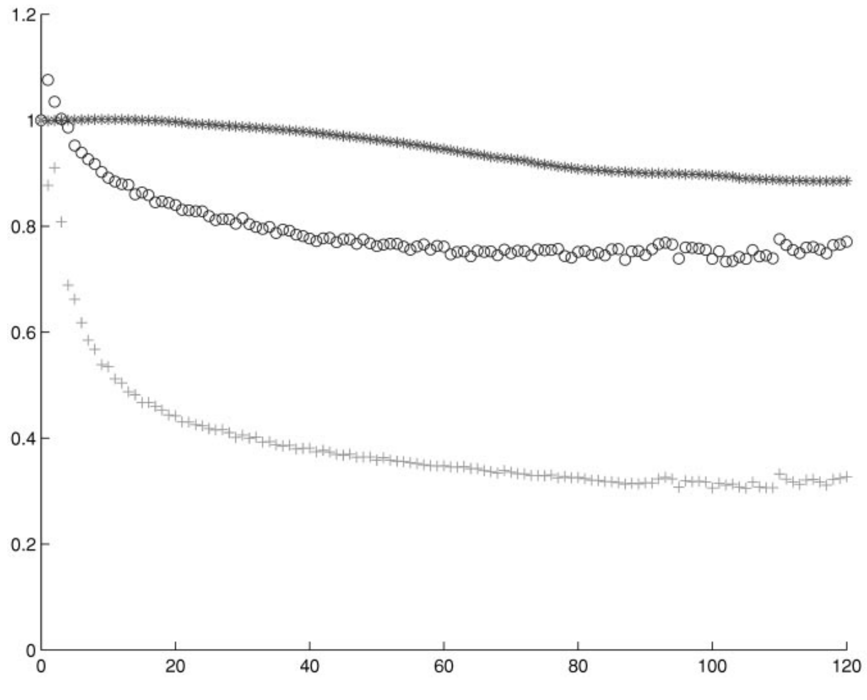


FIG. 18. Maximal stress (crosses), compliance (circles), and weight (stars) for the fillet design example from Figs. 15–17.

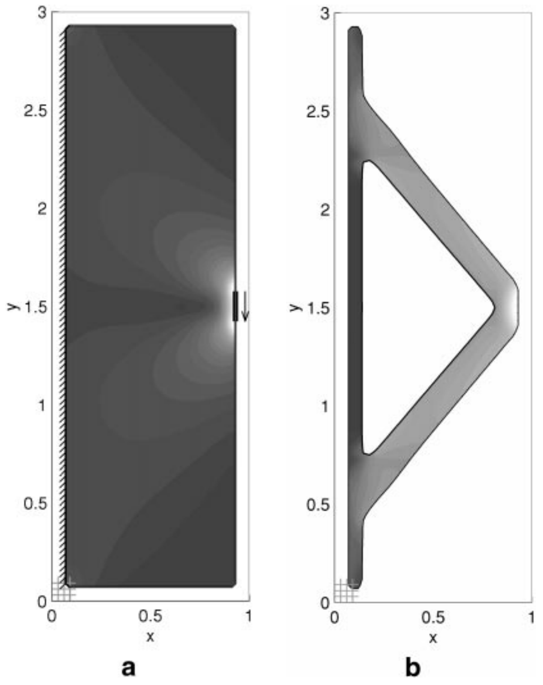


FIG. 19. (a) Clamping, loading, and stresses in the initial design. (b) Stress distribution in an improved design close to the pin-jointed two-truss solution.

6.3.2. *Constrained Design of a Short Cantilever*

A cantilever of ratio 1 : 3 is clamped everywhere on the left boundary and vertically loaded on the mid 6% of the right boundary. The rest of the right boundary and the top and bottom boundaries are traction free, as indicated in Fig. 19.

This problem was chosen because it is a standard test problem for structural boundary design (used, e.g., in [37, 21]), with a known solution for a simpler pin-jointed two-truss problem [22]. The optimal height in that case is twice the width of the structure.

Figure 19a also shows the distribution of the von Mises stress in the original design. Figure 19b shows an improved geometry close to the two-truss solution and stress distribution inside. We have almost achieved a fully stressed design. Except for the left, built-in edge, the cantilever is stressed to at least 40% of the maximal stress of the initial design.

