J Sci Comput
DOI 10.1007/s10915-017-0364-4

Inflow-Based Gradient Finite Volume Method for a Propagation in a Normal Direction in a Polyhedron Mesh

Jooyoung Hahn[1] · **Karol Mikula**[2] · **Peter Frolkovič**[2] · **Branislav Basara**[1]

Received: 21 January 2016 / Revised: 10 January 2017 / Accepted: 13 January 2017

Abstract An inflow-based gradient is proposed to solve a propagation in a normal direction with a cell-centered finite volume method. The proposed discretization of the magnitude of gradient is an extension of Rouy–Tourin scheme (SIAM J Numer Anal 29:867–884, 1992) and Osher–Sethian scheme (J Comput Phys 79:12–49, 1988) in two cases; the first is that the proposed scheme can be applied in a polyhedron mesh in three dimensions and the second is that its corresponding form on a regular structured cube mesh uses the second order upwind difference. Considering a practical application in three dimensional mesh, we use the simplest decomposed domains for a parallel computation. Moreover, the implementation is straightforwardly and easily combined with a conventional finite volume code. A higher order of convergence and a recovery of signed distance function from a sparse data are illustrated in numerical examples on hexahedron or polyhedron meshes.

Keywords Inflow-based gradient · Cell-centered finite volume method · Level set method · Polyhedron mesh · Rouy–Tourin scheme · Propagation in normal direction

Karol Mikula was supported by VEGA 1/0808/15 and APVV-15-0522. Peter Frolkovič was supported by VEGA 1/0728/15 and APVV-15-0522.

✉ Jooyoung Hahn
jooyoung.hahn@avl.com

Karol Mikula
karol.mikula@stuba.sk

Peter Frolkovič
peter.frolkovic@stuba.sk

Branislav Basara
branislav.basara@avl.com

[1] AVL LIST GmbH, Graz, Austria

[2] Department of Mathematics and Descriptive Geometry, Slovak University of Technology, Bratislava, Slovakia

Published online: 24 January 2017

1 Introduction

In this paper, we numerically solve a propagation in a normal direction:

$$\frac{\partial}{\partial t}\phi(\mathbf{x}, t) + F(\mathbf{x})|\nabla\phi(\mathbf{x}, t)| = G(\mathbf{x}), \quad (\mathbf{x}, t) \in \Omega \times [0, T], \tag{1.1}$$

where $\Omega \subset \mathbb{R}^3$ is a computational domain, T is the final time, the speed function F and the force term G are fixed, and the initial condition $\phi(\mathbf{x}, 0) = \phi_0(\mathbf{x})$ is given on Ω. The viscosity solution has been intensively studied and used in many fields such as geometrical optics, control theory, computer vision, and etc. For more details of applications and numerical schemes in steady and non-steady cases, we refer the readers to the literature [3,4] and the references therein. The proposed discretization of (1.1) is based on a cell-centered finite volume method (FVM) and it can be considered as an extension of standard schemes [1,2] in two cases; the first is that the proposed scheme can be applied in a polyhedron mesh in three dimensions (3D) and the second is that its corresponding form on a regular structured cube mesh uses the second order upwind difference.

In contrast to a standard regular structured cube mesh mainly used in level set literature [3, 4], we focus on an extension of numerical scheme into a polyhedron mesh in 3D. A polyhedron mesh in industrial applications has been used because of its shape flexibility and automatic discretization of highly complicated computational domains; see more details in [5] and the references therein. Moreover, in computational fluid dynamics (CFD), the author in [5] shows the advantages of polyhedral meshes compared to tetrahedral meshes. In order to use topological advantages of level set method in industrial and practical CFD problems, it is necessary to investigate a possible extension of classical numerical methods [1,2] into polyhedron meshes. In a steady case of (1.1), the fast marching method (FMM) [4] or the fast sweeping method (FSM) [6] have been used and improved in literature. The extension of FSM into tetrahedron meshes [7] and a parallel computation [8] are studied. However, a correct enforcement of natural causality in a polyhedron mesh with a decomposed domain structure for parallel computation still remains a challenging task. In [9], computing signed distance functions on a polyhedral domain is achieved with the use of mathematical theories in the unsteady Eikonal equation.

The standard numerical schemes in [1,2] capture a nature of propagation by using an upwind discretization. They can be easily extended to multidimensional domains in a dimension-by-dimension manner. A higher order discretization such as well-known weighted essentially non-oscillatory (WENO) schemes [10,11] can be used. A higher order WENO scheme in an unstructured mesh has been also developed in [12,13] and the references therein. However, it is still complicated to adequately collect a stencil structure in decomposed domains of polyhedron mesh for parallel computing and to efficiently reconstruct a function of a required order. An advantage of the proposed scheme is that a complicated reconstruction in a polyhedron mesh is not used and we only need the simplest structure of decomposed domains, that is, the 1-ring face neighborhood structure. The implementation of proposed scheme with a parallel computation is straightforwardly combined with a conventional finite volume code.

A method in [14] to numerically solve (1.1) is used in an unstructured mesh with a vertex-centered FVM. In [15–17], semi-implicit methods for solving advection equations and motions in a normal direction are introduced to remove a time step restriction caused by CFL condition. The main differences between the mentioned methods and the proposed method are a boundary condition and an approximation of gradient. Concerning the first difference, since a Dirichlet boundary condition may not be specified in many real applications, we impose a

linearly extended boundary condition to use the inflow information on the boundary. A zero Neumann boundary condition is not an option to be applied because it forces a distortion of zero level set on the boundary. Concerning the second difference, in contrast to cell-centered gradient in [14–17], a gradient is approximated by an average of constraint face gradients only from inflow sides. A different computation of gradient successfully provides correct upwind discretization of the magnitude of gradient on polyhedron meshes.

The outline of paper is organized as follows. In the rest of introduction, we review the standard numerical schemes [1,2] in one dimensional domain. In Sect. 2, a proposed scheme is introduced in 3D and compared with the standard schemes in one dimension (1D). Numerical properties of proposed scheme are presented in Sect. 3.

1.1 Observation in One Dimension

First of all, we review the standard numerical schemes of discretizing the magnitude of gradient in [1] and [2] to numerically solve a governing Eq. (1.1). Standard schemes are presented with respect to specific cases in 1D because it is easy to understand numerical properties of discretization. Secondly, we address some issues to improve the standard schemes and discuss about how to extend them into 3D polyhedron mesh.

Let us denote a computational domain:

$$\bar{\Omega} = [0,1] \subset \bigcup_{i\in\mathcal{I}} I_i, \quad I_i = \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right], \quad x_{i-\frac{1}{2}} = \left(i - \frac{1}{2}\right)h,$$

where $\bar{\Omega}$ is evenly divided by a small length $h \ll 1$. Using simple coefficients $F = 1$ and $G = 0$ in (1.1), we have:

$$\frac{\partial \phi}{\partial t} + \left|\frac{\partial \phi}{\partial x}\right| = 0. \tag{1.2}$$

Introducing the first order forward and backward differences at an interval I_i

$$\partial_i^{+1}\phi = \frac{\phi_{i+1} - \phi_i}{h}, \quad \partial_i^{-1}\phi = \frac{\phi_i - \phi_{i-1}}{h},$$

where $\phi_i = \phi(x_i)$ and $x_i = ih$, Rouy–Tourin [1] and Osher–Sethian schemes [2] are presented to discretize the magnitude of gradient $\left|\frac{\partial \phi}{\partial x}\right|$ at x_i in (1.2):

$$\text{Rouy–Tourin}: \quad \left|\max(\partial_i^{-1}\phi, -\partial_i^{+1}\phi, 0)\right|$$

$$\text{Osher–Sethian}: \quad \left(\max(\partial_i^{-1}\phi, 0)^2 + \min(\partial_i^{+1}\phi, 0)^2\right)^{\frac{1}{2}}.$$

Now, we consider four representative cases illustrated by Fig. 1 and Rouy–Tourin and Osher–Sethian schemes on each case are written in Table 1. The schemes clearly show a correct upwind directional discretization in Cases 1 and 2. An upwind discretization captures a propagation nature in (1.1) from the correct direction. The schemes can be extended into multidimensional domains in a dimension-by-dimension manner. It means that a mesh type should be restricted to structured cubes which can not be practically used in many real applications. A higher order differences can be used in a structured cube mesh. However, they are not straightforwardly extended into unstructured meshes. In Case 3, the schemes are identically same. In Case 4, all methods show different approximation of possibly nonsmooth points. Rouy–Tourin scheme tries to keep an upwind nature but Osher–Sethian scheme uses derivatives from all directions.

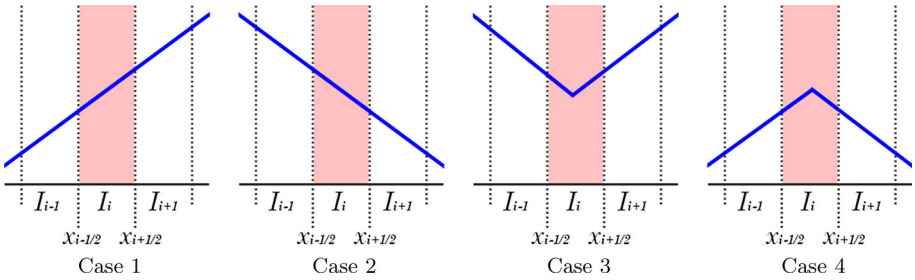


Fig. 1 Four cases of understanding some properties of numerical methods from [1] and [2]

Table 1 A comparison of Rouy–Tourin [1] and Osher–Sethian schemes [2] with respect to special cases in Fig. 1

Cases	Rouy–Tourin	Osher–Sethian
1	$\lvert\partial_i^{-1}\phi\rvert$	$\lvert\partial_i^{-1}\phi\rvert$
2	$\lvert\partial_i^{+1}\phi\rvert$	$\lvert\partial_i^{+1}\phi\rvert$
3	0	0
4	$\max\left\{\lvert\partial_i^{-1}\phi\rvert, \lvert\partial_i^{+1}\phi\rvert\right\}$	$\left((\partial_i^{-1}\phi)^2 + (\partial_i^{+1}\phi)^2\right)^{\frac{1}{2}}$

In this paper, we would like to design a numerical scheme to solve a governing Eq. (1.1) in a 3D polyhedron mesh and improve the previous Rouy–Tourin and Osher–Sethian schemes as follows.

- Second order upwind discretization in a regular structured cube mesh.
- A parallel computation in 1-ring face neighborhood structure.
- A straightforward implementation in a 3D polyhedron mesh.
- Imposing the linear extended boundary condition.

In the rest of section, each point is briefly discussed. In 1D or a structured cube mesh in 3D, the second order forward and backward upwind differences

$$\partial_i^{+2}\phi = \frac{-\phi_{i+2} + 4\phi_{i+1} - 3\phi_i}{2h}, \quad \partial_i^{-2}\phi = \frac{3\phi_i - 4\phi_{i-1} + \phi_{i-2}}{2h}, \tag{1.3}$$

can be easily used by a straightforward dimension-by-dimension extension of Rouy–Tourin and Osher–Sethian schemes. However, in case of unstructured meshes, computing the backward or forward upwind difference is nontrivial because faces of meshes are not simply aligned along the coordinates. The main challenge is to design a scheme of discretizing the magnitude of gradient in 3D polyhedron mesh whose 1D correspondence can be represented by the second order upwind discretization. Another challenge is whether the scheme still provides a second order rate of convergence (EOC) in 3D polyhedron mesh for a smooth solution of (1.1). We solve such a difficulty by proposing an inflow-based gradient defined by constraint face gradients in Sect. 2. The proposed discretization of the magnitude of gradient can be understood as an extension of Rouy–Tourin and Osher–Sethian schemes into a polyhedron mesh. The meaning of extension is that 1D correspondence of proposed scheme can replace the first order difference in Rouy–Tourin and Osher–Sethian schemes with the second order upwind difference of Cases 1 and 2 in Table 1.

