A High-Order and Interface-Preserving Discontinuous Galerkin Method for Level-Set Reinitialization

Jiaqi Zhang[a], Pengtao Yue[a]

[a]*Department of Mathematics, Virginia Tech, Blacksburg, VA 24061-0123, USA*

Abstract

A high-order numerical method for interface-preserving level-set reinitialization is presented in this paper. In the interface cells, the gradient of the level-set function is determined by a weighted local projection scheme and the missing additive constant is determined such that the position of the zero level set is preserved. In the non-interface cells, we compute the gradient of the level-set function by solving a Hamilton-Jacobi equation as a conservation law system using the discontinuous Galerkin method, following the work by Hu and Shu [SIAM J. Sci. Comput. 21 (1999) 660-690]; the missing constant is then recovered by the continuity of the level-set function while taking into account the characteristics. To handle highly distorted initial conditions, we develop a hybrid numerical flux that combines the Lax-Friedrichs flux and the penalty flux. Our method is stable for non-trivial test cases and handles singularities away from the interface very well. When derivative singularities are present on the interface, a second-derivative limiter is designed to suppress the oscillations. At least $(N+1)$th order accuracy in the interface cells and Nth order in the whole domain are observed for smooth solutions when Nth degree polynomials are used. Two dimensional test cases are presented to demonstrate superior properties such as accuracy, long-term stability, interface-preserving capability, and easy treatment of contact lines. We also show some preliminary results on the pinch-off process of a pendant drop, where topological changes of the fluid interface are involved. Our method is readily extendable to three dimensions and adaptive meshes.

Keywords: Hamilton-Jacobi equation, numerical flux, second-derivative limiter, weighted local projection, moving contact line

Email addresses: zjiaqi@vt.edu (Jiaqi Zhang), ptyue@vt.edu (Pengtao Yue)

1. Introduction

Level-set methods, introduced by Osher and Sethian in [1], are popular front capturing techniques which have been used intensively in computational physics and engineering [2, 3, 4, 5]. In a typical simulation, it is preferable or necessary that the solution be initialized to a signed distance function satisfying the Eikonal equation $|\nabla\phi| = 1$, where ϕ is the level-set function. However, it will not remain so in the process of advection, and will become too flat or too steep. To prevent this, the solution needs to be reinitialized to a signed distance function after a certain number of time steps without changing the position of the interface.

There are different types of approaches to perform reinitialization. A popular approach is to directly solve the Eikonal equation by some fast algorithms, such as the fast marching methods [6, 7] and the fast sweeping methods [8, 9]. A second approach, proposed by Sussman *et al.* [5], is to evolve the following Hamilton-Jacobi (HJ) equation to steady state:

$$\phi_t + H(\nabla\phi) = 0, \text{ in } \Omega \times [0,T] \subset \mathbb{R}^n \times \mathbb{R}, \quad \phi(\mathbf{x},0) = \phi_0, \tag{1}$$

where $H(\nabla\phi) = S(\phi_0)\left(|\nabla\phi| - 1\right)$, S is the sign function, ϕ_0 is the initial level-set function, and t is the pseudo time. For numerical stability, the discontinuous sign function is always replaced by

$$S_\eta\left(\phi_0\right) = \frac{\phi_0}{\sqrt{\phi_0^2 + \eta^2}}, \tag{2}$$

where η is the smoothing parameter usually chosen to be the computational cell size h. This PDE-based approach has been widely used in the level-set methods for interfacial flows, e.g, [10, 11, 12, 13], and will be the focus of this work. Another approach is the variational level-set method introduced by Li *et al.* [14], where ϕ is evolved by the gradient flow for an energy functional, the minimization of which leads to $|\nabla\phi| = 1$. Basting and Kuzmin [15] extended this method to the elliptic reinitialization, which is solved by a Ritz-Galerkin finite element method. Extension to a discontinuous Galerkin (DG) method was recently done by Utz et al. [16]. For more recent advances and applications of the level-set methods, the readers are referred to a recent review by Gibou *et al.* [17] and references therein.

It is well known that the zero level set tends to shift when we solve HJ equation (1) numerically. This leads to mass loss in the simulation of interfacial flows. Numerous techniques have been developed to improve mass conservation. For example, Peng *et al.* [18] modified the smooth sign

function such that the interface is confined to one cell during reinitialization. Sussman *et al.* [19, 20] introduced a Lagrange multiplier to enforce the mass conservation in each cell. Russo and Smereka [21] used a subcell fix to correct the shift of the zero level set, which was later improved and extended to higher order by Min [22] and du Chéné *et al.* [23]. Hartman *et al.* [24, 25] proposed a constrained reinitialization scheme that solves a least-squares problem to compute the level-set function in each interface cell. Sophisticated methods are also proposed to correct the level-set function by other mass conserving techniques, such as the particle level-set method [26] and the coupled level-set/volume-of-fluid method [27, 28, 29]. It should be noted that the conservative level-set method [30, 31] proposed by Olsson *et al.* takes a different approach to conserve mass and it is more like a variant of the phase-field method [32, 33]. Some recent developments of this method can be found in [34, 35].

Most high order methods for level-set reinitialization are based on structured meshes, eg., [9, 23, 36]. To achieve high order accuracy on unstructured meshes, DG methods, which have been very successful in solving conservation laws [37, 38, 39], seem to be the top choice. Although Zhang and Shu [40] and Levy *et al.* [41] have successfully constructed finite volume Weighted Essentially Non-Oscillatory (WENO) schemes for the HJ equation on unstructured meshes, DG methods still have the advantages of compactness, easy implementation, and superior scalability. Since the gradient of the HJ equation form a system of conservation laws, DG methods can be readily adapted to solve the HJ equation. Following this idea, Hu and Shu [42] designed the first DG method for the HJ equation, which was later reinterpreted and simplified by Li and Shu [43]. Later, different DG methods are proposed to directly solve the HJ equation [44, 45]. A recent review on DG methods for HJ equations can be found in [46]. In literature, the particular HJ equation (1) for level-set reinitialization was however mostly solved by the finite difference methods or the finite volume methods. Sometimes, people still stick to the more mature finite volume methods for (1) even though they use the DG methods for other equations. For example, Fechter and Munz used a fifth-order WENO scheme in the finite volume subcells of each DG grid cell [47]; Marchandise *et al.* completely avoided reinitialization and relied on a "discontinuous integration" that does not require ϕ to be a signed distance function [48]. There are only a few successful implementations of DG methods, both of which add an additional second-order diffusion term to the right hand side of (1) and use some filtering technique to stabilize the solution [49, 50].

In this paper, we develop an interface-preserving DG method for (1).

The computation cells are divided into interface cells and non-interface cells and the solution of ϕ is decomposed into $\nabla\phi$ and an additive constant. In the interface cells, we construct $\nabla\phi$ using a weighted local projection method and determine the additive constant such that the interface location is preserved. In the non-interface cells, we solve $\nabla\phi$ using the DG method of Hu and Shu [42] and then recover ϕ based on continuity. Our method is very stable and does not need additional diffusion terms or filtering techniques. For smooth ϕ with piecewise Nth degree polynomial space, we can achieve $(N+1)$th order accuracy in the interface cells and Nth order in the whole domain. An additional benefit is that our method can be directly applied to moving contact line simulations without complicated treatments on the boundary [49, 51].

The rest of this paper is organized as follows: in Section 2, we describe the algorithm to compute $\nabla\phi$ in non-interface cells, where a novel hybrid numerical flux is used. In Section 3, we present the interface-preserving reconstruction of level-set function in the interface cells. In Section 4, a second-derivative limiter is developed to stabilize the solution in the extreme case when the interface has singularities. Numerical results are illustrated in Section 5.

2. Discontinuous Galerkin method for Hamilton-Jacobi equation

In this paper, we consider (1) with the smooth sign function (2). If by any chance $|\nabla\phi_0| \ll 1$ or $\gg 1$ on the interface, $S_\eta(\phi_0)$ should be replaced by $S_\eta(\frac{\phi_0}{|\nabla\phi_0|})$ to maintain the thickness of the transition layer.

Following Hu and Shu [42], we can rewrite (1) as a conservation law system by taking the gradient:

$$\mathbf{u}_t + \nabla H(\mathbf{u}) = 0, \text{ in } \Omega \times [0,T] \subset \mathbb{R}^n \times \mathbb{R}, \quad \mathbf{u}(\mathbf{x},0) = \nabla\phi_0, \tag{3}$$

where $\mathbf{u} = \nabla\phi$. It should be noted that components of $\mathbf{u}$ are not completely independent, eg., $\nabla \times \mathbf{u} = 0$ is always satisfied.

In this work, we focus on two dimensions and quadrilateral meshes. But the results can be easily extended to three dimensions and other types of unstructured meshes. We assume that the domain Ω is well approximated by the triangulation $\mathcal{T}_h$ consisting of non-overlapping quadrilaterals with a characteristic mesh size h. As in [43], we introduce two spaces of polynomials:

$$V_h^N = \left\{ v : v \in P^N(K), \forall K \in \mathcal{T}_h \right\}, \tag{4}$$

$$W_h^N = \left\{ \mathbf{w} : \mathbf{w} = \nabla v, v \in V_h^N \right\}, \tag{5}$$

where $P^N(K)$ is the space of polynomials in K that is of degree at most N. It should be noted other polynomial spaces such as Q^N can also be used here. By approximating $\mathbf{u}$ by $\mathbf{u}_h \in W_h^N$, multiplying (3) with the test function $\mathbf{w} \in W_h^N$, and performing integration by parts, we obtain the weak formulation

$$\left(\frac{\partial \mathbf{u}_h}{\partial t}, \mathbf{w}\right)_K - \left(H\left(\mathbf{u}_h\right), \nabla \cdot \mathbf{w}\right)_K + \left(\hat{H}(\mathbf{u}_h^-, \mathbf{u}_h^+), \mathbf{w}\right)_{\partial K} = 0, \ \forall \mathbf{w} \in W_h^N, \quad (6)$$

where $(.,.)_K$ and $(.,.)_{\partial K}$ denote the inner products in the cell K and on its boundary ∂K, respectively, eg.,

$$(\mathbf{u}_h, \mathbf{w})_K = \int_K \mathbf{u}_h \cdot \mathbf{w} \mathrm{d}\mathbf{x}, \quad (7)$$

$$\left(\hat{H}(\mathbf{u}_h^-, \mathbf{u}_h^+), \mathbf{w}\right)_{\partial K} = \sum_{e \in \partial K} \int_e \hat{H}(\mathbf{u}_h^-, \mathbf{u}_h^+) \cdot \mathbf{w} ds. \quad (8)$$

$\hat{H}(\mathbf{u}_h^-, \mathbf{u}_h^+)$ is the numerical flux approximating $H(\mathbf{u}_h)\mathbf{n}$, where $\mathbf{n}$ is the unit outward pointing normal to the cell edge. $\mathbf{u}_h^-$ is the trace from the interior of cell K, while $\mathbf{u}_h^+$ from the interior of the neighboring cell. Details of $\hat{H}(\mathbf{u}_h^-, \mathbf{u}_h^+)$ is given in Sec. 2.1.

As for the finite dimensional space V_h^N, we choose the Legendre polynomial space, whose basis functions are L_2-orthogonal and normalized in the reference cell. Consequently, the first basis function v_0 is constant across the cell (equal to 1 in the reference cell). We approximate ϕ by

$$\phi_h^K = \sum_{i=0}^{m} c_i v_i, \quad (9)$$

where $v_i \in V_h^N$, $i = 0, 1, \ldots, m$, are the basis polynomials, and $m+1$ is the number of degrees of freedom. For the ease of presentation, we drop the superscript K from ϕ_h^K hereinafter. Then the approximation of $\nabla\phi$ is given by

$$\nabla \phi_h = \mathbf{u}_h = \sum_{i=0}^{m} c_i \nabla v_i. \quad (10)$$

Let $\mathbf{w}_i = \nabla v_i \in W_h^N$. Since v_0 is constant, we can further simplify (10) to

$$\nabla \phi_h = \mathbf{u}_h = \sum_{i=1}^{m} c_i \mathbf{w}_i. \quad (11)$$

Substituting (10) into (6), we can obtain

$$\mathbf{A}\frac{d\mathbf{c}}{dt} = \mathbf{F}, \tag{12}$$

where $\mathbf{A} \in \mathbb{R}^{m\times m}$ and $\mathbf{A}_{i,j} = (\mathbf{w}_i, \mathbf{w}_j)_K$, $\mathbf{c} = [c_1, \ldots, c_m]^\intercal$, and $\mathbf{F} \in \mathbb{R}^m$ with

$$\mathbf{F}_i = \left(H\left(\mathbf{u}_h\right), \nabla\cdot\mathbf{w}_{i+1}\right)_K - \left(\hat{H}\left(\mathbf{u}_h^-, \mathbf{u}_h^+\right), \mathbf{w}_{i+1}\right)_{\partial K}. \tag{13}$$

These m equations uniquely determine $\mathbf{u}_h$ in K. Thus the calculation of $\nabla\phi_h$ is completely decoupled from c_0. To recover ϕ_h, we still need c_0, and this additional degree of freedom can be used to preserve the interface, as discussed in Sec. 3.2.

2.1. The hybrid numerical flux

On a cell edge with normal $\mathbf{n}$, as shown in Fig. 1, the flux is $H(\mathbf{u}_h)\mathbf{n}$ and the Jacobian matrix is

$$J = \mathbf{n}(\nabla_{\mathbf{u}_h} H(\mathbf{u}_h))^T = S_\eta(\phi_0)\mathbf{n}\frac{\mathbf{u}_h^T}{|\mathbf{u}_h|}. \tag{14}$$

This matrix has rank one and its only non-zero eigenvalue is

$$a = S_\eta(\phi_0)\mathbf{n}\cdot\frac{\mathbf{u}_h}{|\mathbf{u}_h|}. \tag{15}$$

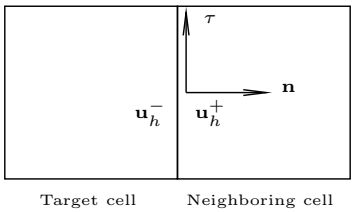


Figure 1: The edge between two cells.

Since the wind direction is readily obtained from the sign of a, the Roe flux can be easily implemented:

$$\hat{H}_{Roe}(\mathbf{u}_h^-, \mathbf{u}_h^+) = \begin{cases} H(\mathbf{u}_h^-)\mathbf{n}, & \text{if } \frac{a^- + a^+}{2} \geq 0, \\ H(\mathbf{u}_h^+)\mathbf{n}, & \text{otherwise.} \end{cases} \tag{16}$$

However it is entropy violating and generates unstable solutions according to our numerical tests. Another choice is the Lax-Friedrichs (LF) flux following [42] with slight modifications:

$$\hat{H}_{LF}(\mathbf{u}_h^-, \mathbf{u}_h^+) = H(\frac{\mathbf{u}_h^- + \mathbf{u}_h^+}{2})\mathbf{n} - \frac{\alpha}{2}[\![\mathbf{u}_h]\!], \tag{17}$$

where $\alpha = \max_{\mathbf{u}_h} |a|$ with the maximum taken over the relevant range and $[\![\mathbf{u}_h]\!] = \mathbf{u}_h^+ - \mathbf{u}_h^-$. We have tested the global LF (maximum taken over the whole computational domain), local LF (maximum taken over the two cells sharing the same edge), and the original local LF in [42]; they all work well for initial conditions that are close to a signed distance function and deliver almost identical solutions. In the rest of this work, we simply choose $\alpha = 1$, which corresponds to the global LF flux.

However, the LF flux falls short if the initial condition is far away from a signed distance function. For example, it takes extremely long time to achieve the steady state in the test case of Sec. 5.1.2. After carefully inspecting the numerical results, we find that broken ϕ_h contours are likely to occur where the cell edge is normal to the ϕ_h contours, i.e., when $\mathbf{n} \cdot \mathbf{u}_h \approx 0$. A further look at $H(\mathbf{u}_h)\mathbf{n}$ reveals that the flux on a cell edge only affects the normal component of $\mathbf{u}_h$, denoted by $\mathbf{u}_n \equiv (\mathbf{u}_h \cdot \mathbf{n})\mathbf{n}$, and has no effect on the tangential component $\mathbf{u}_\tau \equiv \mathbf{u}_h - (\mathbf{u}_h \cdot \mathbf{n})\mathbf{n}$. That is, the Riemann problem on the cell edge is only for $\mathbf{u}_n$ instead of the whole vector $\mathbf{u}_h$, and there is no mechanism to smooth out discontinuities in $\mathbf{u}_\tau$. This is probably another reason why the Roe flux (16) fails so easily. The LF flux (17) performs much better because the $[\![\mathbf{u}_h]\!]$ term has contributions in the tangential direction, which acts as a penalty term to enforce the continuity in $\mathbf{u}_\tau$.

We thus come up with the following hybrid numerical flux which combines the LF flux and the penalty flux:

$$\hat{H}(\mathbf{u}_h^-, \mathbf{u}_h^+) = H(\frac{\mathbf{u}_h^- + \mathbf{u}_h^+}{2})\mathbf{n} - \frac{\alpha}{2}[\![\mathbf{u}_n]\!] - \frac{\beta}{2}[\![\mathbf{u}_\tau]\!], \tag{18}$$

where β is the penalty parameter. If $\beta = \alpha$, (18) reduces to the LF flux (17). To deal with the severely distorted initial conditions, we adjust β according to the wind direction. Let $a^\pm = S_\eta(\phi_0)\,\mathbf{n} \cdot \frac{\mathbf{u}_h^\pm}{|\mathbf{u}_h^\pm|}$, then β is determined as in Algorithm 1. We take $\beta_{\min} = 1$ and $\beta_{\max} \geq \beta_{\min}$. That is, β takes a higher value if there is an expansion wave or the target cell is downwind. The rationale is that $\mathbf{u}_\tau^\pm$ should be changed more in the downwind cell than in the upwind cell toward the final goal $\mathbf{u}_\tau^- = \mathbf{u}_\tau^+$.