We consider the problems of improving the weight of the cantilever for three different constraints on the compliance. The constraints are 1.2, 2.0, and 10.0 times the compliance of the “full” design from Fig. 19. Improved designs are seen in Figs. 20a–20c. It is clear that the shape depends very much on the constraint, and a more lenient constraint allows the use of less material.

Figure 21a shows the evolution of the weight (circles) and compliance (crosses) for the constraint at 2.0. Simple control of the homotopy parameter based on constraint violation leads to a nonmonotone decrease of the weight, but allows significant changes in shape when the constraint is reached and the homotopy parameter is decreased.

Figure 21b shows the number of GMRES iterations needed for the different designs. The iteration count is always in the same range, with a slight increase as the geomerty of the design becomes more complicated.

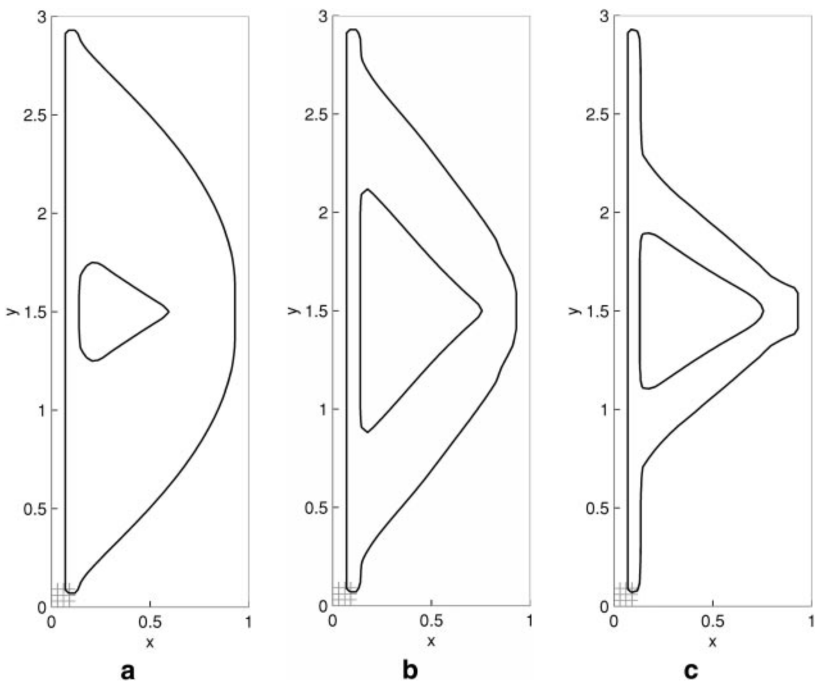


FIG. 20. Improved designs for various constraints on the compliance. (a) 1.2, (b) 2.0, and (c) 10 times the compliance of the initial design in Fig. 19.

6.3.3. *Design of a Long Cantilever from a Perforated Structure*

A cantilever of ratio 2 : 1 is built in near the top and bottom of the left edge and centrally loaded on 5% of the right hand edge. The rest of the right-hand edge and the top and bottom edges are traction free. From theoretical considerations [4, 7], a truss-like structure is expected to develop for optimal low-weight structures in these conditions. These solutions have been observed (see Figs. 22–35) based on a rectangular ground structure both in homogenization, where a weighted sum of weight and compliance are minimized [5]; and in

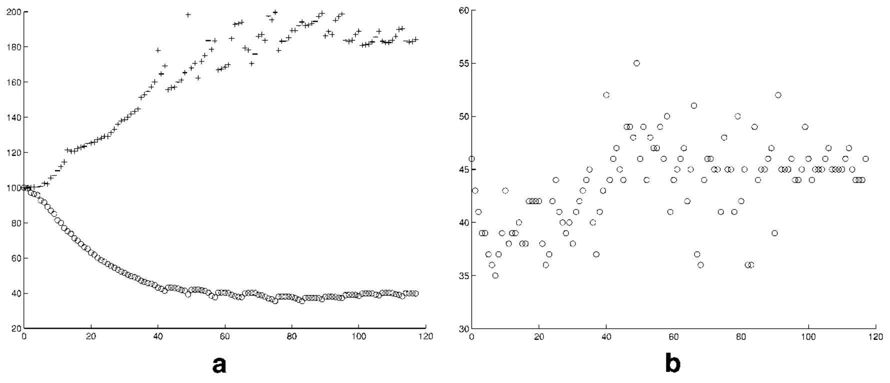


FIG. 21. (a) Compliance (crosses) and weight (circles) of the sequence of designs leading to the one in Fig. 20b. (b) Number of GMRES iterations needed to find the stresses in the designs leading to the one in Fig. 20b.