A FVM is a reasonable choice to easily deal with a standard decomposed domain structure for a parallel computation and diverse mesh types. In decomposed computational domains

for parallel computation in 3D, we use the 1-ring face neighborhood structure which only allows the smallest number of neighbor cells of a cell, whose faces are shared. The 1-ring face neighborhood structure has been used in modern industrial CFD because of its simple structure. The physical limitation of accessing neighbor cells obviously reduce the memory and complexity of obtaining decomposed domains but it directly makes difficult to design a higher order scheme which usually needs a large neighborhood information. The difficulty can be resolved by the locality of computing the proposed inflow-based gradient.

A reason why we use cell-based FVM rather than a vertex-based FVM in [14] and [16] is because of a reasonable boundary condition in many practical applications. The Dirichlet boundary condition can be imposed on special examples where the influx data at the boundary is explicitly given. In general cases of normal flow evolution from arbitrary shapes, it is impossible to obtain unknown inflow data at the boundary. Zero Neumann boundary condition is also not a good choice on inflow regions of the boundary. It always forces a data gradient to be orthogonal to a boundary normal vector and then it is inevitable to have a distortion of zero level set close to the boundary. A reasonable choice to reduce such a distortion is a linearly extended boundary condition. Considering a simple implementation of imposing linearly extended boundary condition, a cell-centered FVM is more preferable choice than a vertex-based FVM because only boundary face values are necessary in the former method. The latter method usually has a difficulty to impose certain boundary conditions such as Neumann or linearly extended types because boundary normal vectors at corners are singular and boundary vertex values should be computed in an underdetermined system; see more details in [18]. It is not necessary to solve such a system in cell-centered FVM to impose linearly extended boundary condition. In Sect. 2, a detailed explanation of proposed scheme is presented. After that, we present 1D correspondence of the proposed method to compare with standard schemes in Table 1.

2 Inflow-Based Gradient Finite Volume Method

In this section, a detailed explanation of the proposed scheme is presented with linearly extended boundary condition. Some notations are defined in Sect. 2.1 for the clear explanation in 3D and then the detail steps of proposed scheme and inflow-based gradient are presented in Sect. 2.2. At the end of section, we rewrite the proposed scheme in 1D in order to show a similarity to previous methods in Sect. 1.1.

2.1 Notations

Let $\bar{\Omega} \subset \mathbb{R}^3$ be a whole computational domain:

$$\bar{\Omega} = \bigcup_{p \in \mathcal{I}} \bar{\Omega}_p,$$

where an open set $\Omega_p \subset \mathbb{R}^3$ with the volume $|\Omega_p| \neq 0$ represents a cell of discretized mesh which is one of hexagonal, tetrahedral, prism, or polyhedral shapes. We denote the set of indices to uniquely indicate cells, faces, and vertices as $\mathcal{I}$, $\mathcal{F}$, and $\mathcal{V}$, respectively.

Since a face of polyhedron mesh is not always a plane, a nonplanar face is tessellated into triangles to make its sub-face planar. From a given face center of a nonplanar face, triangles of tessellation are defined by two consecutive vertices of the face and the face center. The index set $\mathcal{F}$ now indicates all planar faces, respectively sub-faces, of cells.

In order to explain a finite volume method in 3D, a neighbor information of Ω_p, $p \in \mathcal{I}$, should be specified. We define two information; neighbor cells and attached faces to Ω_p. For the neighbor cells, we only indicate cells whose face is shared:

$$\mathcal{N}_p = \{q \in \mathcal{I} : \text{there exists a face } e_f \subset \partial\Omega_q \cap \partial\Omega_p,\ f \in \mathcal{F}\}.$$

In 2D structured rectangular mesh, $\mathcal{N}_p$ indicates five stencil points except p. The attached faces to Ω_p are indicated by two sets:

$$\mathcal{F}_p = \{f \in \mathcal{F} :\ e_f \in \partial\Omega_p\},$$
$$\mathcal{B}_p = \{f \in \mathcal{F}_p :\ e_f \in \partial\Omega_p \cap \partial\Omega\}.$$

Obviously, $\mathcal{B}_p \subset \mathcal{F}_p$, for all $p \in \mathcal{I}$. If Ω_p is a cell whose all faces are not overlapped to the boundary of computational domain, then $\mathcal{B}_p = \emptyset$. Note that the index sets $\mathcal{N}_p$ and $\mathcal{F}_p$ normally have the same cardinality. However, it is not always true because of nonconvex element.

In a cell-centered finite volume method, function values $\phi_p = \phi(\mathbf{x}_p)$ or $\phi_f = \phi(\mathbf{x}_f)$ are evaluated at a cell center $\mathbf{x}_p$ or a face center $\mathbf{x}_f$, $p \in \mathcal{I}$ or $f \in \mathcal{F}$, respectively. For a given vertex index $v \in \mathcal{V}$, we denote $\mathbf{x}_v$ as a position of the vertex of a cell. If a quantity is uniquely defined on a face e_f regardless to cells sharing the face, we use a unique face index $f \in \mathcal{F}$, for instance, a face center $\mathbf{x}_f$ of a face e_f. If a quantity is defined on a face e_f depending on cells sharing the face, we explicitly indicate a cell index at the first subscript and a unique face index at the second subscript. For instance, for a face index $f \in \mathcal{F}_p \subset \mathcal{F}$ attached to the cell Ω_p, $p \in \mathcal{I}$, the outward normal vector to the face is indicated by $\mathbf{n}_{pf}$. If the face e_f is an internal face, there is a cell Ω_q, $q \in \mathcal{I}$, such that $e_f \subset \partial\Omega_p \cap \partial\Omega_q$. Then, obviously,

$$\mathbf{n}_{pf} = -\mathbf{n}_{qf}.$$

We also use the length of the normal vector as the area of the face:

$$|\mathbf{n}_{pf}| = |\mathbf{n}_{qf}| = |e_f|.$$

2.2 Cell-Centered Finite Volume Method

We focus on a spatial discretization of the proposed method at a fixed time and later present a time discretization at the end of subsection. With simple coefficients $F = 1$ and $G = 0$ in the governing Eq. (1.1) and using Gauss's theorem, we have

$$\int_{\Omega_p} \frac{\partial\phi}{\partial t} + \int_{\Omega_p} \nabla \cdot \left(\phi \frac{\nabla\phi}{|\nabla\phi|}\right) - \int_{\Omega_p} \phi\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} = 0,$$

and its spatial discretization is written

$$\int_{\Omega_p} \frac{\partial\phi}{\partial t} + \sum_{f\in\mathcal{F}_p} \phi_{pf} \int_{e_f} \frac{\nabla\phi}{|\nabla\phi|} \cdot \mathbf{n}_{pf} - \sum_{f\in\mathcal{F}_p} \phi_p \int_{e_f} \frac{\nabla\phi}{|\nabla\phi|} \cdot \mathbf{n}_{pf} = 0,$$

where $\mathbf{n}_{pf}$ is an outward normal vector to Ω_p at a face e_f, $f \in \mathcal{F}_p$. Denoting the flux of velocity at a face e_f as

$$a_{pf} = \int_{e_f} \frac{\nabla\phi}{|\nabla\phi|} \cdot \mathbf{n}_{pf} \simeq \frac{\nabla\phi_f}{|\nabla\phi_f|} \cdot \mathbf{n}_{pf}, \tag{2.1}$$

where $\nabla\phi_f$ is a gradient evaluated at the face center, a cell-centered finite volume formulation is presented:

$$\int_{\Omega_p} \frac{\partial \phi}{\partial t} + \sum_{f \in \mathcal{F}_p} \left(\phi_{pf} - \phi_p\right) a_{pf} = 0, \quad p \in \mathcal{I}. \tag{2.2}$$

Similar to many other finite volume methods, the most crucial part is to evaluate a face value ϕ_{pf} depending on a sign of flux a_{pf}. For instance, an upwind scheme is obtained by:

$$\phi_{pf} = \begin{cases} \phi_p, & \text{if } a_{pf} \geq 0, \\ \phi_q, & \text{if } a_{pf} < 0, \end{cases}$$

where there is $q \in \mathcal{N}_p$ such that $e_f \in \partial\Omega_p \cap \Omega_q \neq \emptyset$, for $f \in \mathcal{F}$. In the rest of section, we explain the proposed method into two steps.

- `Step` 1. Define an inflow-based gradient.
- `Step` 2. Compute a face value ϕ_{pf} in (2.2).

Note that the inflow-based gradient in `Step` 1 evaluated at a cell center is used to define a face value ϕ_{pf} in `Step` 2.

`Step`. 1 Define an Inflow-Based Gradient

An inflow-based gradient is a cell-centered vector and defined by constraint face gradients and signs of flux in (2.1). A constraint face gradient is obtained by a constraint minimization from the values close to a face such as cell centers and vertices. A vertex value is interpolated from cell-centered gradients. Note that an inflow-based gradient calculated at cell center is different from a cell-centered gradient.

Now, in order to define an inflow-based gradient, we start to obtain a standard cell-centered gradient with linearly extended boundary condition. Let us fix a cell center $\mathbf{x}_p$ and find a gradient $\nabla\phi(\mathbf{x}_p)$. We define a set of points $\mathcal{S}_p$ at the cell $p \in \mathcal{I}$:

$$\mathcal{S}_p = \begin{cases} \{\mathbf{x}_q \mid q \in \mathcal{N}_p\} & \text{if } \mathcal{B}_p = \emptyset, \\ \{\mathbf{x}_q \mid q \in \mathcal{N}_p\} \cup \{\mathbf{x}_b \mid b \in \mathcal{B}_p\} & \text{if } \mathcal{B}_p \neq \emptyset. \end{cases}$$