Algorithm 1 Determination of β

if $a^- \leq 0$ and $a^+ \geq 0$ **then**
  $\beta = \beta_{\max}$
else if $\frac{a^- + a^+}{2} > 0$ **then**
  $\beta = \beta_{\min}$
else
  $\beta = \beta_{\max}$
end if

It should be noted that the numerical flux (18) is no longer conservative because of the different β values at the two sides of the cell edge. But the scheme is still monotone for piecewise constant $\mathbf{u}_h$, with the modified CFL conditions for forward Euler in time:

$$\Delta t \leq \min_{\Omega} \left(\frac{h}{\alpha + \beta_{\max}} \right) \tag{19}$$

in two dimensions and

$$\Delta t \leq \min_{\Omega} \left(\frac{h}{\alpha + 2\beta_{\max}} \right) \tag{20}$$

in three dimensions.

3. Interface-preserving reconstruction of level-set function

In this section, we will develop a highly accurate approach that utilizes the structure of the DG solution space to preserve the interface.

3.1. Determination of $\nabla\phi$ in interface cells

Denote the zero level set by

$$\Gamma = \{\mathbf{x} \in \Omega : \phi_0(\mathbf{x}) = 0\}, \tag{21}$$

which is also the implicit expression of the interface. Note that ϕ and ϕ_0 share the same zero level set. Then the set of interface cells are denoted by

$$I = \{K \in \mathcal{T}_h : K \cap \Gamma \neq \emptyset\}. \tag{22}$$

Theoretically, $\mathbf{u} = \nabla\phi$ remains normal to the interface and propagates away from the interface along $S_\eta(\phi_0)\nabla\phi$ when we evolve (3). Let $\boldsymbol{\tau}$ be the

unit tangent vector to the interface Γ, then $\boldsymbol{\tau}\cdot\nabla\phi_0 = 0$ at $\mathbf{x}\in\Gamma$. Multiplying (3) by $\boldsymbol{\tau}$, we have

$$\boldsymbol{\tau}\cdot\mathbf{u}_t + \boldsymbol{\tau}\cdot\nabla H(\mathbf{u}) = 0. \tag{23}$$

Since $S(\phi_0) = 0$ for any $\mathbf{x}\in\Gamma$, Γ is also the zero level set of $H(\mathbf{u})$. Consequently, $\boldsymbol{\tau}\cdot\nabla H(\mathbf{u}) = 0$ and $\frac{\partial(\boldsymbol{\tau}\cdot\mathbf{u})}{\partial t} = \boldsymbol{\tau}\cdot\frac{\partial\mathbf{u}}{\partial t} = 0$ on Γ, where we have used the fact that Γ and thus $\boldsymbol{\tau}$ are independent of t. Thus if we start with initial condition $\mathbf{u} = \nabla\phi_0$, we should expect $\boldsymbol{\tau}\cdot\mathbf{u}(\mathbf{x},t) = \boldsymbol{\tau}\cdot\mathbf{u}(\mathbf{x},0) = 0$ for any $\mathbf{x}\in\Gamma$. However, after discretization, $\frac{\partial(\boldsymbol{\tau}\cdot\mathbf{u})}{\partial t}$ could not remain exactly zero on Γ. The errors may accumulate and eventually destroy the zero level set in long-term simulations, as shown later in Sec. 5.2. This kind of instability is common in PDE-based method. For example, it is well-known that the interface tends to shift if the HJ equation (1) is evolved without any constraints [21, 52]. Thus $\mathbf{u}$ in the interface cells has to be determined by a different approach and fixed during the pseudo time evolution.

Ideally, we want ϕ_h to be a signed distance function, i.e., $\phi_h = 0$ on the interface and $|\nabla\phi_h| = 1$ in all cells. The former implies that ϕ_h and ϕ_0 share the same unit normal on the interface. Thus $\mathbf{u}_h$ in an interface cell $K\in I$ needs to satisfy

$$\mathbf{u}_h = \frac{\nabla\phi_0}{|\nabla\phi_0|} \quad \text{on } \Gamma\cap K, \tag{24}$$

and

$$|\mathbf{u}_h| = 1 \quad \text{in } K. \tag{25}$$

Usually, these two conditions can not be satisfied simultaneously. But an optimal $\mathbf{u}_h$ in $K\in I$ can be sought as the minimizer of the energy functional

$$E(\mathbf{u}) = \frac{1}{2}\int_{\Gamma\cap K}\left(\mathbf{u} - \frac{\nabla\phi_0}{|\nabla\phi_0|}\right)^2 \mathrm{d}s + \frac{\lambda}{4}\int_K \left(|\mathbf{u}|^2 - 1\right)^2 \mathrm{d}\mathbf{x}, \tag{26}$$

where λ is a positive penalty parameter controlling the weight of the constraint $|\mathbf{u}| = 1$.

Let $\mathbf{u}_h \in W_h^N$ be the solution to the minimization problem

$$\min_{\mathbf{u}\in W_h^N} E(\mathbf{u}),$$

then $\mathbf{u}_h$ satisfies the variational form

$$\int_{\Gamma\cap K}\left(\mathbf{u}_h - \frac{\nabla\phi_0}{|\nabla\phi_0|}\right)\cdot\mathbf{w}\mathrm{d}s + \lambda\int_K \left(|\mathbf{u}_h|^2 - 1\right)\mathbf{u}_h\cdot\mathbf{w}\mathrm{d}\mathbf{x} = 0, \forall\mathbf{w}\in W_h^N. \tag{27}$$

The first term in (27) requires a surface integral on $\Gamma \cap K$, which is not known explicitly. To make this integral easy to compute numerically, we replace it by a volume integral and rewrite (27) as

$$\int_K \left(\mathbf{u}_h - \frac{\nabla \phi_0}{|\nabla \phi_0|} \right) \cdot \mathbf{w} \bar{\delta}_\epsilon(\phi_0) \mathrm{d}\mathbf{x} + \lambda \int_K \left(|\mathbf{u}_h|^2 - 1 \right) \mathbf{u}_h \cdot \mathbf{w} \mathrm{d}\mathbf{x} = 0, \forall \mathbf{w} \in W_h^N, \tag{28}$$

where $\bar{\delta}_\epsilon$ is a shifted smooth delta function defined as

$$\bar{\delta}_\epsilon(\phi_0) = \begin{cases} \frac{1}{2\epsilon} \left(1 + \cos \left(\frac{\pi}{\epsilon} \frac{\phi_0}{|\nabla \phi_0|} \right) \right) + \frac{\xi}{\epsilon}, & \text{if } \frac{|\phi_0|}{|\nabla \phi_0|} < \epsilon, \\ \frac{\xi}{\epsilon}, & \text{otherwise.} \end{cases} \tag{29}$$

Here ϵ is the half width of the narrow band and ξ is a small positive parameter to avoid singular matrices. The choices of ϵ and ξ will be discussed toward the end of this subsection. It should be noted that ϕ_0 here is rescaled by $|\nabla \phi_0|$ just to take care of the extreme cases with $|\nabla \phi_0| \ll 1$ or $\gg 1$. Since the first term in (28) essentially projects $\frac{\nabla \phi_0}{|\nabla \phi_0|}$ to a gradient space, this method is hereinafter referred to as the weighted local projection (WLP) method.

If the contours of ϕ_0 are parallel lines or concentric circles, $\mathbf{u}_h = \frac{\nabla \phi_0}{|\nabla \phi_0|}$ automatically satisfies the conditions (24) and (25). Note that ϕ_0 does not need to be a signed distance function here. In this case, the second term in (28) plays no role. But for the more general case, $\frac{\nabla \phi_0}{|\nabla \phi_0|}$ does not even reproduce the gradient of any scalar function, because we cannot guarantee that $\nabla \times \left(\frac{\nabla \phi_0}{|\nabla \phi_0|} \right) = 0$ (although $\nabla \times \nabla \phi_0 = 0$ is satisfied). Thus the second (penalty) term in (28) is necessary to maintain the signed distance function in the cell K, which will be further discussed in Sec. 5.1.2. It should be noted that the idea of local projection was first seen in [53], where the author projected $\frac{\phi_0}{|\nabla \phi_0|}$ to ϕ_h in the interface cells. This is however inaccurate unless $|\nabla \phi_0|$ is a constant.

In practice, the quality of the WLP may deteriorate if the length of the interface in cell K is very small. For example, if the interface only cuts a small portion of the cell at one corner, the solution of (28) resembles rays emanating from that corner such that the contours of ϕ_h are concentric circles. This may be totally incorrect. To improve the quality of $\mathbf{u}_h$, the neighboring cells have to be considered as well, which can be very complicated. We take another route to avoid this problem. In fact, we only need to anchor $\mathbf{u}_h$ by the WLP in some interface cells, and $\mathbf{u}_h$ in other interface cells can be maintained by the diffusive numerical flux.

We thus define another set $I_p \subset I$, which only includes cells with sufficient amount of interface:

$$I_p = \{K \in I : |K \cap \Gamma| \geq p\}, \tag{30}$$

where $|K \cap \Gamma|$ is the length of the interface segment in K, and $p < h$ is a positive number. If Γ intersects K at two points, then $|K \cap \Gamma|$ can be approximated by the distance between these two intersections. The details on locating intersections and the complicated case with more than two intersections will be discussed in Sec. 3.2. In practice, we choose $p = h/2$. The WLP method (28) is applied to every cell $K \in I_p$ and the DG method (6) is used in all other cells. Thus the WLP solution in I_p serves as a boundary condition for the DG method, and the accuracy of the WLP affects the solution in the whole computational domain.

We employ Newton's method to solve the non-linear problem (28). Denoting the solution at the kth iteration by $\mathbf{u}_h^k$, the solution at the $(k+1)$th iteration can be written as

$$\mathbf{u}_h^{k+1} = \mathbf{u}_h^k + \delta \mathbf{u}_h, \tag{31}$$

where $\delta \mathbf{u}_h$ is the increment to be determined. Substituting (31) into (28) for $\mathbf{u}_h$ and dropping the higher order terms of $\delta \mathbf{u}_h$, we obtain the linear system for $\delta \mathbf{u}_h$

$$\begin{aligned} \int_K \left[\left(\bar{\delta}_\epsilon(\phi_0) + \lambda \left(|\mathbf{u}_h^k|^2 - 1 \right) \right) \delta \mathbf{u}_h \cdot \mathbf{w} + 2\lambda \left(\mathbf{u}_h^k \cdot \delta \mathbf{u}_h \right) \left(\mathbf{u}_h^k \cdot \mathbf{w} \right) \right] \mathrm{d}\mathbf{x} \\ = - \int_K \left[\left(\mathbf{u}_h^k - \frac{\nabla \phi_0}{|\nabla \phi_0|} \right) \bar{\delta}_\epsilon(\phi_0) + \lambda \left(|\mathbf{u}_h^k|^2 - 1 \right) \mathbf{u}_h^k \right] \cdot \mathbf{w} \mathrm{d}\mathbf{x} \end{aligned} \tag{32}$$

We solve (32) for $\delta \mathbf{u}$ and update $\mathbf{u}^k$ according to (31) repeatedly, until the residual, i.e., the right hand side of (32), is smaller than a prescribed tolerance.

A good initial guess is crucial for Newton's method to succeed, otherwise it may not converge or converge to a wrong solution. An easy way to start is to solve (28) with $\lambda = 0$, which reduces to a linear system, or simply solve

$$\int_K \left(\mathbf{u}_h^0 - \frac{\nabla \phi_0}{|\nabla \phi_0|} \right) \cdot \mathbf{w} \mathrm{d}\mathbf{x} = 0, \forall\ \mathbf{w} \in W_h^N. \tag{33}$$

It should be noted that Newton's method proposed by Chopp [7] to compute the closest point may fail to converge, especially in three dimensions [36, 54]. Our method, however, always converges provided that the solution of (33)

is used as the initial guess. Typically, it takes less than five iterations for Newton's method to converge to a tolerance of 10^{-10}.

The WLP method can be summarized as follows. Firstly, identify the set I_p (30). Secondly, for each cell $K \in I_p$ solve (33) to get the initial guess $\mathbf{u}_h^0$. Finally, solve (28) by Newton's method (32).

We now discuss the choice of parameters in the WLP. To guarantee that the matrix in (32) is non-singular, sufficient quadrature points have to be included in the narrow band. Intuitively, the number of quadrature points in the support of the smooth delta function (i.e., the narrow band) must exceed the degrees of freedom in $\mathbf{u}_h$. A rule of thumb is that the distance between quadrature points should not exceed the bandwidth 2ϵ. On the one hand, limited by the number of quadrature points, ϵ can not be too small. If the $\frac{\nabla\phi_0}{|\nabla\phi_0|}$ is close to the gradient of a signed distance function, a large bandwidth of the interface, such as $\epsilon = 0.1h \sim 1h$, can produce accurate results. This is usually the case when a signed distance function is advected by the flow field only for a few time steps. But highly distorted initial conditions usually require a much smaller bandwidth, such as $\epsilon = 0.001h \sim 0.01h$, as shown in the test cases of Sec. 5.1.2. On the other hand, we use composite quadrature rules with sufficient number of quadrature points in each interface cell $K \in I_p$. In order to maintain sufficient degree of precision and also resolve the interface that almost overlaps with a cell edge, we adopt a composite Gauss-Lobatto quadrature rule. Each cell is divided into $Q \times Q$ subcells, and a $2N$-point Gauss-Lobatto quadrature is used in each cell to accurately integrate (32) with $\delta\mathbf{u}, \mathbf{u}^k \in W_h^N$. This leads to $(2NQ - Q + 1)^2$ quadrature points in total. For example, in order to resolve $\epsilon = 0.01h$ with $N = 3$, we need at least 10×10 subcells with 2601 quadrature points. Oftentimes in practice, this condition can be relaxed to save quadrature points without severely affecting the solution accuracy.

As we have mentioned before, a shift parameter ξ is introduced to further improve conditioning of the matrix. The numerical results are not very sensitive to ξ and we typically choose $\xi = 0.01\epsilon/h$. Extra attention is required on the the choice of λ if $|\nabla\phi_0|$ is highly non-uniform or if a high order DG method is used. A too small λ may end up with a ϕ_h that deviates from a signed distance function while a too big λ may cause the interface to shift. We usually choose $\lambda = 100$.

In the end, we would like to comment on the differences between the WLP method and the minimization-based elliptic reinitialization (ER) meth-

ods [15, 16]. The target functional to be minimized in ER is

$$F(\phi) = \frac{1}{2}\int_{\Omega}(|\nabla\phi| - 1)^2 d\mathbf{x} + \frac{\alpha_{ER}}{2}\int_{\Gamma}\phi^2 ds, \tag{34}$$

where α_{ER} is a parameter similar to our $1/\lambda$. First of all, ER is a global method while the WLP is a local method that is easy to parallelize. Secondly, ER directly enforces $\phi = 0$ while the WLP imposes $\nabla\phi = \frac{\nabla\phi_0}{|\nabla\phi_0|}$ on Γ. Since ϕ varies along the normal direction to the interface, the surface integral of (34) is very sensitive to the location of the interface. Sophisticated techniques such as moment-fitting [55] have to be used for accurate surface integral on Γ. On the contrary, $\nabla\phi_0$ is nearly constant along the interface normal unless ϕ_0 is heavily distorted. Thus the surface integral in the WLP (27) is not that sensitive to the interface location and we can safely replace it with a volume integral as in (28).

3.2. Determination of c_0

The previous sections only determine $c_1, c_2, \cdots, c_m$ in $\mathbf{u}_h$ as shown in (10). But we still need c_0 to recover the complete ϕ_h as shown in (9). Hu and Shu [42] suggested two ways to compute c_0: directly solving

$$\int_K \frac{\phi_h}{\partial t} + \nabla H(\phi_h) = 0, \quad \forall K \in \mathcal{T}_h \tag{35}$$

or integrating $\nabla\phi_h$

$$\phi_h(\mathbf{x}, t) = \phi_h(\mathbf{x}_0, t) + \int_{\mathbf{x}_0}^{\mathbf{x}} \nabla\phi_h \cdot d\boldsymbol{s} \tag{36}$$

along some path from $\mathbf{x}_0$ to $\mathbf{x}$. In the second method, $\phi_h(\mathbf{x}_0, t)$ needs to be computed by the first method in one or a few cells. The first method always leads to the shift of the zero level set, similar to most other PDE-based reinitialization methods. We thus adopt the second method but determine $\phi_h(\mathbf{x}_0, t)$ in an interface-preserving way.

3.2.1. Interface cells

Suppose ϕ^* is the exact solution, i.e., the signed distance function satisfying $|\nabla\phi^*| = 1$ in Ω and $\phi^* = 0$ on Γ. Our goal is to obtain ϕ_h that approximates ϕ^* as accurate as possible. Ideally, $\mathbf{u}_h$ determined from Secs. 2 and 3.1 is already a good approximation of $\nabla\phi^*$. To achieve $\phi_h \approx \phi^*$, we just need to find the appropriate c_0 in each cell K. Apparently, if

$\phi^* \in V_h^N$ and $\mathbf{u}_h = \nabla\phi^*$, then we can easily recover the exact ϕ^* by imposing $\phi_h(\mathbf{x}_0) = \phi^*(\mathbf{x}_0)$ for any point $\mathbf{x}_0 \in K$. For non-interface cells, this method is not practical because ϕ^* is unknown. But for interface cells, we can choose $\mathbf{x}_0$ to be some convenient point on the interface, e.g. the intersections between the interface and the cell edges.

Numerically, we identify the interface cells by checking the ϕ_0 values at the vertices: a cell is a non-interface cell if ϕ_0 at its four vertices (eight vertices for hexagonal cells in 3D) are all positive or all negative, otherwise it is an interface cell. This criterion may miss some interface cells, such as Fig. 2 (d) and (e). But in those cases, either the interface can be taken care of by the neighboring cells or the interface curvature is too high ($> \frac{1}{h}$). For high curvatures, the DG polynomial space could not accurately recover the interface anyway, and mesh refinement is usually the only way to go, which is beyond the scope of this work.