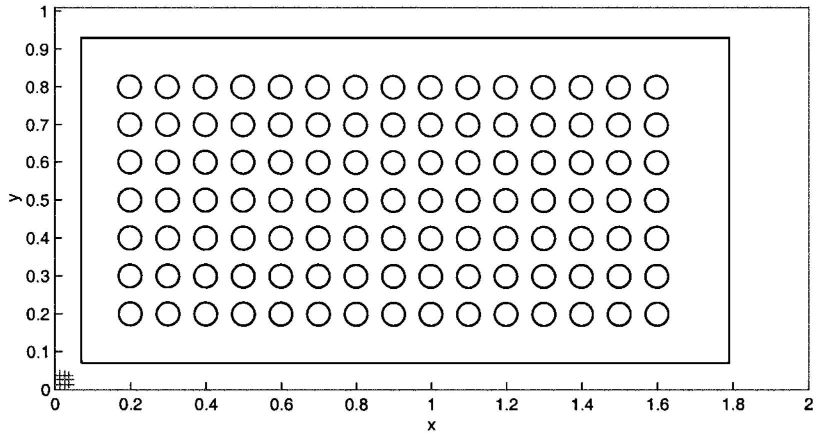


FIG. 22. Initial Design.

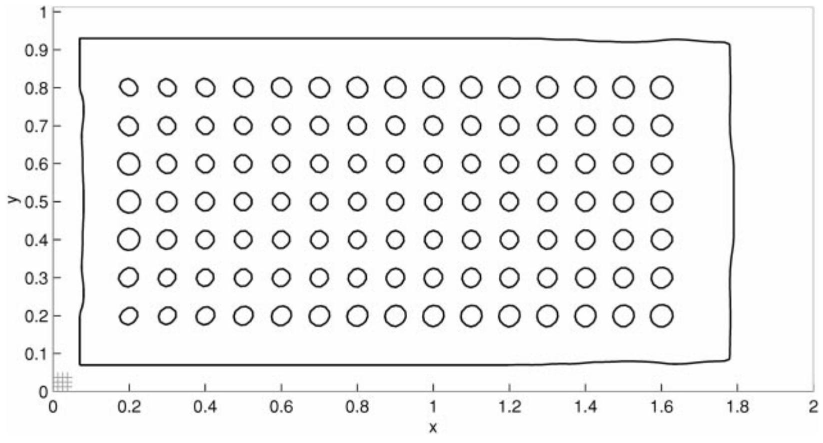


FIG. 23. Design 1.

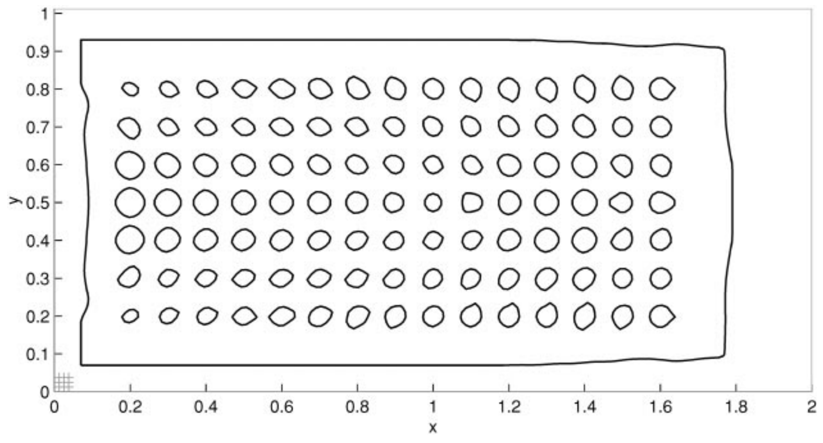


FIG. 24. Design 2.

