Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 2730829, 15 pages
https://doi.org/10.1155/2018/2730829

Research Article

Robust Three-Dimensional Level-Set Method for Evolving Fronts on Complex Unstructured Meshes

Ran Wei, Futing Bao, Yang Liu, and Weihua Hui

Science and Technology on Combustion Internal Flow and Thermal-structure Laboratory, School of Astronautics, Northwestern Polytechnical University, Xi'an 710072, China

Correspondence should be addressed to Weihua Hui; zhongyuancao@163.com

Received 14 June 2018; Accepted 10 September 2018; Published 25 September 2018

Academic Editor: Manuel Pastor

With a purpose to evolve the surfaces of complex geometries in their normal direction at arbitrarily defined velocities, we have developed a robust level-set approach which runs on three-dimensional unstructured meshes. The approach is built on the basis of an innovative spatial discretization and corresponding gradient-estimating approach. The numerical consistency of the estimating method is mathematically proven. A correction technology is utilized to improve accuracy near sharp geometric features. Validation tests show that the proposed approach is able to accurately handle geometries containing sharp features, computation regions having irregular shapes, discontinuous speed fields, and topological changes. Results of the test problems fit well with the reference results produced by analytical or other numerical methods and converge to reference results as the meshes refine. Compared to level-set method implementations on Cartesian meshes, the proposed approach makes it easier to describe jump boundary conditions and to perform coupling simulations.

1. Introduction

We are interested in evolving fronts on complex unstructured meshes in their normal direction using the level-set approach [1]. The level-set approach has been applied to various evolving-front problems. In particular, for evolving in the normal direction, the examples include image segmentation [2] and phase transition of solid materials, such as burning [3, 4] and exploding [5]. The level-set approach uses a higher-dimensional function ϕ to model the evolving front, where the $\{\vec{x} \mid \phi(\vec{x}) = 0\}$ isosurface denotes the front. Since the implicit representation automatically handles topological changes, the level-set approach is capable of processing arbitrary geometry and evolving speed.

The level-set method is usually applied on uniform Cartesian structured meshes. Advantages of Cartesian meshes include that they are easy to generate and that numerical technologies to solve Hamilton-Jacobi equations on Cartesian meshes are mature. When applied to image segmentation, the Cartesian mesh is naturally composed of the pixels.

However, when solid materials are involved in the simulation, there are cases in which non-Cartesian meshes outperform Cartesian ones, such as the following: (a) Describing complex boundary conditions. Jump conditions across the solid boundaries and different surface patches can be critical [6]. In this situation, tetrahedral or hexahedral meshes can be generated to fit the boundaries directly, while Cartesian meshes must be treated specially to handle the complex boundary conditions. One example of this issue is [7], where additional level-set functions are applied to address different boundaries, thus introducing more complexity. (b) Multi-phase coupling simulation. Stress, fluid, and thermal coupling simulations are often required in addition to evolving-front ones. Many such simulations are easier to perform on non-Cartesian meshes. If an evolving-front simulation is run on Cartesian meshes, interpolating among different meshes would be inevitable. (c) For "sparse" solid objects. Ordinary Cartesian meshes fill a rectangular bounding box of the evolving front. For "sparse" solid parts (e.g., the "L" or ring-shaped ones), a number of Cartesian cells, which are outside the object, would never be swept by the front of concern. However, storing, operating, or skipping these cells costs extra computational resources and consequently lowers the operation efficiency.

It is clear from the above discussion that tetrahedral and hexahedral meshes outperform Cartesian ones in evolving fronts with complex boundary conditions or in coupling simulations. Using the level-set method to evolve fronts essentially requires solving the Hamilton-Jacobi equation formed like $\partial\phi/\partial t + H(\nabla\phi, \vec{x}) = 0$. Previous efforts to solve such equations on unstructured meshes have been focused on two-dimensional (2D) triangular meshes. In [6, 8, 9], various numerical methods were proposed to solve the equation. In [10], the authors have studied handling boundary conditions for Hamilton–Jacobi equations on triangular meshes. There have also been studies on high-order numerical methods for the problem: In [11–13], WENO and Hermite WENO reconstruction methodologies were studied separately on 2D unstructured meshes; Augoula [14] developed several high-order numerical approximations for first-order Hamilton Jacobi equations on triangular meshes on the base of the Lax-Friedrichs scheme. Zhang [15] and Rokicki [16] then extended WENO reconstruction to 3D tetrahedral meshes.

Efforts to extend the level-set method to 3D tetrahedral meshes include [17, 18]. Morgan [17] developed 3D level-set methods for evolving fronts on tetrahedral meshes with adaptive mesh refinement, which are suitable for both advecting the level-set field and for evolving a front in the surface normal direction. Fu [18] mainly worked on applying parallel computing technology to speed up level-set solving. We have noticed that both of the aforementioned studies estimate the gradient of ϕ within a tetrahedral cell on the basis of the divergence theorem, i.e., $\nabla\phi = (1/V_{cell})\oint_{S_{cell}}(\phi\vec{n})\mathrm{d}s$. The method, though appearing to be stable and convergent within smooth regions, produces an $O(1)$ error near corners and edges. The error produced is so large that the sharp features would be smeared out during evolving and reinitializing. Another problem caused by sharp features is that the up-wind discretization [17] fails near the angle bisector of edges or corners. On different sides of the angle bisector, $\nabla\phi$ is discontinuous. The discontinuity may lead ϕ_x^- and ϕ_x^+ to differ in sign, and eventually causes simple up-wind gradient estimation to fail to obtain the correct domain of dependence [19] p57. The problem of the up-wind gradient is not significant when the level-set method is applied to flow simulation, since fluids can hardly develop edges or corners during their evolution. However, if we need to evolve fronts with sharp features, the broken up-wind gradient can be a serious problem.

This article addresses development of a robust level-set approach to evolve geometries having sharp features on tetrahedral meshes, but the proposed approach can also be easily extended to hexahedral meshes because each hexahedral cell can be decomposed into five or six tetrahedrons without introducing new vertices [20]. In order to solve the two above-mentioned problems introduced by sharp features, we have developed an innovative spatial discretization scheme and a gradient-estimating technology matching the scheme. The new spatial discretization imitates the majority of behaviors of the Godunov's scheme [21], which ensures numerical stability in discontinuous regions without introducing artificial viscosity like the Lax-Friedrichs or the Roe-Fix schemes. The gradient-estimating technology is more flexible than the divergence theorem-based method, thus allowing us to estimate the gradient of ϕ on any specific side of a vertex. This property is important to the proposed spatial discretization, which requires to perform gradient estimation on specifiable regions.

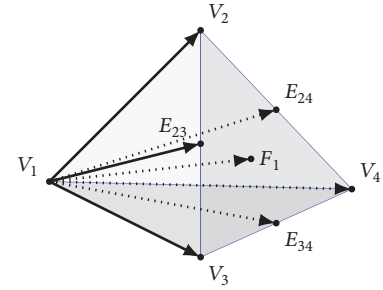


Figure 1: Tetrahedral cell nomenclature demonstration.

Compared to previous works, our level-set approach has the following advantages: (a) It is robust on body-fitted tetrahedral meshes of complex geometries, including those composed of multiple materials; (b) Sharp features are preserved during evolving; (c) No ghost cell or similar tricks are required, meaning that the level-set evolving and coupled physical simulations could be run on the same mesh; (d) The $\nabla\phi$ estimation used for evolving and reinitializing are obtained through unified flow, so the dual grid computation [17] is evaded. These properties imply that our approach has utility in a wide range of applications.

2. Nomenclature

The nomenclature used in this paper is illustrated by Figure 1. Within each cell, the four vertices are denoted by V_i, where $i \in \{1, 2, 3, 4\}$ should be attached to the vertices such that $\overrightarrow{V_2V_1} \cdot (\overrightarrow{V_2V_4} \times \overrightarrow{V_2V_3}) > 0$. In Figure 1, V_1 is the vertex for which we need to estimate $\nabla\phi_{V_1}$. Such a vertex is referred as **target vertex** later. The midpoint between V_i and V_j is referred to as E_{ij}. The barycenter of triangle facet $V_jV_kV_l$ is referred to as F_i, where $\{i\} \cup \{j, k, l\} = \{1, 2, 3, 4\}$. A vector which goes from the target vertex to any of the neighboring vertices, midpoints, or barycenters P is referred as an **outgoing vector** to P. $S(V_1)$ is the set composed of all outgoing vectors that start from V_1.

The signed distance field function of the evolving front is defined on each of the vertices and referred to as $\phi(\vec{V}_i)$. The linear estimation of the directional derivative of ϕ along the outgoing vector to P is referred to as $R_c(P)$ or $R_c(\overrightarrow{V_1P})$, e.g., $R_c(V_2) = (\phi(V_2) - \phi(V_1))/|V_1V_2|$, $R_c(\overrightarrow{V_1F_1}) = (\phi(F_1) - \phi(V_1))/|V_1F_1|$. R_c denotes the changing rate of ϕ.

3. Governing Equations

The level-set equation Eq.(1) describes the movement of the isosurface $\{\vec{x} \mid \phi(\vec{x}) = 0\}$, where $u_n(\vec{x})$ is the normal speed field at which the front evolves.