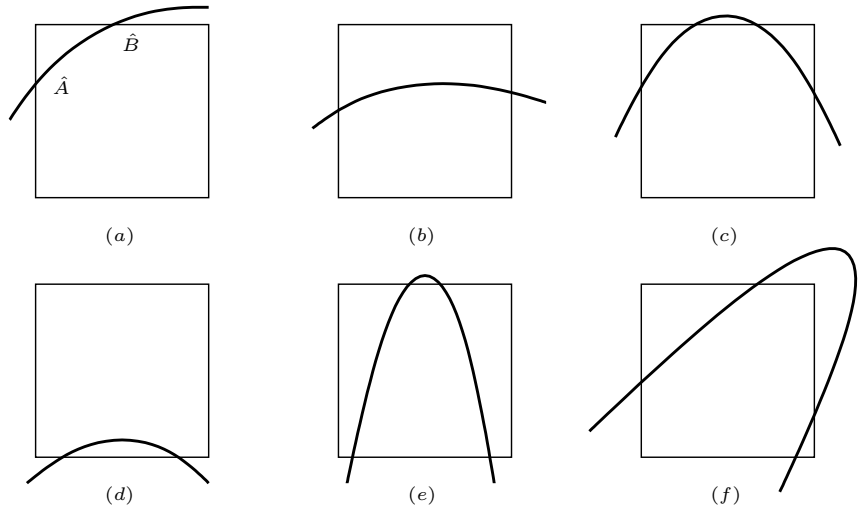


Figure 2: Some typical configurations of interface cells in the reference frame. (a) and (b) are the regular interface cells that can be easily identified. Only two intersections of (c) can be detected and it is treated in the same way as (b). (d) and (e) are not numerically detected as in interface cells. (f) is detected but the interface curvature is too high. Numerically, (d), (e), and (f) are not included in set I. Other cases, including the extremely rare cases of interface passing through one or more vertices, are not included here but considered in our code.

For simplicity, we only discuss the cases with two detected intersections, denoted by $\hat{A}$ and $\hat{B}$ in Fig. 2. We consider the problem in the reference

frame, where the reference cell is the unit square $[0,1]\times[0,1]$. A point (x,y) in the physical cell is mapped to $(\hat{x},\hat{y})$ in the reference cell with $\phi_0(x,y) = \hat{\phi}_0(\hat{x},\hat{y})$, where we have used $\hat{}$ to denote the quantities in reference frame. The exact locations of $\hat{A}$ and $\hat{B}$ can be obtained from $\hat{\phi}_0(\hat{A}) = \hat{\phi}_0(\hat{B}) = 0$. For example, the location of $\hat{A}$ in Fig. 2 (a) can be obtained by solving $\hat{\phi}_0(0,\hat{y}_A) = 0$ using the secant method with starting values 0 and 1 (two ends of the edge). If the iteration diverges or converges to a solution outside $[0,1]$, we switch to the root-bracketing false position method. Then the coordinates of A can be recovered from $\hat{A}$. Since we have two intersections but only one unknown, c_0 can be determined by solving the least squares problem

$$\min_{c_0} \left\| \begin{bmatrix} \phi_h(A) - \phi_0(A) \\ \phi_h(B) - \phi_0(B) \end{bmatrix} \right\|_2 = 0. \tag{37}$$

This problem has a simple solution

$$c_0 = -\frac{\phi_h(A)|_{c_0=0} + \phi_h(B)|_{c_0=0}}{2}, \tag{38}$$

where $\phi_h(\cdot)|_{c_0=0}$ denotes $\phi_h(\cdot)$ evaluated with $c_0 = 0$. To preserve the interface, we perform this procedure on every cell $K \in I$. This method can be easily extended to 3D where we can still use intersections between the interface and the edges (instead of faces of the 3D cell) to determine c_0 .

3.2.2. Non-interface cells

In each cell K, from $\mathbf{u}_h$, we can easily recover $\phi_h(\mathbf{x})|_{c_0=0}$, which differs from the complete $\phi_h(\mathbf{x})$ only by an additive constant c_0. Meanwhile, (36) also implies that $\phi_h(\mathbf{x})$ is continuous across the cells. This leads to a more efficient method than directly integrating (36).

Consider two neighboring cells K_A and K_B. Suppose c_0 in K_A, denoted by $c_{0,A}$, is already known. Then $c_{0,B}$ in K_B can be determined by the continuity across their shared edge. Since we only need one constraint here, we can use the continuity at any point on the shared edge. In this work, we use the edge center E, and $c_{0,B}$ can be determined from

$$c_{0,B} = \phi_{h,A}(E) - \phi_{h,B}(E)|_{c_{0,B}=0}, \tag{39}$$

which is equivalent to (36) with an integration path starting from any point in K_A, ending at any point in K_B, and passing through E. This procedure can be repeated until c_0 is propagated from the interface cells to all non-interface cells. A simple way is to update c_0 based on mesh connectivity: we

first update the direct neighbors of interface cells, then update the neighbors of neighbors, and so on.

Theoretically, the line integral in (36) is path independent. However, this is not the case numerically, especially when the path goes through singularities formed by intersecting characteristics (eg., the center of a circular interface). It makes more sense to update c_0 following the characteristics rather than taking an arbitrary path. For example, we use K_A to update K_B only when the characteristic direction $S_\eta(\phi_0)\nabla\phi$ points from K_A to K_B, i.e., K_A is the upwind cell and K_B is the downwind cell. To make sure all information is taken from the zero level set along the characteristics, we borrow some ideas from the Fast Marching method [6] and the Fast Sweeping method [8], and come up with Algorithm 2. In this algorithm we have used the fact that c_0 is a approximately the cell average of ϕ_h (exact average in the reference cell), and solution propagates from a cell with lower $|c_0|$ to a cell with higher $|c_0|$. The additional condition on $S(\phi_0)\nabla\phi_h \cdot \mathbf{n}$ is just to double confirm that the neighbor is the upwind cell. If the target cell has multiple upwind neighbors, then we take the c_0 with the smallest magnitude. If the target cell does not have any upwind neighbors, which rarely happens in real calculations, we simple keep the c_0 determined from mesh connectivity.

4. Slope limiter for $\nabla\phi_h$

The DG method in Sec. 2 can handle discontinuities in $\nabla\phi_h$ quite well in most cases. However, if the interface has singularities, such as sharp corners on a square interface as shown in Sec. 5.3, numerical oscillations may develop. In this case the limiters are necessary. Since the limiters are rarely used, we simply follow the method by Cockburn and Shu [56]. For better accuracy, the readers are referred to the WENO limiters [57, 58, 59].

The construction of the slope limiter is based on $\mathbf{u}_h$ at vertices, since it is easier to extend to adaptive quadrilateral meshes. Consider two-dimensional quadrilaterals as shown in Fig. 3 where C_i, $i = 0, \ldots, 4$, denote the barycenter of the quadrilateral K_i, and V_i, $i = 1, \ldots, 4$, the vertex of cell K_0. Observe that

$$V_1 - C_0 = \alpha_1 (C_1 - C_0) + \alpha_2 (C_2 - C_0) \tag{40}$$

for some nonnegative coefficients α_1 and α_2 that can be obtained from geometric positions of barycenters and vertices. Then we can compute the difference of $\mathbf{u}_h$ between vertex V_1 and cell center C_0 based on the cell averages:

$$\mathbf{g}_1 = \alpha_1 (\bar{\mathbf{u}}_{h,1} - \bar{\mathbf{u}}_{h,0}) + \alpha_2 (\bar{\mathbf{u}}_{h,2} - \bar{\mathbf{u}}_{h,0}), \tag{41}$$

Algorithm 2 Upwind update of c_0 in non-interface cells.

1 Compute an approximate c_0 in all non-interface cells based on mesh connectivity.
2 Sort all non-interface cells in an increasing order of $|c_0|$.
for all sorted non-interface cells **do**
  Set $c = \text{Inf}$.
  for all edges of the target cell **do**
    Identify the neighboring cell sharing the same edge.
    if $S(\phi_0)\nabla\phi_h \cdot \mathbf{n} < 0$ AND $|c_0^{target}| > |c_0^{neighbor}|$ **then**
      Compute c_0^{new} in the target cell based on the current neighbor
      if $|c_0^{new}| > |c_0^{neighbor}|$ **then**
        Update $c = \min(c, |c_0^{new}|)$.
      end if
    end if
  end for
  if $c \neq \text{Inf}$ **then**
    Set $c_0^{target} = \text{sign}(c_0^{target})c$ in the target cell.
  end if
end for

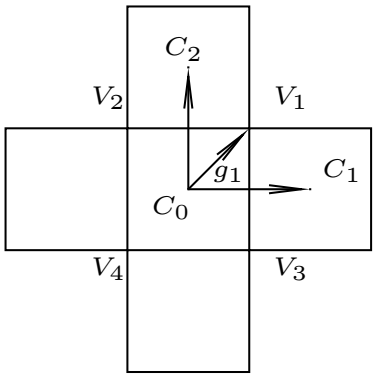


Figure 3: Limiters in a quadrilateral mesh.

where

$$\bar{\mathbf{u}}_{h,i} = \frac{1}{|K_i|}\int_{K_i} \mathbf{u}_h \mathrm{d}\mathbf{x},\ i = 0, \ldots, 4. \tag{42}$$

Here $|K_i|$ denotes the area of cell K_i. Similarly, we can compute $\mathbf{g}_i,\ i = 2, 3, 4$.

Let

$$\Delta_i = \text{minmod}\left(\mathbf{u}_h(V_i) - \bar{\mathbf{u}}_{h,0}, \nu \mathbf{g}_i\right), \tag{43}$$

where the minmod function is defined as

$$\text{minmod}(a,b) = \begin{cases} s\min(|a|,|b|), & \text{if } s = \text{sign}(a) = \text{sign}(b), \\ 0, & \text{otherwise}, \end{cases} \tag{44}$$

and ν is a constant chosen from interval $[1,2]$. If a and b are vectors, then the minmod function is applied component-wise. When $\Delta_i = \mathbf{u}_h(V_i) - \bar{\mathbf{u}}_{h,0}$ for all $i = 1,\ldots,4$, limiting is not necessary in cell K_0. Otherwise, $\mathbf{u}_h = \nabla\phi_h$ with $\phi_h \in P^N(K_0)$ is limited to $\tilde{\mathbf{u}} = \nabla\tilde{\phi}$ with

$$\tilde{\phi} = \tilde{c}_0 + \tilde{c}_1\psi_1(x) + \tilde{c}_2\psi_1(y) + \tilde{c}_3\psi_1(x)\psi_1(y) + \tilde{c}_4\psi_2(x) + \tilde{c}_5\psi_2(y) \in P^2(K_0), \tag{45}$$

where ψ_1 and ψ_2 are the first and second Legendre polynomials. Note that we are only concerned about $\nabla\tilde{\phi}$ here, so $\tilde{c}_0$ does not matter and we only need to find $\tilde{c}_1, \tilde{c}_2, \cdots, \tilde{c}_5$.

If limiting is required, we compute $\tilde{\mathbf{u}}$ that approximates the limited directional derivatives along the two diagonals of K_0. This can be done by solving the least squares problem:

$$\min_{\tilde{c}_3,\tilde{c}_4,\tilde{c}_5} \left\| \begin{bmatrix} (V_4 - V_1)\cdot\nabla\tilde{\mathbf{u}} \\ (V_3 - V_2)\cdot\nabla\tilde{\mathbf{u}} \end{bmatrix} - \begin{bmatrix} 2\text{minmod}\left(\Delta_4, -\Delta_1\right) \\ 2\text{minmod}\left(\Delta_3, -\Delta_2\right) \end{bmatrix} \right\|_2, \tag{46}$$

where $\nabla\tilde{\mathbf{u}} = \begin{bmatrix} 2\tilde{c}_4 & \tilde{c}_3 \\ \tilde{c}_3 & 2\tilde{c}_5 \end{bmatrix}$. In addition, the conservation of $\mathbf{u}$ requires

$$\int_{K_0} \tilde{\mathbf{u}} \mathrm{d}\mathbf{x} = \int_{K_0} \mathbf{u}_h \mathrm{d}\mathbf{x}. \tag{47}$$

For a 2D quadrilateral cell, (46) supplies three constraints and (47) supplies two constraints, which uniquely determines the five coefficients in $\tilde{\mathbf{u}}$ (gradient of a quadratic polynomial). In 3D, $\tilde{\mathbf{u}}$ has nine coefficients ($\tilde{\phi} \in P^2(K_0)$ has ten coefficients) which can be determined from the six constraints in (46) and three in (47). Once $\tilde{\mathbf{u}}$ is obtained, we directly enforce $\mathbf{u}_h = \tilde{\mathbf{u}}$ in K_0.

To avoid overly limiting, we only apply the procedure above when $-S(\phi_0)\nabla^2\phi > \frac{1}{h}$ in K_0, i.e., if the characteristics are converging and the curvature of ϕ contours is very high.

5. Numerical examples

In this section, we show the accuracy, long-term stability, and interface preserving properties of our algorithm. Unless otherwise stated, we choose the following parameters: $N = 3$, $\beta_{\max} = 3$, $\lambda = 100$, $\epsilon = c_\epsilon h$ with $c_\epsilon = 0.1$, $\xi = c_\xi c_\epsilon$ with $c_\xi = 0.01$, and $Q = 5$. The computational domain is a square with 64×64 uniform cells.

Discontinuous Galerkin methods are usually combined with the total variation diminishing (TVD) Runge-Kutta (RK) methods [60] to achieve high orders in both space and time. For convection dominated problems, when a DG method with polynomial degree N is paired with an $(N+1)$th-order TVD RK method, stability requires $CFL \leq \frac{1}{2N+1}$ [61]. For other RK and DG combinations, stability condition has to be established based on numerical tests. For example, the third-order TVD RK is stable for DG polynomial degrees with the following maximum CFL numbers: 0.130 for $N = 3$, 0.089 for $N = 4$, 0.066 for $N = 5$, *etc.* More details can be found in [61] and references therein. In our simulations, we adopt the third-oder TVD RK for time integration and choose $\Delta t = 0.1h$ for $N = 3$, such that $CFL = 0.1$.

The numerical algorithm is summarized as follows:

(i) Prepare initial condition. If ϕ_0 is not given in the DG solution space, project it to $\phi_{h,0} \in V_h^N$ and start the simulation with $\phi_{h,0}$.
(ii) Use the WLP method to determine $\nabla\phi_h$ in I_p (Sec. 3.1).
(iii) Solve the HJ equation using DG for $\nabla\phi_h$ in $\mathcal{T}_h/I_p$ until convergence (Sec. 2). Apply the second-derivative limiter if necessary (Sec. 4).
(iv) Determine c_0 in I and propagate it to all cells (Sec. 3.2). Now we get the complete solution ϕ_h in the computational domain.

The code is developed based on the deal.II finite element library [62, 63].

5.1. Convergence tests

We consider a circular interface in a square domain $\Omega = [-2, 2]^2$. The the initial condition is given by

$$\phi_0 = g(\mathbf{x})\left(\sqrt{x^2 + y^2} - r\right), \tag{48}$$

where $r = 0.9$. The exact solution satisfying (1) is $\phi^* = \sqrt{x^2 + y^2} - r$. We test two different initial conditions with $g(\mathbf{x}) = g_1(\mathbf{x}) = 0.8$ and $g(\mathbf{x}) = g_2(\mathbf{x}) = 0.1 + (x - r)^2 + (y - r)^2$. The former has a uniform $|\nabla\phi_0|$ and $\frac{\nabla\phi_0}{|\nabla\phi_0|}$

directly yields the exact solution $\nabla\phi^*$. While the latter, taken from [24], results in very non-uniform $|\nabla\phi_0|$.

To investigate the accuracy of the WLP method, we define the following errors on the interface. In order to measure the displacement of the interface from its initial position, we introduce

$$E_I = \frac{1}{L}\int_{K\in I} |H_\epsilon(\phi_h) - H_\epsilon(\phi^*)|\,\mathrm{d}\mathbf{x}, \tag{49}$$

where L is the length of the interface, ϕ^* is the exact solution, and H_ϵ is a smooth Heaviside function defined as

$$H_\epsilon(\phi) = \begin{cases} 0, & \text{if } \phi < -\epsilon, \\ 1, & \text{if } \phi > \epsilon, \\ \frac{1}{2}\left(1 + \frac{\phi}{\epsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\phi}{\epsilon}\right)\right), & \text{otherwise.} \end{cases} \tag{50}$$

We also define the averaged L_2 error

$$E_2 = \sqrt{\frac{1}{n_p}\sum_{K\in I_p}\frac{1}{|K|}\int_K (\phi_h - \phi^*)^2\,\mathrm{d}\mathbf{x}} \tag{51}$$

and the L_∞ error

$$E_\infty = \max_{K\in I_p} |\phi_h - \phi^*| \tag{52}$$

to measure the error in the interface cells. Here n_p is the number of cells in I_p and $|K|$ is the area of cell K.