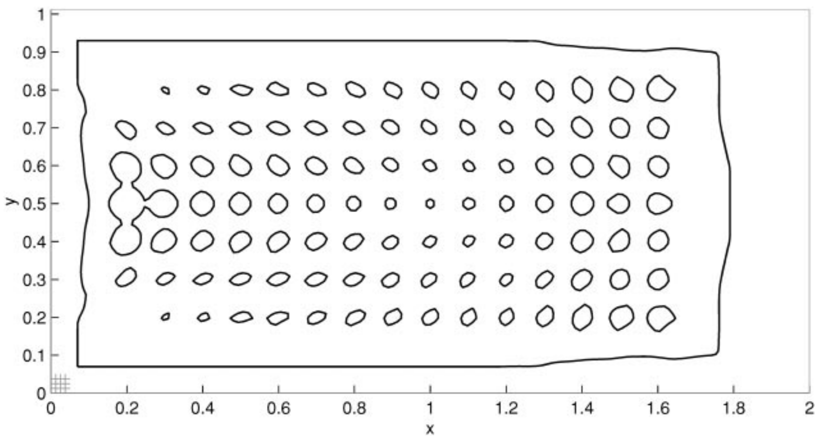


FIG. 25. Design 3.

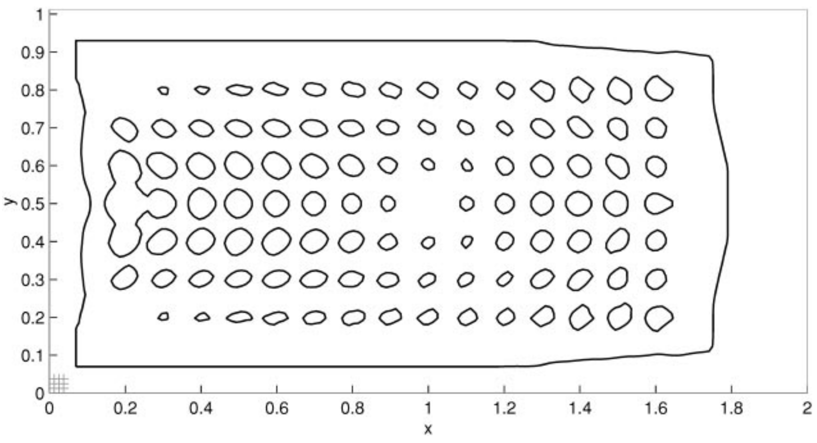


FIG. 26. Design 4.

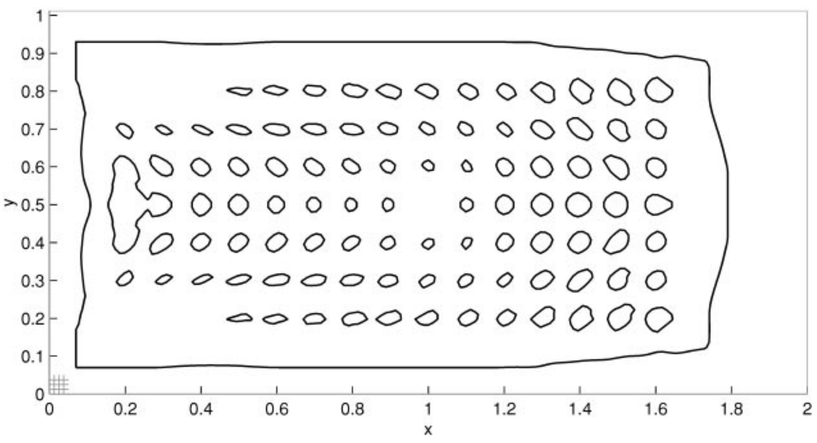


FIG. 27. Design 5.