Usually, $u_n(\vec{x})$ is not constant across the entire field, so ϕ would no longer be an exact signed distance field after each time-forward iteration step, and we have to reinitialize ϕ regularly. The reinitialization is performed via Eq.(2) [22], where τ is artificial time and $S(\phi) = \phi/\sqrt{\phi^2 + \Delta x^2}$ (Δx is the cubic root of the volume of the smallest tetrahedral cell).

$$\frac{\partial \phi}{\partial t} + u_n\left(\vec{x}\right)\left|\nabla\phi\right| = 0 \tag{1}$$

$$\frac{\partial \phi}{\partial \tau} + S\left(\phi\right)\left(\left|\nabla\phi\right| - 1\right) = 0 \tag{2}$$

Equations (1) and (2) can be solved with respect to time via the Euler method or the total variation diminishing Runge-Kutta (TVD-RK) method. Both of the two time-discretization schemes are mature and straightforward, and therefore are not discussed in detail here. The core problem to solve in (1) and (2) is spatial discretization. Since handling sharp features and discontinuity is required, we have not used the simple up-wind scheme, but developed an innovative spatial discretization, which will be discussed in Section 4.1. The newly constructed estimating technologies will be presented in Sections 4.2–4.4.

4. Numerical Method

4.1. Spatial Discretization. The Godunov's spatial discretization scheme is widely used in numerical calculation studies to handle discontinuity. It has different implementations in different situations. The dimension-by-dimension form, which is widely used in level set studies, can be summarized by the following Eq.(3) [19] p58:

$$C = \begin{cases} u_n & \text{During evolving} \\ \phi\left(\vec{x}\right) & \text{During reinitializing} \end{cases}$$
$$\phi_a^2 \approx \begin{cases} \max\left[\max\left(\phi_a^-, 0\right)^2, \min\left(\phi_a^+, 0\right)^2\right] & C > 0 \\ \max\left[\min\left(\phi_a^-, 0\right)^2, \max\left(\phi_a^+, 0\right)^2\right] & C < 0 \end{cases} \tag{3}$$

where a is any of axes, and C is the coefficient of the Hamiltonians. $|\nabla\phi|$ is then computed via:

$$\overline{\left|\nabla\phi\right|} = \sqrt{\sum_{a\in\{x,y,z\}} \phi_a^2} \tag{4}$$

While being widely used in previous studies, the above scheme triggers two problems when applied to our study: (a) With non-Cartesian meshes, Eq.(3) cannot be directly applied due to that the cell edges are not axis-aligned. (b) In the vicinity of sharp features (e.g. 2D corners, 3D edges or corners), Eq.(3) may produce $\phi_x^2 \approx \phi_y^2 \approx \phi_z^2 \approx 1$ on however fine grid, leading Eq.(4) to output $\sqrt{2}$ or $\sqrt{3}$ and smearing sharp geometry features out during reinitialization.

Problem (a) can be settled by a newly developed edge-based estimation method, which will be presented later (Section 4.3). The method works on any specifiable side

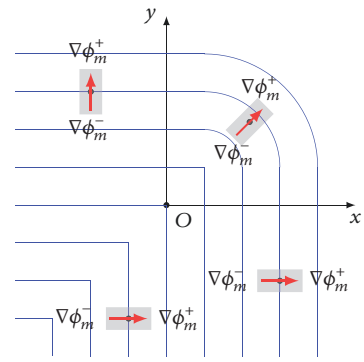


Figure 2: Main axes and domains of dependence of several target vertices.

of a vertex and directly estimate the gradient, instead of computing several directional derivatives like ϕ_x^- and ϕ_x^+. However, in order to solve problem (b) elegantly, we need a new spatial discretization scheme, which has to meet the following criteria:

(C.1) In the vicinity of sharp geometry, the scheme shall not introduce the $O(1)$ error like Eq.(4);

(C.2) At peaks or valleys of the field, the result shall tend to that of Eq.(3);

(C.3) In regions where ϕ_a^- and ϕ_a^+ have the same signs, the result shall tend to that of Eq.(3)-(4);

(C.4) In other regions where the situations on the axes are different (e.g. the field is smooth along one axis while there are peaks or valleys along other axes), the scheme shall perform the estimation within reasonable domains of dependence.

The newly developed scheme is presented below. We will prove that the presented scheme fits the above criteria later.

The proposed scheme defines the direction $\vec{m}$, where ϕ has the maximum directional derivative along $\vec{m}$, as the "main axis". Beside each vertex, two estimating operations are performed on direction $\vec{m}$ and $-\vec{m}$ respectively. Several main axes and corresponding domains of dependence are demonstrated in Figure 2, where thin blue lines are isolines of ϕ, red arrows are main axes, and gray areas denote the domains of dependence where ϕ_m^- and ϕ_m^+ are estimated. Once the two gradient estimation are obtained, the following Eq.(5)-(6) are integrated to finally get $\overline{\nabla\phi}$.

$$t^- = C\left(\vec{m}\cdot\nabla\phi_m^-\right)$$
$$t^+ = C\left(\vec{m}\cdot\nabla\phi_m^+\right) \tag{5}$$

$$\overline{\nabla\phi} = \begin{cases} \nabla\phi_m^+ & t^- < 0 \text{ and } t^+ < 0 \\ \nabla\phi_m^- & t^- > 0 \text{ and } t^+ > 0 \\ \vec{0} & t^- < 0 \text{ and } t^+ > 0 \\ \nabla\phi_m^+ & t^- > 0 \text{ and } t^+ < 0 \text{ and } \left|\nabla\phi_m^+\right| > \left|\nabla\phi_m^-\right| \\ \nabla\phi_m^- & t^- > 0 \text{ and } t^+ < 0 \text{ and } \left|\nabla\phi_m^+\right| < \left|\nabla\phi_m^-\right| \end{cases} \tag{6}$$

For (**C.1**), it can be seen that it is satisfied only if the gradient estimation does not contain $O(1)$ error. In Section 4.4, we have deduced the root cause of the $O(1)$ error and proposed the "explicit correction" technology to ensure (**C.1**) to be satisfied.

For (**C.2**), peaks and valleys can be discussed separately. At peaks of the field, both $\nabla\phi_m^-$ and $\nabla\phi_m^+$ point to the target vertex from their respective dependent domains. Therefore, we always have $\nabla\phi_m^- \cdot \vec{m} > 0$ and $\nabla\phi_m^+ \cdot \vec{m} < 0$. Substituting the values into Eq.(5)-(6), it can be found that the behavior of the proposed scheme is in accordance with that of the Godunov-type scheme: when $C > 0$, either $\nabla\phi_m^-$ or $\nabla\phi_m^+$ is chosen to depend on which one has the maximum norm; when $C < 0$, $\vec{0}$ is outputted as $\overline{\nabla\phi}$. Similarly, at valleys of the field, we have $\nabla\phi_m^- \cdot \vec{m} < 0$ and $\nabla\phi_m^+ \cdot \vec{m} > 0$. When $C > 0$, $\vec{0}$ is outputted; when $C < 0$, the one having larger norm is chosen. It is evident that in both cases concerning (**C.2**), the proposed scheme behaves in the same way as the Godunov-type scheme, and (**C.2**) is satisfied.

In cases concerning (**C.3**), $\forall a \in \{x, y, z\}$, ϕ_a^- and ϕ_a^+ have the same signs. As a result, $\vec{m}$ would be appropriately parallel to $\nabla\phi_m^-$ and $\nabla\phi_m^+$, and $\vec{m} \cdot \nabla\phi_m^- > 0$, $\vec{m} \cdot \nabla\phi_m^+ > 0$. According to Eq.(5)-(6), the proposed scheme outputs $\nabla\phi_m^-$ when $C > 0$ and $\nabla\phi_m^+$ when $C < 0$. Meanwhile, if ϕ_a^- and ϕ_a^+ are both positive, $\vec{m}$ points to the positive side of axis a, thus the component of $\nabla\phi_m^-$ on a is approximately equal to ϕ_a^-, and that of $\nabla\phi_m^+$ is approximately ϕ_a^+. Considering Eq.(3), which takes ϕ_a^- when $C > 0$ and ϕ_a^+ when $C < 0$, it can be seen that the proposed scheme takes the same domain of dependence and gives close result compared to the Godunov-type approach. When ϕ_a^- and ϕ_a^+ are both negative, the case can be proved similarly. The requirement of (**C.3**) is satisfied.

In cases concerning (**C.4**), the proposed scheme operates in a slightly different manner with the Godunov-type scheme, but the behavior can be explained reasonably. Since that there are too many cases to analysis one by one, and that all cases can be explained via similar patterns, we only provide the analysis for two typical 2D cases, which are demonstrated in Figure 3. For both vertex A and B, it is clear that $\phi_x^- < 0$, $\phi_x^+ > 0$, $\phi_y^- > 0$, $\phi_y^+ > 0$. According to Eq.(3)-(4), when $C > 0$, $\sqrt{0^2 + (\phi_y^-)^2}$ will be taken as the estimation of $\overline{|\nabla\phi|}$. For A, ϕ_y equals to the norm of local gradient, so the estimation is reasonable. However, ϕ_y estimated from beside B is significantly smaller than the norm of actual gradient, and the above estimation introduces error.

On the other hand, the proposed scheme processes the two vertices in different manners. For A, $\vec{m}$ is parallel to the corner bisector, so both $\vec{m} \cdot \nabla\phi_m^-$ and $\vec{m} \cdot \nabla\phi_m^+$ are positive. Eq.(6) will choose $\nabla\phi_m^-$ when $C > 0$ and $\nabla\phi_m^+$ when $C < 0$, meaning that the proposed scheme is still in accordance with the Godunov-type scheme. However, for vertex B, $\vec{m}$ is perpendicular to one of the adjacent edges, depending on the numerical disturbance near B. In this situation, $\vec{m} \cdot \nabla\phi_m^+$ is always positive, but the sign of $\vec{m} \cdot \nabla\phi_m^-$ depends on the sharpness of the corner. If the corner is relatively sharp, $\nabla\phi_m^-$ would diverge so far from $\vec{m}$ that $\vec{m} \cdot \nabla\phi_m^- < 0$, and the proposed scheme will treat this situation as a valley. Conversely, if the corner is so smooth that the direction of $\nabla\phi_m^-$ is close to $\vec{m}$, both $\vec{m} \cdot \nabla\phi_m^-$ and $\vec{m} \cdot \nabla\phi_m^+$ would be positive. and the situation will be treated as a smooth region.

