Proceedings of the Project Review, Geo-Mathematical Imaging Group (Purdue University, West Lafayette IN), Vol. 1 (2013) pp. 227-246.

ADAPTIVE DOMAIN PARTITIONING AND DEFORMATION IN REPRESENTATIONS FOR THE WAVESPEED USING UNSTRUCTURED MESHES IN FULL WAVEFORM INVERSION

ZHAOHUI GUO* AND MAARTEN V. DE HOOP†

Abstract. We introduce adaptive domain partitioning, unstructured tetrahedral meshes and their deformation in parametrizations of the wavespeed in full waveform inversion. We develop a two-stage procedure consisting of level-set based segmentation and surface or boundary controlled mesh generation using a current model and residual shape optimization through mesh deformation using the data. The procedure is motivated by available Lipschitz stability estimates for the inverse boundary value problem for the Helmholtz equation, if the wavespeed can be described by piecewise constant functions and a given domain partitioning. We incorporate progressive local mesh refinement based on the gradient in the iterative scheme. We illustrate the procedure with numerical examples.

Key words. Unstructured mesh, shape optimization, mesh deformation, Helmholtz equation, full waveform inversion.

1. Introduction. We consider parametric representations based on unstructured meshes for coefficients of partial differential equations describing (time-harmonic) waves, in particular, the Helmholtz equation in which case the coefficient is the wavespeed. The representations are motivated by the associated inverse boundary value problem, that is, the numerical iterative reconstruction of the coefficients from the data, the Dirichlet-to-Neumann map. Convergence of certain iterative methods has been established for piecewise constant models following a domain partitioning [3, 14]; approximation and compression of the coefficients by such models are then coupled to the radius of convergence. Convergence is derived from conditional stability estimates for the inverse problem, which we expect to also hold under the deformation of domains, in particular, polytopes. Here, unstructured meshes provide a natural framework, not only to adaptively obtain coefficient representations but also to update these through mesh deformation in parallel to coefficient values and, periodically, through re-meshing while increasing the resolution, that is, the number of domains in the partitioning. The constants in the stability estimates are also sensitive to certain geometric parameters which we can constrain in the mesh generation. Rapidly varying values and complex structures are incorporated through local refinement, which hierarchically adapts the meshes.

Unstructured meshes typically allow a flexible structure of node point connections and adjacent elements, which makes them more adaptable to local or global geometrical and topological constraints. Unstructured meshes can be designed to adapt to (geological) subsurface structure by placing nodes along interfaces (possibly identified by large gradients), geobodies, wavespeed isosurfaces, etc. Moreover, these meshes provide explicit representations of surfaces and interior boundaries which enable the carrying out of operations on these such as editing or deformation. The use of unstructured meshes has been proposed for various applications [5, 22]. Recently these have been utilized in traveltime tomography [24, 30]. Here, we introduce such meshes in so-called full waveform inversion. We restrict ourselves to tetrahedral meshes with triangulation of the surfaces and interior boundaries.

In this paper, we propose a region based unstructured mesh representation of subsurface wavespeed models for seismic full waveform inversion. We relate regions to a coarse-scale domain partitioning. Essentially, we combine techniques for generating unstructured tetrahedral meshes with segmentation using level sets to obtain an algorithm and strategy for (i) obtaining a domain partitioning from an image of a current model of coefficients, (ii) identifying large-scale structures (for example, salt bodies), in particular, their geometry using closed surfaces, and patterns (for example, sedimentary layers) through general interfaces while adapting the partitioning under (i), and (iii) deformation avoiding mesh self collisions. The procedure consists of two stages: segmentation of a current model or model update yielding interfaces and interior boundaries, and residual

*Department of Mathematics, Purdue University, West Lafayette, IN 47907 (guo134@math.purdue.edu)
†Department of Mathematics, Purdue University, West Lafayette, IN 47907 (mdehoop@math.purdue.edu)

shape optimization using the (boundary) data. We obtain an image of a model of coefficients by transformation into a Cartesian grid.

Essentially, (i)-(ii) aid in constructing an explicit parametric representation of coefficients from an image, preserving information about the above mentioned interfaces via level sets. In computational inverse boundary value problems based on optimization, the image would be representative of an initial coefficient model or model update (via a gradient); thus the image is used for progressive reparametrization. In (iii), we couple the surface and mesh deformations to gradient flows which are derived from the relevant energy functionals, as in [20]. The gradient flow in shape optimization for the inverse boundary value problem for the Helmholtz equation corresponds with the normal direction of the boundary evolution. In this paper, we employ gradient flows based on triangulated surfaces embedded in a three-dimensional tetrahedral unstructured mesh.

The numerical solution to the boundary value problem makes typically use of a mesh which is significantly finer than the mesh defining the parametric representation of coefficients. However, the computational mesh is directly related to the parametric representation via mesh refinement, which is locally determined by the value of the wavespeed. Periodically, the iterative method is restarted upon generating an image, using segmentation and re-meshing, which periodically involves refinement. The updating comprises (residual) shape optimization based mesh deformation for a given wavespeed distribution, and wavespeed optimization for a given mesh or domain partitioning.

The outline of the paper is as follows. In Section 2, we present the generation of unstructured tetrahedral meshes following the geometry inferred from level-set based segmentation and horizon tracking on coefficient images. We illustrate the process using an image of SEAM 3D model on a Cartesian grid. In Section 3 we present general gradient flows for triangulated surfaces. In Section 4, we describe (residual) shape optimization based mesh deformation, in particular, the self-intersection problem and topological adaptation for mesh deformation. The shape optimization is tied to the regions mentioned above. We review the derivative of the shape functional for residual shape optimization in terms of solutions of the Helmholtz equation using the adjoint state method. In Section 5 we carry out numerical experiments.

2. Generation of a 3D unstructured mesh based on segmentation. The adaptive domain partitioning starts with generating an image of the current coefficient model. The image is given on a regular Cartesian grid. Our procedure could be generalized to images generated on an adaptive Cartesian grid with the hierarchical data structure of an octree in 3D [2].

We invoke a segmentation, but first discuss the generation of an unstructured tetrahedral mesh while focussing on methods which provide explicit boundary mesh expressions. Conventional methods include Delaunay triangulation [39] and the advancing front method [32]. Both methods require explicit representations of the geometry of the boundaries as an input, and, hence, are not suitable for our problem. We make use of the mesh generation algorithm DistMesh [28] for creating unstructured surface triangular meshes using implicit geometries and Tetgen, a traditional three-dimensional mesh generator [35], for creating interior tetrahedral meshes based on the generated surface triangular meshes. We initiate our procedure with obtaining the geometric, boundary information from an image of the current model or model update by segmentation and edge detection.

2.1. Image segmentation and regions. We introduce an image-based segmentation to obtain a partitioning of the (current) wavespeed model. In the process, we use intermediate Cartesian-grid based level set functions to implicitly represent subdomains and the boundaries. Each subdomain can have a distinct character expressed in terms of the local wavespeed variations.

We introduce level set functions [27] to implicitly describe interior boundaries and interfaces. The level set method employs an implicit representation of images by defining the boundary of a region by the zero level set of an embedding function $\phi(x)$, which is referred to as the level set function. The points inside the region are identified by $\{x \mid \phi(x) < 0\}$ and the points outside the region by $\{x \mid \phi(x) > 0\}$.

We view the wavespeed model as an image $I(x)$. The segmentation is implemented by the minimization of an energy functional, $\mathcal{E}(\phi)$, defined on a level set function. One can derive the

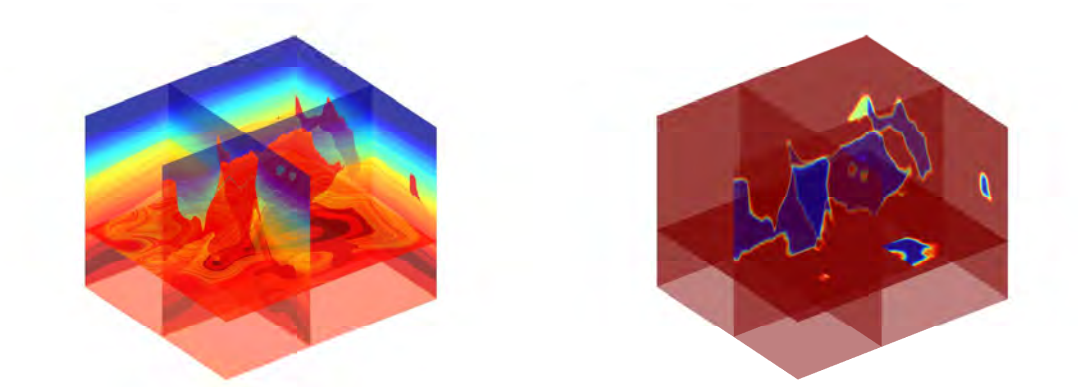

FIG. 1. *The SEAM 3D model (left) and the level set function of salt segmentation (right).*

gradient flow equation,

$$\frac{\partial \phi}{\partial t} = -\frac{\partial \mathcal{E}(\phi)}{\partial \phi}, \tag{2.1}$$

and obtain the steady state solution iteratively to compute the minimizer of the energy function $\mathcal{E}(\phi)$, while t is the artificial evolution time. There are many possible choices for the energy functional $\mathcal{E}(\phi)$. Here, we introduce an edge-based active contour model with a distance regularized level set implementation [25]. The segmentation functional takes the form

$$\mathcal{E}(\phi) = \int p(|\nabla \phi|)\, \mathrm{d}x + \gamma \int g\delta(\phi)|\nabla \phi|\, \mathrm{d}x + \mu \int gH(\phi)\, \mathrm{d}x, \tag{2.2}$$

where δ and H denote the Dirac measure and the Heaviside function. The potential function p for distance regularization is given by $p(s) = \frac{1}{2}(s-1)^2$. The g acts as an edge detector function based on the gradient of $I(x)$, which takes on small values near boundaries,

$$g = \frac{1}{1+\beta|\nabla I|^2}. \tag{2.3}$$

We may use a convolution of the image with a Gaussian kernel to mitigate any noise. The level set method naturally allows changes in topology.