We use a weighted minimization to find a cell-centered gradient of $\phi(\mathbf{x}_p)$, $p \in \mathcal{I}$:

$$\nabla\phi(\mathbf{x}_p) = \underset{\mathbf{y} \in \mathbb{R}^3}{\arg\min} \left\{ \mathbf{y} \in \mathbb{R}^3 \mid \sum_{\mathbf{x} \in \mathcal{S}_p} w_p(\mathbf{x}) \left| \mathbf{y} \cdot (\mathbf{x} - \mathbf{x}_p) - (\phi(\mathbf{x}) - \phi(\mathbf{x}_p)) \right|^2 \right\},$$

where a weight function is defined by

$$w_p(\mathbf{x}) = \frac{1}{|\mathbf{x} - \mathbf{x}_p|^2}.$$

Applying the Gâteaux derivative to the above energy functional, we have:

$$\left(\sum_{\mathbf{x} \in \mathcal{S}_p} \frac{\mathbf{d}_p(\mathbf{x}) \otimes \mathbf{d}_p(\mathbf{x})}{|\mathbf{d}_p(\mathbf{x})|^2} \right) \nabla\phi(\mathbf{x}_p) = \sum_{\mathbf{x} \in \mathcal{S}_p} \frac{\mathbf{d}_p(\mathbf{x})}{|\mathbf{d}_p(\mathbf{x})|^2} \big(\phi(\mathbf{x}) - \phi(\mathbf{x}_p)\big), \tag{2.3}$$

where $\mathbf{d}_p(\mathbf{x}) = \mathbf{x} - \mathbf{x}_p$. In case of $\mathcal{B}_p = \emptyset$, the matrix Eq. (2.3) is well-defined and it has a unique solution. However, in case of $\mathcal{B}_p \neq \emptyset$, we have to know the value $\phi(\mathbf{x}_b)$ at a boundary

face e_b, $b \in \mathcal{B}_p$. Since linearly extended boundary condition is used, we apply Newton's method to approximate a boundary face value:

$$\phi(\mathbf{x}_b) = \phi(\mathbf{x}_p) + \nabla\phi(\mathbf{x}_p) \cdot (\mathbf{x}_b - \mathbf{x}_p), \quad b \in \mathcal{B}_p.$$

Inserting the Newton's approximation into (2.3), the terms from $\mathbf{x}_b \in \mathcal{S}_p$ and $b \in \mathcal{B}_p$ are canceled out because of a simple algebraic relation:

$$\frac{\mathbf{d}_p(\mathbf{x}_b)}{|\mathbf{d}_p(\mathbf{x}_b)|^2}\big(\phi(\mathbf{x}_b) - \phi(\mathbf{x}_p)\big) = \frac{\mathbf{d}_p(\mathbf{x}_b)}{|\mathbf{d}_p(\mathbf{x}_b)|^2}\big(\nabla\phi(\mathbf{x}_p) \cdot \mathbf{d}_p(\mathbf{x}_b)\big) = \frac{\mathbf{d}_p(\mathbf{x}_b) \otimes \mathbf{d}_p(\mathbf{x}_b)}{|\mathbf{d}_p(\mathbf{x}_b)|^2}\nabla\phi(\mathbf{x}_p).$$

Therefore, the matrix Eq. (2.3) is rewritten:

$$\left(\sum_{\mathbf{x}\in\mathcal{S}_p^*} \frac{\mathbf{d}_p(\mathbf{x}) \otimes \mathbf{d}_p(\mathbf{x})}{|\mathbf{d}_p(\mathbf{x})|^2}\right)\nabla\phi(\mathbf{x}_p) = \sum_{\mathbf{x}\in\mathcal{S}_p^*} \frac{\mathbf{d}_p(\mathbf{x})}{|\mathbf{d}_p(\mathbf{x})|^2}\big(\phi(\mathbf{x}) - \phi(\mathbf{x}_p)\big), \tag{2.4}$$

where $\mathcal{S}_p^* = \{\mathbf{x}_p \mid q \in \mathcal{N}_p\}$. Consequently, the boundary value $\phi(\mathbf{x}_b)$, $b \in \mathcal{B}_p$, is not practically used to compute a gradient if a linearly extended boundary condition is imposed. Whenever $|\mathcal{S}_p^*| > 2$ which is most of cases in 3D mesh structure, the coefficient matrix in (2.3) is always invertible because a mesh in 3D is not flat. Unfortunately, a cell of $|\mathcal{S}_p^*| \le 2$ possibly exists in given boundary cells, for example, a corner cell of tetrahedron mesh. Such a boundary cell Ω_p can be tessellated to satisfy with $|\mathcal{S}_p^*| > 2$. Note that the cell-centered gradient (2.3) on five stencil points in 2D structured rectangular mesh is the gradient calculated by a centered difference scheme.

As long as a cell-centered gradient is available, a vertex value $\phi(\mathbf{x}_v)$ can be approximated by an inverse distance average from adjacent cells. For a fixed vertex index $v \in \mathcal{V}$, let us denote $\mathcal{N}_v$ as a subset of cell index where a cell $\bar{\Omega}_p$ contains the vertex $\mathbf{x}_v$:

$$\mathcal{N}_v \equiv \left\{p \in \mathcal{I} \mid \mathbf{x}_v \in \partial\Omega_p\right\}, \quad v \in \mathcal{V}.$$

Then, a vertex value from cell-centered values is approximated by Newton's approximation and inverse distance average:

$$\phi(\mathbf{x}_v) = \frac{\displaystyle\sum_{p\in\mathcal{N}_v} \frac{1}{|\mathbf{d}_{pv}|}(\phi(\mathbf{x}_p) + \nabla\phi(\mathbf{x}_p) \cdot \mathbf{d}_{pv})}{\displaystyle\sum_{p\in\mathcal{N}_v} \frac{1}{|\mathbf{d}_{pv}|}}, \quad \mathbf{d}_{pv} = \mathbf{x}_v - \mathbf{x}_p, \ p \in \mathcal{N}_v, \ v \in \mathcal{V}. \tag{2.5}$$

Note that a vertex value $\phi(\mathbf{x}_v)$ on a boundary face is averaged by an inverse distance from linearly extended values from cell centers and cell centered gradients which already contain the linearly extended boundary condition. In case of computing (2.5) in 1-ring face neighborhood structure for parallel computation, a summation routine in message passing interface (MPI) is only necessary on vertices shared by decomposed domains.

Before we define an inflow-based gradient, a constraint face gradient should be computed in order to obtain a flux in (2.1) at a face e_f, $f \in \mathcal{F}$. Let us denote $\mathcal{P}_f$ as a set of points around the face center $\mathbf{x}_f$. For a precise definition of the point set $\mathcal{P}_f$, we need to denote vertices of a face e_f:

$$\mathbf{V}_f = \{\mathbf{x}_{k_i} \mid i = 1, \ldots, |\mathcal{V}_f|\}, \quad \mathcal{V}_f \subset \mathcal{V}$$

where $\mathcal{V}_f$ is a set of vertex indices of face e_f. Then, a set of points $\mathcal{P}_f$ at the face $f \in \mathcal{F}$ is defined:

$$\mathcal{P}_f = \begin{cases} \{\mathbf{x}_p, \mathbf{x}_q\} \cup \mathbf{V}_f & \text{if } \exists!\ p, q \in \mathcal{I} \text{ such that } e_f \in \partial\Omega_p \cap \partial\Omega_q, \\ \{\mathbf{x}_p\} \cup \mathbf{V}_f & \text{if } \exists!\ p \in \mathcal{I} \text{ such that } f \in \mathcal{B}_p \neq \emptyset. \end{cases}$$

Note that $\mathcal{P}_f$ is a generalization of diamond-cell strategy in a regular structured cube mesh [15]. We use a constraint minimization to find constraint face values α_f and gradients $\boldsymbol{\beta}_f$:

$$(\alpha_f, \boldsymbol{\beta}_f) = \underset{|\mathbf{b}_f| \le 1}{\arg\min} \left\{ (a_f, \mathbf{b}_f) \in \mathbb{R}^4 \mid \sum_{\mathbf{x} \in \mathcal{P}_f} w_f(\mathbf{x}) \left| a_f + \mathbf{b}_f \cdot (\mathbf{x} - \mathbf{x}_f) - \phi(\mathbf{x}) \right|^2 \right\}, \tag{2.6}$$

where a weight function is defined by

$$w_f(\mathbf{x}) = \frac{1}{|\mathbf{x} - \mathbf{x}_f|^2}.$$

Note that the values $\phi(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{P}_f$ are already computed in previous steps and the minimization is well-defined because of $|\mathcal{P}_f| \ge 4$. From the minimization functional, the constraint face value α_f on a boundary face is linearly extended value. A reason of using inequality constraint is because a norm of level set gradient is preferable to be 1 in order to keep a signed distance function. The unity constraint may be also reasonable but it is numerically challenging because it is nonconvex minimization. The inequality constraint (2.6) is convex and the minimizer $(\alpha_f, \boldsymbol{\beta}_f)$ is obtained uniquely [19].

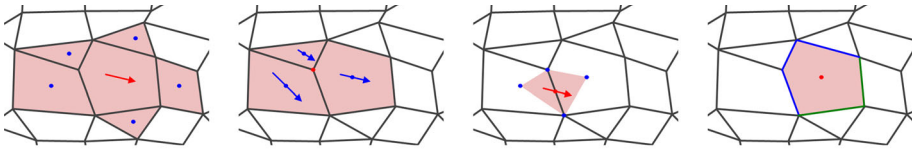
Finally, we define an inflow-based gradient as an inverse distance average of constraint face gradients only from inflow sides. In order to present inflow sides more carefully, we are using signed disjoint index sets:

$$\begin{aligned} \mathcal{B}_p^- &= \{b \in \mathcal{B}_p \mid a_{pb} < 0\} \quad \text{and} \quad \mathcal{B}_p^+ = \mathcal{B}_p \setminus \mathcal{B}_p^-, \\ \mathcal{F}_p^- &= \{f \in \mathcal{F}_p \setminus \mathcal{B}_p \mid a_{pf} < 0\} \quad \text{and} \quad \mathcal{F}_p^+ = (\mathcal{F}_p \setminus \mathcal{B}_p) \setminus \mathcal{F}_p^-. \end{aligned} \tag{2.7}$$

Note that a face index set $\mathcal{F}_p$ can include both boundary and internal faces but signed face index sets $\mathcal{F}_p^+$ or $\mathcal{F}_p^-$ do not include boundary faces. Now, the definition of inflow-based gradient is given by constraint face gradients only from inflow sides:

$$\mathcal{D}_p^- \phi = \begin{cases} \dfrac{\displaystyle\sum_{f \in \mathcal{F}_p^- \cup \mathcal{B}_p^-} \dfrac{1}{|\mathbf{d}_{pf}|} \boldsymbol{\beta}_f}{\displaystyle\sum_{f \in \mathcal{F}_p^- \cup \mathcal{B}_p^-} \dfrac{1}{|\mathbf{d}_{pf}|}} & \text{if } \mathcal{F}_p^- \cup \mathcal{B}_p^- \neq \emptyset, \\ 0 & \text{if } \mathcal{F}_p^- \cup \mathcal{B}_p^- = \emptyset. \end{cases} \tag{2.8}$$

A schematic procedure to compute inflow-based gradient in 2D is illustrated in Fig. 2. The inflow-based gradient in 1D case is derived at the end of section and it is forward or backward difference depending on a sign of flux. In the rest of subsection, we use the inflow-based gradient to compute a face value and present the proposed algorithm with time discretization.



(a) (2.4) **(b)** (2.5) **(c)** (2.6) **(d)** (2.8)

Fig. 2 A schematic procedure to compute an inflow-based gradient in 2D is illustrated from (**a**) to (**d**). The *red* color means the information to compute and the *blue* or *green* color presents given information. In **d**, the *blue* and *green* edges present inflow and outflow fluxes, respectively (Color figure online)

Step. 2 Compute a Face Value ϕ_{pf} in (2.2)

The finite volume formulation in (2.2) is divided into the signed face index sets (2.7):

$$\begin{aligned}\int_{\Omega_p} \frac{\partial \phi}{\partial t} + \sum_{f\in\mathcal{F}_p^-} \left(\phi_{pf} - \phi_p\right) a_{pf} + \sum_{f\in\mathcal{F}_p^+} \left(\phi_{pf} - \phi_p\right) a_{pf} \\ + \sum_{b\in\mathcal{B}_p^-} \left(\phi_{pb} - \phi_p\right) a_{pb} + \sum_{b\in\mathcal{B}_p^+} \left(\phi_{pb} - \phi_p\right) a_{pb} = 0.\end{aligned} \tag{2.9}$$

A time discretization is shown in details after the proposed spatial discretization is completed. The only remaining thing is to compute a face value ϕ_{pf} in disjoint sets (2.7) with linearly extended boundary condition.