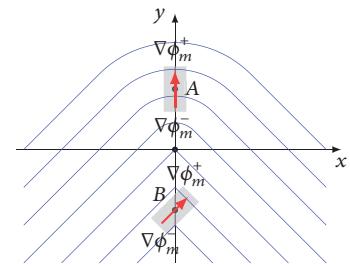


Figure 3: A sample case concerning (**C.4**): semi-concave region.

To sum up, the proposed scheme treats the vertex as a valley when "valley" is the dominant property, and otherwise chooses either side of the bisector as the information source to evolve the target vertex. Considering that the target vertex lies extremely close to the angle bisector, it may be influenced by the information propagated from either side. In some ways, choosing one side can be viewed as shifting the vertex towards the chosen side by a tiny distance, and thus ensuring the chosen side to be the theoretically correct one. Although the behavior introduces error for potentially incorrect choices, when the scheme is used for solving Eq.(1)-(2), the introduced error is small due to the symmetry across both sides of the bisector. Numerical experiments and validations show that the produced result is stable and accurate enough, implying the proposed scheme to satisfy (**C.4**) within the scope of this article.

A problem which remains unsolved is about establishing the main axis. Theoretically, $\vec{m}$ shall be established according to the central-difference estimation of the gradient. However, such operation requires building dual grid structure [17] and costs extra computational resource. In this paper, we establish $\vec{m}$ such that $\vec{m}$ is parallel to the outgoing vector $\vec{M}$ which has the maximum $|R_c|$:

$$\vec{M} = \arg\max_{\vec{VP} \in S(V)} R_c\left(\vec{VP}\right) \tag{7}$$

For vertices adjacent to the $\phi = 0$ isosurface, since the computing is aimed at evolving the $\phi = 0$ isosurface, information is supposed to propagate from the isosurface to other regions of the mesh. Therefore, the correct domains of dependence are always on the nearer sides to the $\phi = 0$ isosurface, and the "explicit correction" technology is applied to obtain proper estimations for such vertices.

4.2. Cell-Based Gradient Estimation. Here, we briefly illustrate the problem of the cell-based approach before introducing our approach. The core of the cell-based estimating approach can be summarized by Eq. (8). The cell-centered

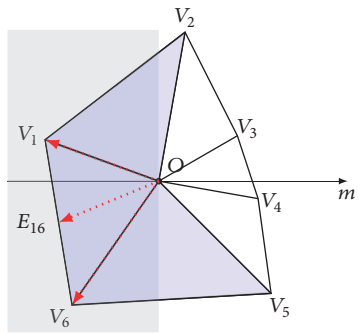


FIGURE 4: Domain of dependence of the edge- and cell-based approaches. The gray region in the figure shows the ideal domain of dependence with which to estimate $\nabla_m \phi$. $\triangle OV_1V_2$, $\triangle OV_1V_6$, and $\triangle OV_5V_6$ overlap with the gray region. Including cells $\triangle OV_1V_2$ and $\triangle OV_5V_6$ in the calculation introduces information from V_2 and V_5, which are not supposed to be included since they lie outside the gray region. Excluding $\triangle OV_1V_2$ and $\triangle OV_5V_6$, however, results in a lack of robustness.

estimations of $\nabla\phi$ obtained from (8) can then be integrated into various difference schemes, such as the up-wind approach [17] and the WENO approach [15, 16]:

$$\nabla\phi = \frac{1}{V_{cell}} \oint_{S_{cell}} \left(\phi \vec{n}\right) \mathrm{d}s. \tag{8}$$

Such a cell-based approach, though used in a large portion of previous research, shows two major disadvantages when applied to feature-rich geometries: (a) The orientations of cells with respect to the target vertex cannot be well defined, making it difficult to apply the above-presented spatial discretization. The problem is illustrated in Figure 4. (b) As we have proved in the Appendix, cell-based estimation methods rely on a pre-condition that the gradient within one cell tends to be uniform as the mesh grows finer. The pre-condition is true only if the zero level set $\{\vec{x} \mid \phi(\vec{x}) = 0\}$ is $c1$ continuous or smoother. Near sharp features (where the zero level set is $c0$ continuous), the cell-based approach produces an $O(1)$ error.

In this article, we have developed the edge-based gradient estimation approach to solve problem (a), and the "explicit correction" technology to settle problem (b). The two methods is described in the following Sections.

4.3. Edge-Based Gradient Estimation. Assuming that $\nabla\phi$ is uniform within one differential volume, we know from the Appendix that $\forall\overrightarrow{VP}$ within this region, $R_c(\overrightarrow{VP}) \equiv \nabla\phi \cdot \overrightarrow{VP}/|\overrightarrow{VP}|$. Conversely, in N-dimensional space, we can use N non-linear correlative vectors and corresponding changing-rate values to uniquely identify $\nabla\phi$, as shown in Figure 5. This approach, which uses outgoing vectors instead of cells as sampling primitives to fetch gradient information, is referred to as the edge-based approach.

Since outgoing vectors are smaller primitives than cells, and all outgoing vectors pass the target vertex, the edge-based approach allows finer control over the domain of dependence compared to the cell-based approach. In Figure 4, the red vectors demonstrate acceptable outgoing vectors to estimate $\nabla_m \phi$. Obviously, for each outgoing vector, it is easy to unambiguously tell whether the outgoing vector belongs to the desired domain of dependence or not.

Let $\vec{e}$ be the vector that points to the desired domain of dependence. Linear equation set (9) estimates $\nabla\phi$ at point $\vec{O}$ upon the desired domain of dependence, where T_G is the threshold value that controls how close the select outgoing vectors should be to $\vec{e}$. For the outgoing vectors that end at midpoints or barycenters, the values of ϕ on their ends can be computed via interpolation.

$$\vec{V} \cdot \nabla\phi = \left|\vec{V}\right| R_c\left(\vec{V}\right) \quad \forall\vec{V} \in S(O), \text{ and } \vec{V} \cdot \vec{e} < T_G \tag{9}$$

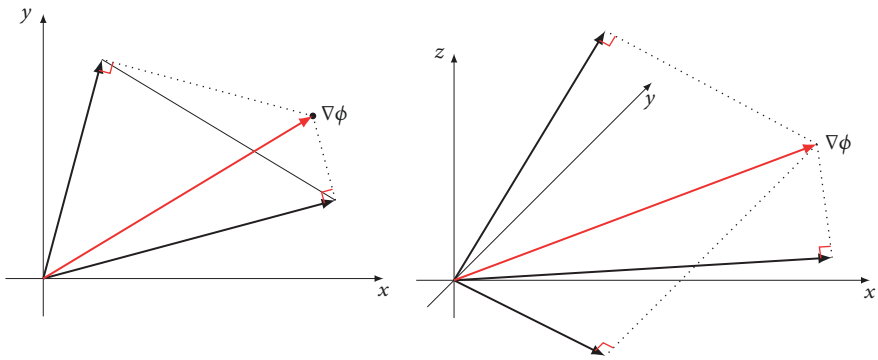
In order to solve (9) in N-dimensional space, N outgoing vectors meeting the threshold condition are required. Practically, there are often more outgoing vectors than required, especially in 3D meshes. In this case, an overdetermined linear system can be established to include all the available outgoing vectors. The system can then be solved via the least-squares method. For overdetermined linear systems, multiplying a constant value on both sides on an equation is equivalent to setting the weight of this equation in the system. We assign larger weights to the outgoing vectors that are closer to $\vec{e}$ by using $\vec{V} \cdot \vec{e}$ as the weight. The complete form of the estimating equation set can be described by (10):

$$\begin{aligned} w &= \vec{V} \cdot \vec{e} \\ w\vec{V} \cdot \nabla\phi &= w\left|\vec{V}\right| R_c\left(\vec{V}\right) \\ &\qquad \forall\vec{V} \in S(O), \text{ and } w < T_G \end{aligned} \tag{10}$$

Numerical consistency is a required property for discrete Hamiltonians [14]. The analysis below will demonstrate that Eq. (10) provides an exact estimation for linear fields. Letting ϕ_L be a linear field, we have

$$\phi_L\left(\vec{x}\right) = \phi_L\left(\vec{z}\right) + \left(\vec{x} - \vec{z}\right) \cdot \vec{g}, \tag{11}$$

where $\vec{z}$ is an arbitrary constant point and $\vec{g}$ is the gradient of field ϕ. We rewrite Eq. (10) as the matrix form $A\vec{r} = b$, where $\vec{r}$ is the unknown gradient estimation obtained from (10). Let V_0 be the target vertex and $V_1, V_2, \ldots, V_i$ be the ends of all



(a) 2D (b) 3D

FIGURE 5: Edge-based $\nabla\phi$ estimating.