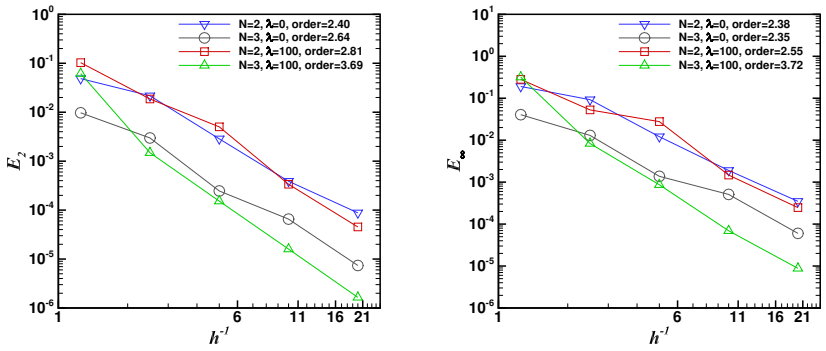
To investigate the accuracy of the whole method in the computational domain, we define the L_2 error

$$e_2 = \sqrt{\sum_{K\in\Omega_0} (\phi_h - \phi^*)^2}, \tag{53}$$

and the L_∞ error

$$e_\infty = \max_{K\in\Omega_0} |\phi_h - \phi^*|. \tag{54}$$

Here we take $\Omega_0 = [-2,2]^2 \setminus [-0.4,0.4]^2$ to exclude the center of the circular interface, which is a singular point. The cell size of the uniform grids are $h = \frac{0.8}{2^l}$, where $l = 0, 1, \cdots, 4$ is the level of refinement. To guarantee that the steady state is achieved, we compare $\nabla\phi_h$ every 200 pseudo time steps. If the L_2-norm of the difference in Ω_0 is below 10^{-15}, then we stop the computation; otherwise, the problem is computed to the pseudo time $t = 15$.



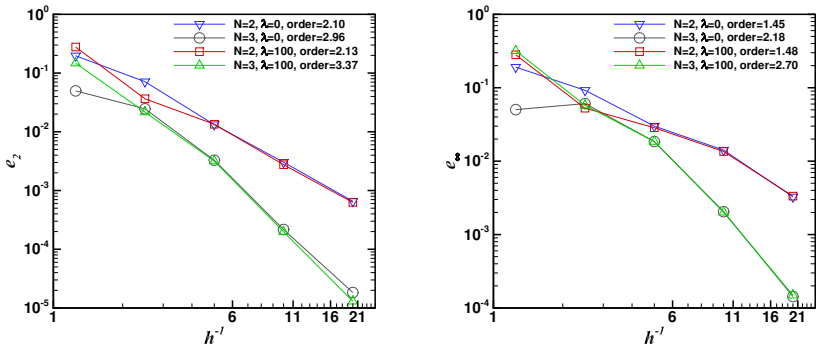
(a) E_2 of $(\phi_h)_x$ (b) E_∞ of $(\phi_h)_x$

Figure 4: Convergence of the WLP method for the circular interface with $g = g_1$. The errors are evaluated in the interface cells I_p. $c_\epsilon = 0.1$ and $Q = 5$.

5.1.1. Convergence tests for $g(\mathbf{x}) = g_1(\mathbf{x})$

For $g_1(\mathbf{x}) = 0.8$, $\frac{\nabla\phi_0}{|\nabla\phi_0|}$ is uniform and we do not need a small ϵ in the WLP. We fix $c_\epsilon = 0.1$ and $Q = 5$, and test $\lambda = 0$ and 100. Owing to the symmetry in $\nabla\phi_h$, we only compute the errors of $(\phi_h)_x$ (derivative with respect to x), which are given in Fig. 4. The orders in the legends, in this figure and all the following figures, are computed based on a power fitting without considering the data points from the coarsest mesh. For polynomial degree N (roughly degree $N-1$ for $\nabla\phi_h$) we can achieve at least order N for $\nabla\phi_h$. The only exception is the case with $N = 3$ and $\lambda = 0$. Since $\left|\frac{\nabla\phi_0}{|\nabla\phi_0|}\right| = 1$ is automatically satisfied, theoretically, the value of λ in (28) should not matter. But after we project ϕ to $\phi_{h,0} \in V_h^N$, $\left|\frac{\nabla\phi_0}{|\nabla\phi_0|}\right|$ is no longer one. Thus the numerical results are still dependent on λ. For $N = 3$, $\lambda = 100$ gives better results than $\lambda = 0$.

The solution from the WLP is then fed to the Hamilton-Jacobi equation as "boundary conditions". For both $N = 2$ and 3, the steady state is achieved within $t = 6$. The errors of $\nabla\phi_h$ in the region with the central singularity removed are shown in Fig. 5. In L_2-norm, the convergence orders of $\nabla\phi_h$ as shown in Fig. 5(a) are slightly lower than those in the WLP, but are still essentially of order N. This is expected from the $(N-1)$th degree polynomials for $\nabla\phi_h$. In L_∞-norm, the convergence order drops to around $N - \frac{1}{2}$, probably because of some individual cells that are poorly resolved. Note that e_∞ of $(\phi_h)_x$ even increases at the first two data points of the



(a) e_2 of $(\phi_h)_x$ (b) e_∞ of $(\phi_h)_x$

Figure 5: Convergence of the DG method for the circular interface with $g = g_1$. The errors are evaluated in $\Omega_0 = [-2, 2]^2/[-0.4, 0.4]^2$.

$N = 3$ and $\lambda = 0$ curve in Fig. 5(b). This is because in the coarsest 5×5 mesh, most cells in Ω_0 are interface cells which do not participate in the DG computation. This kind of abnormal behavior occurs frequently for errors evaluated in Ω_0, which is also the reason why we exclude the first data point in computing the convergence orders.

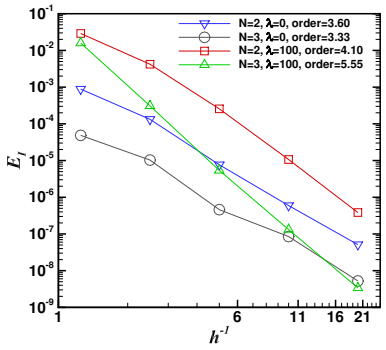
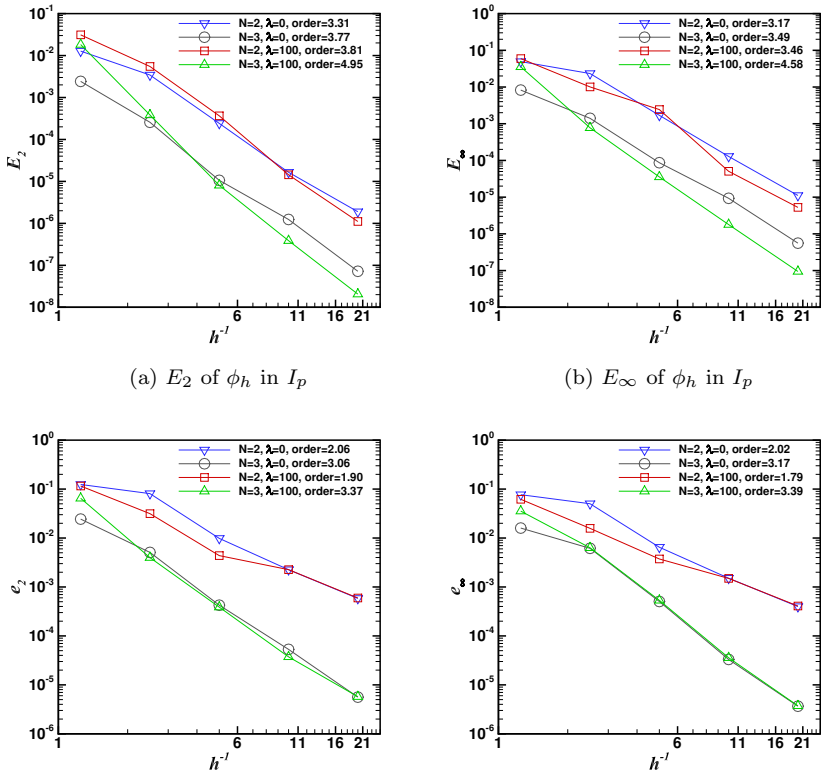


Figure 6: Interface displacement, measured by E_I in the interface cells I, for the circular interface with $g = g_1$.

The c_0's in all interface cells are directly computed by (38) and the displacement of the zero level set is given in Fig. 6. For $\lambda = 100$, $|\nabla \phi_h| = 1$ is enforced in the whole interface cell at the cost of the accuracy on the

interface; thus the values of E_I with $\lambda = 100$ are higher than those with $\lambda = 0$ in most situations. However, the error with $\lambda = 100$ quickly catches up as mesh refines and the convergence order is at least $N + 2$. For all λ and N values, the convergence orders of E_I are above N, which is sufficient for the overall Nth order convergence for ϕ_h to be discussed later.



(a) E_2 of ϕ_h in I_p

(b) E_∞ of ϕ_h in I_p

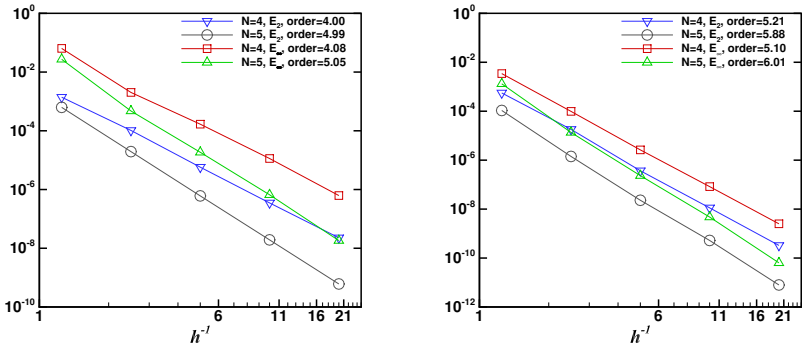
(c) e_2 of ϕ_h in Ω_0

(d) e_∞ of ϕ_h in Ω_0

Figure 7: Errors of ϕ_h for the circular interface with $g = g_1$.

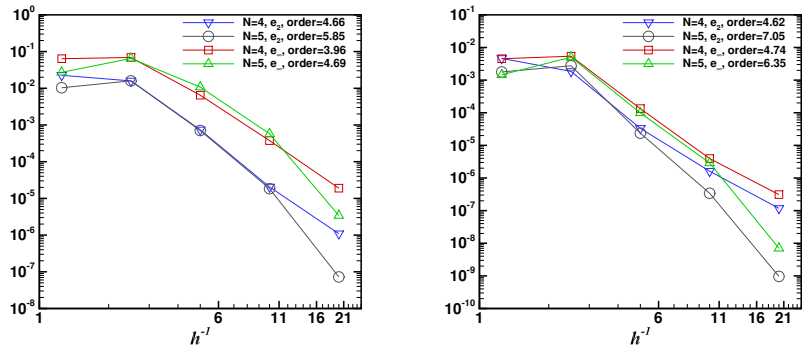
The errors of ϕ_h are given in Fig. 7. Since ϕ_h is essentially the line integral of $\nabla\phi_h$, the order of convergence should be one order higher than $\nabla\phi_h$ if the path length is $O(h)$ and of the same order as $\nabla\phi_h$ if the path length is $O(1)$. This is confirmed by the convergence order $N + 1$ in I_p and the convergence order N in Ω_0.

For higher polynomial degree N, our method can still maintain the optimal convergence orders: order N for $\nabla\phi_h$, order $N + 1$ for ϕ_h in interface



(a) Errors of $(\phi_h)_x$ in I_p

(b) Errors of ϕ_h in I_p



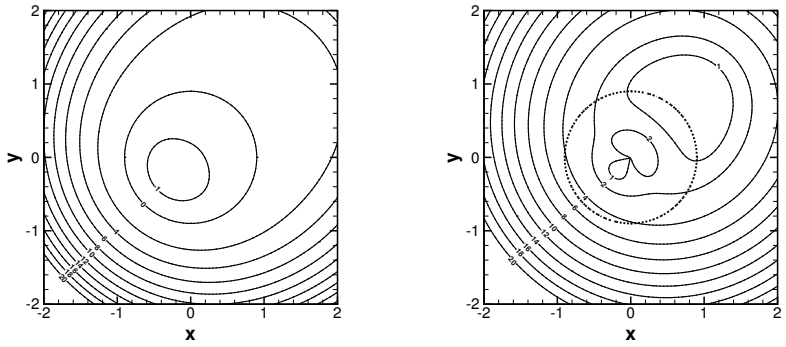
(c) Errors of $(\phi_h)_x$ in Ω_0

(d) Errors of ϕ_h in Ω_0

Figure 8: Convergence tests of higher order methods for circular interface with $g = g_1$. $\lambda = 0$, $Q = 5$, $c_\epsilon = 0.1$.

cells, and (at least) order N for ϕ_h in the whole domain. The results for $N = 4$ and 5 are presented in Fig. 8. To reduce the sources of error, the exact initial condition (55) is directly used without projecting onto V_h^N in the WLP of these tests.

It should be noted that the closest point algorithm by Saye [36] can also achieve arbitrarily high order. A key component of this algorithm is the least squares polynomial approximation of ϕ_0 around the interface, which always leads to very wide stencils in the finite difference framework. In principle, this algorithm can be adapted to DG and become another option for high order accuracy on unstructured meshes.

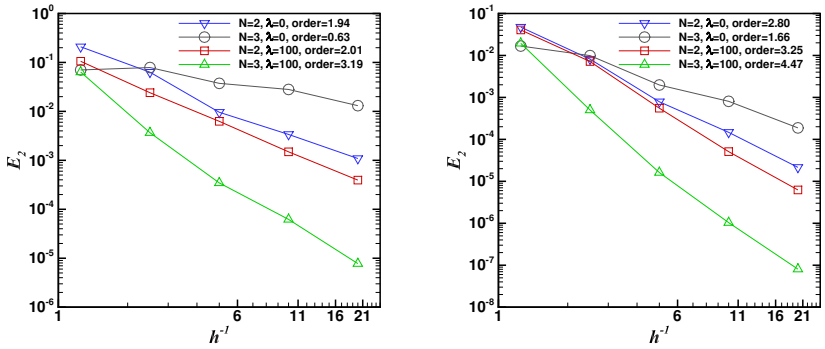


(a) ϕ_0 (b) $|\nabla\phi_0|$

Figure 9: The initial condition for circular interface with $g(\mathbf{x}) = g_2(\mathbf{x})$.

5.1.2. Convergence tests for $g(\mathbf{x}) = g_2(\mathbf{x})$

When the signed distance function is disturbed by $g_2(\mathbf{x})$, $\frac{\nabla\phi_0}{|\nabla\phi_0|}$ is far from the exact solution $\nabla\phi^*$ except on the interface. The initial condition, as shown in Fig. 9, exhibits highly non-uniform $\nabla\phi_0$, and $|\nabla\phi_0|$ ranges from 0.239 to 4.82 in interface cells. Thus extra care has to be taken in the choice of parameters such as the penalty parameter λ in the WLP, the bandwidth parameter $c_\epsilon = \epsilon/h$ of the smooth delta function, and β_{max} in the numerical flux.

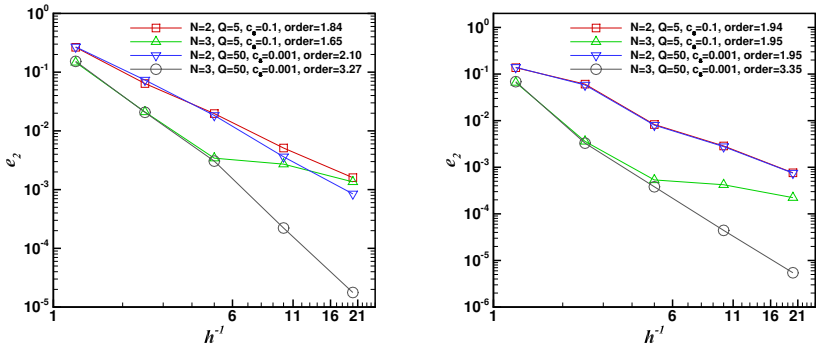


(a) E_2 of $(\phi_h)_x$ (b) E_2 of ϕ_h

Figure 10: Effect of λ for circular interface with $g = g_2$. The errors are evaluated in the interface cells I_p. $c_\epsilon = 0.001$, $Q = 50$, and $\beta_{max} = 3$.

We first investigate the effect of λ in the WLP. To make sure that the

surface integral on the interface is accurately evaluated, we use a very small bandwidth with $c_\epsilon = 0.001$ and each cell is divided into 50×50 subcells ($Q = 50$) in numerical integration. We test $\lambda = 0$ and 100 for polynomial degrees $N = 2$ and 3. The errors in the interface cells are given in Fig. 10. When $\lambda = 100$, the optimal convergence orders are achieved for both polynomial degrees: Nth order for $\nabla\phi_h$ and $(N+1)$th order for ϕ_h. $\lambda = 100$ delivers better results than $\lambda = 0$, especially for $N = 3$. This is due to the fact that the Q^3 polynomial space has more degrees of freedom to better satisfy $\nabla\phi_h = \frac{\nabla\phi_0}{|\nabla\phi_0|}$ on the interface, but at the cost of violating $|\nabla\phi_h| = 1$ in the interface cells to a higher degree. Thus it is necessary to impose the second term in (28) to enforce $|\nabla\phi_h| = 1$ if ϕ_0 is far from a signed distance function, especially for a high polynomial degree N.



(a) e_2 of $(\phi_h)_x$ (b) e_2 of ϕ_h

Figure 11: Effect of ϵ for the circular interface with $g = g_2$. The errors are evaluated in $\Omega_0 = [-2, 2]^2 \setminus [-0.4, 0.4]^2$. $\lambda = 100$, $\beta_{\max} = 3$.

We next investigate the influence of the bandwidth ϵ in the smooth delta function. We test $c_\epsilon = 0.1$ and compare with $c_\epsilon = 0.001$, as shown in Fig. 11. To avoid repeating the $\epsilon = 0.001$ curves in Fig. 10, we use the errors evaluated in Ω_0 here. The errors in the interface cells lead to the same conclusion. When $c_\epsilon = 0.001$, the surface integral on the interface is well approximated and optimal convergence order is achieved: order N for both ϕ_h and $\nabla\phi_h$ in Ω_0. For $N = 2$, $c_\epsilon = 0.1$ yields almost the same errors as $c_\epsilon = 0.001$. This is because $\nabla\phi_h$ (and also $\frac{\nabla\phi_{h,0}}{|\nabla\phi_{h,0}|}$ after projection onto V_h^N) is roughly linear at $N = 2$. In this case, integration of $\nabla\phi_h$ in the narrow band is only dependent on the values at the center of the narrow band, i.e., the interface; thus the results for $N = 2$ are insensitive to c_ϵ. For $N = 3$, $\nabla\phi_h$ is no longer linear and $c_\epsilon = 0.1$ produces much larger errors

than $c_\epsilon = 0.001$.