Firstly, when a face value is computed at an internal face, it is computed straightforwardly:

$$\begin{aligned} f \in \mathcal{F}_p \setminus \mathcal{B}_p,\ p \in \mathcal{I} \ &\Rightarrow\ \exists!\, q \in \mathcal{N}_p \ \text{ such that } \ e_f \in \partial\Omega_p \cap \partial\Omega_q \\ &\Rightarrow\ \phi_{pf} = \begin{cases} \phi_p + \mathcal{D}_p^- \phi \cdot (\mathbf{x}_f - \mathbf{x}_p) & \text{if } a_{pf} \geq 0, \\ \phi_q + \mathcal{D}_q^- \phi \cdot (\mathbf{x}_f - \mathbf{x}_q) & \text{if } a_{pf} < 0. \end{cases} \end{aligned} \tag{2.10}$$

The flux value a_{pf} in (2.1) is computed by a constraint face gradient:

$$a_{pf} \simeq \frac{\nabla \phi_f}{|\nabla \phi_f|} \cdot \mathbf{n}_{pf} \simeq \frac{\boldsymbol{\beta}_f}{\sqrt{\varepsilon + |\boldsymbol{\beta}_f|^2}} \cdot \mathbf{n}_{pf},$$

where $\varepsilon = 10^{-24}$ is fixed for all examples in Sect. 3. It is numerically more stable to compute (2.1) with a regularization in order to completely make sure $\left|\frac{\nabla \phi_f}{|\nabla \phi_f|}\right| \leq 1$. Note that the internal constraint face value α_f, $f \in \mathcal{F}_p \setminus \mathcal{B}_p$, in (2.6) is different from a face value ϕ_{pf} in (2.10). When a cell-centered gradient is used in (2.10) instead of inflow-based gradient, we show how the whole method in 1D is changed in Sect. 2.3. From (2.8), an inflow-based gradient can be understood as an approximation of a gradient at cell center only from inflow direction. The magnitude of inflow-based gradient is less than 1 because of $|\boldsymbol{\beta}_f| \leq 1$, which gives a limitation of slope to approximate a face value ϕ_{pf}. At the next subsection, we show that such a limitation can be interpreted as a slope limiter in case of 1D domain.

Secondly, when a face value is computed at a boundary face, we formally introduce a ghost cell value $\phi_{q^*} = \phi(\mathbf{x}_{q^*})$:

$$b \in \mathcal{B}_p(\neq \emptyset),\ p \in \mathcal{I} \ \Rightarrow\ \phi_{pb} = \begin{cases} \phi_p + \mathcal{D}_p^- \phi \cdot (\mathbf{x}_b - \mathbf{x}_p) & \text{if } a_{pb} \geq 0, \\ \phi_{q^*} + \boldsymbol{\beta}_b \cdot (\mathbf{x}_b - \mathbf{x}_{q^*}) & \text{if } a_{pb} < 0, \end{cases} \tag{2.11}$$

where $\mathbf{x}_{q^*} = \mathbf{x}_b + h\mathbf{n}_b$, $h > 0$ and $\mathbf{n}_b$ is an outward normal vector to the boundary face. The value $\phi(\mathbf{x}_{q^*})$ should be computed in advance, where $\mathbf{x}_{q^*}$ is located at outside of computational domain Ω. Applying linearly extended boundary condition at $\mathbf{x}_b$, the value $\phi(\mathbf{x}_{q^*})$ can be computed by Newton's approximation with a constraint boundary value and gradient (2.6):

$$\phi(\mathbf{x}_{q^*}) = \alpha_b + \boldsymbol{\beta}_b \cdot (\mathbf{x}_{q^*} - \mathbf{x}_b).$$

Combining the above formulation and (2.11), the boundary face value is computed:

$$b \in \mathcal{B}_p(\neq \emptyset), \quad p \in \mathcal{I} \;\Rightarrow\; \phi_{pb} = \begin{cases} \phi_p + \mathcal{D}_p^- \phi \cdot (\mathbf{x}_b - \mathbf{x}_p) & \text{if } a_{pb} \geq 0, \\ \alpha_b & \text{if } a_{pb} < 0. \end{cases} \tag{2.12}$$

Note that the boundary constraint face value $\alpha_b, b \in \mathcal{B}_p$ is obtained by consistently imposing linearly extended boundary condition.

Inserting (2.10) and (2.12) into (2.9), we finally have the spatial discretization of proposed method:

$$\int_{\Omega_p} \frac{\partial \phi}{\partial t} = \mathcal{L}(\phi_p), \tag{2.13}$$

where

$$\begin{aligned} \mathcal{L}(\phi) = &- \sum_{f \in \mathcal{F}_p^-} \left(\phi_q + \mathcal{D}_q^- \phi \cdot \mathbf{d}_{qf} - \phi_p \right) a_{pf} - \sum_{f \in \mathcal{F}_p^+} \left(\mathcal{D}_p^- \phi \cdot \mathbf{d}_{pf} \right) a_{pf} \\ &- \sum_{b \in \mathcal{B}_p^-} \left(\alpha_b - \phi_p \right) a_{pb} - \sum_{b \in \mathcal{B}_p^+} \left(\mathcal{D}_p^- \phi \cdot \mathbf{d}_{pb} \right) a_{pb}, \end{aligned}$$

$\mathbf{d}_{qf} = \mathbf{x}_f - \mathbf{x}_q$, and for each $f \in \mathcal{F}_p \setminus \mathcal{B}_p$, $p \in \mathcal{I}$, there exists an index $q \in \mathcal{N}_p$ such that $e_f \subset \partial\Omega_p \cap \partial\Omega_q$. Let us denote a time discretization as an ordered set:

$$\{t_0 = 0 < t_1 < \ldots < t_k = T\} \quad \text{and} \quad \Delta t_n = t_{n+1} - t_n, \; n = 0 \ldots k-1.$$

Since spatial discretizations in (2.11) and (2.12) provide the second order scheme, we compatibly use the second order total variation diminishing (TVD) Runge–Kutta method [20,21] in explicit time discretization. That is, re-writing (2.13)

$$\phi_p^{n^*} = \phi_p^n + \frac{\Delta t_n}{|\Omega_p|} \mathcal{L}\left(\phi_p^n\right),$$

the numerical solution at the next time step is obtained by

$$\phi_p^{n+1} = \frac{1}{2}\phi_p^n + \frac{1}{2}\left(\phi_p^{n^*} + \frac{\Delta t_n}{|\Omega_p|} \mathcal{L}\left(\phi_p^{n^*}\right) \right).$$

Since the propose scheme (2.13) is explicit, the time step Δt_n should be restricted and we use the same method as [14]:

$$\Delta t_n = CFL \cdot \min_{p \in \mathcal{I}} \tau_p, \quad \tau_p = |\Omega_p| \left(\sum_{f \in \mathcal{F}^- \cup \mathcal{B}^-} |a_{pf}| \right)^{-1},$$

where $CFL = 0.9$ is fixed for all examples in Sect. 3.

2.3 Similarities to Previous Schemes

In this subsection, we would like to show how the proposed inflow-based gradient provides a second order upwind discretization in 1D as an extension of well-known schemes introduced in Sect. 1.1.

In order to present the proposed scheme in 1D domain, a constraint face value and gradient (2.6) should be calculated and they are a value at the interface between two intervals. From (2.6), we obtain

$$\alpha_{i+\frac{1}{2}} = \frac{\phi_i + \phi_{i+1}}{2} \quad \text{and} \quad \beta_{i+\frac{1}{2}} = \delta\left(\partial_{i+1}^{-1}\phi\right),$$

where δ is defined by

$$\delta(x) = \begin{cases} \dfrac{x}{|x|} & \text{if } |x| > 1, \\ x & \text{otherwise.} \end{cases}$$

Then, an inflow-based gradient in 1D is obtained by (2.8):

$$\mathcal{D}_i^- \phi = \begin{cases} \beta_{i+\frac{1}{2}} & \text{if } a_{i+\frac{1}{2}} < 0 \text{ and } a_{i-\frac{1}{2}} \geq 0, \\ \beta_{i-\frac{1}{2}} & \text{if } a_{i+\frac{1}{2}} \geq 0 \text{ and } a_{i-\frac{1}{2}} < 0, \\ 0 & \text{if } a_{i+\frac{1}{2}} \geq 0 \text{ and } a_{i-\frac{1}{2}} \geq 0, \\ \dfrac{1}{2}\left(\beta_{i+\frac{1}{2}} + \beta_{i-\frac{1}{2}}\right) & \text{if } a_{i+\frac{1}{2}} < 0 \text{ and } a_{i-\frac{1}{2}} < 0, \end{cases} \tag{2.14}$$

where $a_{i+\frac{1}{2}}$ and $a_{i-\frac{1}{2}}$ are fluxes in (2.1) at two end points of an interval I_i. We would like to emphasize that the inflow-based gradient contains derivative information at the end points of I_i only from inflow side. However, assuming $\delta(x) = x$ to simplify an observation, a central difference gradient at the center of I_i used in [15,17]:

$$\mathcal{C}_i\phi = \partial_i^0\phi = \frac{\phi_{i+1} - \phi_{i-1}}{2h} = \frac{1}{2}\left(\partial_{i+1}^-\phi + \partial_i^-\phi\right) = \frac{1}{2}\left(\beta_{i+\frac{1}{2}} + \beta_{i-\frac{1}{2}}\right) \tag{2.15}$$

shows a mixture of derivative information at the end points of I_i regardless of signs of flux. The inflow-based gradient uses the same derivative information but it has four different cases depending on signs of flux.