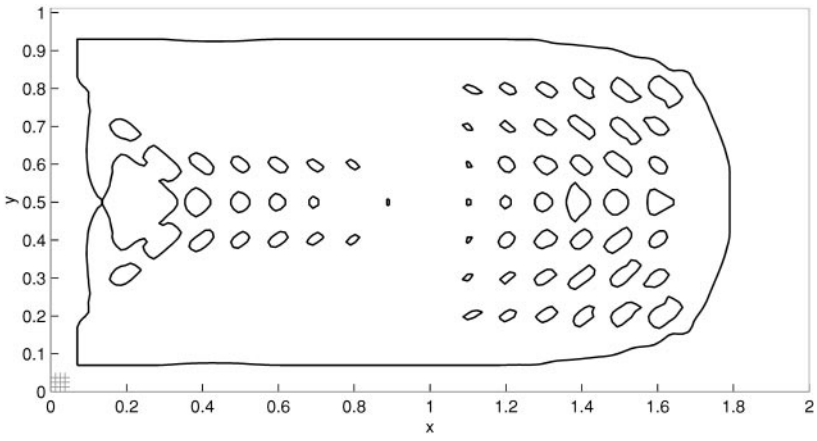


FIG. 28. Design 10.

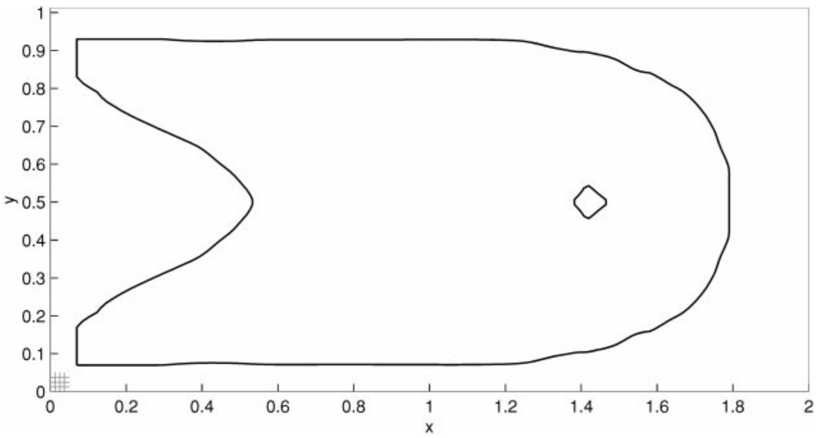


FIG. 29. Design 30.

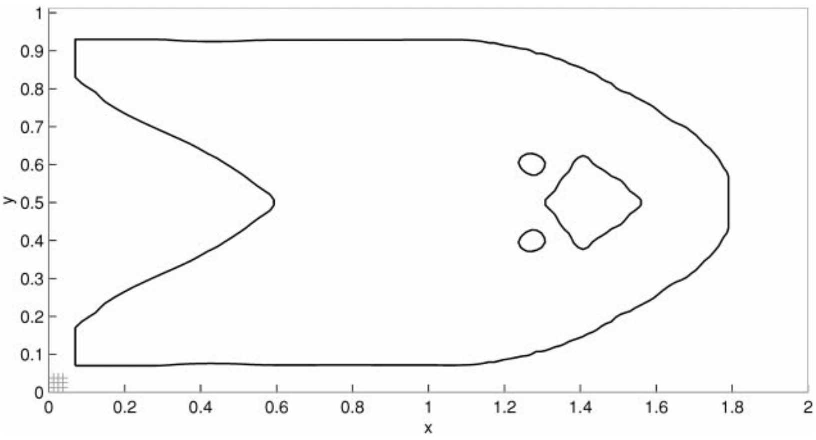


FIG. 30. Design 50.

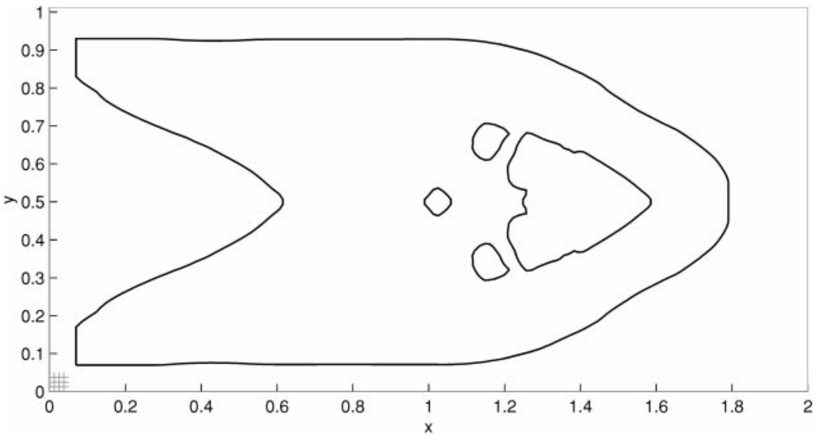


FIG. 31. Design 70.