We finally investigate the influence of $\beta_{\max}$ in the numerical flux. The convergence results for $\beta_{\max} = 1$, 2, and 3 are given in Fig. 12. When $\beta_{\max} = 1$ the solution does not converge at pseudo time $t = 15$ as we refine the mesh, while $\beta_{\max} = 2$ and 3 provides similar convergence order. The pseudo time to reach the steady state for $\beta_{\max} = 2$ and 3 are $t = 14.55$ and 7.24, respectively. After we apply the WLP, there is a big jump between cells in I_p and their neighboring cells, which would generate oscillations. With larger β_{max}, these oscillations can be dissipated away more quickly. In the test, we set the maximum pseudo time to $t = 15$; therefore it is possible that, given enough time, $\beta_{\max} = 1$ can also generate the desired convergence order. Figure 13 shows the contours of $(\phi_h)_x$. It is obvious that $\beta_{\max} = 3$ almost reproduces the exact solution, $\beta_{\max} = 2$ still has oscillations at the center, and $\beta_{\max} = 1$ exhibits the most severe oscillations. We further present the contour plots of ϕ_h for $\beta_{\max} = 2$ and 3 in Fig. 14, where we can see that $\beta_{\max} = 3$ reproduces the exact solution at the center at an earlier stage. It should be noted that it takes longer than $t = 1$ for the ϕ_h contours to reach steady state because we use the smooth sign function (2), which causes the speed of propagation along the characteristics to be smaller than one near the interface.

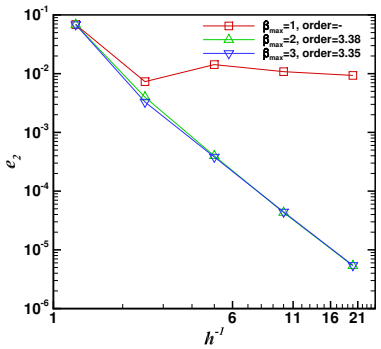
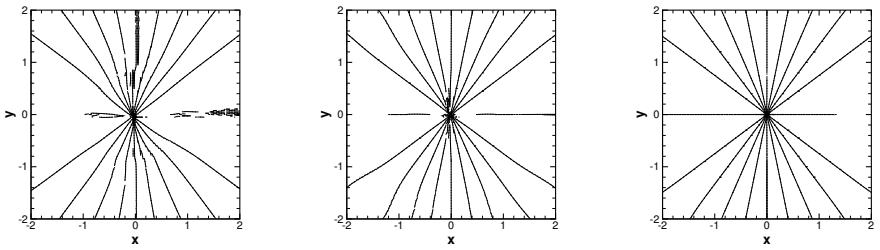


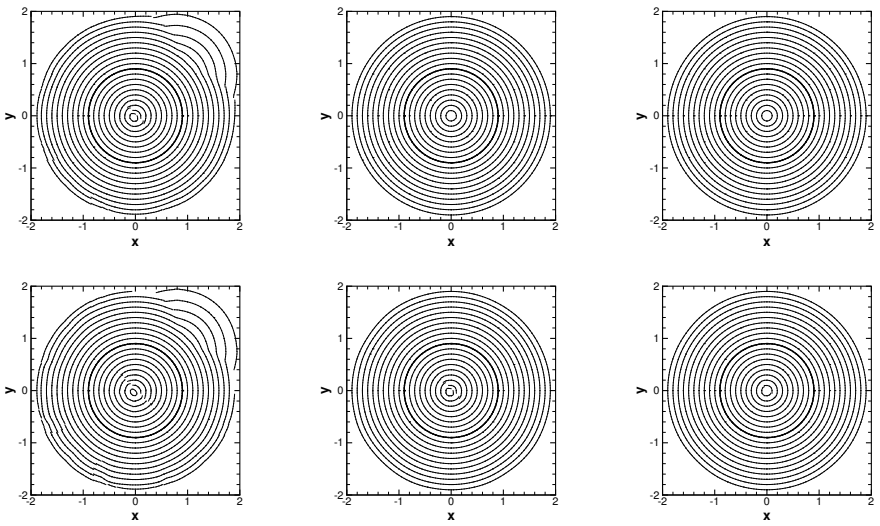
Figure 12: Effect of $\beta_{\max}$ for the circular interface with $g = g_2$. The errors are e_2 of ϕ_h evaluated in $\Omega_0 = [-2, 2]^2/[-0.4, 0.4]^2$. $N = 3$, $\lambda = 100$, $c_\epsilon = 0.001$, and $Q = 50$.

red



(a) $\beta_{\max} = 1$ (b) $\beta_{\max} = 2$ (c) $\beta_{\max} = 3$

Figure 13: $(\phi_h)_x$ at $t = 3$ for $g = g_2$ using different $\beta_{\max}$. Contours run from -1 to 1 with interval 0.2. $N = 3$, $\lambda = 100$, $c_\epsilon = 0.001$, $Q = 50$, and $h = 0.05$.



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$

Figure 14: Evolution of ϕ_h for $g = g_2$ using different $\beta_{\max}$. $\beta_{\max} = 3$ for the top row and $\beta_{\max} = 2$ for the bottom row. Contours run from -0.8 (center) to 1 with interval 0.1. The thick line denotes the interface. $N = 3$, $\lambda = 100$, $c_\epsilon = 0.001$, $Q = 50$, and $h = 0.05$.

5.1.3. Reinitialization in three dimensions

Our method can be easily extended to 3D and the same set of parameters still work. It should be noted that the deal.II library allows for dimension-independent programming by including the number of dimensions as a template parameter. Thus, in terms of coding, there is almost no difference between 2D and 3D.

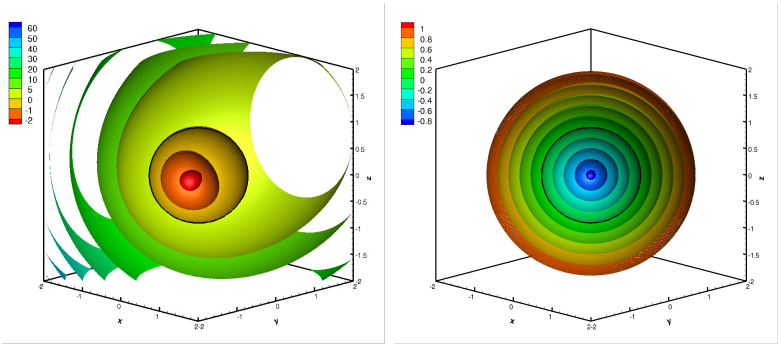
We consider a spherical interface in $\Omega = [-2, 2]^3$ with the initial condition

$$\phi_0 = g_3(\mathbf{x}) \left(\sqrt{x^2 + y^2 + z^2} - r \right), \tag{55}$$

where

$$g_3(\mathbf{x}) = 0.1 + (x - r)^2 + (y - r)^2 + (z - r)^2 .$$

Figure 15 shows the 3D results using the same parameters as in 2D computations. There is only one exception: we may need a smaller pseudo time step in 3D because the stability requirement (20) in 3D more restrictive than Eq. (19) in 2D. The results in Fig. 15 are obtained using $\Delta t = 0.05h$. According to our tests, $\Delta t = 0.1h$ still works for $\beta_{\max} \leq 2$.



(a) $t = 0$ (b) $t = 1.5$

Figure 15: Reinitialization in 3D. $N = 3$, $\beta_{\max} = 3$, $\lambda = 100$, $c_\epsilon = 0.1$, $c_\xi = 0.01$, and $Q = 5$. The mesh size is $h = 1/16$, which corresponds to 64^3 cells. The thick line denotes the interface.

5.2. Elliptic interface

In this example, we would like to show the long-term stability of our method. We reinitialize an elliptic interface in a unit square. The initial condition is given by

$$\phi_0(x, y) = (x - 0.5)^2 + 6\,(y - 0.5 - 0.5h)^2 - 0.1, \tag{56}$$

as shown in Fig. 16. To make the problem more challenging, the interface is shifted in y-direction by half cell size such that its major axis, where discontinuities in $\nabla\phi$ occur, does not align with cell edges.

We solve the PDE (3) in all cells by the DG method and determine c_0 still using the interface location. The results are illustrated in Fig. 17 together

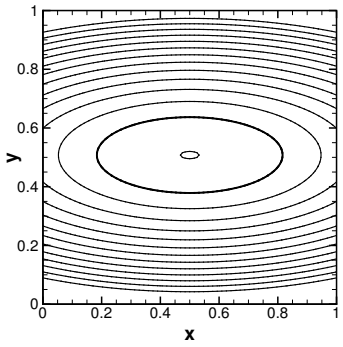
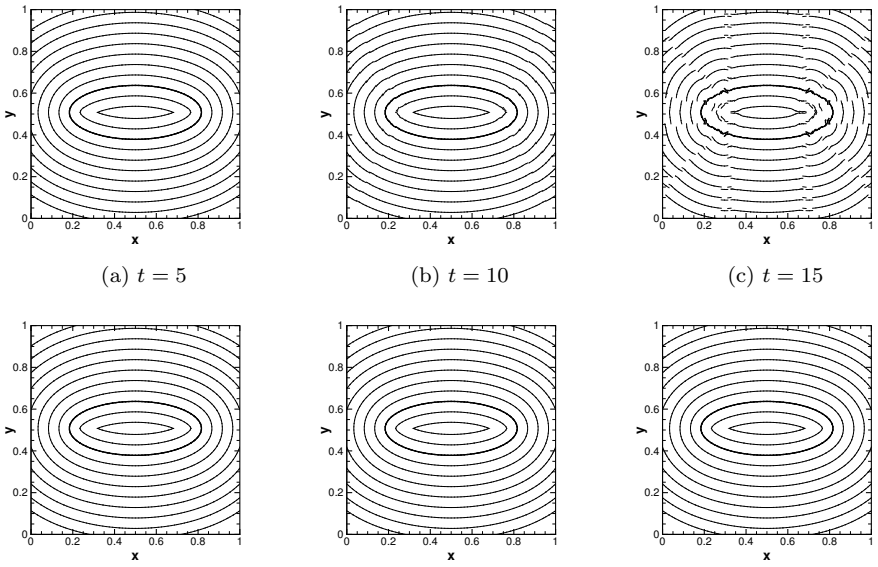


Figure 16: Initial condition ϕ_0 for the elliptic interface. The thick line denotes the interface. Contours run from -0.1 to 1.2 with interval 0.1.



(a) $t = 5$ (b) $t = 10$ (c) $t = 15$

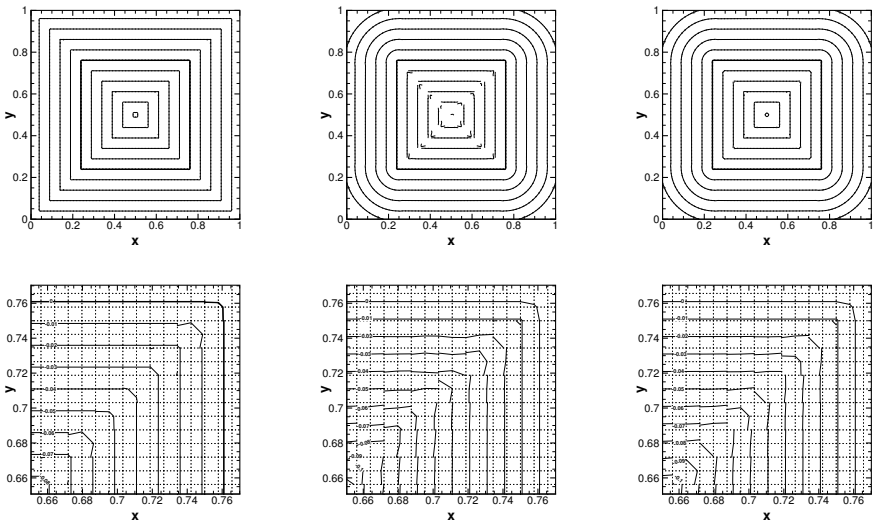
(d) $t = 5$ (e) $t = 10$ (f) $t = 15$

Figure 17: Evolution of ϕ_h for the elliptic interface. The interface cells are solved by evolving the PDE (3) in the top row, while the interface cells in I_p are fixed by the WLP method in the bottom row. Contours run from -0.1 to 0.4 with interval 0.05. The thick line is $\phi_h = 0$. $N = 3$ and $h = 1/64$.

with the WLP method for interface cells. At $t = 5$, both methods generate the same signed distance function and the derivative discontinuities inside

the interface are well captured. However, if we continue the calculation, oscillations start to appear if the interface cells are solved by the PDE. The reason is that $\nabla\phi_h$ in the interface cells are not influenced by any neighbors and the error may grow without bound. The WLP method resolves this issue by anchoring $\nabla\phi_h$ in I_p, a subset of all interface cells.

5.3. Square interface



(a) $t = 0$ (b) $t = 0.4$ without limiter (c) $t = 0.4$ with limiter

Figure 18: Reinitialization of the square interface without (b) and with (c) limiters. The top row shows the solutions in the whole domain, with contours running from -0.25 to 0.25 with interval 0.05. The bottom row shows the close-up views of ϕ_h near the upper-right corner of the interface. For visualization purposes, each actual computational cell is divided into 2×2 cells demarcated by the dotted grid lines. $N = 3$ and $h = 1/64$.

Generally, we do not need to apply limiters. However when the interface has extremely high or singular curvatures, such as the corners of a square interface, it is necessary to consider the limiter. In this example, we consider the initial condition

$$\phi_0(x, y) = 0.8\,(\max\{|x - 0.5|, |y - 0.5|\} - R_0) \tag{57}$$

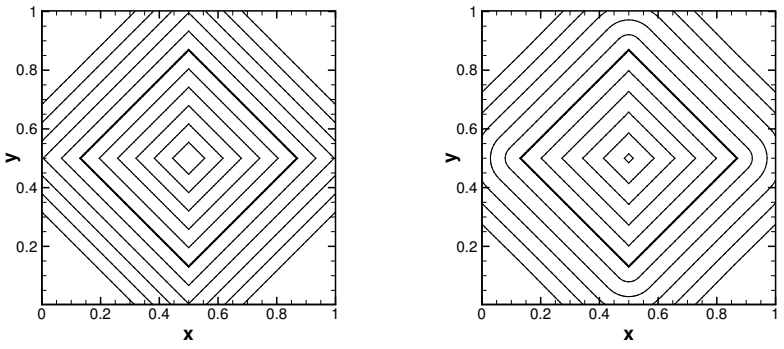
in a unit square, where $R_0 = 16.7h \approx 0.26$ such that all the interface cells are included in I_p. Note that after we project the initial condition to $\phi_{h,0} \in V_h^N$ as shown in Fig. 18(a), the corner is no longer sharp, and the the zero level set

is broken. This numerical oscillation is typical when polynomials, especially those of high degrees, are used to represent non-smooth interfaces. A similar deficiency at the interface corner was also reported in [36]. In practice, a better way to treat the curvature singularity is to smooth it out to a rounded corner, and then refine the mesh to resolve the high curvature. This is out of the scope of this paper. In Fig. 18 (b), the solution without limiters develops oscillations along the diagonals of the square, where characteristics from different interface segments converge. These oscillations are damped and a much better signed distance function is obtained when the limiter is applied, as shown in Fig. 18(c), regardless of the singularities at the corners of the interface.

In the special case when the kinks are located on cells edges, the numerical flux can deal with the the discontinuities without limiting. For example, if we rotate the square interface 45° clockwise, then a diamond interface

$$\phi_0(x, y) = 0.8 \left(|x - 0.5| + |y - 0.5| - \sqrt{2} R_0 \right) \tag{58}$$

is obtained, as shown in Fig. 19(a). In this case, the perfect signed distance function, as depicted in Fig. 19(b), can be easily obtained without any limiter.



(a) $t = 0$ (b) $t = 0.4$

Figure 19: Reinitialization of a square interface with all kinks on the cell edges. No limiter is applied. Contour levels run from -0.25 to 0.25 with interval 0.05. $N = 3$ and $h = 1/64$.

5.4. Contact line

In contact line problems, the fluid interface intersects the solid wall and may result in boundary segments where boundary conditions are required

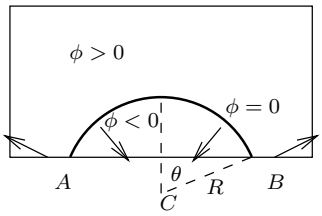


Figure 20: Schematic of a drop sitting on the lower wall of a rectangular domain $[-2, 2] \times [0, 2]$. The interface is a part of a circle, which is given by $|\mathbf{x} - \mathbf{x}_0| = R$ with $\mathbf{x}_0 = (0, -2 - R\cos(\theta))^\intercal$. R is the radius of the circle and θ is the contact angle. A and B are the contact lines (points in 2D) where the interface meets the solid wall. The arrows denote the directions of the characteristics.

for reinitialization. For example, for the interface as shown in Fig. 20, we have

$$S_\eta(\phi_0)\frac{\nabla\phi}{|\nabla\phi|}\cdot\mathbf{n}_w > 0$$

on AB, and

$$S_\eta(\phi_0)\frac{\nabla\phi}{|\nabla\phi|}\cdot\mathbf{n}_w < 0$$

elsewhere on the lower wall with the outward unit normal $\mathbf{n}_w$. As a result, the characteristics go into the computational domain from the lower wall outside AB, where boundary condition has to be supplied. We refer to this part of the boundary as inflow boundary and denote it by

$$\partial\Omega_{in} = \{\mathbf{x} \in \partial\Omega : S(\phi_0)\nabla\phi \cdot \mathbf{n}_w < 0\}. \tag{59}$$

Different approaches have been proposed to supply boundary conditions on $\partial\Omega_{in}$. The first approach is to use ghost cells, where the ϕ values are obtained by extrapolation[51, 64]. However, the use of ghost cells is very difficult to extend to unstructured mesh and curved boundary. The second approach is to compute ϕ on $\partial\Omega_{in}$ by solving a reinitialization problem but only on the boundary [49]. Another approach is to solve a relaxation equation in the first layer of cells along the inflow boundary such that $\frac{\nabla\phi}{|\nabla\phi|}\cdot\mathbf{n}_w$ is fixed [65]. But this approach is dependent on the quality of ϕ_0 away from the interface. In our method, since we solve for $\nabla\phi$, the treatment of inflow boundary is much simpler: Dirichlet conditions for $\nabla\phi$ can be directly imposed on $\partial\Omega_{in}$.