The resulting level set function, ϕ, can be reinitialized [37] as a signed distance function ψ, which is used in our volumetric mesh generation via the surface meshes. The signed distance function has the property that $|\nabla \phi| = 1$ while it returns the signed distance from each point to the closest boundary. A standard way to obtain the signed distance function ψ of ϕ is to solve the following evolution equation using the artificial evolution time t,

$$\frac{\partial \psi}{\partial t} = \mathrm{sign}(\phi)\,(1 - |\nabla \psi|), \tag{2.4}$$

for a short period of 'time'. Alternatively, one can employ the fast marching method [33].

Segmentation of an image of the SEAM 3D model. We illustrate our segmentation using an image of the SEAM 3D model, on a 876 by 1001 by 751 Cartesian grid consisting of about 659 million points. The model is depicted in the left image of Figure 1, contains a highly irregular salt body, and is strongly heterogeneous. Here, we segment the salt body and obtain the level set function ϕ shown in the right image of Figure 1. The region of negative values (blue) indicates the salt body in the model. Finally we compute the signed distance function ψ of ϕ, which is shown in Figure 2. All these provide the geometric information needed in the mesh generation.

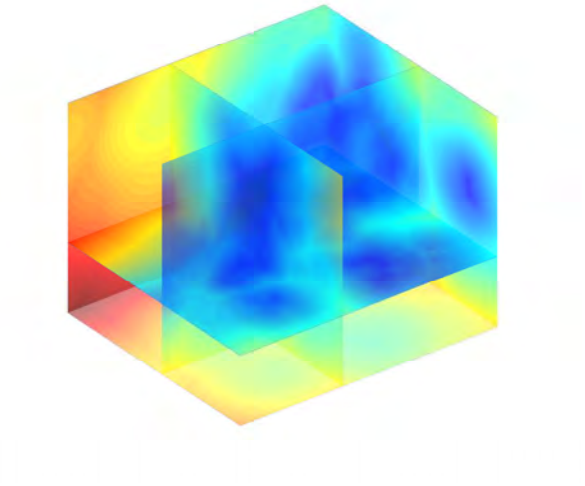

FIG. 2. *The generated signed distance function.*

2.2. Horizon and edge detection. Beside salt bodies, the model might contain other important features. The preprocessing of this information is necessary because the domain partitioning depends on the large scale structures in the model. Various methods have been developed to address subsurface horizon picking and interpolation. We mention the (2D) edge detection method [6], the 3D volume interpretation method [7] which instead of processing 2D layers detects entire 3D layers in the volume data, and the horizon fragments union method [19] which uses a 6-directional connectivity technique to combine detected fragments to form larger horizon fragments. In [21], the author presents a structure-oriented metric tensor field to perform image-guided interpolation. Here, we consider the edge detection method restricted to layers which are strongly reflective and laterally consistent. If necessary, we artificially extend interfaces to the entire image volume.

First, a boundary detector is used to identify the strong reflectors in the image. We employ the multi-step Canny edge detection procedure [9], which finds the edges by picking local maxima of the gradients. The algorithm is separated into five steps: smoothing with a Gaussian filter to suppress noise; computing gradients where the intensity changes rapidly; non-maximum suppression to mark the local maxima; double thresholding to determine the potential edges; and edge tracking by hysteresis to detect the weak edges which are connected to strong edges. We apply the edge detection to the image of the SEAM 3D model with properly chosen thresholding parameters. The result of a (200 by 200 by 200) part of the model is illustrated in Figure 3. Indeed, we recover the strong reflectors. The resulting edge is described in a '0-1' matrix, with '1' indicating the locations of edge pixels.

Next, we perform a boundary tracking technique to pick edge points on the same surface or boundary. The method utilizes a robot that walks in a sinusoidal path along the edge detected in the previous step. Similar 'walking' methods have been used for tracking environmental boundaries [10, 23]. The robot starts at a given position, and then tracks pixels of intensity 1 on the same surface reflectors. The method uses an ordered search through a neighborhood of each picked pixel in the iteration and changes 'walking' directions and search window if no pixel of intensity 1 is found. The boundary tracking algorithm works well especially for the strong and continuous layers. We connect all the edge points belonging to the same layer and directly construct a surface. Figure 4(a) displays an example of two-dimensional cross section of the SEAM 3D model as well as a bottom layer. Figure 4(b) shows the surface constructed using the edge points obtained by tracking.

Surface construction is also necessary when the layer structure is not continuous. We show an example in Figure 5(a), where the constructed layers are shown in white on a cross section. Because

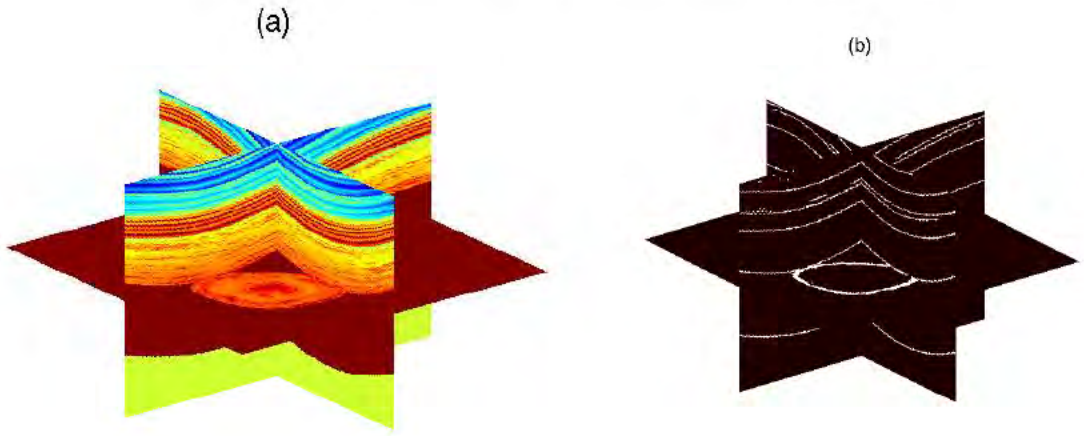

FIG. 3. *(a): Part of the SEAM 3D model; (b) the outcome of edge detection.*

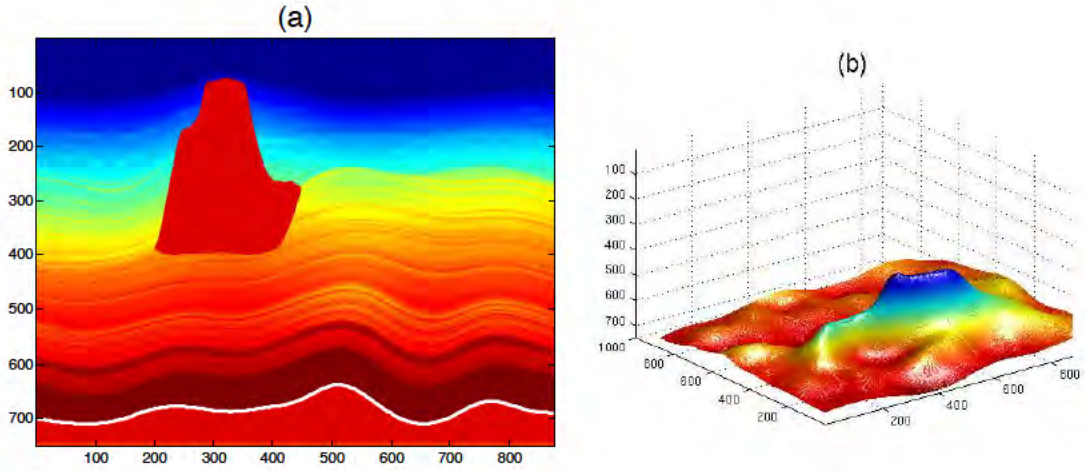

FIG. 4. *(a): A vertical cross section of the SEAM 3D model with one horizon displayed as a white line; (b): surface reconstruction of this horizon.*

these are constructed from picked points, one of the layers seems to cross the salt body when actually it should not. However, such layers aid in segmenting the domain nonetheless. Figure 5(b) shows the final layer partitioning.