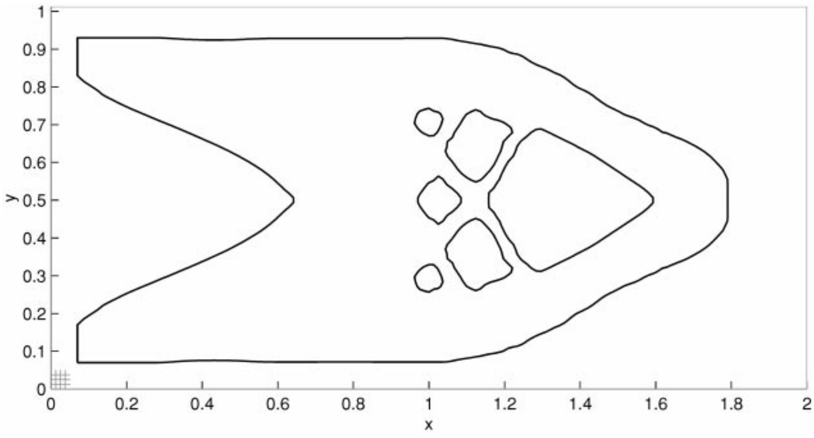


FIG. 32. Design 90.

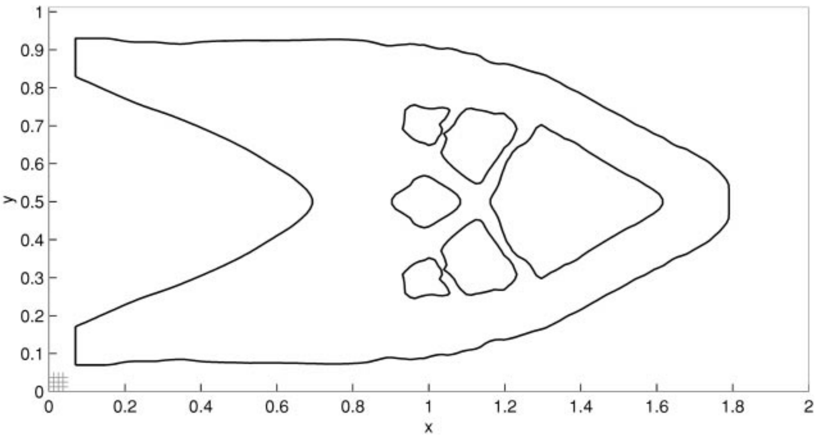


FIG. 33. Design 110.

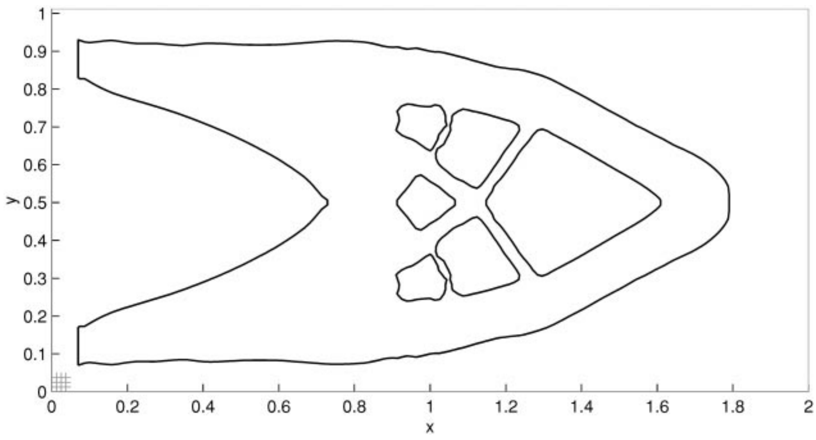


FIG. 34. Design 130.

evolutionary structural optimization, where material below a certain stress level is removed [22, 37]. Using unconstrained weight reduction by homotopy on the stress, we observe the same phenomenon based on a perforated ground structure. The homotopy parameter is initially rather small, so that the small holes tend to disappear. After the stable configuration in Fig. 29 is reached, the homotopy parameter is increased slowly and a truss-like structure develops as more material is removed.