acceptable outgoing vectors; then, A and b can be expressed by the following partitioned matrix:

$$A = \begin{bmatrix} w_1\overrightarrow{V_0V_1} \\ w_2\overrightarrow{V_0V_2} \\ \cdots \\ w_i\overrightarrow{V_0V_i} \end{bmatrix} \tag{12a}$$

$$\begin{aligned} b &= \begin{bmatrix} w_1 \left|V_0V_1\right| R_c\left(\overrightarrow{V_0V_1}\right) \\ w_2 \left|V_0V_2\right| R_c\left(\overrightarrow{V_0V_2}\right) \\ \cdots \\ w_i \left|V_0V_i\right| R_c\left(\overrightarrow{V_0V_i}\right) \end{bmatrix} \\ &= \begin{bmatrix} w_1\left[\phi_L\left(\overrightarrow{V_1}\right) - \phi_L\left(\overrightarrow{V_0}\right)\right] \\ w_2\left[\phi_L\left(\overrightarrow{V_2}\right) - \phi_L\left(\overrightarrow{V_0}\right)\right] \\ \cdots \\ w_1\left[\phi_L\left(\overrightarrow{V_i}\right) - \phi_L\left(\overrightarrow{V_0}\right)\right] \end{bmatrix} \end{aligned} \tag{12b}$$

Substituting (11) into (12b), we have

$$b = \begin{bmatrix} w_1\left(\overrightarrow{V_1} - \overrightarrow{V_0}\right)\cdot\overrightarrow{g} \\ w_2\left(\overrightarrow{V_2} - \overrightarrow{V_0}\right)\cdot\overrightarrow{g} \\ \cdots \\ w_1\left(\overrightarrow{V_i} - \overrightarrow{V_0}\right)\cdot\overrightarrow{g} \end{bmatrix} = Ag \tag{13}$$

As discussed above, we solve (10) via the least-squares method,

$$\overrightarrow{r} = \left(A^TA\right)^{-1}A^Tb \tag{14}$$

Substituting (13) into (14) we have

$$\overrightarrow{r} = \left(A^TA\right)^{-1}\left(A^TA\right)g = g \tag{15}$$

From (15), it is obvious that the proposed edge-based approach gives an exact estimation result for linear fields.

4.4. Gradient Estimation Near Sharp Features. While the edge-based method settles the problem of estimating $\nabla\phi$ upon the desired domain of dependence, the problem of the $O(1)$ error near sharp features remains unresolved (although using smaller T_G eases the problem). To achieve higher accuracy, it is obvious from the above discussion that the uniform gradient assumption must be discarded.

The root reason of the problem is that decreasing the cell size near sharp features does not improve the uniformity of the gradient within single cells. Since only the cells near sharp features (or more broadly, near the zero level set) are affected, a reasonable solution is integrating another estimating approach that does not rely on the uniform gradient assumption for these cells. One useful property of signed distance fields is that the zero level set is perpendicular to the local gradient. In discontinuous regions such as the outer sector of the corner in Figure 17(a), the direction of gradient starts from the corner and ends at the target vertex. Generally, for a vertex V that locates near the $\phi = 0$ isosurface, letting the nearest point to V on the isosurface be N, the corresponding gradient ϕ_V is parallel to $\overrightarrow{NV}$ or $\overrightarrow{VN}$. Assuming that the gradient distributes uniformly on line segment NV, the norm of the gradient equals to the changing rate on NV. The above process can be summarized by Eq. (16):

$$\begin{aligned} \overrightarrow{N} &= \underset{\forall\overrightarrow{N}\mid\phi(\overrightarrow{N})=0}{\arg\min}\left|NV\right|, \\ \frac{\overline{\nabla\phi}}{\left|\overline{\nabla\phi}\right|} &= \begin{cases} \dfrac{\overrightarrow{NV}}{\left|\overrightarrow{NV}\right|} & \phi\left(\overrightarrow{V}\right) > 0, \\ -\dfrac{\overrightarrow{NV}}{\left|\overrightarrow{NV}\right|} & \phi\left(\overrightarrow{V}\right) < 0, \end{cases} \\ \overline{\left|\nabla\phi\right|} &= \left|\frac{\phi\left(\overrightarrow{V}\right)}{NV}\right|. \end{aligned} \tag{16}$$

Input: $\overrightarrow{V}, \phi$

1: Cell set $C = \{C_i \mid \overrightarrow{V} \in C_i\}$
2: Vertex set $S = \{\overrightarrow{V_i} \mid \overrightarrow{V_i} \in C\}$
3: **if** $\forall \overrightarrow{V_i} \in S$, $\text{sign}(\phi(\overrightarrow{V_i})) = \text{sign}(\phi(\overrightarrow{V}))$ **then**
4: V is not adjacent to the level set. Quit procedure and launch edge-based approach.
5: **end if**
6: **for all** Cells $C_i \in C$ **do**
7: Find the line segment (2D) or facet (3D) Z_i in C_i belonging to $\{\overrightarrow{x} \mid \phi(\overrightarrow{x}) = 0\}$ via linear interpolation.
8: Record the minimum distance D and corresponding nearest point N
9: **end for**
10: **if** $\exists \overrightarrow{V_i} \in S$ that $|SN| < D$ **then**
11: Add neighboring vertices of V_i to S
12: Repeat step 6 - 11 until step 10 returns **false**.
13: **end if**
14: **return** $\overrightarrow{NV}, D$

Procedure 1: Search nearest point to $\overrightarrow{V}$ on $\{\overrightarrow{x} \mid \phi(\overrightarrow{x}) = 0\}$.

The problem is now reduced to locating the nearest point $\overrightarrow{N}$ for each target vertex near the level set. In [23], a brute-force approach is proposed to search $\overrightarrow{N}$ in all adjacent cells in an explicit manner. We have employed this approach, together with a minor improvement, to search $\overrightarrow{N}$ more reliably. The approach used in this article can be summarized by Procedure 1.

Compared to its original form, in Procedure 1 we have added steps 10 - 13 to ensure that the outputted nearest point is the globally nearest one. Other details about the procedure can be found in [23] and are not discussed here. The method proposed here is referred to as "explicit correction (EC)" below.

4.5. Overall Flow. The three core modules of our evolving approach, namely the spatial discretization, the edge-based method, and the explicit correction approach, were discussed above. Combining them, we have a robust approach to solve Eq.(1)-(2). Other aspects of our evolving flow are straightforward. The entire flow is demonstrated in Figure 6.

The time integration we have used to solve Eq.(1)-(2) is the two-stage Runge-Kutta method, which can be summarized by equation (17):

$$\begin{aligned} \phi_V^{n+1/2} &= \phi_V^n - \frac{1}{2}\Delta t H_V^n, \\ \phi_V^{n+1} &= \phi_V^n - \Delta t H_V^{n+1/2}. \end{aligned} \tag{17}$$

5. Validation

The following problems are tested to validate the accuracy and robustness of the proposed level-set method:

(i) (2D) Diffusion into a notched square. This example demonstrates the ability to evolve fronts at non-uniform evolving velocity on irregular regions.

(ii) (2D) Three merging circles. This example demonstrates the ability to handle topological changes.

(iii) (2D) Rate stick [5]. This example demonstrates the compatibility with slender and long regions.

(iv) (3D) Burning and erosion in a solid rocket motor. This example demonstrates the ability to evolve fronts at non-uniform evolving velocity on 3D irregular regions.

(v) (3D) Reconstructing a signed distance function [24]. This example demonstrates the convergence of the reinitializing operation.

For the merging-circles example, the reinitializing operation is not integrated since the evolving speed distributes uniformly. Other details of each test problem will be discussed in the corresponding subsections. We have already published the simulation code of all 2D validating problems [25].

The error metric we have used to measure the accuracy is the averaged and maximum errors of ϕ near the zero level set, which are defined as equation (18),

$$\begin{aligned} S &= \left\{\overrightarrow{V_i} \mid \phi_{exact}\left(\overrightarrow{V_i}\right) < 5\Delta x\right\}, \\ E_a &= \frac{1}{n(S)}\sum_{i=1}^{n(S)} \left|\phi_{sim}\left(\overrightarrow{V_i}\right) - \phi_{exact}\left(\overrightarrow{V_i}\right)\right|, \\ E_m &= \max_{\overrightarrow{V_i}\in S} \left|\phi_{sim}\left(\overrightarrow{V_i}\right) - \phi_{exact}\left(\overrightarrow{V_i}\right)\right|, \end{aligned} \tag{18}$$

where Δx is the scale of cells. Unless otherwise specified, $\Delta x = 0.01$ in the following examples. The 2D meshes used for simulation are generated by the mesh2d [26] toolkit, while the 3D meshes are generated by Gmsh [27].

5.1. Diffusion into a Notched Square. This problem describes the diffusion starting at the upper edge of a U-notched square.