From a formal expression of FVM of (1.1) on an interval I_i with $F = 1$ and $G = 0$,

$$\int_{I_i} \frac{\partial \phi}{\partial t} + \int_{I_i} \left|\frac{\partial \phi}{\partial x}\right| = 0,$$

where

$$\int_{I_i} \left|\frac{\partial \phi}{\partial x}\right| \simeq \mathcal{M}_i\phi = \frac{1}{h}\left(\left(\phi_{i+\frac{1}{2}} - \phi_i\right) a_{i+\frac{1}{2}} + \left(\phi_{i-\frac{1}{2}} - \phi_i\right) a_{i-\frac{1}{2}}\right), \tag{2.16}$$

face values in the proposed schemes are computed:

$$\begin{cases} a_{i+\frac{1}{2}} \geq 0 \;\Rightarrow\; \phi_{i+\frac{1}{2}} = \phi_i + \frac{h}{2}\mathcal{D}_i^-\phi, \\ a_{i+\frac{1}{2}} < 0 \;\Rightarrow\; \phi_{i+\frac{1}{2}} = \phi_{i+1} - \frac{h}{2}\mathcal{D}_{i+1}^-\phi, \end{cases} \quad \text{and} \quad \begin{cases} a_{i-\frac{1}{2}} \geq 0 \;\Rightarrow\; \phi_{i-\frac{1}{2}} = \phi_i - \frac{h}{2}\mathcal{D}_i^-\phi, \\ a_{i-\frac{1}{2}} < 0 \;\Rightarrow\; \phi_{i-\frac{1}{2}} = \phi_{i-1} + \frac{h}{2}\mathcal{D}_{i-1}^-\phi. \end{cases}$$

Now, let us compare all cases in Fig. 1 in details. In the Case 1, the proposed scheme in 1D on an interval I_i is written:

$$\mathcal{M}_i\phi = \frac{1}{h}\left(\left(\phi_i + \frac{h}{2}\beta_{i-\frac{1}{2}} - \phi_i\right)\cdot(1) + \left(\phi_{i-1} + \frac{h}{2}\beta_{i-\frac{3}{2}} - \phi_i\right)\cdot(-1)\right).$$

Now, by using a different representation of $\beta_{i+\frac{1}{2}}$,

$$\beta_{i+\frac{1}{2}} = \delta\left(\partial_{i+1}^{-1}\phi\right) = \rho_{i+1}\partial_{i+1}^{-1}\phi,$$

where

$$\rho_{i+1} = \begin{cases} \dfrac{1}{|\partial_{i+1}^{-1}\phi|} & \text{if } |\partial_{i+1}^{-1}\phi| > 1, \\ 1 & \text{otherwise,} \end{cases} \tag{2.17}$$

it is easy to observe that a nonlinear slope limiter ρ is used to approximate face values $\phi_{i+\frac{1}{2}}$ and $\phi_{i-\frac{1}{2}}$ of the interval I_i:

$$\mathcal{M}_i\phi = \frac{1}{h}\left(\left(\phi_i + \frac{h}{2}\rho_i\partial_i^{-1}\phi - \phi_i\right)\cdot(1) + \left(\phi_{i-1} + \frac{h}{2}\rho_{i-1}\partial_{i-1}^{-1}\phi - \phi_i\right)\cdot(-1)\right).$$

If we assume that ϕ is a signed distance function, then the proposed scheme for the Case 1 in Fig. 1 exactly provides the second order upwind discretization because of $\rho = 1$:

$$\mathcal{M}_i\phi = \frac{3\phi_i - 4\phi_{i-1} + \phi_{i-2}}{2h} = |\partial_i^{-2}\phi|.$$

With a similar computation, the spatial discretization (2.16) of proposed scheme for all cases in Fig. 1 can be written:

$$\mathcal{M}_i\phi = \begin{cases} |\partial_i^{-2}\phi| & \text{Case 1,} \\ |\partial_i^{+2}\phi| & \text{Case 2,} \\ 0 & \text{Case 3,} \\ \frac{h}{2}\partial_i^2\phi & \text{Case 4,} \end{cases} \tag{2.18}$$

where $\partial_i^{-2}\phi$ and $\partial_i^{+2}\phi$ are the second order upwind differences in (1.3) and $\partial_i^2\phi$ is a second order central difference of second derivative:

$$\partial_i^2\phi = \frac{\phi_{i+2} - 3\phi_{i+1} + 4\phi_i - 3\phi_{i-1} + \phi_{i-2}}{h^2} \simeq \left.\frac{\partial^2\phi}{\partial x^2}\right|_{x=x_i} + \mathcal{O}(h^2).$$

If the proposed scheme (2.18) is compared with Rouy–Tourin and Osher–Sethian schemes in Table 1, an upwind discretization is clearly improved to the second order difference. An inflow-based gradient essentially provides the second order upwind discretization. In the formulation (2.10), if a central difference gradient (2.15) is used instead of inflow-based gradient, then the main part of scheme (2.16) for a Case 1 in Fig. 1 is changed as an average of central difference and the second order backward difference:

$$\mathcal{M}_i\phi = \frac{|\partial_i^0\phi| + |\phi_i^{-2}\phi|}{2}. \tag{2.19}$$

It is not purely upwind discretization compared to the Case 1 in the proposed scheme (2.18). Note that we numerically observe an instability in some examples in Sect. 3 when (2.19) is

used instead of (2.18). In Case 3, all methods are identically same and we have a second order central difference of second derivative in Case 4. When a level set is smooth enough to have bounded derivatives, the Case 4 is numerically ignored as $h \to 0$. We may roughly understand the Case 4 in the proposed scheme as localized numerical viscosity solution. Since the proposed scheme is based on FVM, it can be used in polyhedron meshes, which is not straightforwardly possible to Rouy–Tourin and Osher–Sethian schemes.

3 Numerical Experiments

In the following subsections, various experiments are presented to show numerical properties of proposed scheme. The examples are computed in different shapes of meshes generated by AVL FIRE® and we always use decomposed computational domains with 1-ring face neighborhood structure.

3.1 Bidirectional Flow

We present an example which can be applied to accurately find a signed distance function or reinitialize the level set function. We assume that a closed surface Π is given to divide a computational domain into two open sets Π^+ and Π^-:

$$\Pi = \partial\Pi^+ \cap \partial\Pi^-, \quad \bar{\Pi}^+ \cup \bar{\Pi}^- = \bar{\Omega}, \quad \Pi^+ \cap \Pi^- = \emptyset.$$

Then, we solve two governing equations, so-called bidirectional flow:

$$\begin{aligned} &\frac{\partial}{\partial t}\phi(\mathbf{x},t) + |\nabla\phi(\mathbf{x},t)| = 1, \quad (\mathbf{x},t) \in \Pi^+ \times [0,T], \\ &\frac{\partial}{\partial t}\phi(\mathbf{x},t) - |\nabla\phi(\mathbf{x},t)| = -1, \quad (\mathbf{x},t) \in \Pi^- \times [0,T], \end{aligned} \tag{3.1}$$

where linearly extended boundary condition is imposed on $\partial\Omega$. Note that it is formally same as well-known reinitialization equation in [22]:

$$\frac{\partial}{\partial t}\phi(\mathbf{x},t) + sgn(\phi_0(\mathbf{x}))(|\nabla\phi(\mathbf{x},t)| - 1) = 0, \quad (\mathbf{x},t) \in \Pi^0 \times [0,T],$$

where $\Pi^0 = \Omega \setminus \Pi$, $sgn(\mathbf{x})$ is a sign function, and $\phi_0(\mathbf{x})$ is a continuous function whose zero level set is Π and its value on Π^+ and Π^- is positive and negative, respectively. In level set literature in surface evolution, it is a useful equation to recover a signed distance function. However, it is also notoriously difficult to numerically solve on unstructured meshes especially with finite volume method [23].

We assume that a given surface Π is analytically represented or a triangulated surface. In a discretized domain, a level set function value is fixed on a cell contains a part of surface. Such a value can be analytically obtained or approximated from a triangulated surface. We denote three types of cells in a discretized computational domain:

$$\bar{\Omega} = \bigcup \left\{ \bar{\Omega}_p : p \in \mathcal{I}^+ \cup \mathcal{I}^0 \cup \mathcal{I}^- = \mathcal{I} \right\},$$

where $\mathcal{I}^+$, $\mathcal{I}^0$, and $\mathcal{I}^-$ are mutually disjoint and $\mathcal{I}^+ \cap \mathcal{I}^0 \cap \mathcal{I}^- = \emptyset$. If all corners of a cell $p \in \mathcal{I}$ are located in Π^+ or Π^-, then $p \in \mathcal{I}^+$ or $p \in \mathcal{I}^-$, respectively. The index set $\mathcal{I}^0$ is the complement set of $\mathcal{I}^+ \cup \mathcal{I}^-$. An initial level set is given on the discrete domains:

$$\phi_0(\mathbf{x}_p) = \begin{cases} d^+, & p \in \mathcal{I}^+, \\ d_0(\mathbf{x}_p), & p \in \mathcal{I}^0, \\ d^-, & p \in \mathcal{I}^-. \end{cases} \tag{3.2}$$

The value d_0 is either an exact value from an analytically represented signed distance function or an approximated signed distance value from a point to a triangulated surface. The values d^+ and d^- can be any positive and negative values, respectively. In the examples of Figs. 4 and 5, $d^+ = 0.002$ and $d^- = -0.002$ are used. The values d_0 are fixed along the time evolution, that is, $\phi(\mathbf{x}_p, t) = d_o(\mathbf{x}_p)$ for all $p \in \mathcal{I}^0$ and $t > 0$. The initial values d_0 are extremely sparse in 3D computational domain and they are propagated into a whole domain to recover a signed distance function as $t \to \infty$ in (3.1). Note that a method [24] can be used to compute the values d_0 if analytic values are not available and a triangulated surface of Π is given.

Now, we consider computational domains of two shapes: a box shape

$$\bar{\Omega} = \bigcup_{p \in \mathcal{I}} \bar{\Omega}_p = [-0.05, 0.05]^3 \subset \mathbb{R}^3$$

and a cylinder shape whose height is 0.1 and radius is 0.05. The discrete cell Ω_p is either a hexahedron or polyhedron mesh in Fig. 3. In order to check an EOC, four levels of meshes in Table 2 are used and we approximately generate 8 times smaller volume if one level gets higher. In the examples of bidirectional flow from an analytically given initial function, since the zero level set of an initial function is frozen and an exact solution is known, it is interested to see the global errors when the final time T is large:

$$L^\infty = \max_{\mathbf{x} \in \Omega} \left\{ \left| \phi(\mathbf{x}, T) - \phi^e(\mathbf{x}) \right| \right\} \simeq \max_{p \in \mathcal{I}} \left\{ \left| \phi(\mathbf{x}_p, T) - \phi^e(\mathbf{x}_p) \right| \right\}, \tag{3.3}$$

$$L^1 = \frac{1}{|\Omega|} \int_\Omega \left| \phi(\mathbf{x}, T) - \phi^e(\mathbf{x}) \right| \simeq \frac{1}{|\Omega|} \sum_{p \in \mathcal{I}} \left| \phi(\mathbf{x}_p, T) - \phi^e(\mathbf{x}_p) \right| |\Omega_p|, \tag{3.4}$$

where ϕ^e is an exact solution. Considering the size of computational domain and the speed of propagation, $T = 0.2$ is large enough to propagate all initial values through the whole domain. The level set functions at $T = 0.2$ of bidirectional flow (3.1) from a sphere and a cube are illustrated in Fig. 4.