The computational setup is given in Fig. 20 and the initial condition is given by

$$\phi_0 = \sqrt{x^2 + (y + 2 + R\cos\theta)^2} - R. \tag{60}$$

The computational domain is meshed into 64×32 uniform square cells.

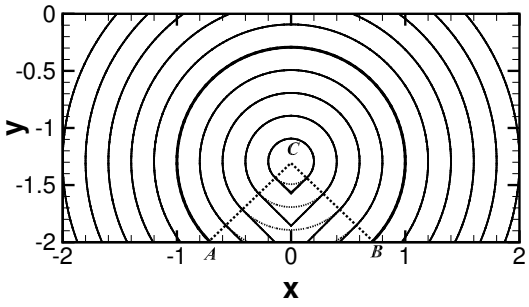


Figure 21: Reinitialization of a drop with a contact angle $\theta = 3\pi/4$. The solid lines are ϕ_h contours at $t = 1$, while the dotted lines are ϕ_0. Contours run from -0.8 to 1 with increment 0.2. $N = 3$ and $h = 1/16$.

We test two cases with different contact angles $\theta = 3\pi/4$ and $\pi/6$. In the first one we set $R = 1$ and $\theta = 3\pi/4$. The inflow boundary is the portion inside a drop: $\Omega_{in} = [-1, 1] \times 0$. Based on the interface normals at A and B, we impose the the boundary condition:

$$\mathbf{u} = \begin{cases} \left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)^{\mathsf{T}}, & \text{if } -1 \le x \le 0,\ y = 0, \\ \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)^{\mathsf{T}}, & \text{if } 0 < x \le 1,\ y = 0. \end{cases} \tag{61}$$

As shown in Fig. 21, the solution ϕ_h at $t = 1$ is exactly the same as ϕ_0 outside the triangle ΔABC. But the solution in ΔABC is determined by the boundary condition (61). In this region, the contours are straight lines intersecting the boundary at the angle $\theta = 3\pi/4$.

The numerical method works equally well for the second case with an acute contact angle $\theta = \pi/6$ and $R = 2$. In this case, $\partial\Omega_{in} = ([-2, -1] \cup [1, 2]) \times 0$, and the boundary condition for (3) is given by

$$\mathbf{u} = \begin{cases} \left(-\frac{1}{2}, \frac{\sqrt{3}}{2}\right)^{\mathsf{T}}, & \text{if } -2 \le x \le -1,\ y = 0, \\ \left(\frac{1}{2}, \frac{\sqrt{3}}{2}\right)^{\mathsf{T}}, & \text{if } 1 \le x \le 2,\ y = 0. \end{cases} \tag{62}$$

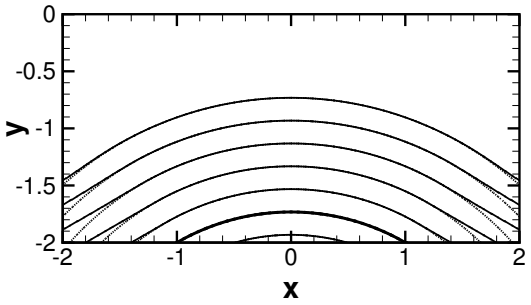


Figure 22: Reinitialization of a drop with a contact angle $\theta = \pi/6$. The solid lines are ϕ_h contours at $t = 1$, while the dotted lines are ϕ_0. Contours run from -0.2 to 1 with increment 0.2. $N = 3$ and $h = 1/16$.

As shown in Fig. 22, the level sets in the region above the inflow boundary are straight lines intersecting the boundary at the angle $\theta = \pi/6$.

It should be noted that the $\nabla\phi$ condition on the inflow boundary is artificial, and the choice is not unique. The bottom line is that the boundary condition should maintain $|\nabla\phi| = 1$ and produce smooth level sets. For example, for circular interfaces in Figs. 21 and 22, it is difficult to say whether the solution ϕ_h at pseudo time $t = 1$ is better than ϕ_0 with concentric level sets. But in the general case, when the interface shape is arbitrary, imposing $\nabla\phi$ condition based on the contact angle seems the most feasible. For three-dimensional problems where the solid wall is a two-dimensional surface, the angle that the level sets make with the wall can be obtained by solving the normal extension equation on that surface [64].

5.5. Interface deformation in a swirling vortex

In this example, we demonstrate the interface-preserving performance of our method in moving interface problems.

5.5.1. Short-time vortex test

Following [66, 67, 30], we consider the deformation of a circular interface in a swirling flow with the velocity field

$$\mathbf{v}(\mathbf{x}, T) = \begin{bmatrix} -\sin^2(\pi x)\sin(2\pi y) \\ \sin^2(\pi y)\sin(2\pi x) \end{bmatrix} \tag{63}$$

which is reversed at $T = 0.5$, in a square domain of size $[0, 1]^2$. Note that T is the flow time, which should be distinguished from the pseudo time t

used in the level-set reinitialization. Therefore, the interface will return to the initial position at $T = 1$. The initial condition is

$$\phi_0(x, y) = \sqrt{(x - 0.5)^2 + (y - 0.75)^2} - 0.15 \tag{64}$$

and the computation domain is a square of size $[0, 1]^2$.

The level-set equation

$$\frac{\partial \phi}{\partial T} + \nabla \cdot (\phi \mathbf{v}) = 0 \tag{65}$$

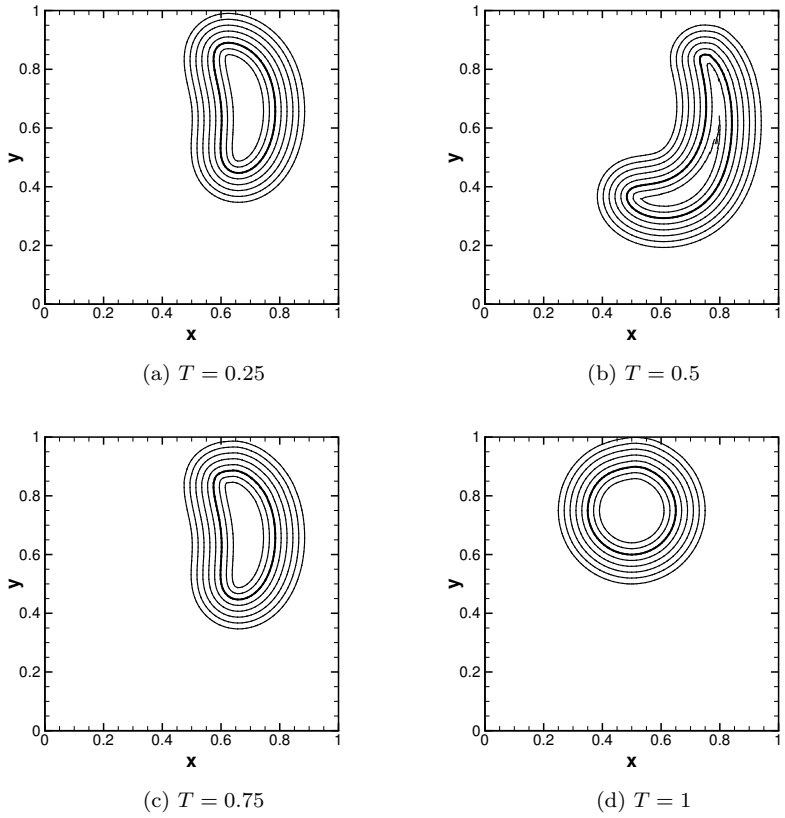
is solved by the DG method [39] with a local LF flux for spatial discretization and the third order TVD Runge-Kutta method for time integration. The time step is chosen to be $\Delta T = 0.1h$ such that the corresponding CFL number is 0.1.

The solutions at different time instants are shown in Fig. 23. Our method produces a signed distance function for different interface shapes with good quality. The comparison between numerical and exact solutions is presented in Fig. 24. At $T = 1$, the circle is accurately recovered (see the dashed line) except for a small portion near the top, which is caused by the error at the high-curvature tip as shown in Fig. 24(a).

Conservation of mass, i.e., the area bounded by the zero level set, is given in Fig. 25(a). We first reinitialize the level-set function every time step, which amounts to 640 reinitializations. The maximum error is -0.39%, which is comparable to that of the conservative level-set method in [30] at the same mesh resolution. The interface displacement as illustrated in Fig. 24 is however at least at the same level as $h = 1/128$ in [30]. It should be noted that the conservative level-set method, by design, conserves the integral of ϕ, but that does not necessarily preserve the exact interface location.

In practice it is usually not necessary to perform reinitialization every time step, especially for small time steps. For example, the reinitialization is performed every 10 steps in [52] and even 100 steps in [47]. Our simulation with reinitialization every 10 steps gives a much better result: the relative error is -0.045%, nearly $\frac{1}{10}$ of the original one. The interfaces at $T = 0.5$ and 1 are almost the same as the exact ones as shown in Fig. 24.

Before the flow reversal, the maximum of the interface curvature increases with time and reaches 32 (i.e., $\frac{1}{2h}$) at $T = 0.28$, as shown in Fig. 25(b). The relative errors of area are -7.07×10^{-6} and -5.66×10^{-6} at $T = 0.28125$ for reinitializations every 1 step and 10 steps, respectively. This indicates that our method barely causes any mass loss for a smooth interface with a low curvature ($\lesssim \frac{1}{2h}$). The mass loss mostly occurs during $T \in [0.28, 0.72]$,

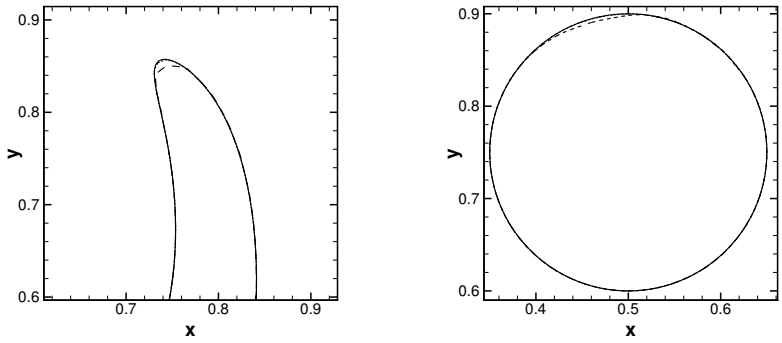


(a) $T = 0.25$ (b) $T = 0.5$

(c) $T = 0.75$ (d) $T = 1$

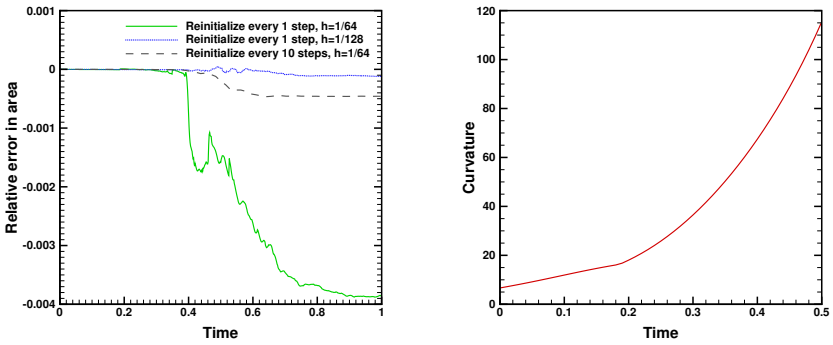
Figure 23: Circular interface sheared by a vortex that is reversed at $T = 0.5$. Reinitialization is performed every time step till a pseudo time $t = 0.1$. ϕ_h contours run from -0.04 to 0.1 with interval 0.02. The thick line is the zero level set. $N = 3$ and $h = 1/64$.

when curvature is above $\frac{1}{2h}$. This is expected because the polynomial space cannot accurately resolve the curvature radius that is comparable to or smaller than the cell size h. If the curvature is too high, there is also a possibility that the interface cell is not detected by our method, such as Fig. 2(e). One way to resolve high curvature is mesh refinement. The results with $h = 1/128$ is included in Fig 25(a) for comparison. Although 1280 reinitializations are performed during $T \in [0, 1]$, the relative error is amazingly small: -0.012%. This error can be further reduced if we perform reinitialization every few time steps.



(a) $T = 0.5$ (b) $T = 1$

Figure 24: Comparison of the interface. The dashed and dotted lines represent the numerical solutions obtained by reinitialization every 1 time step and reinitialization every 10 time steps, respectively. The solid lines represent the exact solution, which almost overlap with the dotted lines. $N = 3$ and $h = 1/64$.



(a) (b)

Figure 25: (a) Relative error in area bounded by the interface. (b) Maximum curvature of the exact interface.

5.5.2. Long-time vortex test

To demonstrate the capability of our method in handling long filaments, we consider a slightly modified velocity field

$$\mathbf{v}\left(\mathbf{x}, T\right) = \begin{bmatrix} -\sin^2\left(\pi x\right)\sin\left(2\pi y\right)\cos(\pi t/8) \\ \sin^2\left(\pi y\right)\sin\left(2\pi x\right)\cos(\pi t/8) \end{bmatrix} \tag{66}$$

following [68, 54, 34, 35]. The interface gets fully stretched at $T = 4$ and restores to initial state at $T = 8$. Thanks to the easy implementation of adaptive mesh refinement in the DG framework, we use an adaptive mesh with finest mesh size $h_{\min} = 1/512$ at the interface. More details of on the adaptive mesh can be found in Sec. 5.7. For stability, we choose $\Delta T = 2.5 \times 10^{-4}$ for $N = 2$ and $\Delta T = 10^{-4}$ for $N = 3$. We perform 20 reinitializations with every time unit.

As shown in Fig. 26, the circle is accurately recovered at $T = 8$, except for some oscillations at the drop tip. The area loss of $N = 2$ (0.15%) is comparable to those obtained by Gómez *et al.* [68] (0.11% to 0.197%) on a 256^2 main grid and Herrmann [54] (0.28%) with $h = 1/1024$, both using finite difference. Since Gómez *et al.* divided each cell around the interface into 4^2 subcells, their finest mesh size is $h_{\min} = 1/1024$, which is comparable to our $N = 2$ in terms of degrees of freedom. However, our results with $N = 3$ is even worse than that with $N = 2$, which is probably because a higher order method is more prone to numerical oscillations when the solution is non-smooth. This is consistent with the consensus that one should use p-refinement for the smooth part and h-refinement for the singular part of the solution in hp-finite element methods.

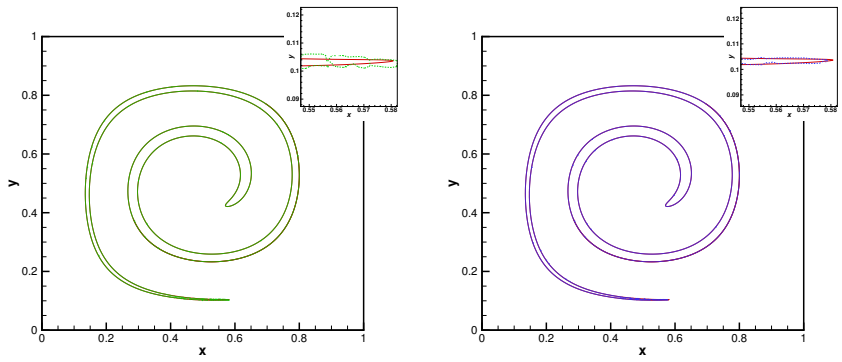
5.5.3. Vortex test in 3D

We consider a sphere with radius $r = 0.15$ centered at $(0.35, 0.35, 0.35)$ in a divergence free velocity field [69, 26, 70, 35]

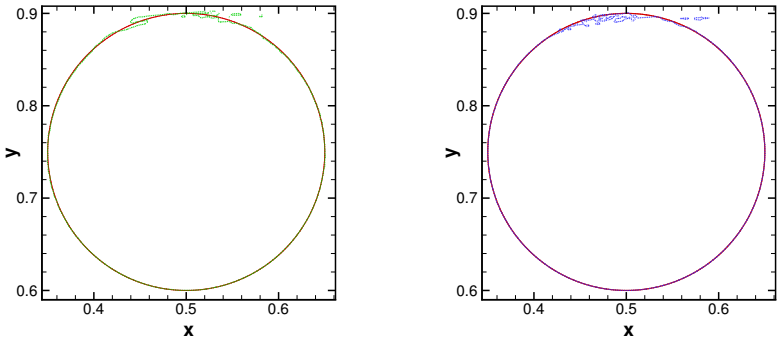
$$\mathbf{v}(\mathbf{x}, T) = \begin{bmatrix} 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi T/3) \\ -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi T/3) \\ -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi T/3) \end{bmatrix}. \tag{67}$$

The sphere gets fully stretched at $T = 1.5$ and restores to its initial shape at $T = 3$. We use an adaptive mesh with $h_{\min} = 1/256$ and DG with $N = 2$. The time step is chosen to be 0.00025 and we perform 40 reinitializations within every time unit. The other parameters are the same as those in 2D calculations.