2.3. Mesh generation – domain partitioning. The approach we propose, here, is a surface based mesh approximation of an (intermediate) wavespeed model based on geometrical information obtained from image segmentation. Given the model surface information, we first employ DistMesh [28] to generate the surface meshes. We use the signed distance function is derived from level set functions in the previous subsection 2.1. The implicit form of ψ is used to compute the distance from the points to the nearest boundaries, as well as the normal vector at the boundaries,

$$n = \frac{\nabla\psi}{|\nabla\psi|} = \nabla\psi, \tag{2.5}$$

as the signed distance function satisfies $|\nabla\psi| = 1$. The initial node positions are chosen along the zero level set of ψ. We decide whether nodes are inside or outside the boundary from the sign of ψ

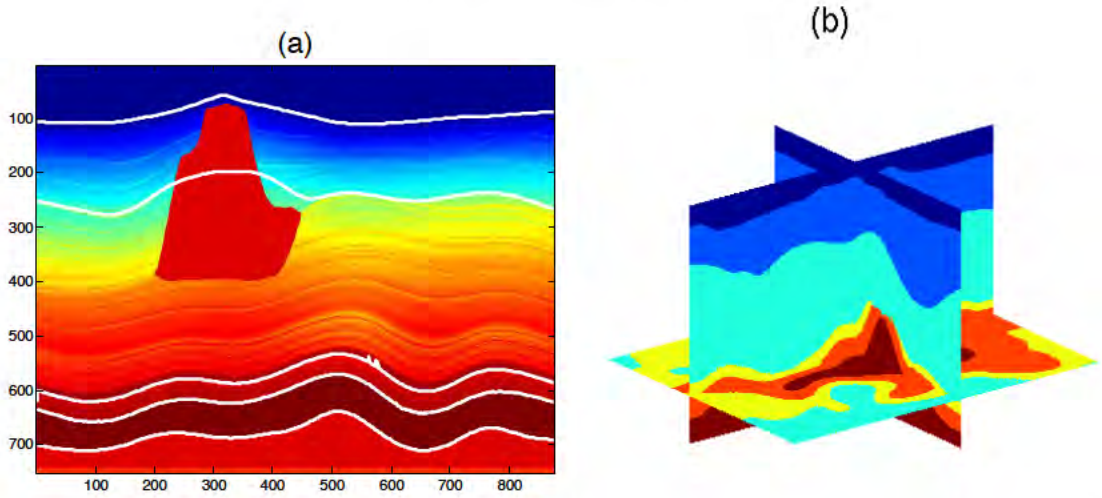

FIG. 5. *(a): A vertical cross section of the SEAM 3D model with all the selected horizons displayed as white lines; (b): the region partitioning.*

and project them back to the closest boundary along the normal vector.

The optimal location of nodes in the mesh can be obtained by solving a force equilibrium with piecewise linear force-displacement relations in an iterative manner. The idea is to treat vectors of an element edge between two nodes as springs, with current length l and ideal length l_0. A linear repulsive force is given as

$$f(l, l_0) = \begin{cases} k(l_0 - l) & : \quad l < l_0 \\ 0 & : \quad l \geq l_0 \end{cases}. \tag{2.6}$$

Each node is assigned the repulsive forces from all the edges connected to it. Thus the forces summed at node x_i result in

$$F(x_i) = \sum_{(i,j)\in e} f(l_{i,j}, l_0). \tag{2.7}$$

Here, e indicates all the edges connected with node x_i, and $l_{i,j}$ is the length of edge connecting nodes x_i and x_j.

The equilibrium state can be obtained by solving the system $F(x) = 0$; however, it is common to find this solution starting from an initial distribution $x(0) = x_0$ and letting it evolve in 'time' according to

$$\frac{\partial x}{\partial t} = F(x). \tag{2.8}$$

Then if a stationary solution is found, it satisfies $F(x) = 0$. To solve this ODE system, we follow a discretized forward Euler scheme,

$$x_i^{n+1} = x_i^n + \tau F(x_i^n), \tag{2.9}$$

where τ is the step size.

The ideal length l_0 is specified by a predefined mesh size function, which can be uniform or nonuniform and adaptive to model features. The underlying mesh is naturally subjected to an adaptive refinement method, which is one of the main features of our approach. To illustrate the procedure, we generate a triangular surface mesh for the salt body in the SEAM 3D model based on

the above obtained surface and interior boundary information. We make use of the signed distance function $\psi(x)$ displayed in Figure 2 to generate the surface meshes, which are shown in Figure 6. The left image in Figure 6 shows the surface mesh of the salt body generated using a uniform mesh size function; the right one shows the surface mesh generated using a curvature based nonuniform mesh size function.

Next, we generate the volumetric mesh based on the salt region surface mesh using Tetgen [35], which utilize a boundary constrained Delaunay tetrahedral mesh method. We can keep the surface nodes unchanged by only adding interior volume points. The placement of these nodes is chosen to improve the quality of the mesh, or to perform Delaunay refinement. Together they make the mesh representation for the region based model. Figure 7 shows the generated volumetric tetrahedral meshes for the salt body region and the SEAM 3D model in a cutaway view. In Figure 7, the tetrahedral mesh is shown with only the interface of salt body. Figure 8 shows the mesh with all the regions.

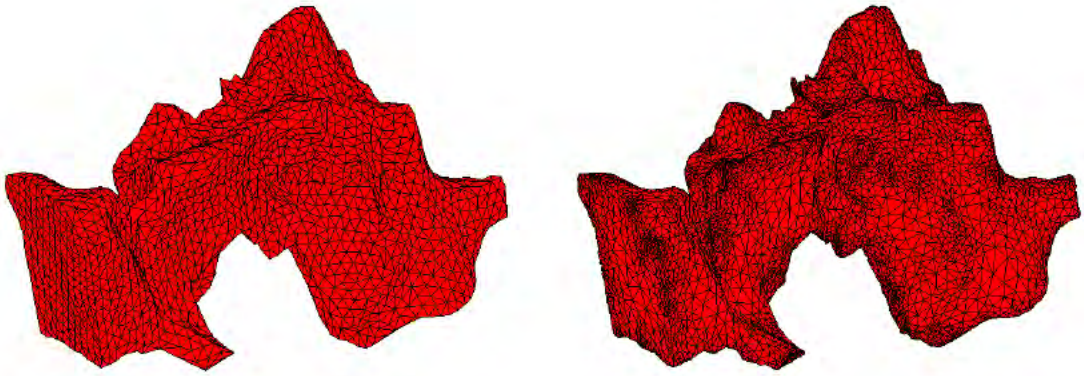

FIG. 6. *The triangular surface mesh for the salt body segmented from the SEAM 3D model with a uniform mesh size function (left) and a curvature based nonuniform mesh size function (right).*

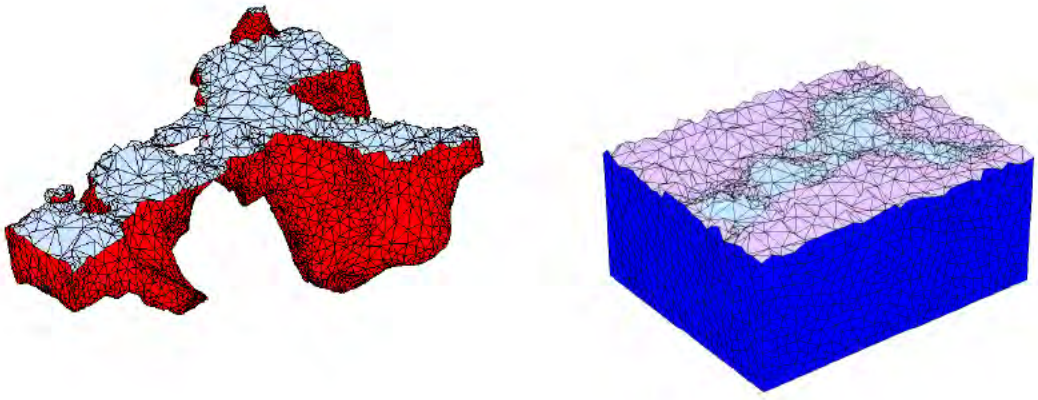

FIG. 7. *The cutaway views of the tetrahedral mesh for the salt body region (left) and the entire model (right).*

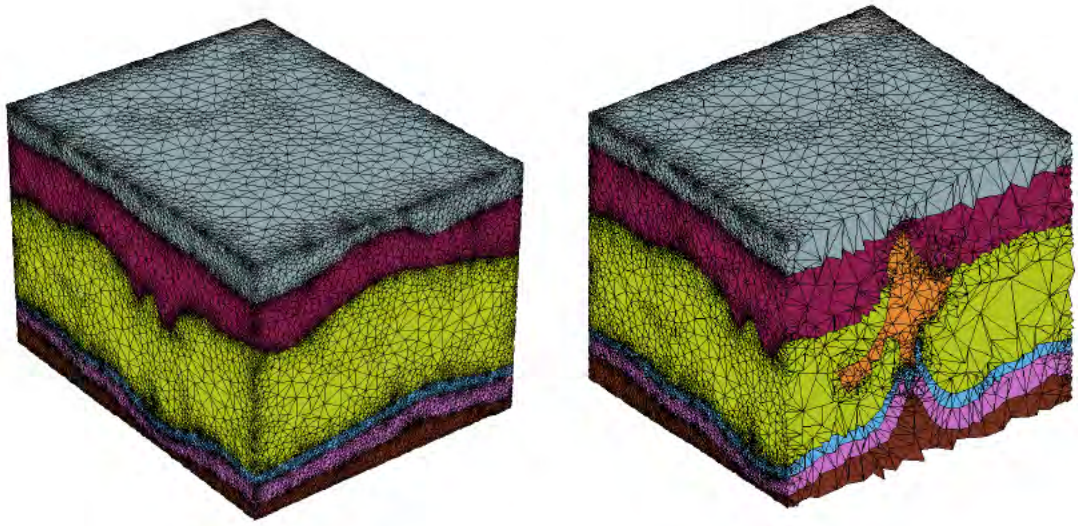

FIG. 8. *Tetrahedral mesh (510k elements) obtained from the SEAM 3D model constrained by salt and layers (left), and the cutaway view (right).*

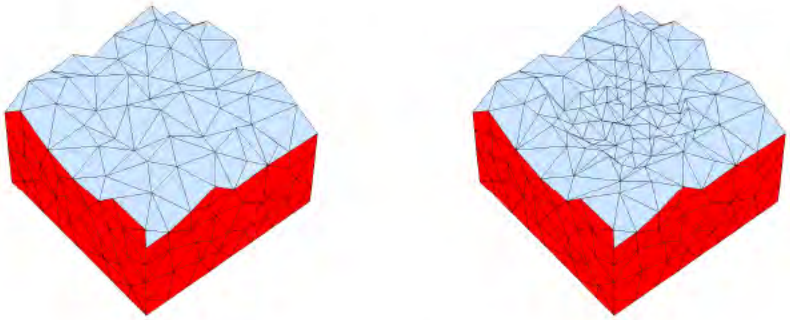

FIG. 9. *A basic example of local mesh refinement.*

Local refinement. Techniques for local mesh refinement [4, 29] were developed to provide local improved resolution without utilizing computations involving the entire domain. One can start with a coarse mesh and adaptively increase the resolution based on certain prescribed requirements. With the constraints of boundary conforming and quality measures, Delaunay mesh refinement has been extensively studied; for implementations in 2D, see [11, 13, 31] and in 3D, see, for example, [12, 34]. Here, we utilize the method developed in Tetgen, which involves the insertion of additional nodes. We employ the idea of splitting the segments and facets by inserting a node at the midpoint or at the circumcenter. In Figure 9, we show a basic example of a local mesh refinement.