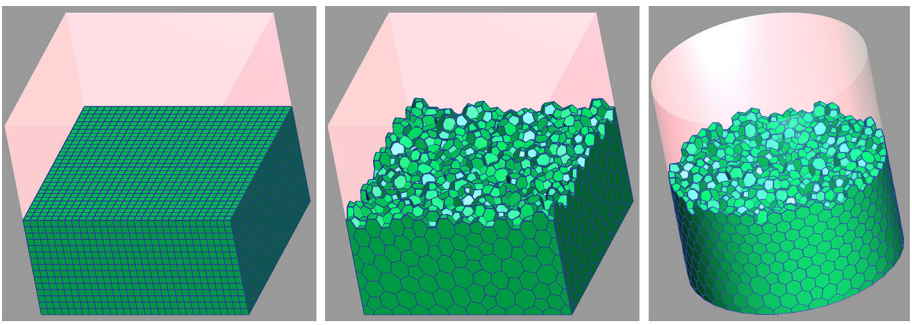


Fig. 3 The types of cells in $\mathcal{H}_b$ and $\mathcal{P}_b$ or $\mathcal{P}_c$ meshes are hexahedrons and polyhedrons, respectively. The inside structures of *box* and *cylindirical* shape are illustrated

Table 2 The number of cells(c), faces(f), and vertices(v) of meshes in Fig. 3 are enumerated

level	c	f	v
	Hexahedrons in a box ($\mathcal{H}_b$)		
1	27, 000	83, 700	29, 791
2	216, 000	658, 800	226, 981
3	1, 728, 000	5, 227, 200	1, 771, 561
4	13, 824, 000	41, 644, 800	13, 997, 521
	Polyhedrons in box ($\mathcal{P}_b$)		
1	4, 033	28, 458	24, 428
2	30, 683	222, 745	192, 065
3	241, 726	1, 782, 338	1, 540, 615
4	1, 914, 579	14, 227, 431	12, 312, 855
	Polyhedrons in cylinder ($\mathcal{P}_c$)		
1	3, 947	27, 643	23, 695
2	28, 410	206, 829	178, 418
3	224, 548	1, 658, 925	1, 434, 376
4	1, 788, 209	13, 301, 701	11, 513, 491

We approximately generate 8 times smaller volume if one level gets higher

In the case of bidirectional flow from a sphere, a partial L^1_ϵ and L^∞_ϵ errors are additionally computed using (3.3) and (3.4) on a subdomain $\Omega_\epsilon = \{\mathbf{x} \in \Omega \mid |\mathbf{x}| > \epsilon\}$, $\epsilon = 0.01$. Since no singularities of solution occur in the subdomain, the EOC of proposed method exhibits second order accuracy on hexahedron and polyhedron meshes; see Table 3. We note that the L^1_ϵ and L^1 errors provide nearly same results of EOC as they are not sensitive to the singularity of solution at the center of sphere. However, the EOC from L^∞_ϵ and L^∞ errors shows a clear difference with only the L^∞_ϵ error showing second order accuracy. In the case of bidirectional flow from a cube we present only the EOC using the global L^1 or L^∞ errors for which one observes the first order accuracy; see Table 4.

For given triangulated surfaces, a qualitative comparison is presented to show a practical robustness of the proposed method. We use the Stanford Bunny and Dragon from the Stanford 3D scanning repository.[1] A meshing toolkit `AVL FAME` is applied to slightly modify the original triangulated surfaces in order to make a closed and oriented surface.

For the Dragon example in Fig. 5, hexahedron and polyhedron meshes are generated in a box shape to enclose the entire triangulated surfaces of the Dragon and we solve the bidirectional flow to obtain a signed distance function. In order to obtain a proper initial condition (3.2), we start with finding the first layer of cells which have an intersection with a given triangulated surface. After that, from an octree structure generated on a triangulated surface, we can efficiently search a triangle which has a minimum distance to a vertex of cell. The angle weighted pseudonormal [25] is used to verify topological information whether a vertex of cells in the first layer is located outside or inside of a triangulated surface. For the other cells which are not in the first layer, we use a simple flood fill algorithm to decide the topological status from the first layer. A cell center value in the first layer is calculated by weighted least square algorithms from the vertex values of the cell. We illustrate a given triangulated surface in Fig. 5a. The isocurves of numerical solution on xy, xy, and zx-planes are shown in Fig. 5b. The isosurfaces at the different level values are plotted from hexahedron mesh in Fig. 5c and polyhedron mesh in Fig. 5d. The results from two different mesh types

[1] http://graphics.stanford.edu/data/3Dscanrep.

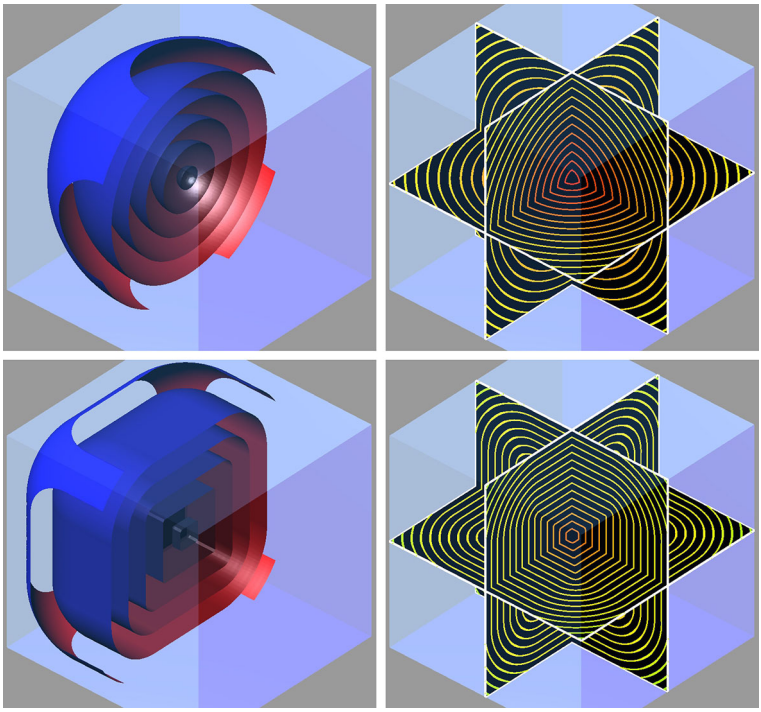


Fig. 4 Results of bidirectional flow (3.1) at $T = 0.2$ from a *sphere* and a *cube*: The *first column* is the isosurfaces of level set functions. The *second column* is the isocurves on x, y, z-planes

have almost no noticeable visual differences and it shows that the proposed algorithm robustly work in 3D polyhedron meshes.

For the Stanford bunny example in Fig. 6, we use the modified triangulated surface to generate hexahedron or polyhedron meshes into the Standford bunny and only one side of bidirectional flow is solved to obtain a distance function from the triangulated surface. Note that there is small perturbation of initial surface when the meshes are generated and it slightly changes the location of surface boundary shown in Figs. 6a, c. The cell shape in (a) is hexahedron. A dominant cell shape in (c) is polyhedron, but there are also prism and hexahedron shapes. The isosurfaces at the different level values are plotted from hexahedron mesh in Fig. 6b and polyhedron mesh in Fig. 6d. Even though the cell shapes are irregular in both cases and there is a mixture of cells in (c), results computed by the proposed method are stable.

3.2 Shrinking and Expanding Shapes

In this subsection, from an analytically given surface shape, we illustrate the surfaces which shrinks or expands along the surface normal direction. The expanding or shrinking shapes are obtained by solving (1.1) with $F = \pm 1$ and $G = 0$:

$$\frac{\partial}{\partial t}\phi(\mathbf{x}, t) \pm |\nabla\phi(\mathbf{x}, t)| = 0, \quad (\mathbf{x}, t) \in \Omega \times [0, T],$$

Table 3 Experimental order of convergence (EOC) of bidirectional flow from a sphere in Fig. 4

Sphere	$\mathcal{H}_b$		$\mathcal{P}_b$		$\mathcal{P}_c$	
Level	L_ϵ^∞	EOC	L_ϵ^∞	EOC	L_ϵ^∞	EOC
1	$7.32 \cdot 10^{-5}$	–	$4.23 \cdot 10^{-4}$	–	$3.83 \cdot 10^{-4}$	–
2	$1.92 \cdot 10^{-5}$	1.93	$1.87 \cdot 10^{-4}$	1.18	$1.77 \cdot 10^{-4}$	1.12
3	$4.88 \cdot 10^{-6}$	1.97	$4.97 \cdot 10^{-5}$	1.91	$4.54 \cdot 10^{-5}$	1.96
4	$1.24 \cdot 10^{-6}$	1.98	$1.51 \cdot 10^{-5}$	1.72	$1.31 \cdot 10^{-5}$	1.79
Level	L^∞	EOC	L^∞	EOC	L^∞	EOC
1	$3.41 \cdot 10^{-4}$	–	$7.57 \cdot 10^{-4}$	–	$5.38 \cdot 10^{-4}$	–
2	$1.73 \cdot 10^{-4}$	0.98	$4.80 \cdot 10^{-4}$	0.66	$3.78 \cdot 10^{-4}$	0.51
3	$9.52 \cdot 10^{-5}$	0.86	$2.45 \cdot 10^{-4}$	0.97	$2.04 \cdot 10^{-4}$	0.89
4	$4.11 \cdot 10^{-5}$	1.21	$9.97 \cdot 10^{-5}$	1.29	$9.87 \cdot 10^{-5}$	1.05
Level	L_ϵ^1	EOC	L_ϵ^1	EOC	L_ϵ^1	EOC
1	$1.55 \cdot 10^{-5}$	–	$1.55 \cdot 10^{-4}$	–	$1.38 \cdot 10^{-4}$	–
2	$3.85 \cdot 10^{-6}$	2.01	$4.98 \cdot 10^{-5}$	1.64	$4.08 \cdot 10^{-5}$	1.76
3	$9.53 \cdot 10^{-7}$	2.01	$1.35 \cdot 10^{-5}$	1.88	$1.13 \cdot 10^{-5}$	1.86
4	$2.38 \cdot 10^{-7}$	2.00	$3.56 \cdot 10^{-6}$	1.92	$2.96 \cdot 10^{-6}$	1.93
Level	L^1	EOC	L^1	EOC	L^1	EOC
1	$1.58 \cdot 10^{-5}$	–	$1.56 \cdot 10^{-4}$	–	$1.38 \cdot 10^{-4}$	–
2	$3.91 \cdot 10^{-6}$	2.02	$5.02 \cdot 10^{-5}$	1.64	$4.13 \cdot 10^{-5}$	1.75
3	$9.65 \cdot 10^{-7}$	2.02	$1.36 \cdot 10^{-5}$	1.88	$1.14 \cdot 10^{-5}$	1.85
4	$2.40 \cdot 10^{-7}$	2.00	$3.59 \cdot 10^{-6}$	1.92	$3.00 \cdot 10^{-6}$	1.93

The information of $\mathcal{H}_b$, $\mathcal{P}_b$, and $\mathcal{P}_c$ meshes are listed in Table 2. The L^∞ and L^1 errors are computed by (3.3) and (3.4), respectively. The L_ϵ^∞ and L_ϵ^1 errors are computed by (3.3) and (3.4) with $\Omega_\epsilon = \{\mathbf{x} \in \Omega \mid |\mathbf{x}| > \epsilon\}$ and $\epsilon = 0.01$, respectively

with an initial condition $\phi_0(\mathbf{x}) = \phi(\mathbf{x}, 0)$. Similar to examples in [15], spherical and octahedron shapes are used to obtain an initial level set ϕ_0 in a normal directional motion:

$$\Gamma_s(\mathbf{c}_1, \mathbf{c}_2, r) = \{\mathbf{x} : h_s(\mathbf{x}, \mathbf{c}_1, r) = 0 \text{ or } h_s(\mathbf{x}, \mathbf{c}_2, r) = 0\},$$
$$\Gamma_o(\mathbf{c}_1, \mathbf{c}_2, r) = \{\mathbf{x} : h_o(\mathbf{x}, \mathbf{c}_1, r) = 0 \text{ or } h_o(\mathbf{x}, \mathbf{c}_2, r) = 0\},$$

where shape functions h_s and h_o are defined by

$$h_s(\mathbf{x}, \mathbf{c}, r) = |\mathbf{x} - \mathbf{c}| - r \quad \text{and} \quad h_o(\mathbf{x}, \mathbf{c}, r) = \sum_{l=1}^{3} |x_l - c_l| - r. \tag{3.5}$$