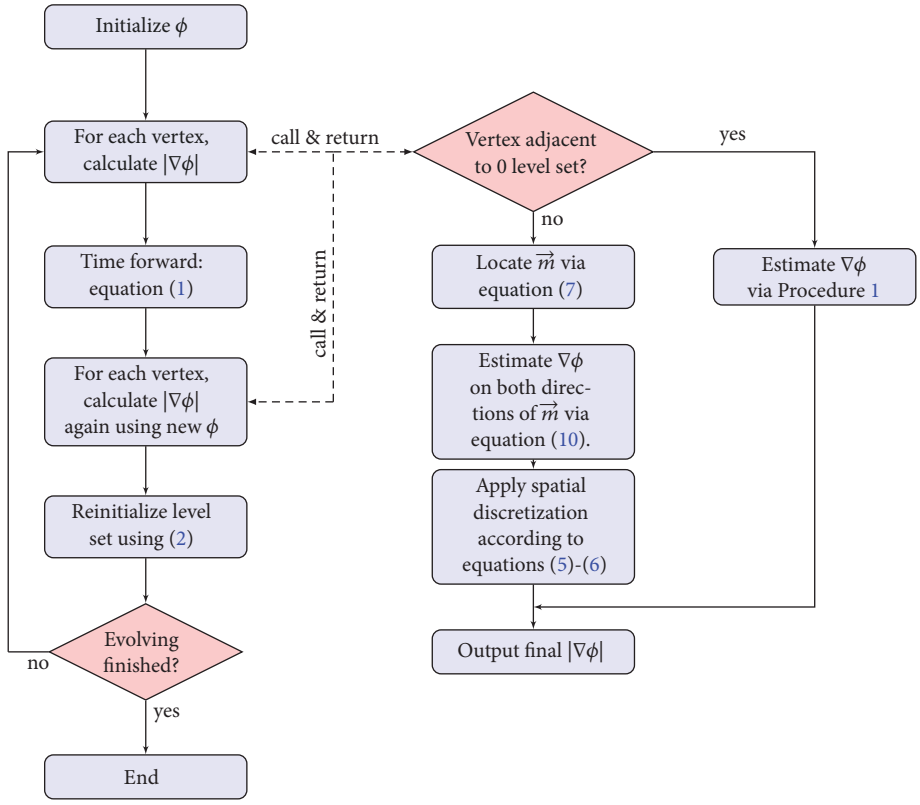


Figure 6: Overall flow.

Figure 7 shows the computational mesh of the notched square, where the red line is the initial location of the field being diffused. Evolving velocity is defined as

$$u_n\left(\vec{V}\right) = \begin{cases} 0.25 & 0 \le \vec{V_x} < 0.15, \\ 0.5 & 0.15 \le \vec{V_x} < 0.6, \\ 1 & 0.6 \le \vec{V_x} \le 1, \end{cases} \tag{19}$$

where $\vec{V_x}$ denotes the x coordinate of $\vec{V}$.

In order to demonstrate the improvement introduced by explicit correction, in Figure 8(a), we show two sets of transient results that are generated by the algorithm with and without explicit correction. The analytical solution is obtained via artificial geometry analysis. It can be seen that sharp geometric features are successfully preserved. The smearing-out effects near sharp features are not completely eliminated, but the explicit correction largely helps in relieving the effect. The error data, plotted in Figure 8(b), imply that the averaged error near the zero level set is kept below $0.7\Delta x$ at the end of the simulation. The peaks in the two E_m curves at $100 \sim 115$ steps are the product of the merging procedure of two sharp features near $[0.15, 0.8]$.

5.2. Three Merging Circles. This example includes three circles, the centers of which are located at $[0.3, 0.3]$, $[0.7, 0.3]$, and $[0.5, 0.7]$, respectively. The radiuses of the three circles

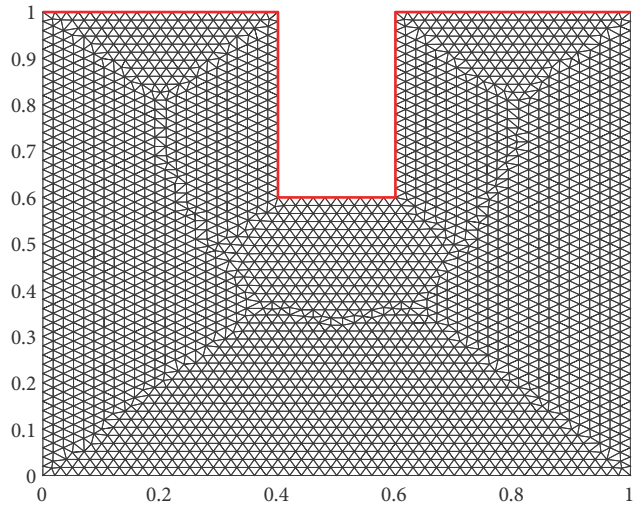
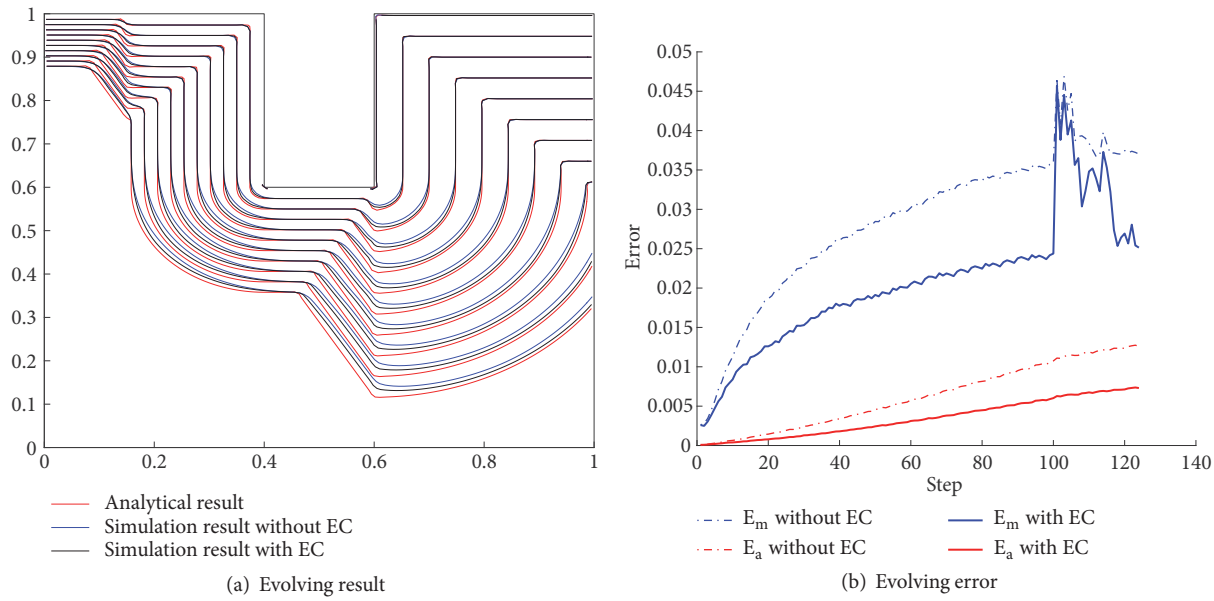


Figure 7: Computational mesh of the notched square.

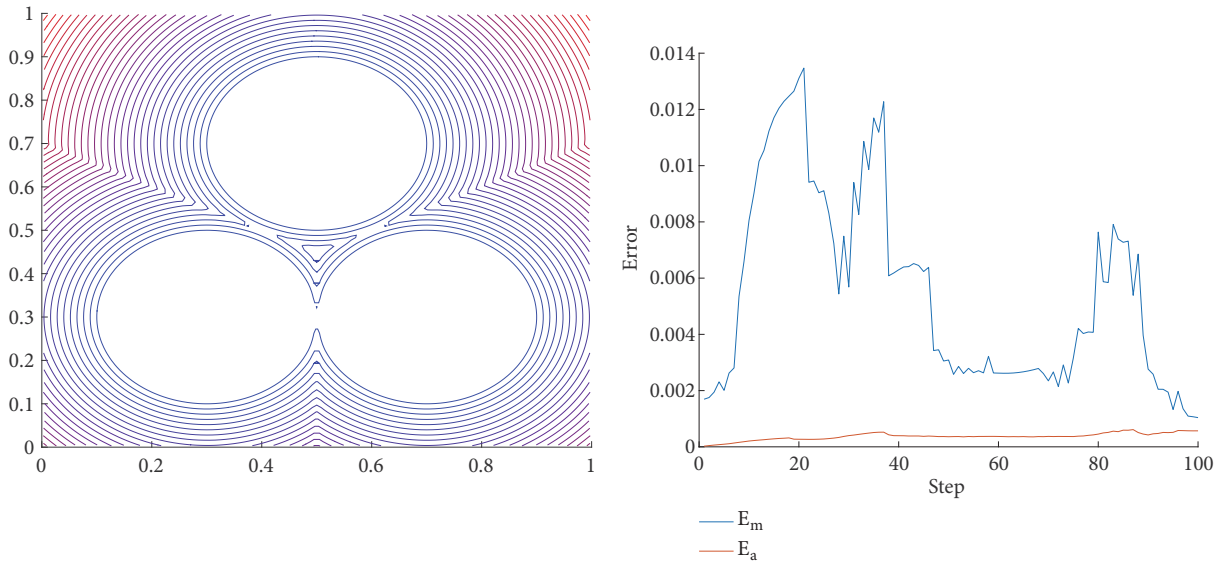
are initially 0.2 and grow uniformly at speed 1. We introduce this example to verify the ability of the proposed method to smoothly handle topological changes, especially the peak of ϕ located in the middle of the three circle centers.

The result of this example is plotted in Figure 9. It can be seen that the proposed method is able to solve the merging-circles problem with high accuracy. The averaged error is kept below $0.06\Delta x$ at the end of the simulation.



(a) Evolving result (b) Evolving error

Figure 8: Result of notched square example.