3. Gradient flows. In this section, we discuss gradient flows used in our adaptive domain partitioning procedure. We consider an energy functional, $\mathcal{J}(\Omega)$, based on a connected domain Ω in $\mathbb{R}^3$, while the boundary is $\Gamma = \partial\Omega$. We determine the associated gradient descent direction. A shape deformation can be expressed as $\frac{\mathrm{d}x_t}{\mathrm{d}t} = V$ on Γ, where V is a given vector field. Here, Ω represents a single region, but the method developed here can be generalized to the case of multiple, connected regions. The deformed domains and boundaries can be written as

$$\Omega_t = \{x_t \mid x \in \Omega\}, \quad \Gamma_t = \{x_t \mid x \in \Gamma\}. \tag{3.1}$$

As in [8, 16, 20], we use a variational formulation.

The shape derivative, $D\mathcal{J}(\Omega)\,V$ is defined as in [36]

$$D\mathcal{J}(\Omega)\,V = \lim_{t\to 0} \frac{\mathcal{J}(\Omega_t) - \mathcal{J}(\Omega)}{t}. \tag{3.2}$$

We let n denote the outward normal to Γ. Essentially, the shape derivative can be expressed in the following way

$$D\mathcal{J}(\Omega)\,V = (\rho, v)_{L^2(\Gamma)}, \tag{3.3}$$

for a normal velocity $v = V \cdot n$ on Γ and some $\rho \in L^2(\Gamma)$. This explicit expression provides a way to find a descent direction for the shape of the (sub)domain in the direction V of maximal decrease of the functional $\mathcal{J}(\Omega)$. This directly yields a gradient flow evolution, which can be computed.

3.1. Triangular surface representation. We use a triangular mesh to discretize the boundary, Γ. Such meshes have been studied extensively in geometric modeling and computer graphics, and provide a 'data' structure involving vertex nodes connected by elements of line segments and triangles to form a parametric framework for interpolation and numerical computation. Here, we assume that Γ is compact, connected, oriented and has no boundary. Thus a properly triangulated surface can produce a manifold triangular mesh that is a piecewise flat surface and a good approximation of the original surface.

Let $\mathcal{S} = (X, E, T)$ denote the piecewise flat triangular representation of Γ, where $X = \{x_k \mid k = 1, 2, \ldots, n\}$ is the set of nodes, $E = \{(x_i, x_j) \mid 1 \le i, j \le n\}$ is the set of edge segments, and $T = \{(x_i, x_j, x_k) \mid 1 \le i, j, k \le n\}$ is the set of triangles represented by a triplet of nodes. We let φ_k denote piecewise linear basis function, where $\varphi_k(x_k) = 1$ and $\varphi_k(x_i) = 0$, for $i = 1, 2...k-1, k+1, \ldots, n$. The deformable triangulated surface $\mathcal{S}$ is represented by

$$\mathcal{S}(x) = \sum_k x_k \varphi_k(x).$$

A discrete vector field, V, on $\mathcal{S}$ can be linearly interpolated using the basis functions:

$$V(x) = \sum_k V_k \varphi_k(x).$$

In this way, the surface shape evolution with vector field V is consistent with the motion of nodes x_k in the direction of vector field V. In Figure 10, we show an example of surface vertex evolution in the normal direction. The normal vector $n(x_k)$ at x_k is approximated by the average of normal vectors of surrounding triangles,

$$n(x_k) = \frac{1}{|\sum_{x_k \in T_i} n(T_i)|} \sum_{x_k \in T_i} n(T_i).$$

where $n(T_i)$ is the normal vector to the triangle T_i.

3.2. Gradient flows on triangulated surfaces. We introduce a vector field V for the evolution of the shape of $\mathcal{S}$ that decreases the energy functional $\mathcal{J}(\Gamma)$, which is approximated by $\mathcal{J}(\mathcal{S})$. The shape derivative along the vector field V is represented as

$$D\mathcal{J}(\mathcal{S})\,V = \lim_{t\to 0} \frac{\mathcal{J}(\mathcal{S}(x_t)) - \mathcal{J}(\mathcal{S})}{t}. \tag{3.4}$$

If we can write the shape derivative in the form (3.3), the standard choice of the gradient flow follows from the inner product in $L^2(\Gamma)$: $V = -\rho n$ on Γ. Then we have for the shape derivative

$$D\mathcal{J}(\Omega)\,V = (\rho, v)_{L^2(\Gamma)} = -(\rho, \rho)_{L^2(\Gamma)} \le 0. \tag{3.5}$$

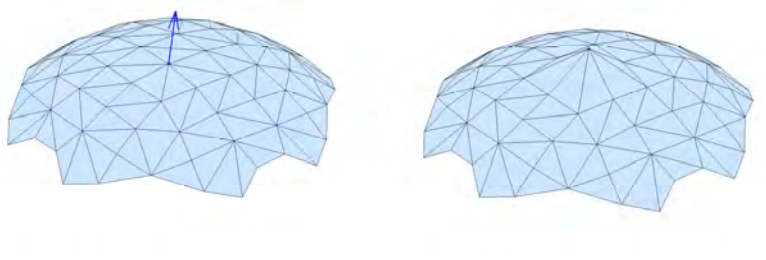

FIG. 10. *Surface mesh evolution, at x_k in the direction of V_k, indicated by the arrow.*

Assuming that the surface Γ is approximated by a triangular mesh, $\mathcal{S}$, the inner product in $L^2(\Gamma)$ between two vector fields V^1 and V^2 is evaluated as

$$\begin{aligned}(V^1, V^2)_{L^2(\mathcal{S})} &= \int_{\mathcal{S}} V^1 \cdot V^2 \mathrm{d}\sigma \\ &= \int_{\mathcal{S}} \sum_k V_k^1 \varphi_k(x) \cdot \sum_k V_k^2 \varphi_k(x) \mathrm{d}\sigma \\ &= U_1^T M U_2,\end{aligned} \tag{3.6}$$

where M is a sparse symmetric positive definite matrix with entries

$$M_{ij} = \mathrm{Id}_3 \int_{\mathcal{S}} \varphi_i(x)\varphi_j(x)\mathrm{d}\sigma.$$

According to [15], we can use a diagonal mass lumping matrix to approximate M, where M_{ii} is roughly computed by the area of the Voronoi dual cell of x_i. Here we use a 1-ring triangle area of x_i to approximate it as in [18]. Thus the discrete gradient flow is given by

$$\frac{\mathrm{d}\mathcal{S}(t)}{\mathrm{d}t} = -M^{-1}V. \tag{3.7}$$

In the case that smoothing is desirable, we can consider the Laplacian of the surface, which simulates the surface tension; it is defined as the linear combination of the 1-ring edge of nodes on the surface mesh,

$$\mathcal{L}(x_k) = \sum w_{kl}(x_k - x_l) = x_k - \sum w_{kl}x_l,$$

where w_{kl} is the weight corresponding to the mesh edge (x_k, x_l) with the requirement of $\sum_k w_{kl} = 1$. A common choice of w_{kl} is the uniform and cotangent weights. Then the numerical computation of surface evolution is rewritten as

$$\mathcal{S}_{n+1} = \mathcal{S}_n - \mathrm{d}t(M^{-1}V + \mathcal{L}(\mathcal{S}_n)). \tag{3.8}$$

However, to preserve edges of geobodies, we can choose to apply the Laplacian smoothing only when desired.

During the computational evolution, we use a boundary indicator to mark all the surface nodes, which will be used later by the self-intersection checking of mesh deformation. In addition, Tetgen generates the tetrahedral mesh with certain constraints for volume size and radius-edge ratio [26], which measures the radius of the circumsphere and the length of shortest edge. With the employment of Tetgen, we can perform local refinement and global re-meshing during the evolution to avoid bad quality triangles.

4. Mesh deformation controlled by shape optimization.

4.1. Mesh deformation and topological adaptation. Here, we discuss mesh deformation controlled by the evolution of surface nodes in the gradient descent direction following shape optimization. For the non-surface nodes, as in [28], we assume a simple mechanical (spring) relation between them. The edges of elements are controlled by a linear force-displacement as in (2.7). Thus the gradient flow acts as an external force in the normal direction at the surface nodes, while the positions of the remaining nodes are obtained by solving a force equilibrium in a truss structure. During this procedure, nodes are moving to an equilibrium position. The optimal mesh can reach a high quality, and avoid very thin and long elements. In addition, we can assign stiffness parameters to the nodes that are a distance away from the evolving boundary. By setting them to infinity, we can fix nodes that are far away from the boundaries.