In order to check an EOC, four levels of meshes in Table 2 are used. Since the numerical results are meaningful only on the zero level set in these examples, we measure the local error instead of using the global norms in (3.4) or (3.3):

Table 4 Experimental order of convergence (EOC) of bidirectional flow from a cube in Fig. 4

Cube	$\mathcal{H}_b$		$\mathcal{P}_b$		$\mathcal{P}_c$	
Level	L^∞	EOC	L^∞	EOC	L^∞	EOC
1	$2.14 \cdot 10^{-3}$	–	$3.39 \cdot 10^{-3}$	–	$3.46 \cdot 10^{-3}$	
2	$1.16 \cdot 10^{-3}$	0.87	$1.87 \cdot 10^{-3}$	0.86	$1.91 \cdot 10^{-3}$	0.86
3	$5.23 \cdot 10^{-4}$	1.16	$6.53 \cdot 10^{-4}$	1.51	$6.60 \cdot 10^{-4}$	1.53
4	$2.89 \cdot 10^{-4}$	0.85	$3.74 \cdot 10^{-4}$	0.81	$3.77 \cdot 10^{-4}$	0.81
Level	L^1	EOC	L^1	EOC	L^1	EOC
1	$6.74 \cdot 10^{-4}$	–	$7.51 \cdot 10^{-4}$	–	$5.46 \cdot 10^{-4}$	
2	$3.56 \cdot 10^{-4}$	0.92	$3.39 \cdot 10^{-4}$	1.15	$2.94 \cdot 10^{-4}$	0.89
3	$1.58 \cdot 10^{-4}$	1.18	$1.56 \cdot 10^{-4}$	1.13	$1.12 \cdot 10^{-4}$	1.39
4	$8.44 \cdot 10^{-5}$	0.90	$8.01 \cdot 10^{-5}$	0.96	$5.43 \cdot 10^{-5}$	1.04

The information of $\mathcal{H}_b$, $\mathcal{P}_b$, and $\mathcal{P}_c$ meshes are listed in Table 2. The L^∞ and L^1 errors are computed by (3.3) and (3.4), respectively

$$L^1_{loc} = \frac{1}{|\Gamma|}\int_\Gamma \left|\phi(\mathbf{x}, T) - \phi^e(\mathbf{x})\right| \simeq \frac{1}{\sum_{p\in\mathcal{J}\subsetneq\mathcal{I}} |\Omega_p|} \sum_{p\in\mathcal{J}\subsetneq\mathcal{I}} \left|\phi(\mathbf{x}_p, T) - \phi^e(\mathbf{x}_p)\right| |\Omega_p|, \tag{3.6}$$

where ϕ^e is an exact solution, Γ is a zero level set of ϕ^e, and $\mathcal{J}$ is a set of cell index whose signs of vertex values of the exact solution are not identical.

For a shrinking motion, we use $F = -1$ and $G = 0$ in (1.1) and the initial level set ϕ_0 is a signed distance function from a surface

$$\Gamma_s(\mathbf{c}_1, \mathbf{c}_2, r) \text{ or } \Gamma_o(\mathbf{c}_1, \mathbf{c}_2, r) \text{ with } \begin{cases} \mathbf{c}_1 = (-0.025, 0, 0), \\ \mathbf{c}_2 = (0.025, 0, 0), \\ r = 0.02. \end{cases} \tag{3.7}$$

Note that ϕ_0 is negative inside of shape and positive outside of shape. The final time $T = 0.005$ is fixed and then the exact solution is $\phi^e = \phi_0 + T$ only on a zero level set. The numerical results of level set function at $T = 0.005$ from the proposed scheme are illustrated in Fig. 7.

The EOC of L^1_{loc} error shown in Table 5 is around 2 for shrinking spheres as the error is computed only on a smooth part of solution. As expected the first order rate of convergence is observed for shrinking octahedrons because of the discontinuous gradient of solution along edges. However, the EOC seems to be higher than 1.

For an expanding motion, we use $F = 1$ and $G = 0$ in (1.1) and the initial level set is a signed distance function from a surface

$$\Gamma_s(\mathbf{c}_1, \mathbf{c}_2, r) \text{ or } \Gamma_o(\mathbf{c}_1, \mathbf{c}_2, r) \text{ with } \begin{cases} \mathbf{c}_1 = (-0.025, 0, 0), \\ \mathbf{c}_2 = (0.025, 0, 0), \\ r = 0.024. \end{cases} \tag{3.8}$$

The value r is purposely set to be larger than shrinking cases in order to have a merged shape at the same final time $T = 0.005$. Since ϕ_0 is negative inside of shape and positive outside

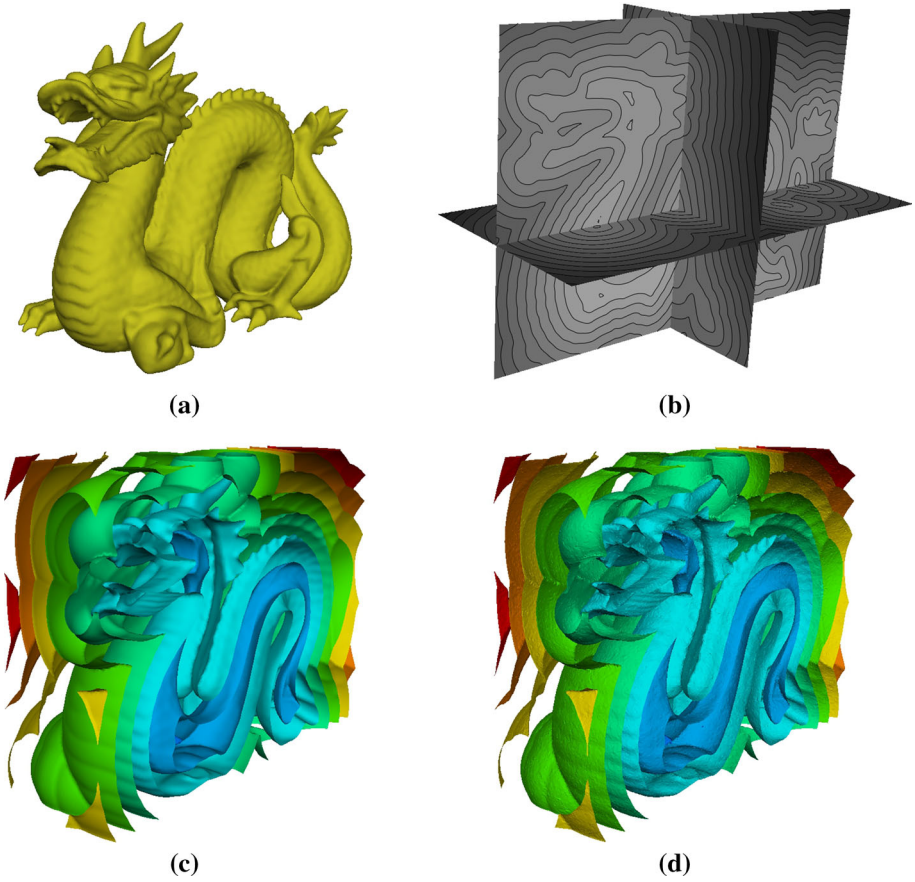


Fig. 5 **a** A given triangulated surface. **b** Isocurves of (3.1) on xy, xy, and zx-planes. **c** Isosurfaces of a result computed on a hexahedron mesh. **d** Isosurfaces of a result computed on a polyhedron mesh

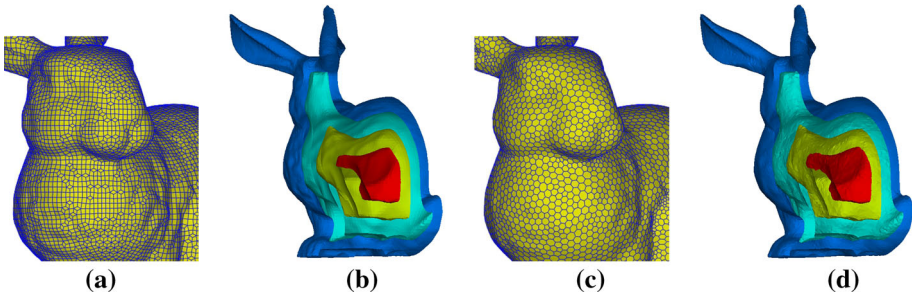


Fig. 6 **a** A hexahedron mesh generated by a triangulated surface. **b** Isosurfaces of a result computed by proposed method on the mesh (**a**). **c** A polyhedron mesh generated by a triangulated surface. **d** Isosurfaces of a result computed by proposed method on the mesh (**c**)

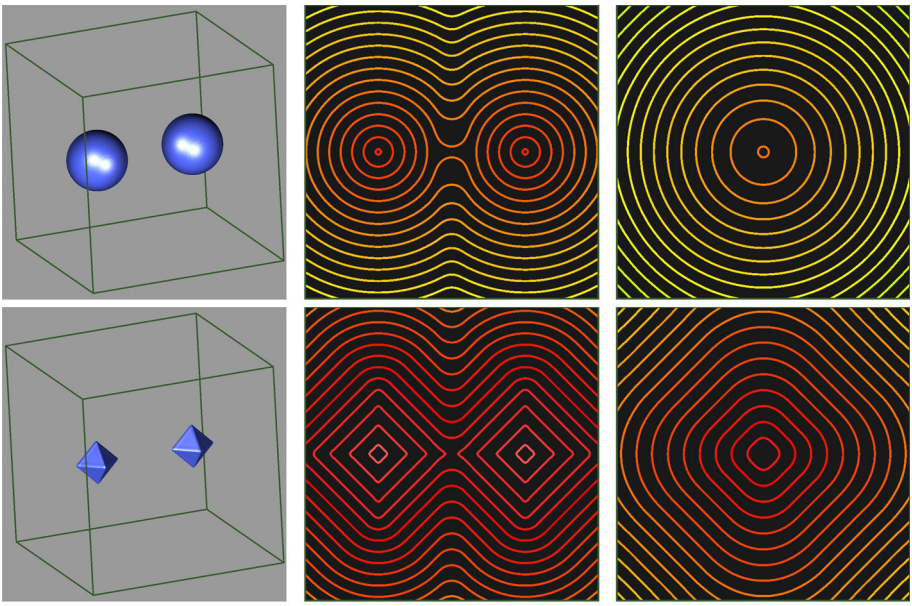
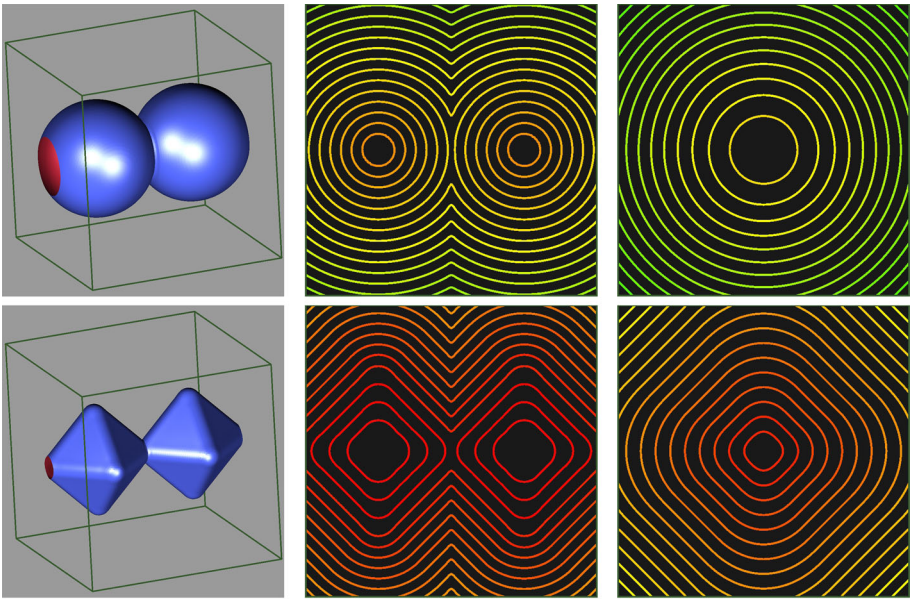