7. CONCLUSION AND OUTLOOK

We have presented a numerical algorithm for structural boundary design. The technique accurately solves the Lamé equations and accurately differences the displacements for the stresses in arbitrary domains. It can be used to find basically optimal solutions to constrained minimization problems in some simple, well-studied test cases. We expect to extend the work to three dimensions and more complex situations. A large collection of case studies and further refinements may be found in [30].

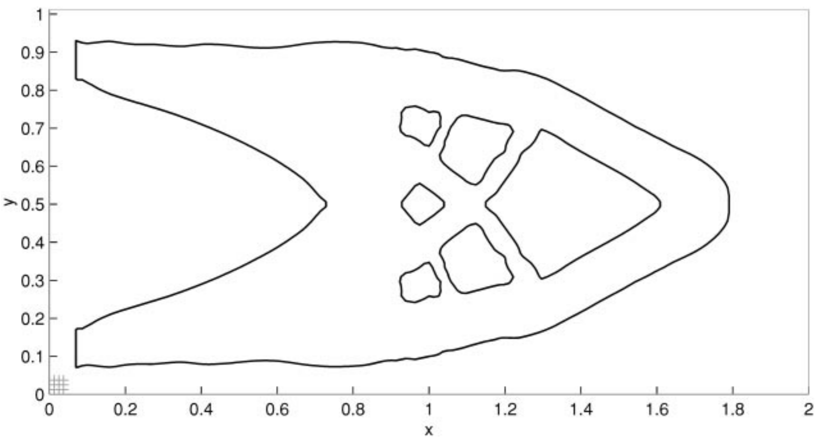


FIG. 35. Design 135.

ACKNOWLEDGMENTS

The authors thank A. Sarti, D. Adalsteinsson, and D. Kessler for their comments and suggestions. Valerie Heatlie helped with typesetting parts of this paper. All calculations were performed at the Lawrence Berkeley National Laboratory. A Web page describing level set and fast marching methods may be found at http://www.math.berkeley.edu/∼sethian.