The mesh parameterization that we use provides explicit and visual surface representations, but raises two big issues for the evolution over 'time': mesh self-intersection and changing in topology. To overcome these challenges, we introduce a topology adaptation method [1] in surface deformable models in 3D. The algorithm includes two key elements: mesh self-intersection detection and topological completion for an open surface mesh. Given a closed surface mesh in evolution, the flow of the vertex nodes is to be stopped if self-intersection has been detected; neighboring nodes of a self-intersection part are deleted to get an open mesh; the open mesh is completed with another mesh that is topologically equivalent to a sphere with holes. Then the evolution step continues.

One begins with identifying mesh self intersection, in particular, the edges of the triangular mesh which intersect. For a small number of triangles, one would perform one-by-one checking to verify whether two triangles intersect with one another. However, the cost of this checking increases dramatically when the number of facets becomes large. The authors in [1] introduced an algorithm to check self intersection in linear excepted time. The idea behind the algorithm is based on the projections of nodes on gridded boxes in three directions with a chosen grid length l. Theoretically, we can choose an optimal l such that nodes with a distance smaller than $\frac{l}{2}$ can be identified with an intersection of triangles. Moreover, nodes that are closer than $\frac{l}{2}$ must be contained in at least one common box. The task becomes to check the existence of a common box for each of the nodes.

The algorithm is based on a spatial hashing table [38] that defines eight hash functions $h_{i_1,i_2,i_3}(x_k)$ for each vertex $x_k = (v_x, v_y, v_z)$,

$$h_{i_1,i_2,i_3}(x_k) = \left\{ r\left(\frac{v_x}{l}, i_1\right) p_1 + r\left(\frac{v_y}{l}, i_2\right) p_2 + r\left(\frac{v_x}{l}, i_3\right) p_3 \right\} \bmod(H+1), \tag{4.1}$$

for a given function

$$r(m,n) = \begin{cases} \lfloor \frac{m}{l} \rfloor l & i = 0, \\ \lfloor \frac{m}{l} + \frac{l}{2} \rfloor l & i = 1, \end{cases}$$

where $i_1, i_2, i_3 \in \{0,1\}$, p_1, p_2, p_3 are three large prime numbers, H is the hash table size and $\lfloor\rfloor$ indicates the largest integer that is smaller than the value inside. We restate two relevant theorems in [1] in the following

THEOREM 4.1. *[1] If $|x_k - x_l| < \frac{l}{2}$, then there exists at least one hash functions, $h_{i_1,i_2,i_3}(\cdot)$, such that*

$$h_{i_1,i_2,i_3}(x_k) = h_{i_1,i_2,i_3}(x_l).$$

THEOREM 4.2. *If the triangles $T = (x_1, x_2, x_3)$ and $S = (y_1, y_2, y_3)$ in the surface mesh intersect, then there exist i and j, such that*

$$|x_i - y_j| < \sqrt{\frac{2}{3}} L_{\max},$$

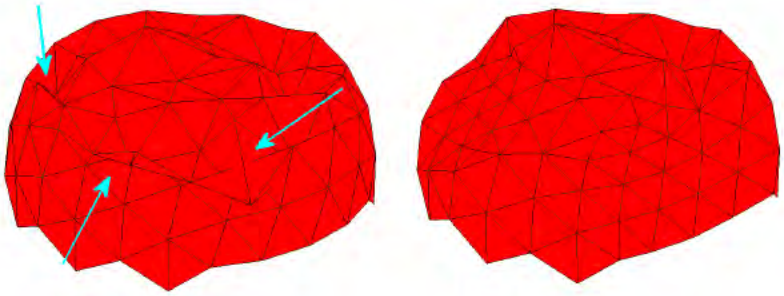

FIG. 11. *One body mesh: self-intersection detection and topology adaptation. Left: the self-intersected mesh indicated by the arrows; right: the mesh after application of topology adaptation.*

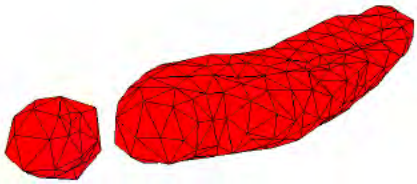

FIG. 12. *The two body mesh before evolution.*

where $L_{\max}$ is the bound of the edge length in the mesh.

If we choose $l > 2\sqrt{\frac{2}{3}}L_{\max}$, there are at least two nodes which hold the same hash key. With this information, triangle intersection is to be searched based on the nodes holding the same hash keys.

The next step is to remove the nodes of intersecting triangles and the adjacent mesh elements to get an open mesh and to reconnect the remaining nodes by inserting edges and triangles. The open mesh is arranged in a way that the boundary consists of a simple polygon. The reconnection is performed using a handle mesh which is topologically equivalent to a sphere with holes. Each choice of handle gives a possible connection with the open mesh. We have to choose the best one which does not contain the intersection, also with the shortest edge lengths. After the completion of an open surface mesh, we implement the remeshing step to adjust the topological change of the surface mesh and volumetric elements.

We employ this method in our mesh evolution to remove self intersections and apply a correction to the topology where appropriate. The meshes we encounter might contain sharp corners; these can give rise to erroneous evolution directions and, hence, intersections. To investigate the effectiveness of the algorithm, we first test a simple example of a surface mesh on a ball with no change in topology. With an incorrect deformation, we detect self intersections, see Figure 11 (left). We remove the intersections and the resulting mesh is shown in Figure 11 (right). Here, we keep the topology of th original mesh.

The second example shown in Figure 12 illustrates a changing topology of two body meshes. We artificially choose a large step size in the mesh evolution. Two bodies intersect one another,

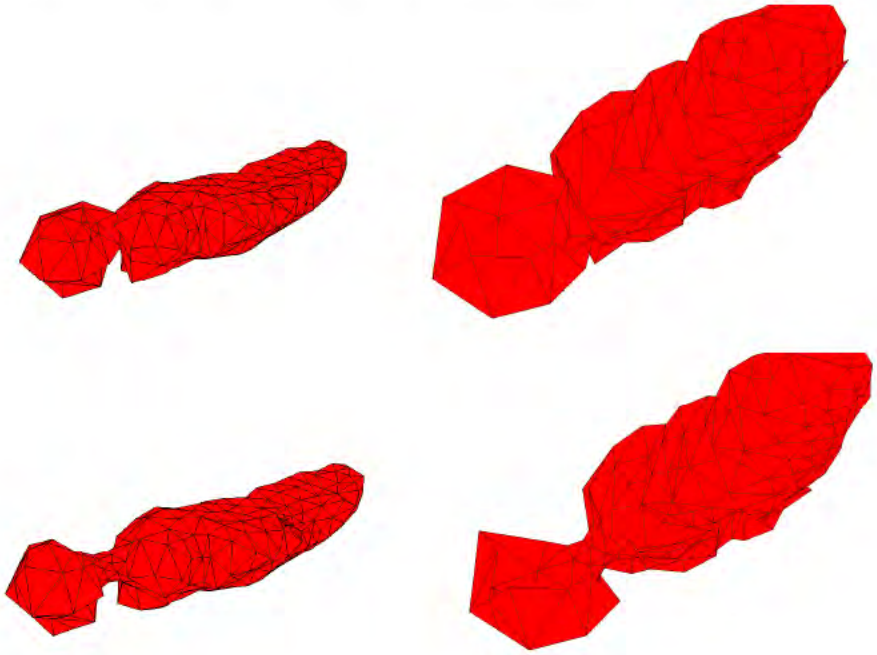

FIG. 13. *Two body meshes: self-intersection detection and topology adaptation. Top row: the self-intersected mesh. Bottom row: the resulting mesh after application of topology adaptation.*

as shown in the top row left image in Figure 13. We detect the self intersection and apply the topology adaptation algorithm. The bottom row left images in Figure 13 shows the resulting mesh. To better visualize the result, we also display the cutaway views of surface meshes of the original self-intersected mesh and the resulting mesh in Figure 13.

4.2. Residual shape optimization using time-harmonic boundary data. We consider the inverse boundary value problem for the Helmholtz equation on the connected domain Ω. The wavespeed is assumed to be piecewise constant on a given domain partitioning, with subdomains Ω_j. Thus the residual shape optimization problem amounts to determining the boundaries, $\Gamma_j = \partial\Omega_j$, which define conormal singularities.

For one subdomain, which we indicate by Ω for simplicity of notation, we assume that the wavefield u has limits u^+ and u^- from the exterior and interior of Γ, while $q(x) = \omega^2 c^{-2}(x)$ takes the values $q_1(x)$ and $q_2(x)$, yielding the following equations

$$
\left\{
\begin{array}{rcll}
(-\Delta - \quad q_0)u & = & 0, & x \in \Theta' \quad , \\
(-\Delta - q_2(x))u & = & f, & x \in \Theta \backslash \overline{\Omega} \quad , \\
(-\Delta - q_1(x))u & = & 0, & x \in \Omega \quad , \\
u^- - u^+ & = & 0, & x \in \Gamma \quad , \\
\dfrac{\partial u^-}{\partial \nu} - \dfrac{\partial u^+}{\partial \nu} & = & 0, & x \in \Gamma \quad , \\
\lim\limits_{r\to\infty} \left(r\dfrac{\partial u}{\partial r} - \mathrm{i}\sqrt{q_0}u \right) & = & 0 & \quad .
\end{array}
\right. \tag{4.2}
$$

We denote a bounded domain Θ and a open connected domain D with $\Omega_j \subset D \subset \Theta$. Let Σ be a open portion of ∂D, and suppose $\mathrm{supp}(f) \subseteq \Theta \backslash \overline{D}$. The energy functional is expressed with

difference between the solution operator $u_t = \mathcal{S}(\Omega_t; q_1, q_2)f$ and data d_f,

$$\mathcal{J}(\Omega_t, q_1, q_2) = \frac{1}{2}\|\mathcal{P}(\mathcal{S}(\Omega_t; q_1, q_2)f - d_f)\|^2_{L^2(\Sigma)}. \tag{4.3}$$

where $\mathcal{P}$ is an elliptic operator such that $\|\mathcal{P}\, d\|_{L^2(\Sigma)} = \|d\|_{H^{3/2}(\Sigma)}$.