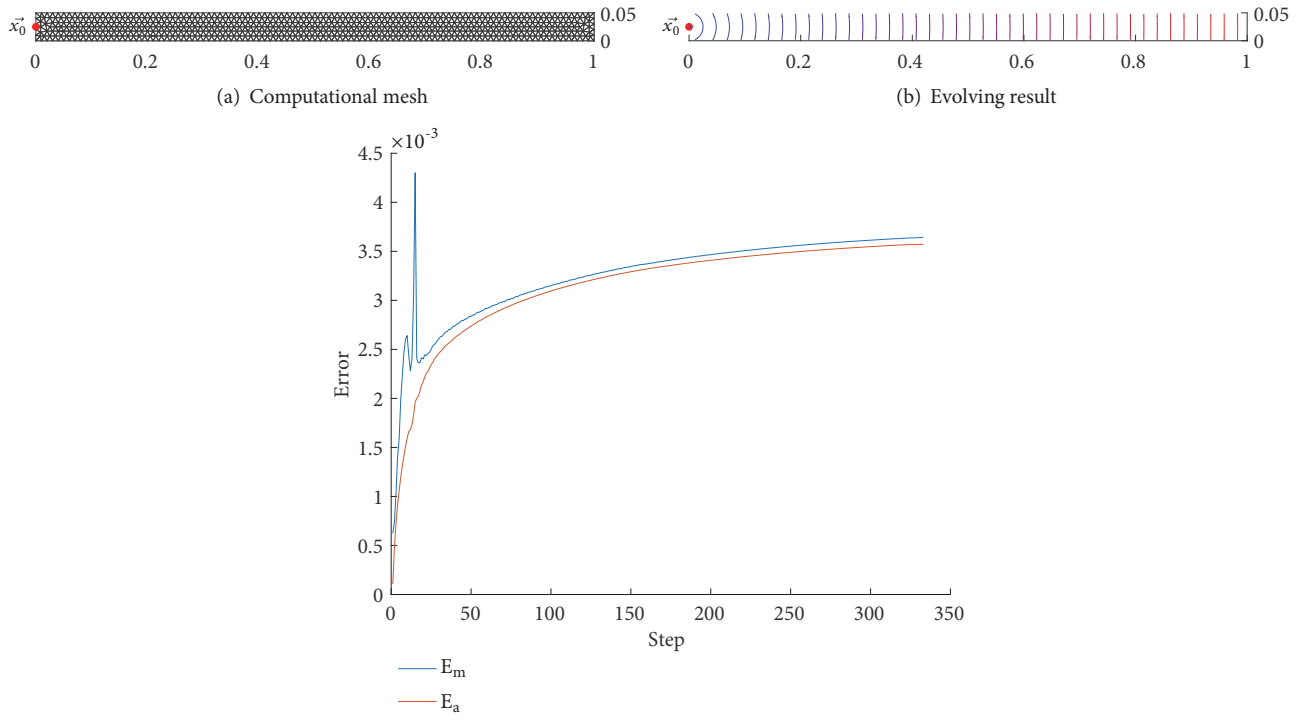
(a) Evolving result (b) Evolving error

Figure 9: Result of merging-circles example.

5.3. Rate Stick. In the rate-stick problem, a front that starts from a single point and propagates through a slender and long tube is the object of study. The case is similar to the high-explosives (HE) rate-stick experiment described in [5]. Figure 10(a) demonstrates the mesh and initial status. The red point $\overrightarrow{x_0}$, which denotes a zero-radius circle such that $\phi(\overrightarrow{x_0}) = 0$, is the starting point of the front. $u_n \equiv 1$ across the entire region. The mesh scale used in the example is $\Delta x = 0.005$.

The slender computational region means that the boundaries must be handled smoothly to evade distortion. The zero-radius initial region requires the proposed method to handle the case where the scale of the initial feature is smaller than Δx. Figures 10(b) and 10(c) show the result and error, respectively. The result shows that the proposed method works well on the rate-stick problem. The peak in the E_m curve is produced when the evolving circle is tangential to the mesh boundary.

5.4. Burning and Erosion in a Solid Rocket Motor. This problem contains real engineering components: the solid-rocket-motor grain and the corresponding thermal insulation layer. During the working procedure of the motor, the grain, which is made of solid propellant, burns and transforms into high-temperature gas. As the grain retreats, the thermal



(a) Computational mesh

(b) Evolving result

(c) Evolving error

FIGURE 10: Rate-stick problem.

insulation layer is exposed to the gas and is slowly eroded. To summarize, the gas propagates into the grain and the insulation at different velocities. Figure 11 demonstrates the working procedure on one section view of the motor, where the red lines outline the propagation of the front of the gas, and the gray and blue hatching mark the sections of the thermal insulation layer and the grain, respectively.

The computational mesh for this problem is shown in Figure 12. In actual solid rocket motors the propagation speed is determined by local environmental parameters such as temperature and pressure. In this article, however, we set the speed as shown by Eq. (20) to ease the verification:

$$u_n\left(\vec{x}\right) = \begin{cases} 1 & \vec{x} \in grain, \\ 0.1 & \vec{x} \in insulation. \end{cases} \tag{20}$$

Accordingly, the mesh for the insulation is finer than that for the grain to better capture the relatively slower front movement inside the insulation. Letting $d_t(\vec{x})$ be the minimum distance from the insulation to $\vec{x}$, the mesh scale Δx is defined by Eq. (21):

$$\Delta x\left(\vec{x}\right) = \begin{cases} 2.4 & d_t\left(\vec{x}\right) < 2, \\ 1.8 d_t\left(\vec{x}\right) - 1.2 & 2 \le d_t\left(\vec{x}\right) \le 6, \\ 6 & 6 < d_t\left(\vec{x}\right). \end{cases} \tag{21}$$

Since the structure of the motor is so complex that an analytical solution is unavailable, we used the level-set method implementation described in [28] to produce a reference result and further compute the error via tri-linear interpolation across meshes. The mesh scale used to generate the reference result is $\Delta x = \Delta y = \Delta z = 0.6$. Figure 13(a) shows two patches of the resulting isosurfaces (the coordinate systems to plot the two patches are aligned): the green piece is the right half of the zero level set at the 25th step and the red piece is the left half of the zero level set at the 80th step. The demonstrated patches are in line with the predictions made in Figure 11. The error data shown in Figure 13(b) reveal that the proposed method is consistent with the level-set method on Cartesian grids.

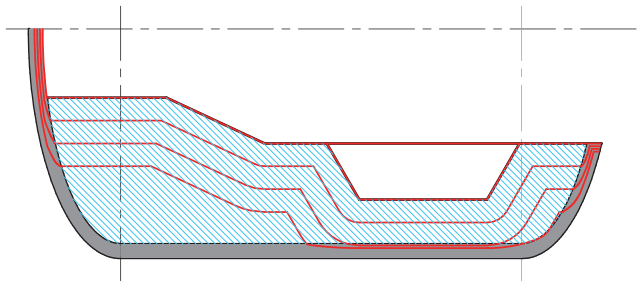


FIGURE 11: Working procedure of a solid rocket motor.

5.5. Reconstructing a Signed Distance Function. This test is aimed at constructing an exact signed distance field from a distorted one. In this article, the source geometry of the sign

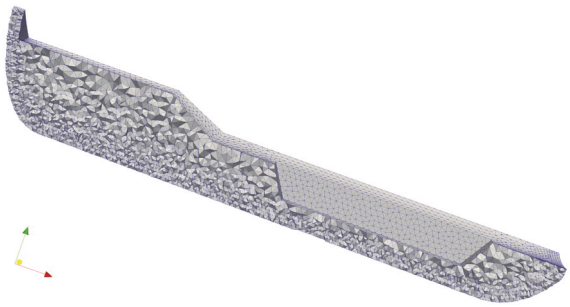
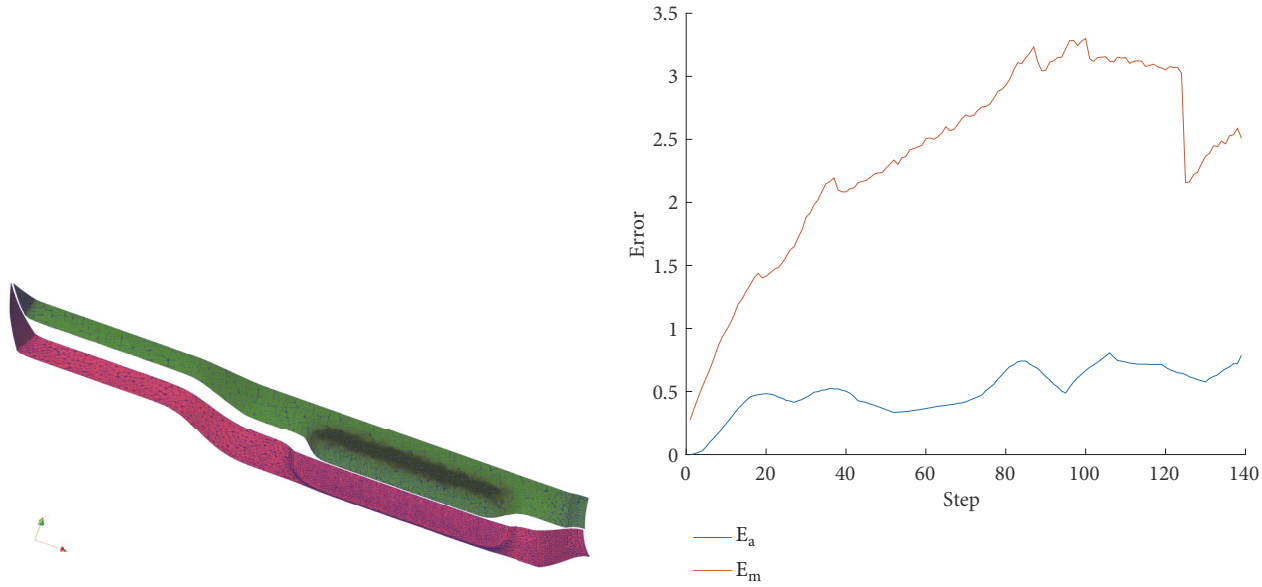


Figure 12: Computational mesh of a solid rocket motor.