REFERENCES

1. D. Adalsteinsson and J. A. Sethian, A fast level set method for propagating interfaces, *J. Comput. Phys.* **118**(2), 269 (1995).
2. D. Adalsteinsson and J. A. Sethian, The fast construction of extension velocities in level set methods, *J. Comput. Phys.* **148**, 2 (1999).
3. G. Allaire, The homogenization method for topology and shape optimization, in G. I. N. Rozvany (Ed.), *Topology Optimization in Structural Mechanics* (CISM, 1997), p. 101.
4. G. Allaire and R. V. Kohn, Optimal design for minimum weight and compliance using extremal microstructures. *Eur. J. Mech. A/Solids* **12**, 839 (1993).
5. G. Allaire and R. V. Kohn, Topology optimization and optimal shape design using homogenization. in M. P. Bendsøe and C. A. Mota Soares (Eds.), *Topology Design of Structures*, volume 227 of *NATO ASI Series, Series E* (Kluwer, 1993), p. 207.
6. M. P. Bendsøe, *Optimization of Structural Topology, Shape and Material* (Springer, Berlin, 1997).
7. M. P. Bendsøe and R. Haber, The Michell layout problem as a low volume fraction limit of the homogenization method for topology design: An asymptotic study, *J. Struct. Optimization* **6**, 263 (1993).
8. M. P. Bendsøe and N. Kikuchi, Generating optimal topologies in structural design using a homogenisation method, *Comput. Methods Appl. Mech. Eng.* **71**, 197 (1988).
9. B. L. Buzbee, F. W. Dorr, J. A. George, and G. H. Golub, The direct solution of the discrete Poisson equation on irregular regions, *SIAM J. Numer. Anal.* **8**(4), 722 (1971).
10. S. Chen, B. Merriman, S. Osher, and P. Smereka, A simple level set method for solving stefan problems, *J. Comput. Phys.* **138**, 8 (1997).
11. D. L. Chopp, Computing minimal surfaces via level set curvature flow, *J. Comput. Phys.* **106**, 77 (1993).
12. H. A. Eschenauer, H. A. Kobelev, and A. Schumacher, Bubble method for topology and shape optimization of structures, *J. Struct. Optimization* **8**, 142 (1994).
13. H. A. Eschenauer and A. Schumacher, Topology and shape optimization procedures using hole positioning criteria, in G. I. N. Rozvany (Ed.), *Topology Optimization in Structural Mechanics* (CISM, 1997), p. 135.
14. R. J. LeVeque and Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, *SIAM J. Numer. Anal.* **31**, 1019 (1994).
15. Z. Li, A fast iterative algorithm for elliptic interface problems, *SIAM J. Numer. Anal.* **35**(1), 230 (1998).
16. Z. Li, H. Zhao, and H. Gao, A numerical study of electron-migration voiding by evolving level set functions on a fixed cartesian grid, *J. Comput. Phys.* **152**(1), 281 (1999).
17. R. A. Lorentz, *Multivariate Birkhoff Interpolation* (Springer, 1992).
18. F. D. Murnaghan, *Finite Deformation of an Elastic Solid* (New York, Wiley, 1951).
19. S. J. Osher and J. A. Sethian, Fronts propagating with curvature dependent speed: Algorithms based on the Hamilton–Jacobi formulation, *J. Comput. Phys.* **79**, 12 (1988).
20. W. Proskurowski and O. Widlund, On the numerical solution of Helmholtz's Equation by the capacitance matrix method, *Math. Comp.* **30**(135), 433 (1976).
21. D. Reynolds, J. McConnachie, P. Bettess, W. C. Christie, and J. W. Bull, Reverse adaptivity—A new evolutionary tool for structural optimization, *Int. J. Numer. Meth. Engng.* **45**, 529 (1999).
22. G. I. N. Rozvany, M. P. Bendsøe, and U. Kirsch, Layout optimisation of structures, *Appl. Mech. Rev.* **48**(2), 41 (1995).
23. Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **7**(3), 856 (1986).

24. J. A. Sethian, *An Analysis of Flame Propagation* (Dept. of Mathematics, University of California, Berkeley, 1982).

25. J. A. Sethian, Curvature and the evolution of fronts, *Comm. Math. Phys.* **101**, 487 (1985).

26. J. A. Sethian, Numerical methods for propagating fronts, in *Variational Methods for Free Surface Interfaces*, edited by Concus and Finn, Proceedings of the 1985 Vallambrosa Conference (Springer–Verlag, NY, 1987), p. 155.

27. J. A. Sethian, A fast marching level set method for monotonically advancing fronts, *Proc. Nat. Acad. Sci.* **93**(4), 1591 (1996).

28. J. A. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Material Science* (Cambridge University Press, 1996).

29. J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Sciences* (Cambridge University Press, 1999).

30. J. A. Sethian and A. Wiegmann, Construction of efficient designs through evolving interfaces, *Int. J. Numer. Meth. Engng.*, in press.

31. S. Timoshenko, *Theory of Elasticity*, 3rd ed. (New York, McGraw-Hill, 1970).

32. H. A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi–CG for the solution of nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **13**, 631 (1992).

33. A. Wiegmann, *The Explicit–Jump Immersed Interface Method and Interface Problems for Differential Equations* (University of Washington, 1998).

34. A. Wiegmann, Fast Poisson, fast Helmholtz and fast linear elastostatic solvers on rectangular parallelepipeds, Technical Report LBNL-43565, Lawrence Berkeley National Laboratory, MS 50A-1148, Berkeley (1999). *SIAM J. Sci. Comput.*, submitted.

35. A. Wiegmann, Improved scaling in 2D and 3D for reflection based fast solvers (1999), in preparation.

36. A. Wiegmann and K. P. Bube, The explicit–jump immersed interface method: finite difference methods for PDE with piecewise smooth solutions. *SIAM J. Numer. Anal.*, in press.

37. Y. M. Xie and G. P. Steven, *Evolutionary Structural Optimization* (Springer, 1997).

38. Z. Yang, *A Cartesian Grid Method for Elliptic Boundary Value Problems in Irregular Regions* (University of Washington, 1996).