The derivative with respect to t is given by,

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{J}(\Omega_t, q_1, q_2)\,\Big|_{t=0} = \mathrm{Re}\int_\Sigma [\mathcal{P}^*\mathcal{P}(\mathcal{S}(\Omega_t; q_1, q_2)f - d_f)]\ \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{S}(\Omega_t; q_1, q_2)f\,\Big|_{t=0}\,\mathrm{d}\sigma. \tag{4.4}$$

Using an augmented Lagrangian and employing the adjoint state method, we obtain

$$D\mathcal{J}(\Omega, q_1, q_2)\,V = \mathrm{Re}\left\{\int_\Gamma (q_1 - q_2)\, u\bar{w}\,\vec{n}\cdot V\,\mathrm{d}\sigma\right\} = \int_\Gamma \rho v\,\mathrm{d}\sigma, \tag{4.5}$$

where $\bar{w} \in H^1(\Theta)$ is the solution of the equations

$$\left\{\begin{array}{rcll} (-\Delta - \quad q_0)\bar{w} &=& 0, \quad w \in \Theta' & , \\ (-\Delta - q_2(x))\bar{w} &=& [\mathcal{P}^*\mathcal{P}(\mathcal{S}(\Omega_t; q_1, q_2)f - d_f)]\ \delta_\Sigma, \quad w \in \Theta\backslash\overline{\Omega} & , \\ (-\Delta - q_1(x))\bar{w} &=& 0, \quad w \in \Omega & , \\ \bar{w}^- - \bar{w}^+ &=& 0, \quad w \in \Gamma & , \\ \dfrac{\partial \bar{w}^-}{\partial \nu} - \dfrac{\partial \bar{w}^+}{\partial \nu} &=& 0, \quad w \in \Gamma & , \\ \lim\limits_{r\to\infty}\left(r\dfrac{\partial \bar{w}}{\partial r} - \mathrm{i}\sqrt{q_0}\bar{w}\right) &=& 0 & . \end{array}\right. \tag{4.6}$$

On a triangular surface mesh we use the discretization introduced in Subsection 3.2.

In the optimization, we simultaneously update q_1 and q_2, the updating of which can be denoted by

$$\frac{\mathrm{d}}{\mathrm{d}t}q_t\,\Big|_{t=0} \quad \text{with} \quad v.$$

Where q_t is the updating family of wavespeed models with $q_0 = q$, v is normal velocity on Γ. Then, in the shape derivative, $\rho = -u\bar{w}$. In our case of piecewise constant models, we assume a basis $\{\psi_\alpha\}_{\alpha=1}^{N_q}$ adapted to Ω

$$q(x) = \sum_{\alpha=1}^{N_q} \gamma_\alpha \psi_\alpha(x).$$

Then

$$D\mathcal{J}(\Omega)\,V = -\,\mathrm{Re}\sum_{\alpha=1}^{N_q}\left(\int_\Theta u_k \bar{w}_k\,\psi_\alpha\,\mathrm{d}x\right)v,$$

and

$$\frac{\mathrm{d}}{\mathrm{d}t}\gamma_{\alpha,t}\,\Big|_{t=0} \quad \text{with} \quad v \in \mathbb{R}^{N_q},$$

which gives the expression for ρ based on the piecewise constant basis.

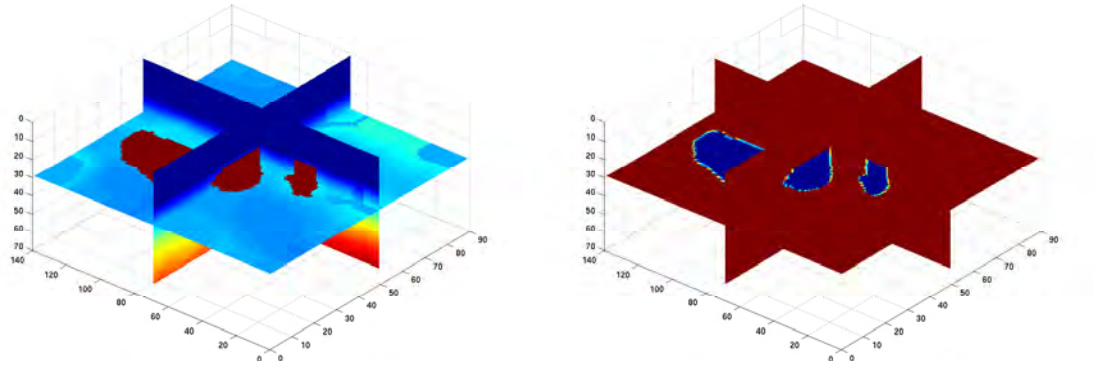

FIG. 14. *Image of the louro3d wavespeed model (left) and salt bodies segmentation function (right).*

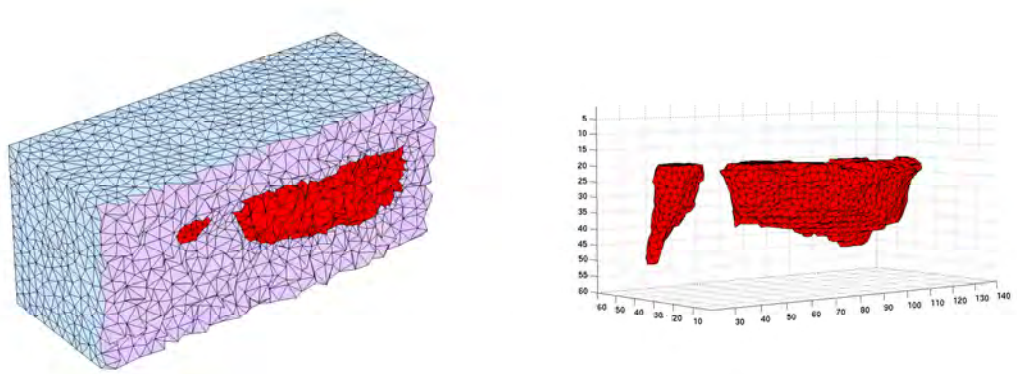

FIG. 15. *Louro3d model mesh: the tetrahedral mesh cutaway view (62k elements)(left) and the salt shape mesh (right).*

5. Numerical experiments. We carry out numerical experiments with a three-dimensional model, which we refer to a louro3d (courtesy Total) and which contains two salt bodies of different sizes. An image of the model is shown in Figure 14 (left); the wavespeed varies from 1517 m/s to 4500 m/s. The level set function providing the segmentation of the salt bodies is shown in Figure 14 (right). With the salt bodies' boundary information, we generate the tetrahedral mesh displayed in Figure 15 (left) using different colors to indicate the salt and non-salt regions; the salt bodies' mesh is displayed in Figure 15 (right).

5.1. Shape optimization controlled mesh deformation: geobodies. To obtain an initial shape Ω_0 in the shape optimization procedure, we carry out a coarse-scale full waveform inversion for wavespeed using a structured, regular grid. The result is displayed in Figure 16.

We then carry out an initial model based segmentation and mesh generation, and perform a residual shape optimization via mesh deformation using the data. In Figure 17, we display the direct mesh generation of the initial shape Ω_0, including the salt bodies' mesh and the tetrahedral mesh. Then we employ the iterative mesh evolution with the descent direction derived from the shape optimization using the data and three frequencies from low to high with 5 iterations for each frequency. The results are shown in Figure 18, 19 and 20. We perform salt surface mesh refinement and tetrahedral mesh remeshing at each new frequency. The zoom-in cutaway views in the final result are shown in Figure 21.

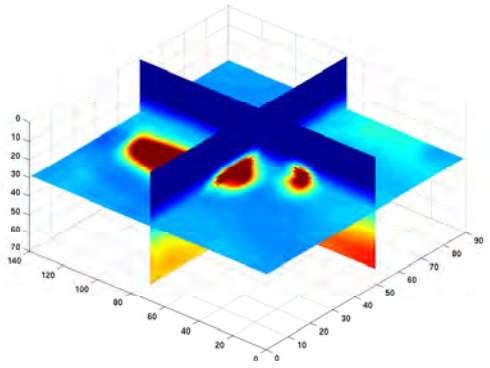

FIG. 16. *FWI inverted model as the initial coefficients.*

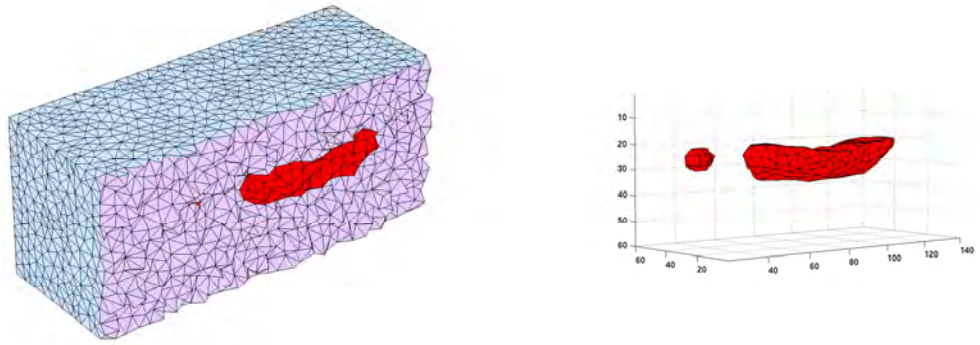

FIG. 17. *Initial mesh for shape optimization: the tetrahedral mesh cutaway view (left) and the initial salt mesh (right).*