(a) Evolving result

(b) Evolving error

Figure 13: Result of solid-rocket-motor example.

distance fields is a cube. The initial distorted field is generated via Eq. (22a),(22b), and (22c),

$$h = 0.5, \tag{22a}$$

$$\phi'\left(\vec{V}\right) = -\min\left\{h + V_x, h - V_x, h + V_y, h - V_y, h + V_z, h - V_z\right\}, \tag{22b}$$

$$\phi\left(\vec{V}\right) = \phi'\left(\vec{V}\right) \cdot \left[\left(V_x - 0.3\right)^2 + \left(V_y - 0.2\right)^2 + \left(V_z - 0.1\right)^2 + 0.05\right], \tag{22c}$$

where h is the half-side-length of the cube. Equation (22b) generates a signed field that contains discontinuity and its zero level set denotes the cube. Equation (22c) is the equation used by [24] to further distort the field. One section of the finally generated field is shown in Figure 14, where the thick red line marks its $\phi = 0$ isosurface. It can be seen that the output field contains two desired properties, namely the sharp features and the non-uniform distortion, which are generated on purpose to verify the ability of the proposed approach to reconstruct an exact signed distance function. The mesh scale used to generate the 3D mesh is $\Delta x = 0.05$.

The result after 80 reinitializing passes is shown in Figure 15. It can be seen that the signed distance field is correctly reconstructed and that the initial $\phi = 0$ isosurface is left unmoved. The error computed from the resulting ϕ field is $E_a = 0.082\Delta x$, $E_m = 0.54\Delta x$. Such a result implies that the proposed approach can be used as a robust and accurate method to construct signed distance fields on unstructured meshes.

5.6. Error Convergence. Here, we demonstrate the convergence of error as the mesh refines. The overall error across the evolving procedure is defined as Eq. (23),

$$E_o = \frac{1}{n(T)} \sum_{i=1}^{n(T)} E_a(i), \tag{23}$$

where $n(T)$ is the number of evolving steps and $E_a(i)$ denotes the E_a computed at step i. For the example that reconstructs

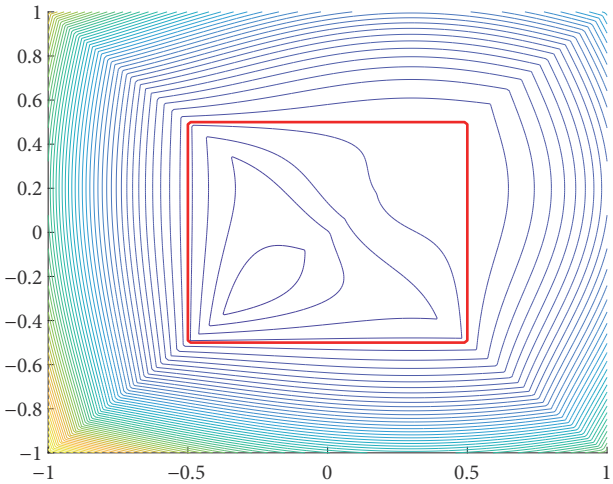


Figure 14: Contours of initial ϕ at section $z = 0$.

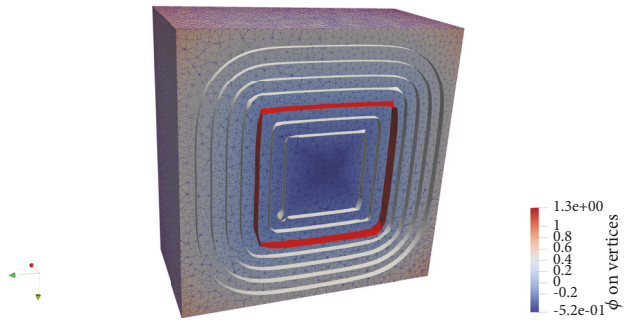


Figure 15: Result of reconstructing the signed distance field.

a signed distance function, since the only error of concern is that of the last step, we use the corresponding $E_a(n(T))$ to measure its error.

The error-convergence data is shown in Figure 16. Since the above examples have different mesh scales, we have unified the mesh-scale axis in Figure 16 using the above-used Δx in corresponding examples. For the solid-rocket-motor example, the Δx used for unifying is 6. The error data of the solid rocket motor example corresponding to 4 are missing because $\Delta x = 4 \times 6 = 24$ is so large a scale that the unstructured mesh cannot be properly generated. It can be seen from Figure 16 that, as the mesh refines, the absolute E_o decreases. However, $E_o/\Delta x$ has slightly increased at the same time. This implies that the error decreases more slowly than the mesh scale. Since the edge-based gradient-estimating approach is proved to have first-order accuracy, there must be other error sources that produce lower-order error and consequently cause $E_o/\Delta x$ to increase as the mesh refines. Examples of such error sources include floating-point error, the remaining smearing-out effect near sharp features that cannot be completely eliminated by the explicit correction, or the error produced when converging explicit analytical results to implicit ones.

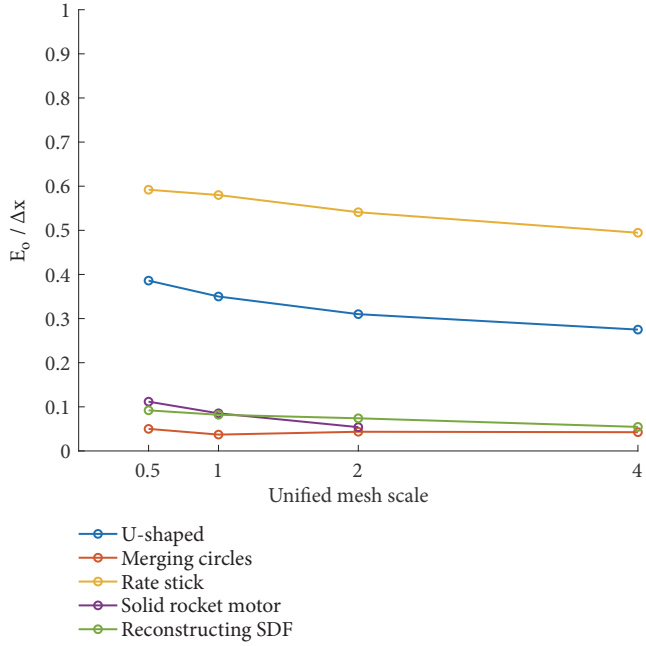
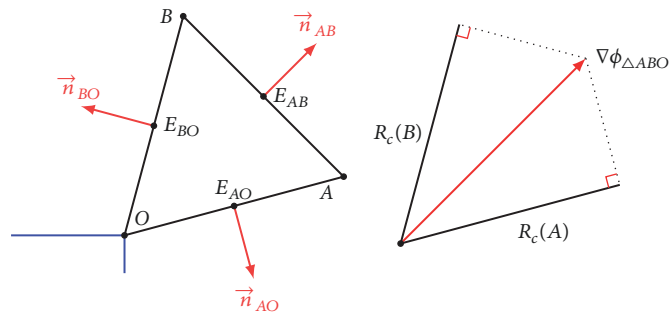


Figure 16: Error convergence data.



(a) (b)

Figure 17: Cell-based $\nabla\phi$ estimation.

Table 1: Error-converging orders.

Test example	Converging order
Notched square	0.8357
Three merging circles	0.9528
Rate stick	0.9121
Reconstructing signed distance fields	0.7586

In order to quantify the convergence, we have performed Richardson's analysis [29] modeled by Eq. (24), where C is a constant and p is the unknown order of convergence. By substituting the error data into (24), the order of convergence p of each example problem can be solved via the least-squares method. The results are collected in Table 1. The solid-rocket-motor example is excluded from the table since computing the order of convergence form one numerical method to another makes no sense. It can be seen from Table 1 that the implementation's order of convergence is near 1. Theoretically, arbitrarily small error can be achieved by continually refining the mesh:

$$E_o = C\,\Delta x^p + o(\Delta x). \tag{24}$$

6. Conclusions

In this article, we developed a robust approach to evolve fronts in their normal direction on unstructured meshes. The approach is built on an innovative spatial discretization scheme and a gradient-estimating technology matching the scheme. An explicit correction approach is introduced to improve the accuracy near the zero level set. Results of the validating examples show that the proposed method handles sharp geometric features, discontinuous velocity field, and topological changes smoothly and accurately, on both 2D and 3D unstructured meshes. The method theoretically also applies to structured or hybrid meshes, not only due to that hexahedral cells can be decomposed into tetrahedrons, but also because the proposed gradient-estimating technology only relies on the outgoing vectors, which are available in all kinds of meshes.

In practice, there are also cases where advecting fronts using externally generated velocity fields is required. We have also studied the advecting problem during this research. Unfortunately, it turned out that solving some of the advecting problems with acceptable accuracy requires higher-order spatial discretization schemes, which is beyond the scope of this article. In future research, we would study possible routes to integrate the proposed method with higher-order schemes and further expand the range of application of the method.