As shown in Fig. 27, numerical oscillations appear at the equator of the sphere at $T = 3$. This is likely due to the lack of mesh resolution to describe the thin film: the smallest film thickness at $T = 1.5$ is less than $h_{\min}$. Our mass loss 1.62% is slightly greater than the second order level-set method by Min and Gibou, who reported a volume loss of 0.74% with $h_{\min} = 1/512$, but with a slightly different velocity field. The DG conservative level-set method by Jibben and Herrmann [35] seem to perform much better with a volume loss of 0.25% on a 128^2 uniform mesh with N=2. But they only



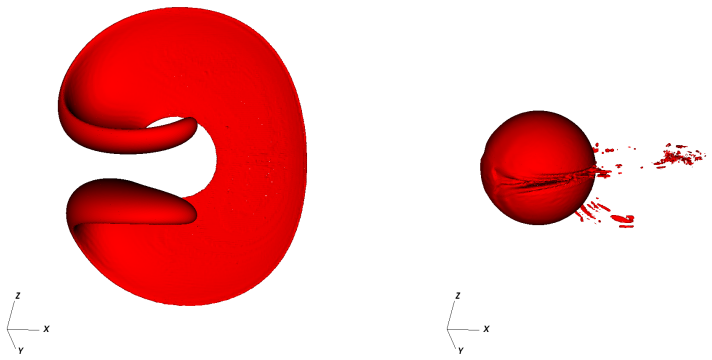
(a) $N = 2,\ T = 4$ (b) $N = 3,\ T = 4$



(c) $N = 2,\ T = 8$ (d) $N = 3,\ T = 8$

Figure 26: Circular interface under long-time shear. The solid line (red) is the exact solution , while the dashed line is the numerical solution. The insets show the zooms of the tail tip. Relative error in area bounded by the interface are 0.15% and 0.29% for $N = 2$ and $N = 3$, respectively. An adaptive mesh with $h_{\min} = 1/512$ is used and the maximal number of cells (at T=4) is around 25000, which amounts to a 158^2 uniform mesh.

performed three reinitializations for the whole simulation while we did 120 reinitializations. This makes the numbers not directly comparable. We have to point out that the conservative level-set methods may perform better for non-smooth solutions, because their formulation has a diffusion term that regularizes the level-set function, especially when discontinuity at the film center develops. But the conservative level-set function has a steep variation at the interface and thus looses the numerical convenience of the signed distance function. Further discussions between the conservative level-set methods and the classical level-set methods are beyond the scope of this work.



(a) $T = 1.5$ (b) $T = 3$

Figure 27: Deformation of a sphere under shear. The relative error in volume is 1.62% at $T = 3$. The maximal number of cells is 460400, which amounts to a 77^3 uniform mesh. $N = 2$, $\beta_{\max} = 3$, $\lambda = 100$, $c_\epsilon = 0.1$, $c_\xi = 0.01$, $Q = 5$, and $h_{min} = 1/256$.

5.6. Rotation of a slotted disk

We consider the rigid boby rotation of a slotted disk [71] in the unit square $[0, 1]^2$. The velocity is

$$\mathbf{v}(\mathbf{x}, T) = \begin{bmatrix} \pi(0.5 - y)/3.14 \\ \pi(x - 0.5)/3.14 \end{bmatrix}, \tag{68}$$

such that the period of rotation is 6.28. A circular disk of radius 0.15 is initially centered at $(0.5, 0.75)$, with a slot of width 0.05 and length 0.25 being cut off. The initial condition for ϕ is obtained based on the closest distance to the interface. We test two uniform meshes, $h = 1/128$ and $1/256$, with time steps $\Delta T = 0.001$ and 0.0005. In both cases, we reinitialize the level-set function every 10 time steps so that the frequency of reinitialization is comparable to that in [24]. The level-set equation is solved in the same way as in Sec. 5.5, and all the other parameters are kept the same. No slope limiter is used in this problem because the sharp corner quickly gets smoothed.

The interface shapes are given in Fig. 28. In the left column, the interfaces after different full revolutions almost overlap. Close-up views at a sharp corner are shown in the right column. The initial interface here is the zero level set after projecting the exact signed distance function to the DG space. It is thus discontinuous at the corner. After rotation, the corner is

eventually rounded to a smooth curve with curvature radius being approximately $2h$. The relative mass losses after three revolutions, as shown in Figure 29, are around 0.1% and 0.06% for $h = 1/128$ and $1/256$, respectively. These are much smaller than 0.86% on a 256^2 mesh and 0.43% on a 512^2 mesh obtained by Hartmann *et al.* [24] using a fifth-order upstream central scheme. It should be noted that our DG method with $N = 3$ has ten degrees of freedom (DOF) in each cell. If we compare the DOF, our 128^2 mesh still has less DOF than the 512^2 finite difference mesh. More importantly, the majority of the mass loss occurs in the initial stage when the sharp corner cannot be well approximated by the DG polynomial space. Once the corner is rounded to such an extent that the curvature can be resolved by the computational mesh, further rotation causes very little mass loss. The mass loss in [24], however, grows almost linearly with the number of revolutions. In comparison, our method preserves interface better, especially in long-time simulations.

5.7. Pinch-off of a pendant drop

In this subsection, we show the capability of our method in handling topological changes. Following [72, 73, 74], we consider the growth and pinch-off of a drop from a capillary tube, with radius a, into an ambient fluid, as illustrated in Fig. 31(a). In the following, we only briefly explain the governing equations and the numerical methods. More details and code validations will be presented in a follow-up paper.

Following the standard treatment in level-set literature, the two-phase flow is governed by a single set of equations:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial T} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = \nabla\cdot\left(-p\mathbf{I} + \boldsymbol{\tau} + \sigma\delta_\omega(\phi)|\nabla\phi|\left(\mathbf{I} - \frac{\nabla\phi\otimes\nabla\phi}{|\nabla\phi|^2}\right)\right) - \rho g\mathbf{e}_z \tag{69}$$

and

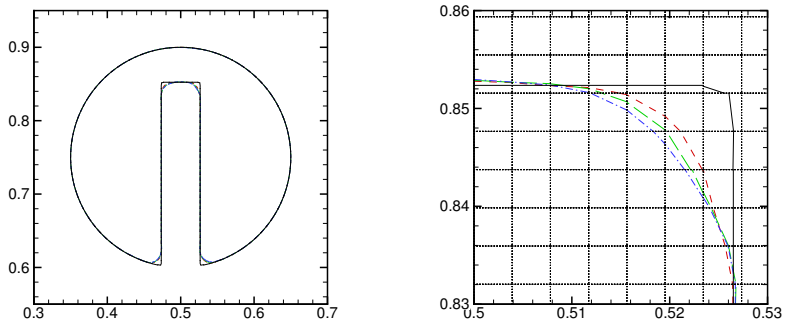
$$\nabla\cdot\mathbf{u} = 0, \tag{70}$$

where $\boldsymbol{\tau} = \mu\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^\intercal\right)$ is the viscous stress, g is the gravitational acceleration, and σ is the surface tension. The density ρ and the viscosity μ of the mixture are given by

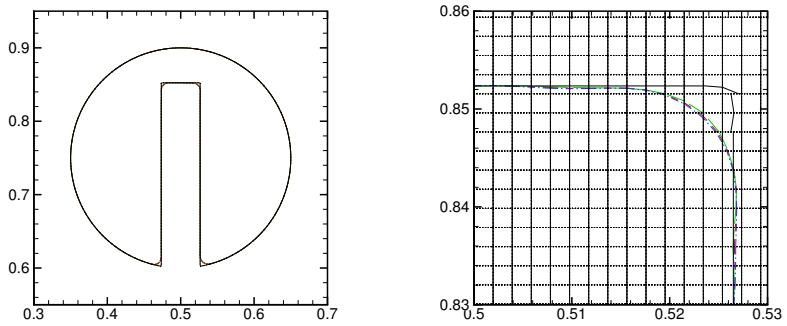
$$\rho = H_\omega(\phi)\rho_1 + (1 - H_\omega(\phi))\rho_2, \tag{71}$$

$$\mu = H_\omega(\phi)\mu_1 + (1 - H_\omega(\phi))\mu_2, \tag{72}$$

where $H_\omega(\phi)$ is the smooth Heaviside function, as defined in Eq. (50), but with ϵ replaced by the half-width ω of the fluid interface. The subscripts



(a) $h = 1/128$



(b) $h = 1/256$

Figure 28: Rotation of the slotted disk. The solid (black) line represent the initial interface. The dashed (red), long-dashed (green), and dash-dotted (blue) lines represent the interfaces after one, two, and three revolutions. The right column shows the zoom of the upper-right corner of the slot. For visualization purposes, each actual computational cell is divided into 2×2 cells demarcated by the dotted grid lines. $N = 3$. (color online)

$_1$ and $_2$ denote the fluid properties inside ($\phi > 0$) and outside ($\phi < 0$) the drop, respectively. $\delta_\omega(\phi)$ is the smooth delta function obtained by taking the derivative of $H_\omega(\phi)$. The flow equations (69,70) are solved using a mixed finite element method with Q^2 for velocity and Q^1 for pressure. A Crank-Nicolson scheme is adopted for temporal discretization. The discretized linear system is solved by the sparse direct solver UMFPACK [75].

Since the flow is axisymmetric, we only compute half of the meridian plane. The computational domain is $[0, 4a] \times [0, 10a]$ in the r-z plane. The mesh is refined at the interface until a minimum mesh size $h_{\min} = \frac{1}{64}a$ is reached and coarsened far away from the interface until a maximum mesh

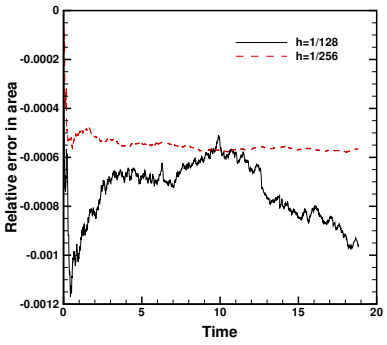


Figure 29: Relative error in the area of the slotted disk.

size $h_{\max} = \frac{1}{2}a$ is reached, as illustrated in Fig 30. We take $\omega = 1.5h_{\min}$.

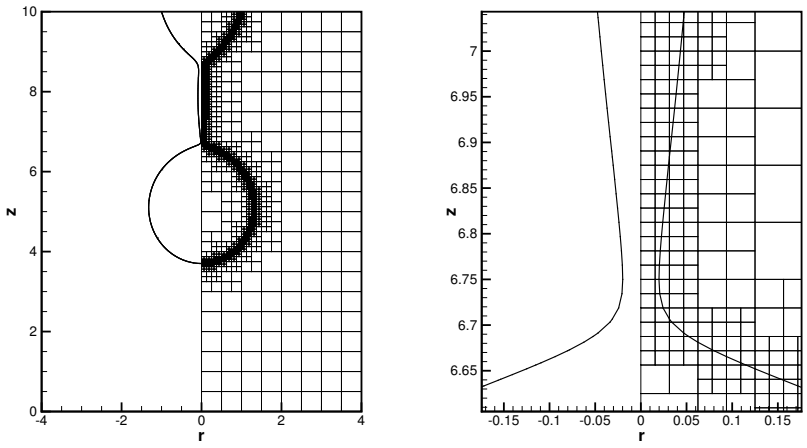


Figure 30: Computational mesh before pinch-off occurs. $\bar{T} = \frac{TV}{a} = 3.3478$. The right panel shows a zoom of the mesh around the neck. The thick lines are the $\phi = 0$ level set.

On the upper boundary, we impose the inflow condition:

$$\mathbf{u} = \begin{cases} -2V\left(1 - \left(\frac{r}{a}\right)^2\right)\mathbf{e}_z, & \text{if } r < a \\ \mathbf{0}, & \text{otherwise,} \end{cases} \tag{73}$$

where V is the average velocity in the capillary tube. Symmetry, slip, and zero stress conditions are imposed on the left, right, and lower boundaries, respectively. The interface is initially hemispherical. We take $a = 1$, $\rho_1 = 1$, $\rho_2 = 0.5$, $\mu_1 = \mu_2 = 0.178$, $\sigma = 1$, $g = 0.930$, and $V = 0.0172$ such that

the dimensionless groups match those of [74]: density ratio $\frac{\rho_1}{\rho_2} = 2$, viscosity ratio $\frac{\mu_1}{\mu_2} = 1$, Capillary number $\mathrm{Ca} = \frac{\mu_1 V}{\sigma} = 3.05 \times 10^{-3}$, Bond number $\mathrm{Bo} = \frac{(\rho_1 - \rho_0) g a^2}{\sigma} = 0.465$, and Weber number $\mathrm{We} = \frac{\rho_1 V^2 a}{\sigma} = 2.95 \times 10^{-4}$. To present the transient results, we define a dimensionless time $\bar{T} = \frac{TV}{a}$.

The time step is determined adaptively based on the mesh size and the fluid velocity. Limited by the DG method for the level-set equation, the CFL number is set to 0.1. We reinitialize ϕ every 50 time steps before the pinch-off, but when it gets close to pinch-off ($\bar{T} \approx 3.35$), ϕ is reinitialized every time step in order to better capture the detachment of the drop. The pseudo stopping time is 4ω, i.e., a signed distance function is maintained in the narrow band $|\phi| < 6h_{\min}$. Since ϕ is always close to a signed distance function, there is no need to use a large λ and we simply take $\lambda = 1$. Other parameters for reinitialization are $N = 3$, $c_\epsilon = 0.1$, $Q = 5$, $\beta_{max} = 3$.

The evolution of the interface is given in Fig. 31. The pinch-off occurs at $\bar{T} = 3.3495$, after which a primary drop detaches and several satellite drops form. At pinch-off, the horizontal radius of the primary drop is $1.33a$, very close to the $1.3a$ extracted from Fig. 2 of [74]. The ϕ contours in the neck region near the instant of pinch-off are given in Fig. 32. After pinch-off, the tip of the thin filament above the primary drop quickly retracts upwards. The topological transitions are well captured by the current method, although the curvature at filament tip is beyond the mesh resolution. Meanwhile, capillary waves develop and the filament eventually breaks up, as shown in Fig. 32(c,d). The radii of the filament in Fig. 32(d) and the bulb at its lower end are about $0.1h_{\min}$ and $h_{\min}$, respectively.

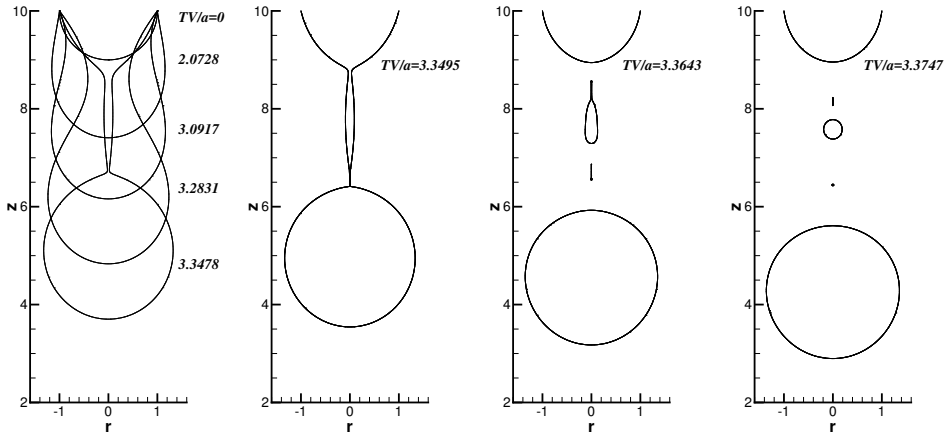
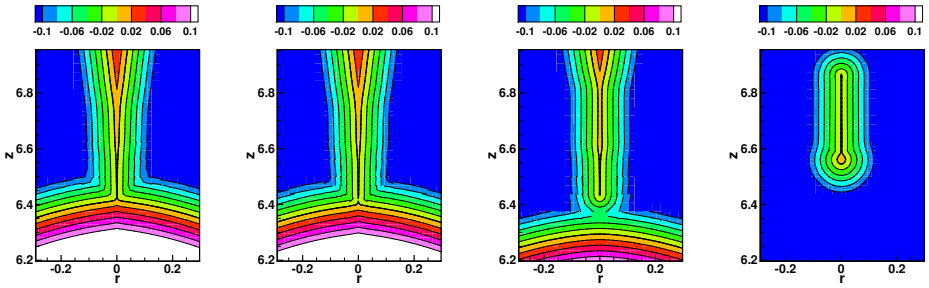


Figure 31: Snapshots of the pinch-off process of a pendant drop.



(a) $\bar{T} = 3.3495$ (b) $\bar{T} = 3.3500$ (c) $\bar{T} = 3.3529$ (d) $\bar{T} = 3.3643$

Figure 32: ϕ contours near the instant of pinch-off. The thick line denotes the interface.

We have to point out that the successful capture of the filament much thinner than the cell size in this test case is fortuitous: the shock wave of ϕ inside the filament lies exactly on the axis of symmetry. In the general case, the WLP method may have trouble in capturing sub-cell structures for the following two reasons. First, if both sides of the filament passes through the same cell, similar to Fig. 2(f), our method fails to identify that cell as an interface cell. Of course, this can be resolved if we consider all the possibilities of interface cells. But the second reason is more substantial: the polynomial approximation suffers from large errors in case of shock waves or other sub-cell structures inside the cell. This limitation applies to all methods based on high-degree polynomial approximation. In interfacial flows, mesh refinement is probably the only way to go because the flow field also needs to be resolved.