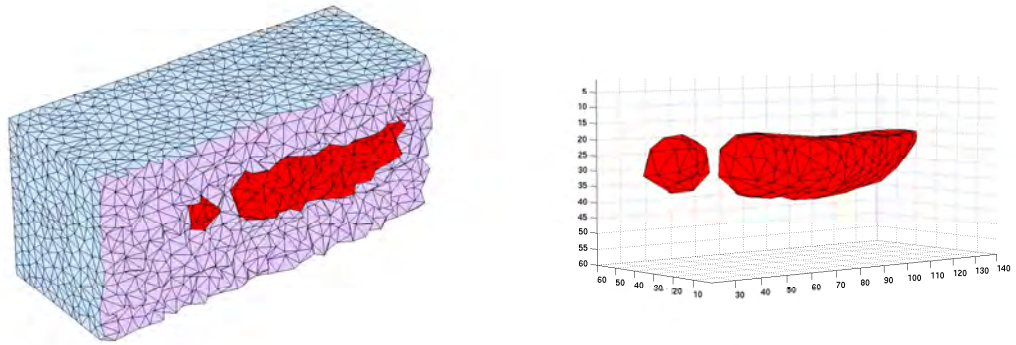

FIG. 18. *Shape and mesh after the first frequency: the tetrahedral mesh cutaway view (left) and the evolved salt mesh (right).*

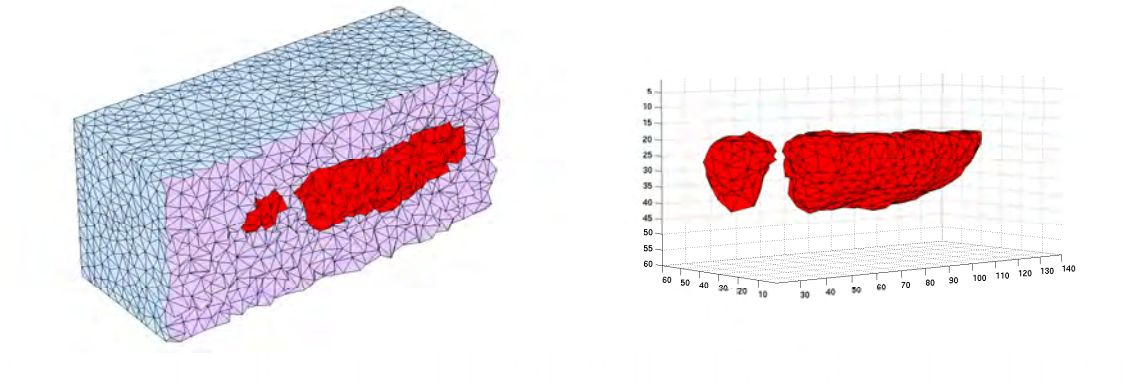

FIG. 19. *Shape and mesh after the second frequency: the tetrahedral mesh cutaway view (left) and the evolved salt mesh (right).*

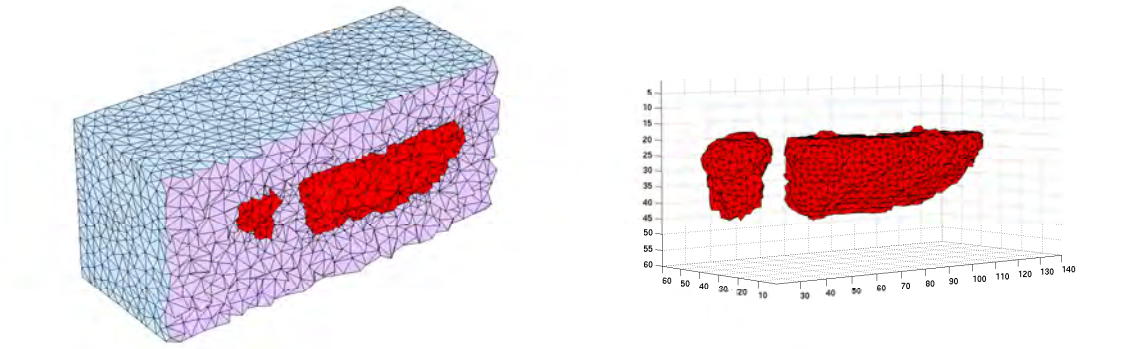

FIG. 20. *Shape and mesh after the third frequency: the tetrahedral mesh cutaway view (left) and the evolved salt mesh (right).*

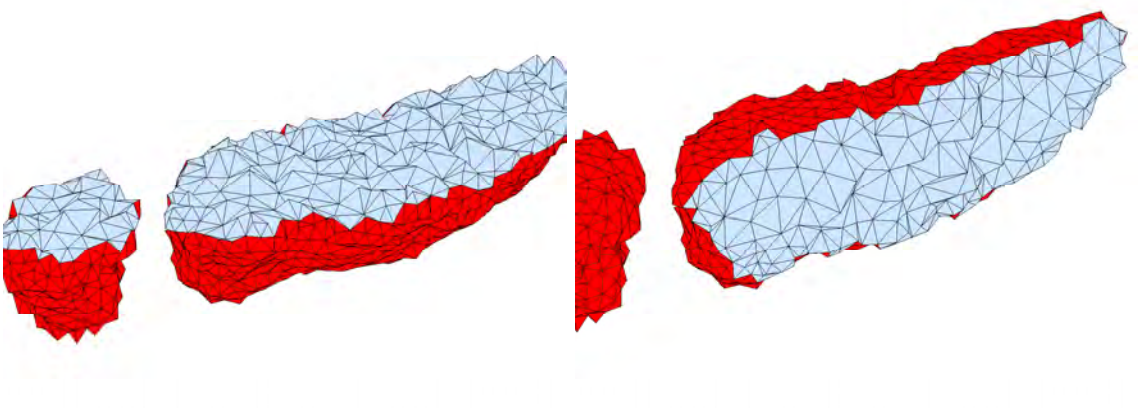

FIG. 21. *The zoom-in cutaway views of the resulting mesh.*

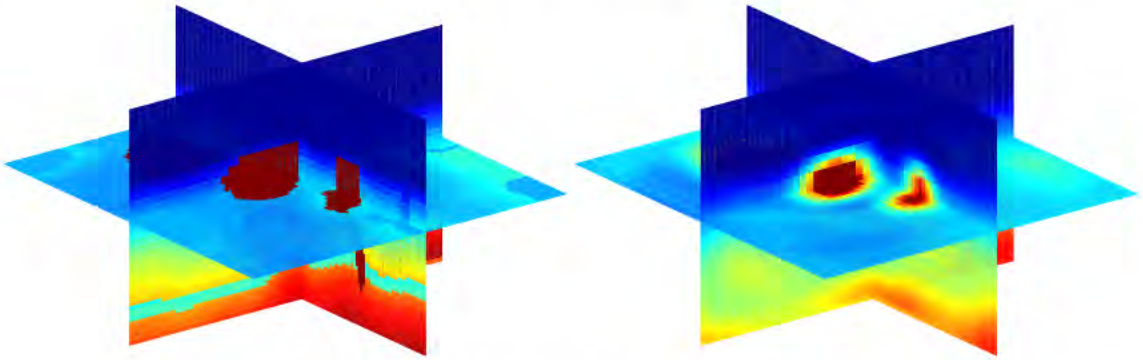

FIG. 22. *The modified louro3d wavespeed model (left) and initial model (right).*

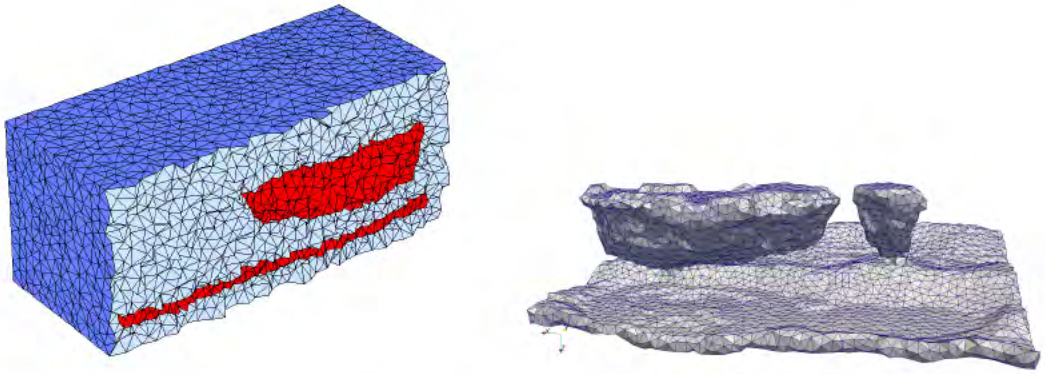

FIG. 23. *The modified louro3d model mesh: cutaway view (left) and the salt-layer surface mesh (right).*

5.2. Shape optimization controlled mesh deformation: geobody and layer. Next, we test our procedure with a salt body and a layer. We modify the louro3d model by adding a layer at the bottom; Figure 22 (left) shows an image of the modified louro3d wavespeed model. Figure 23 shows the mesh of the modified louro3d model. We follow the same procedure as in the first example by employing FWI to obtain an initial model. Figure 22 (right) shows an image of the outcome of FWI, that is, our initial model. We then perform the shape optimization based mesh deformation. The results are shown in Figure 24.

6. Discussion. We introduced adaptive domain partitioning via the generation of unstructured tetrahedral meshes based on segmentation, and their deformation based on (residual) shape optimization, in the parametrizations of the wavespeed in seismic full waveform inversion. The unstructured meshes are designed to obtain piecewise constant approximations to the ('true') wavespeed model. Our approach aims to update these (periodically) during the iterations in the inversion. It can be integrated in a multi-level, multi-frequency scheme [14] via progressive local mesh refinement, starting from a coarse mesh. Mesh refinement also enables the direct use of these unstructured meshes in finite element computations both in time, for example, following a discontinuous Galerkin method [17, 40, 41], and in frequency, following a continuous Galerkin method.