Fig. 7 **a**: The results of shrinking shapes at $T = 0.005$ from the initial shapes Γ_s and Γ_o (3.7) on $\mathcal{H}_b$ mesh in Fig. 3. **b** and **c**: Isocurves on xy-plane and yz-plane, respectively

Table 5 Experimental order of convergence (EOC) of shringking shapes in Fig. 7

Level	Shrinking spheres					
	$\mathcal{H}_b$		$\mathcal{P}_b$		$\mathcal{P}_c$	
	L^1_{loc}	EOC	L^1_{loc}	EOC	L^1_{loc}	EOC
1	$2.90 \cdot 10^{-5}$	–	$1.03 \cdot 10^{-4}$	–	$1.09 \cdot 10^{-4}$	–
2	$5.37 \cdot 10^{-6}$	2.43	$4.49 \cdot 10^{-5}$	1.19	$4.21 \cdot 10^{-5}$	1.37
3	$7.53 \cdot 10^{-7}$	2.83	$1.02 \cdot 10^{-5}$	2.14	$8.74 \cdot 10^{-6}$	2.27
4	$1.39 \cdot 10^{-7}$	2.43	$2.15 \cdot 10^{-6}$	2.24	$1.89 \cdot 10^{-6}$	2.21
Level	Shrinking octahedrons					
	$\mathcal{H}_b$		$\mathcal{P}_b$		$\mathcal{P}_c$	
	L^1_{loc}	EOC	L^1_{loc}	EOC	L^1_{loc}	EOC
1	$3.63 \cdot 10^{-4}$	–	$6.64 \cdot 10^{-4}$	–	$5.53 \cdot 10^{-4}$	–
2	$2.00 \cdot 10^{-4}$	0.86	$3.63 \cdot 10^{-4}$	0.87	$3.42 \cdot 10^{-4}$	0.69
3	$6.79 \cdot 10^{-5}$	1.56	$1.61 \cdot 10^{-4}$	1.17	$1.41 \cdot 10^{-4}$	1.28
4	$1.32 \cdot 10^{-5}$	2.36	$4.81 \cdot 10^{-5}$	1.74	$4.31 \cdot 10^{-5}$	1.71

The information of $\mathcal{H}_b$, $\mathcal{P}_b$, and $\mathcal{P}_c$ is listed in Table 2. The L^1_{loc} error (3.6) is computed at $T = 0.005$



(a) **(b)** **(c)**

Fig. 8 **a**: The results of expanding shapes at $T = 0.005$ from the initial shapes Γ_s and Γ_o (3.8) on $\mathcal{H}_b$ mesh in Fig. 3. **b** and **c**: Isocurves on xy-plane and yz-plane, respectively

Table 6 Experimental order of convergence (EOC) of expanding shapes in Fig. 8

Level	Expanding spheres					
	$\mathcal{H}_b$		$\mathcal{P}_b$		$\mathcal{P}_c$	
	L^1_{loc}	EOC	L^1_{loc}	EOC	L^1_{loc}	EOC
1	$5.67 \cdot 10^{-5}$	–	$1.37 \cdot 10^{-4}$	–	$1.16 \cdot 10^{-4}$	–
2	$1.64 \cdot 10^{-5}$	1.78	$4.35 \cdot 10^{-5}$	1.66	$4.25 \cdot 10^{-5}$	1.44
3	$4.26 \cdot 10^{-6}$	1.91	$1.37 \cdot 10^{-5}$	1.67	$1.25 \cdot 10^{-5}$	1.77
4	$1.11 \cdot 10^{-6}$	1.97	$3.59 \cdot 10^{-6}$	1.93	$3.23 \cdot 10^{-6}$	1.95
Level	Expanding octahedrons					
	$\mathcal{H}_b$		$\mathcal{P}_b$		$\mathcal{P}_c$	
	L^1_{loc}	EOC	L^1_{loc}	EOC	L^1_{loc}	EOC
1	$2.08 \cdot 10^{-4}$	–	$5.06 \cdot 10^{-4}$	–	$4.47 \cdot 10^{-4}$	–
2	$1.24 \cdot 10^{-4}$	0.75	$2.45 \cdot 10^{-4}$	1.04	$2.31 \cdot 10^{-4}$	0.95
3	$5.12 \cdot 10^{-5}$	1.27	$1.11 \cdot 10^{-4}$	1.14	$1.02 \cdot 10^{-4}$	1.18
4	$2.59 \cdot 10^{-5}$	0.98	$4.63 \cdot 10^{-5}$	1.27	$4.30 \cdot 10^{-5}$	1.24

The information of $\mathcal{H}_b$, $\mathcal{P}_b$, and $\mathcal{P}_c$ is listed in Table 2. The L^1_{loc} error (3.6) is computed at $T = 0.005$

of shape, the exact solution is $\phi^e = \phi_0 - T$ only on a zero level set. The numerical results of level set function at $T = 0.005$ from the proposed scheme are illustrated in Fig. 8.

The EOC of L^1_{loc} error for expanding spheres in Table 6 is close to 2. As expected the EOC of L^1_{loc} error for expanding octahedrons in Table 6 is around 1 because of the presence of discontinuous gradient of solution.

4 Conclusion

In this paper, we propose a numerical scheme to solve a propagation in a normal direction with a linearly extended boundary condition. The proposed scheme is based on a cell-centered finite volume method and it is designed to be used in a 3D polyhedron mesh which is common in real CFD applications. An inflow-based gradient provides a second order upwind discretization of the magnitude of gradient in the case of a regular structured cube mesh. From numerical examples, the second order rate of convergence is observed experimentally for smooth solutions in appropriate norms.

Acknowledgements We would like to thank to Dr. Peter Priesching in AVL LIST GmbH, Graz, Austria and Dr. Kiwan Jeon in National Institute for Mathematical Sciences, Daejeon, South Korea for and to Dr. Peter Sampl and MSc. Dirk Martin in AVL LIST GmbH, Graz, Austria, for generating polyhedron and hexahedron meshes of the Stanford bunny and Dragon examples. We sincerely appreciate valuable comments and feedback from anonymous referee.

References

1. Rouy, E., Tourin, A.: A viscosity solutions approach to shape-from-shading. SIAM J. Numer. Anal. **29**, 867–884 (1992)
2. Osher, S., Sethian, J.A.: Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formultaions. J. Comput. Phys. **79**, 12–49 (1988)
3. Osher, S., Fedkiw, R.: Level Set Methods and Dynamic Implicit Surfaces. Springer, Berlin (2000)
4. Sethian, J.A.: Level Set Methods and Fast Marching Methods, Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materical Science. Cambridge University Press, New York (1999)
5. Perič, M.: Flow simulation using control volumes of arbitrary polyhedral shape. ERCOFTAC Bull. **62**, 25–29 (2004)
6. Zhao, H.K.: A fast sweeping method for Eikonal equations. Math. Comput. **74**, 603–627 (2005)
7. Qian, J., Zhang, Y.-T., Zhao, H.K.: Fast sweeping methods for Eikonal equations on triangular meshes. SIAM J. Numer. Anal. **45**, 83–107 (2007)
8. Detrixhe, M., Gibou, F., Min, C.: A parallel fast sweeping method for the Eikonal equation. J. Comput. Phys. **237**, 46–55 (2013)
9. Dapogny, C., Frey, P.: Computation of the signed distance function to a discrete contour on adapted triangulation. Calcolo **49**, 193–219 (2012)
10. Liu, X.-D., Osher, S., Chan, T.: Weighted essentially non-oscillatory schemes. J. Comput. Phys. **115**, 200–212 (1994)
11. Jiang, G., Shu, C.-W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. **126**, 202–228 (1996)
12. Zhang, Y.-T., Shu, C.-W.: High order WENO schemes for Hamilton–Jacobi equations on triangular meshes. SIAM J. Sci. Comput. **24**, 1005–1030 (2003)
13. Tsoutsanis, P., Titarev, V., Drikakis, D.: WENO schemes on arbitrary mixed-element unstructured meshes in three space dimensions. J. Comput. Phys. **230**, 1585–1601 (2011)
14. Frolkovič, P., Mikula, K.: High-resolution flux-based level set method. SIAM J. Sci. Comput. **29**, 579–597 (2007)
15. Mikula, K., Ohlberger, M.: A new level set method for motion in normal direction based on a semi-implicit forward–backward diffusion approach. SIAM J. Sci. Comput. **32**, 1527–1544 (2010)

16. Frolkovič, P., Mikula, K., Urbán, J.: Semi-implicit finite volume level set method for advective motion of interfaces in normal direction. Appl. Numer. Math. **95**, 214–228 (2015)
17. Mikula, K., Ohlberger, M., Urbán, J.: Inflow-implicit/outflow-explicit finite volume methods for solving advection equations. Appl. Numer. Math. **85**, 16–37 (2014)
18. Bertolazzi, E., Manzini, G.: A unified treatment of boundary conditions in least-square based finite-volume methods. Comput. Math. Appl. **49**, 1755–1765 (2005)
19. Tai, X.-C., Hahn, J., Chung, G.J.: A fast algorithm for Euler's elastica model using augmented Lagrangian method. SIAM J. Imaging Sci. **4**, 313–344 (2011)
20. Shu, Chi-Wang, Osher, Stanley: Efficient implementation of essentially non-oscillatory shock-capturing schemes. J. Comput. Phys. **77**, 439–471 (1988)
21. Gottlieb, Sigal, Shu, Chi-Wang: Total variation diminishing runge–kutta schemes. Math. Comput. **67**, 73–85 (1998)
22. Sussman, M., Smereka, P., Osher, S.: A level set approach for computing solutions to incompressible two-phase flow. J. Comput. Phys. **114**, 146–159 (1994)
23. Lv, X., Zou, Q., Zhao, Y., Reeve, D.: A novel coupled level set and volume of fluid method for sharp interface capturing on 3D tetrahedral grids. J. Comput. Phys. **229**, 2573–2604 (2010)
24. Ausas, R.F., Dari, E.A., Buscaglia, G.C.: A geometric mass-preserving redistancing scheme for the level set function. Int. J. Numer. Methods Fluids **65**, 989–1010 (2011)
25. Baerentzen, J.A., Aanaes, H.K.: Signed distance computation using the angle weighted pseudonormal. IEEE Trans. Vis. Comput. Gr. **11**, 243–253 (2005)