In the end, we would like to comment on the computational cost of our reinitialization method. On a regular Cartesian mesh, we have to admit that the DG method is usually much slower than its finite-difference or finite-volume counterparts. There are several reasons, eg., DG uses many quadrature points for numerical integration and DG needs to deal with more degrees of freedom on the same computational mesh. In our case, there is an additional reason: our code is developed for general unstructured quadrilateral and hexagonal meshes, and it does not save any computational cost on a regular Cartesian mesh. But DG has its own advantages on stencil compactness, high order accuracy, *hp*-refinement, unstructured mesh, and parallel performance. In interfacial flows, the cost of reinitialization is not a concern as long as it much less the flow solver. On a single core of a Intel Xeon E5 2.4G processor, our code spends 1.01 s and 3.02 s (wall time) on average on matrix assembly and direct sparse solver, respectively, for one time step of

flow equations in the pendant drop problem (typically around 7500 cells). In comparison, it only takes around 0.18 s to perform one pseudo time step of level-set reinitialization, which is much less than the flow solver. The cost of reinitialization is almost negligible if we only perform reintialization every many flow time steps. On eight cores, the computation times of matrix assembly and level-set reinitialization are brought down to 0.16 s and 0.034 s, respectively, while the computational time for the direct sparse solver almost remain the same. The cost of reinitialization can be further reduced if we only perform reinitialization within a narrow band of the interface [18, 68].

6. Concluding remarks

We have developed a high-order level-set reinitialization method that preserves the zero level set. The major conclusions are summarized as follows.

(i) For the Nth degree piecewise polynomial space, both the weighted local projection and the discontinuous Galerkin method can achieve the optimal Nth order convergence in $\nabla\phi$ for smooth solutions. The convergence order of ϕ is at least $N+1$ in the interface cells and N in the whole domain. The interface displacement may even achieve order $N+2$.

(ii) The penalty flux is necessary to produce smooth solutions, especially when the initial ϕ is highly distorted.

(iii) The numerical method is stable in most cases and handles the discontinuities in $\nabla\phi$ with ease. But in the extreme case with singularities on the interface, the second-derivative limiter may be needed.

(iv) Since we compute $\nabla\phi$ instead of ϕ in the discontinuous Galerkin method, the boundary condition for the Hamilton-Jacobi equation is easy to set up for contact line problems.

(v) The mass loss is negligible if the highest interface curvature can be resolved by the computational mesh. Mesh refinement is suggested if interface curvature exceeding $\frac{1}{2h}$ or other sub-cell structures need to be resolved. In the moving interface problems, mass conservation can be further improved if we preform reinitialization only once every few time steps.

There are many parameters in the proposed method, the tuning of which may be necessary when the initial ϕ is highly distorted. But for general interfacial flows, where ϕ is usually reinitialized before it gets too distorted,

these parameters are no longer sensitive and we recommend the following values: $\beta_{\max} = 3$, $\epsilon = 0.1h$, $\lambda = 1$, $\xi = 0.01\epsilon/h$, and $Q = 5$.

It should be noted that the use of Runge-Kutta discontinuous Galerkin method for the convection dominated level-set equation always leads to a small CFL number. This may be too restrictive, especially, if the level-set method is coupled with an implicit flow solver. One solution to this issue is to use multiple sub-steps for the level-set equation within each time step for the flow solver [34].

The method proposed in this paper can be easily extended to other types of unstructured meshes and complex geometry. The coupling with a finite-element flow solver for moving contact line problems is currently ongoing.

Acknowledgements

This work was supported by the National Science Foundation (Grant DMS-1522604). The authors thank Prof. Chi-Wang Shu at Brown University for stimulating discussions. The coding in this work is based on the open source finite element library deal.II and we would like to thank all of its developers. We also acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results in this paper.

References

[1] S. Osher and J. Sethian, Fronts propagating with curvature dependent speed: algorithms based on hamilton-jacobi formulations *J. Comput. Phys.*, vol. 79, pp. 12–49, 1988.

[2] D. Adalsteinsson and J. Sethian, A fast level set method for propagating interfaces *J. Comput. Phys.*, vol. 118(2), pp. 269–277, 1995.

[3] T. Hou, Z. Li, S. Osher, and H. Zhao, A hybrid method for moving interface problems with application to the heleshaw flow *J. Comput. Phys.*, vol. 134(2), pp. 236–252, 1997.

[4] W. Mulder, S. Osher, and J. Sethian, Computing interface motion in compressible gas dynamics *J. Comput. Phys.*, vol. 100(2), pp. 209–228, 1992.

[5] M. Sussman, P. Smereka, and S. Osher, A level set approach for computing solutions to incompressible two-phase flow *J. Comput. Phys.*, vol. 114, pp. 146–159, 1994.

[6] J. Sethian, A fast marching level set method for monotonically advancing fronts *Proc. Natl. Acad. Sci.*, vol. 93 (4), pp. 1591–1595, 1996.

[7] D. L. Chopp, Some improvements of the fast marching method *SIAM J. on Sci. Comput.*, vol. 23, no. 1, pp. 230–244, 2001.

[8] H. Zhao, A fast sweeping method for eikonal equations *Math. Comp.*, vol. 74, pp. 603–627, 2005.

[9] Y.-T. Zhang, H.-K. Zhao, and J. Qian, High order fast sweeping methods for static hamilton–jacobi equations *J. Sci. Comput.*, vol. 29, no. 1, pp. 25–56, 2006.

[10] M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, An adaptive level set approach for incompressible two-phase flows *J. Comput. Phys.*, vol. 148, no. 1, pp. 81 – 124, 1999.

[11] S. Pillapakkam and P. Singh, A level-set method for computing solutions to viscoelastic two-phase flow *J. Comput. Phys.*, vol. 174, no. 2, pp. 552 – 578, 2001.

[12] X. Zheng, J. Lowengrub, A. Anderson, and V. Cristini, Adaptive unstructured volume remeshing – ii: Application to two- and three-dimensional level-set simulations of multiphase flow *J. Comput. Phys.*, vol. 208, no. 2, pp. 626 – 650, 2005.

[13] J.-J. Xu, Z. Li, J. Lowengrub, and H. Zhao, A level-set method for interfacial flows with surfactant *J. Comput. Phys.*, vol. 212, no. 2, pp. 590 – 616, 2006.

[14] C. Li, C. Xu, C. Gui, and M. Fox, Level set evolution without re-initialization: a new variational formulation. in ieee computer society conference on computer vision and pattern recognition *IEEE Comput. Soc. Conf. on Comput. Vis. Pattern Recognit.*, vol. 1, pp. 430–436, 2005.

[15] C. Basting and D. Kuzmin, A minimization-based finite element formulation for interface-preserving level set reinitialization *Comput.*, vol. 95(1), pp. 13–25, 2012.

[16] T. Utz, F. Kummer, and M. Oberlack, Interface-preserving level-set reinitialization for dg-fem *Int. J. Num. Meth. Fluid*, vol. 84, pp. 183–198, 2017.

[17] F. Gibou, R. Fedkiw, and S. Osher, A review of level-set methods and some recent applications *J. Comput. Phys.*, 2017.

[18] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, A pde-based fast local level set method *J. Comput. Phys.*, vol. 155, no. 2, pp. 410 – 438, 1999.

[19] M. Sussman, E. Fatemi, P. Smereka, and S. Osher, An improved level set method for incompressible two-phase flows *Comput. Fluids*, vol. 27, no. 5, pp. 663 – 680, 1998.

[20] M. Sussman and E. Fatemi, An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow *SIAM J. on scientific computing*, vol. 20, no. 4, pp. 1165–1191, 1999.

[21] G. Russo and P. Smereka, A remark on computing distance functions *J. Comput. Phys.*, vol. 163, no. 1, pp. 51 – 67, 2000.

[22] C. Min, On reinitializing level set functions *J. computational physics*, vol. 229, no. 8, pp. 2764–2772, 2010.

[23] A. du Chéné, C. Min, and F. Gibou, Second-order accurate computation of curvatures in a level set framework using novel high-order reinitialization schemes *J. Sci. Comput.*, vol. 35, no. 2-3, pp. 114–131, 2008.

[24] D. Hartmann, M. Meinke, and W. Schroder, Differential equation based constrained reinitialization for level set methods *J. Comput. Phys.*, vol. 227(14), pp. 6821–6845, 2008.

[25] D. Hartmann, M. Meinke, and W. Schroder, The constrained reinitialization equation for level set methods *J. Comput. Phys.*, vol. 229, no. 5, pp. 1514 – 1535, 2010.

[26] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, A hybrid particle level set method for improved interface capturing *J. Comput. Phys.*, vol. 183, no. 1, pp. 83 – 116, 2002.

[27] M. Sussman and E. G. Puckett, A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows *J. Comput. Phys.*, vol. 162, no. 2, pp. 301 – 337, 2000.

[28] S. P. van der Pijl, A. Segal, C. Vuik, and P. Wesseling, A mass-conserving level-set method for modelling of multi-phase flows *Int. J. for Numer. Methods Fluids*, vol. 47, no. 4, pp. 339–361, 2005.

[29] X. Yang, A. J. James, J. Lowengrub, X. Zheng, and V. Cristini, An adaptive coupled level-set/volume-of-fluid interface capturing method for unstructured triangular grids *J. Comput. Phys.*, vol. 217, no. 2, pp. 364 – 394, 2006.

[30] E. Olsson and G. Kreiss, A conservative level set method for two phase flow *J. Comput. Phys.*, vol. 210(1), pp. 225–246, 2005.

[31] E. Olsson, G. Kreiss, and S. Zahedi, A conservative level set method for two phase flow ii *J. Comput. Phys.*, vol. 225, no. 1, pp. 785 – 807, 2007.

[32] D. Jacqmin, Calculation of two-phase navier–stokes flows using phase-field modeling *J. Comput. Phys.*, vol. 155, no. 1, pp. 96 – 127, 1999.

[33] T. Biben and C. Misbah, Tumbling of vesicles under shear flow within an advected-field approach *Phys. Rev. E*, vol. 67, p. 031908, 2003.

[34] M. Owkes and O. Desjardins, A discontinuous galerkin conservative level set scheme for interface capturing in multiphase flows *J. Comput. Phys.*, vol. 249, pp. 275 – 302, 2013.

[35] Z. Jibben and M. Herrmann, An arbitrary-order runge–kutta discontinuous galerkin approach to reinitialization for banded conservative level sets *J. Comput. Phys.*, vol. 349, pp. 453 – 473, 2017.

[36] R. Saye *et al.*, High-order methods for computing distances to implicitly defined surfaces *Commun. Appl. Math. Comput. Sci.*, vol. 9, no. 1, pp. 107–141, 2014.

[37] B. Cockburn and C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws. II: General framework *Math. Comput.*, vol. 52, pp. 411–435, 1989.

[38] B. Cockburn, S. Y. Lin, and C.-W. Shu, Tvb Runge-Kutta local projection discontinuous galerkin finite element method for conservation laws III: one-dimensional systems *J. Comput. Phys.*, vol. 84, pp. 90–113, 1989.

[39] B. Cockburn, S. Hou, and C.-W. Shu, The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV: the multidimensional case *Math. Comput.*, vol. 54, pp. 545–581, 1990.

[40] Y.-T. Zhang and C.-W. Shu, High-order weno schemes for hamilton–jacobi equations on triangular meshes *SIAM J. on Sci. Comput.*, vol. 24, no. 3, pp. 1005–1030, 2003.

[41] D. Levy, S. Nayak, C.-W. Shu, and Y.-T. Zhang, Central weno schemes for hamilton–jacobi equations on triangular meshes *SIAM J. on Sci. Comput.*, vol. 28, no. 6, pp. 2229–2247, 2006.

[42] C. Hu and C.-W. Shu, A discontinuous galerkin finite element method for Hamilton-Jacobi equations *SIAM J. Sci. Comput.*, vol. 21, pp. 666–690, 1999.

[43] F. Li and C.-W. Shu, Reinterpretation and simplified implementation of a discontinuous Galerkin method for Hamilton-Jacobi equations *Appl. Math. Lett.*, vol. 18, pp. 1204–1209, 2005.

[44] Y. Cheng and C.-W. Shu, A discontinuous galerkin finite element method for directly solving the Hamilton-Jacobi equations *J. Comput. Phys.*, vol. 223, pp. 398–415, 2007.

[45] J. Yan and S. Osher, A local discontinuous galerkin method for directly solving hamilton-jacobi equations *J. Comput. Phys.*, vol. 239, pp. 232–244, 2011.

[46] C.-W. Shu, Survey on discontinuous galerkin methods for hamilton-jacobi equations *Contemp. Math.*, vol. 586, pp. 323–330, 2013.

[47] S. Fechter and C.-D. Munz, A discontinuous Galerkin-based sharp-interface method to simulate three-dimensional compressible two-phase flow *Int. J. for Numer. Methods Fluids*, vol. 78, no. 7, pp. 413–435, 2015.

[48] E. Marchandise, J.-F. Remacle, and N. Chevaugeon, A quadrature-free discontinuous galerkin method for the level set equation *J. Comput. Phys.*, vol. 212, pp. 338–357, Feb. 2006.

[49] J. Grooss and J. Hesthaven, A level set discontinuous Galerkin method for free surface flows *Comput. Methods Appl. Mech. Eng.*, vol. 195, no. 25, pp. 3406 – 3429, 2006.

[50] A. Karakus, T. Warburton, M. Aksel, and C. Sert, A GPU accelerated level set reinitialization for an adaptive discontinuous Galerkin method *Comput. Math. Appl.*, vol. 72, no. 3, pp. 755 – 767, 2016.

[51] P. D. M. Spelt, A level-set approach for simulations of flows with multiple moving contact lines with hysteresis *J. Comput. Phys.*, vol. 207, pp. 389–404, 2005.

[52] R. F. Ausas, E. A. Dari, and G. C. Buscaglia, A geometric mass-preserving redistancing scheme for the level set function *Int. J. for Numer. Methods Fluids*, vol. 65, no. 8, pp. 989–1010, 2011.

[53] N. Parolini, Computational fluid dynamics for naval engineering problems *PhD thesis, EPFL Lausanne*, 2004.

[54] M. Herrmann, A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids *J. Comput. Phys.*, vol. 227, no. 4, pp. 2674 – 2706, 2008.

[55] B. Muller, F. Kummer, and M. Oberlack, Highly accurate surface and volume integration on implicit domains by means of moment-fitting *Int. J. for Numer. Methods Eng.*, vol. 96, no. 8, pp. 512–528, 2013.

[56] B. Cockburn and C.-W. Shu, The Runge-Kutta discontinuous galerkin method for conservation laws V: multidimensional systems *J. Comput. Phys.*, vol. 141, pp. 199–224, 1998.

[57] J. Qiu and C.-W. Shu, Runge–Kutta discontinuous Galerkin method using WENO limiters *SIAM J. on Sci. Comput.*, vol. 26, no. 3, pp. 907–929, 2005.

[58] J. Zhu, J. Qiu, C.-W. Shu, and M. Dumbser, Runge–Kutta discontinuous Galerkin method using WENO limiters ii: Unstructured meshes *J. Comput. Phys.*, vol. 227, no. 9, pp. 4330 – 4353, 2008.

[59] J. Zhu, X. Zhong, C.-W. Shu, and J. Qiu, Runge–Kutta discontinuous Galerkin method using a new type of WENO limiters on unstructured meshes *J. Comput. Phys.*, vol. 248, pp. 200 – 220, 2013.

[60] S. Gottlieb and C.-W. Shu, Total variation diminishing runge-kutta schemes *Math. Comp.*, vol. 67, pp. 73–85, 1998.

[61] B. Cockburn and C.-W. Shu, Runge-Kutta discontinuous Galerkin finite element methods for convection-dominated problems *J. Sci. Comput.*, vol. 16, pp. 173–261, 2001.

[62] W. Bangerth, R. Hartmann, and G. Kanschat, deal.II – a general purpose object oriented finite element library *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 24/1–24/27, 2007.

[63] W. Bangerth, D. Davydov, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and D. Wells, The `deal.II` library, version 8.4 *J. Numer. Math.*, vol. 24, 2016.

[64] S. Xu and W. Ren, Reinitialization of the level-set function in 3d simulation of moving contact lines *Commun. Comput. Phys.*, vol. 20, no. 5, pp. 1163–1182, 2016.

[65] G. Della Rocca and G. Blanquart, Level set reinitialization at a contact line *J. Comput. Phys.*, vol. 265, pp. 34 – 49, 2014.

[66] J. Bell, P. Colella, and H. Glaz, A second-order projection method for the incompressible navier-stokes equations *J. Comput. Phys.*, vol. 85(2), pp. 257–283, 1989.

[67] W. Rider and D. Kothe, Reconstructing volume tracking *J. Comput. Phys.*, vol. 141(2), pp. 112–152, 1998.

[68] P. Gomez, J. Hernandez, and J. Lopez, On the reinitialization procedure in a narrow-band locally refined level set method for interfacial flows *Int. journal for numerical methods engineering*, vol. 63, no. 10, pp. 1478–1512, 2005.

[69] R. J. LeVeque, High-resolution conservative algorithms for advection in incompressible flow *SIAM J. on Numer. Analysis*, vol. 33, no. 2, pp. 627–665, 1996.

[70] C. Min and F. Gibou, A second order accurate level set method on non-graded adaptive cartesian grids *J. Comput. Phys.*, vol. 225, no. 1, pp. 300 – 321, 2007.

[71] S. T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids *J. Comput. Phys.*, vol. 31, no. 3, pp. 335–362, 1979.

[72] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, and S. Zaleski, Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows *J. Comput. Phys.*, vol. 152, no. 2, pp. 423 – 456, 1999.

[73] E. D. Wilkes, S. D. Phillips, and O. A. Basaran, Computational and experimental analysis of dynamics of drop formation *Phys. fluids*, vol. 11, no. 12, pp. 3577–3598, 1999.

[74] C. Zhou, P. Yue, and J. J. Feng, Formation of simple and compound drops in microfluidic devices *Phys. fluids*, vol. 18, no. 9, p. 092105, 2006.

[75] T. A. Davis, Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method *ACM Trans. Math. Softw.*, vol. 30, pp. 196–199, June 2004.