Appendix

Error Analysis of Cell-Based Gradient Estimation

Taking the 2D cell $\triangle OAB$ in Figure 17(a) for example, the target vertex O lies on a rectangular corner of the level set (denoted by blue). All edges of $\triangle OAB$ are within the outer sector of the corner, so the point on the front that is closest to A, B is O, $\phi_A = |AO|$, $\phi_B = |BO|$, $\phi_O = 0$. Vectors $\vec{n}_{IJ} (I, J \in \{A, B, O\})$ are outer normals of $\triangle OAB$, where $|\vec{n}_{IJ}| = |IJ|$ and $\vec{n}_{IJ} \cdot \vec{IJ} = 0$. According to (8), we have

$$\begin{aligned}\nabla\phi_{\triangle ABO} &= \frac{1}{S_{\triangle ABO}}\left(\phi_O\frac{\vec{n}_{AO}}{2} + \phi_O\frac{\vec{n}_{BO}}{2} + \phi_A\frac{\vec{n}_{AO}}{2}\right.\\ &\quad\left. + \phi_A\frac{\vec{n}_{AB}}{2} + \phi_B\frac{\vec{n}_{BO}}{2} + \phi_B\frac{\vec{n}_{AB}}{2}\right)\\ &= \frac{1}{2S_{\triangle ABO}}\left[\vec{n}_{AO}(\phi_A + \phi_O) + \vec{n}_{BO}(\phi_B + \phi_O)\right.\\ &\quad\left. + \vec{n}_{AB}(\phi_A + \phi_B)\right].\end{aligned} \tag{A.1}$$

Letting $\vec{OA} = (x_A, y_A)$, $\vec{OB} = (x_B, y_B)$, O is to the left-hand side of $\vec{AB}$; therefore, we have $x_A y_B > x_B y_A$, and that the outer normals $\vec{n}_{AO} = (y_A, -x_A)$, $\vec{n}_{BO} = (-y_B, x_B)$, and $\vec{n}_{AB} = (y_B - y_A, x_A - x_B)$. We also know that $\phi(A) = \sqrt{{x_A}^2 + {y_A}^2}$, $\phi(B) = \sqrt{{x_B}^2 + {y_B}^2}$. The triangle area $S_{\triangle ABO} =$

$(1/2)|\overrightarrow{OA} \times \overrightarrow{OB}| = (1/2)\sqrt{(x_A y_B - x_B y_A)^2}$. Substituting the above formulas into Eq. (A.1), we have

$$\begin{aligned}\nabla\phi_{\triangle ABO} &= \frac{1}{S_{\triangle ABO}}\left(\phi_O\frac{\overrightarrow{n}_{AO}}{2} + \phi_O\frac{\overrightarrow{n}_{BO}}{2} + \phi_A\frac{\overrightarrow{n}_{AO}}{2}\right.\\ &\left. + \phi_A\frac{\overrightarrow{n}_{AB}}{2} + \phi_B\frac{\overrightarrow{n}_{BO}}{2} + \phi_B\frac{\overrightarrow{n}_{AB}}{2}\right)\\ &= \frac{1}{\sqrt{(x_A y_B - x_B y_A)^2}}\left(-y_A\sqrt{x_B^2 + y_B^2}\right.\\ &\left. + y_B\sqrt{x_A^2 + y_A^2}, x_A\sqrt{x_B^2 + y_B^2} - x_B\sqrt{x_A^2 + y_A^2}\right).\end{aligned} \tag{A.2}$$

The projection lengths of $\nabla\phi_{\triangle ABO}$ to $\overrightarrow{OA}$ and $\overrightarrow{OA}$ (as shown in Figure 17(b)) are

$$\begin{aligned}\nabla\phi_{\triangle ABO}\cdot\frac{\overrightarrow{OA}}{|OA|} &= \frac{x_A y_B - x_B y_A}{\sqrt{x_A^2 y_B^2 - 2x_A x_B y_A y_B + x_B^2 y_A^2}}\\ &= 1 = R_c(A),\\ \nabla\phi_{\triangle ABO}\cdot\frac{\overrightarrow{OB}}{|OB|} &= \frac{x_A y_B - x_B y_A}{\sqrt{x_A^2 y_B^2 - 2x_A x_B y_A y_B + x_B^2 y_A^2}}\\ &= 1 = R_c(B).\end{aligned} \tag{A.3}$$

Taking equilateral triangles as an example, the magnitude of the resulting $\overline{\nabla}\phi$ is $2\sqrt{3}/3 \approx 1.1547$, which deviates from the true value (i.e., 1) by 15.47% no matter how small the scale of $\triangle ABO$ is. Although the above discussion is based on the 2D case, the 3D cell-based estimation method shows a similar problem near sharp features.

Data Availability

The source code and validation data used to support the findings of this study have been deposited in the GitHub repository https://github.com/metorm/Unstructured-Level-Set-Method.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, no. 1, pp. 12–49, 1988.

[2] L. A. Vese and T. F. Chan, "A multiphase level set framework for image segmentation using the Mumford and Shah model," *International Journal of Computer Vision*, vol. 50, no. 3, pp. 271–293, 2002.

[3] D.-H. Wang, Y. Fei, F. Hu, and W.-H. Zhang, "An integrated framework for solid rocket motor grain design optimization," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 228, no. 7, pp. 1156–1170, 2014.

[4] K. Albarado, A. Shelton, and R. J. Hartfield, "SRM simulation using the level set method and higher order integration schemes," in *Proceedings of the 48th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit 2012*, USA, August 2012.

[5] J. A. Zukas and W. P. Walters, *Explosive Effects and Applications. Shock Wave and High Pressure Phenomena*, Springer New York, New York, NY, 1998.

[6] T. J. Barth and J. A. Sethian, "Numerical schemes for the Hamilton-Jacobi and level set equations on triangulated domains," *Journal of Computational Physics*, vol. 145, no. 1, pp. 1–40, 1998.

[7] T. D. Aslam, J. B. Bdzil, and D. S. Stewart, "Level set methods applied to modeling detonation shock dynamics," *Journal of Computational Physics*, vol. 126, no. 2, pp. 390–409, 1996.

[8] R. Abgrall, "Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes," *Communications on Pure and Applied Mathematics*, vol. 49, no. 12, pp. 1339–1373, 1996.

[9] X. Li, W. Yan, and C. K. Chan, "Numerical schemes for Hamilton-Jacobi equations on unstructured meshes," *Numerische Mathematik*, vol. 94, no. 2, pp. 315–331, 2003.

[10] R. Abgrall, "Numerical discretization of boundary conditions for first order Hamilton-Jacobi equations," *SIAM Journal on Numerical Analysis*, vol. 41, no. 6, pp. 2233–2261, 2003.

[11] Y.-T. Zhang and C.-W. Shu, "High-order WENO schemes for Hamilton-Jacobi equations on triangular meshes," *SIAM Journal on Scientific Computing*, vol. 24, no. 3, pp. 1005–1030, 2002.

[12] C. Hu and C.-W. Shu, "Weighted essentially non-oscillatory schemes on triangular meshes," *Journal of Computational Physics*, vol. 150, no. 1, pp. 97–127, 1999.

[13] J. Zhu and J. Qiu, "Hermite WENO schemes for Hamilton-Jacobi equations on unstructured meshes," *Journal of Computational Physics*, vol. 254, pp. 76–92, 2013.

[14] S. Augoula and R. Abgrall, "High order numerical discretization for Hamilton-Jacobi equations on triangular meshes," *Journal of Scientific Computing*, vol. 15, no. 2, pp. 197–229, 2000.

[15] Y.-T. Zhang and C.-W. Shu, "Third order WENO scheme on three dimensional tetrahedral meshes," *Communications in Computational Physics*, vol. 5, no. 2-4, pp. 836–848, 2009.

[16] J. Rokicki and R. Wieteska, "High Order Weno Schemes on Unstructured Tetrahedral Meshes," in *Proceedings of the In ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics*, p. 12, 2006.

[17] N. R. Morgan and J. I. Waltz, "3D level set methods for evolving fronts on tetrahedral meshes with adaptive mesh refinement," *Journal of Computational Physics*, vol. 336, pp. 492–512, 2017.

[18] Z. Fu, S. Yakovlev, R. M. Kirby, and R. T. Whitaker, "Fast parallel solver for the levelset equations on unstructured meshes," *Concurrency Computation*, vol. 27, no. 7, pp. 1639–1657, 2015.

[19] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, vol. 153, Springer, New York, NY, USA, 2003.

[20] D. Hacon and C. Tomei, "Tetrahedral decompositions of hexahedral meshes," *European Journal of Combinatorics*, vol. 10, no. 5, pp. 435–443, 1989.

[21] S. K. Godunov, "A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics," *Matematicheskii Sbornik*, vol. 47 (89), pp. 271–306, 1959.

[22] M. Sussman, P. Smereka, and S. Osher, "A level set approach for computing solutions to incompressible two-phase flow," *Journal of Computational Physics*, vol. 114, no. 1, pp. 146–159, 1994.

[23] O. Fortmeier and H. M. BÜcker, "Parallel re-initialization of level set functions on distributed unstructured tetrahedral grids," *Journal of Computational Physics*, vol. 230, no. 12, pp. 4437–4453, 2011.

[24] C. Min, "On reinitializing level set functions," *Journal of Computational Physics*, vol. 229, no. 8, pp. 2764–2772, 2010.

[25] R. Wei, *Unstructured Level Set Method*, May 2018, https://github.com/metorm/Unstructured-Level-Set-Method.

[26] D. Engwirda, *Locally Optimal Delaunay-Refinement and Optimisation-Based Mesh Generation [Ph.D. thesis]*, University of Sydney, November 2014.

[27] C. Geuzaine and J. F. Remacle, "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[28] R. Wei, F. Bao, Y. Liu, and W. Hui, "Combined Acceleration Methods for Solid Rocket Motor Grain Burnback Simulation Based on the Level Set Method," *International Journal of Aerospace Engineering*, vol. 2018, Article ID 4827810, 12 pages, 2018.

[29] E. Christiansen and H. G. Petersen, "Estimation of convergence orders in repeated Richardson extrapolation," *BIT Numerical Mathematics*, vol. 29, no. 1, pp. 48–59, 1989.