7. Acknowledgements. This research was partially supported by the National Science Foundation under grant CMG-1025259 and partially by the members of GMIG at Purdue University, BGP, ExxonMobil, PGS, Statoil and Total.

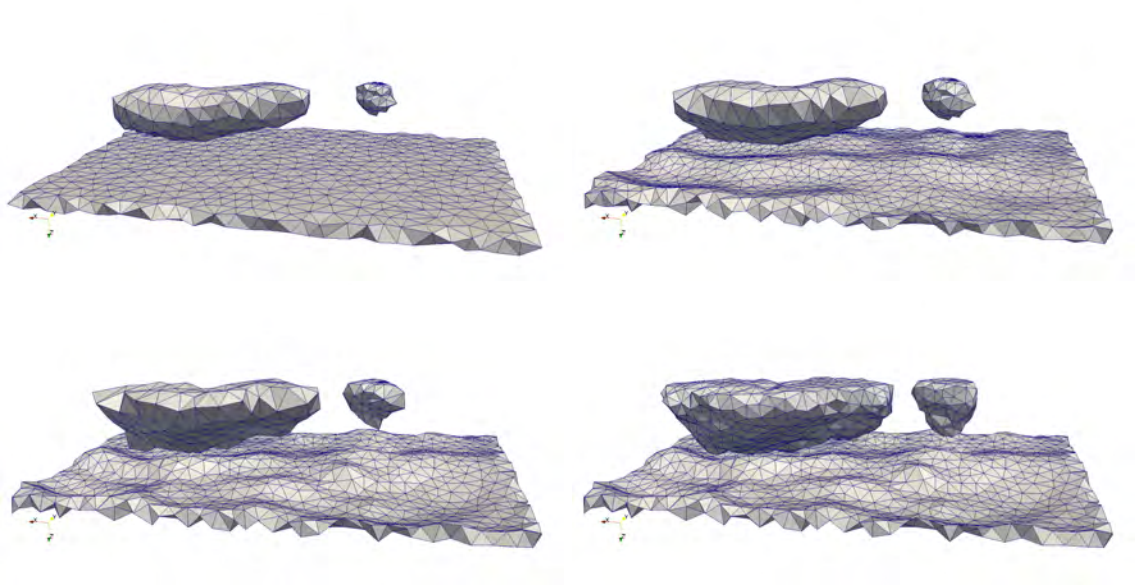

FIG. 24. *The initial and the first 3 iterations of the shape optimization and salt-layer mesh deformation.*

REFERENCES

[1] J. ABHAU AND O. SCHERZER, *A combinatorial method for topology adaptations in 3d deformable models*, Int. J. of Computer Vis., 87(3) (2010), pp. 304–315.

[2] M. J. AFTOSMIS, M. J. BERGER, AND J. E. MELTON, *Adaptive cartesian mesh generation*, 1999.

[3] E. BERETTA, M. DE HOOP, AND L. QIU, *Lipschitz stability of an inverse boundary value problem for a schrödinger type equation*. preprint (2013).

[4] M. J. BERGER, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 82 (1989), pp. 64–84.

[5] G. BÖHM, P. GALUPPO, AND A. VESNAVER, *3d adaptive tomography using delaunay triangles and voronoi polygons*, Geophysical Prospecting, 48 (2000), p. 723C744.

[6] I. BONDAR, *Seismic horizon detection using image processing algorithms*, Geophysical Prospecting, 40 (1992), pp. 785–800.

[7] ———, *Surface slice generation and interpretation: A review*, Lead Edge, 15 (1996), pp. 818–819.

[8] M. BURGER, *A framework for the construction of level set methods for shape optimization and reconstruction*, Interfaces and free boundaries, 5 (2002), pp. 301–329.

[9] J. CANNY, *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8 (1986), pp. 679–698.

[10] A. CHEN, T. WITTMAN, A. TARTAKOVSKY, AND A. BERTOZZI, *Efficient boundary tracking through sampling*, Appl. Math. Res. Express., 2011 (2011), pp. 182–214.

[11] L. CHEW, *Guaranteed-quality triangular meshes*, Technical Report TR-89-983, Department of Computer Science, Cornell University, (1989).

[12] ———, *Guaranteed-quality delaunay meshing in 3d*, 1993, pp. 391–393.

[13] ———, *Guaranteed-quality mesh generation for curved surfaces*, 1993, pp. 274–280.

[14] M. DE HOOP, L. QIU, AND O. SCHERZER, *A convergence analysis of a multi-level projected steepest descent iteration for nonlinear inverse problems in banach spaces subject to stability constraints*. preprint (2012).

[15] A. DELAUNOY AND E. PRADOS, *Gradient flows for optimizing triangular mesh-based surfaces: Applications to 3d reconstruction problems dealing with visibility*, International Journal of Computer Vision, (2011).

[16] O. DORN, E.L. MILLER, AND C.M. RAPPAPORT, *A shape reconstruction method for electromagnetic tomography using adjoint fields and level sets*, Inverse Problems, 16 (2000), pp. 1119–1156.

[17] M. DUMBSER AND M. KÄSER, *An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes ii. the three-dimensional isotropic case*, Geophysical Journal International, 167 (2006), pp. 319–336.

[18] I. ECKSTEIN, J.-P. PONS, Y. TONG, C.-C. J. KUO, AND M. DESBRUN, *Generalized surface flows for mesh processing*, 2007.

[19] M. FARAKLIOTI AND M. PETROU, *Horizon picking in 3d seismic images*, in SCIA'03 Proc. of the 13th Scandinavian conference on Image analysis, 2003, pp. 216–222.

[20] Z. Guo and M. de Hoop, *Shape optimization in seismic inverse problems with sparse blocky model representations*, Geo-Mathematical Imaging Group (GMIG):2012 Project Review, (2012).
[21] D. Hale, *Image-guided blended neighbor interpolation of scattered data*, CWP Report, 634 (2009), p. 634.
[22] E. Hinz and J. Bradford, *Ground-penetrating-radar reflection attenuation tomography with an adaptive mesh*, Geophysics, 75 (2010), pp. WA251–WA261.
[23] Z. Jin and A. Bertozzi, *Environmental boundary tracking and estimation using multiple autonomous vehicles*, 46th IEEE Conference on Decision and Control, (2007), pp. 4918–4923.
[24] P. Lelièvre, C. Farquharson, and C. Hurich, *Computing first-arrival seismic traveltimes on unstructured 3-d tetrahedral grids using the fast marching method*, Geophys. J. Int., 184 (2011), pp. 885–896.
[25] C. Li, C. Xu, C. Gui, and M.D. Fox, *Distance regularized level set evolution and its application to image segmentation*, IEEE Trans. Image Process., 19 (2010), pp. 3243–3254.
[26] G. L. Miller, D. Talmor, S.-H. Teng, N. Walkington, and H. Wang, *Delaunay based numerical method for three dimensions: Generation, formulation, and partition*, 1995.
[27] S. Osher and J. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations*, J. Comput. Phys., 70 (1988), pp. 12–49.
[28] P. Persson, *Mesh generation for implicit geometries*, PhD thesis, Massachusetts Institute of Technology, (2005).
[29] T. Plewa, t. Linde, and V. Weirs, *Adaptive Mesh Refinement, Theory and Applications*, Mathematics, Springer, 2005.
[30] A. Rüger and D. Hale, *Meshing for velocity modeling and ray tracing in complex velocity fields*, Geophysics, 71 (2006), pp. U1–U11.
[31] J. Ruppert, *A delaunay refinement algorithm for quality 2-dimensional mesh generation*, Journal of Algorithms, 18(3) (1995), pp. 548–585.
[32] J. Schöberl, *Netgen - an advancing front 2d/3d mesh generator based on abstract rules*, Comput. and Vis. in Science,, 1 (1997), pp. 41–52.
[33] J. Sethian, *A fast marching level set method for monotonically advancing fronts*, J. Comput. Phys., 114 (1994), pp. 146–159.
[34] J. Shewchuk, *Tetrahedral mesh generation by delaunay refinement*, in Proc. 14th Annu. ACM Sympos. Comput. Geom, 1998, pp. 86–95.
[35] H. Si, *Tetgen: A quality tetrahedral mesh generator and three-dimensional delaunay triangulator*.
[36] J. Sokolowski and J.P. Zolesio, *Introduction to shape optimization: shape sensitivity analysis*, vol. 16 of computational mathematics, Springer-Verlag, Berlin, 1992.
[37] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solutions to incompressible two-phase flow*, J. Comput. Phys., 114 (1994), pp. 146–159.
[38] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross, *Optimized spatial hashing for collision detection of deformable objects*, in In Proceeding of vision, modeling, visualization, 2003, pp. 47–54.
[39] N.P. Weatherill and O. Hassan, *Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints*, Int. J. for Num. Meth. in Eng., 37 (1994), pp. 2005–2039.
[40] L. Wilcox, G. Stadlera, C. Bursteddea, and O. Ghattas, *A high-order discontinuous galerkin method for wave propagation through coupled elasticcacoustic media*, Journal of Computational Physics, 229 (2010), pp. 9373–9396.
[41] R. Ye, C. Petrovitch, M. de Hoop, L.J. Pyrak-Nolte, L. Wilcox, and Y. Situ, *Parallel discontinuous galerkin method for modelling acoustic-elastic waves: from microstructure to seismology*. preprint (2013).