Computer Physics Communications 184 (2013) 1547–1554

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

A new method for the level set equation using a hierarchical-gradient truncation and remapping technique

Haruhiko Kohno [a,*], Jean-Christophe Nave [b]

[a] Plasma Science and Fusion Center, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA
[b] The Department of Mathematics and Statistics, McGill University, 805 Sherbrooke W., Montreal, QC, H3A 2K6, Canada

ARTICLE INFO

Article history:
Received 17 January 2012
Received in revised form
17 January 2013
Accepted 4 February 2013
Available online 9 February 2013

Keywords:
Level set equation
Interface capturing
Advection problem
Semi-Lagrangian scheme

ABSTRACT

We present a novel numerical method for solving the advection equation for a level set function. The new method uses hierarchical-gradient truncation and remapping (H-GTaR) of the original partial differential equation (PDE). Our strategy reduces the original PDE to a set of decoupled linear ordinary differential equations with constant coefficients. Additionally, we introduce a remapping strategy to periodically guarantee solution accuracy for a deformation problem. The proposed scheme yields nearly an exact solution for a rigid body motion with a smooth function that possesses vanishingly small higher derivatives and calculates the gradient of the advected function in a straightforward way. We will evaluate our method in one- and two-dimensional domains and present results to several classical benchmark problems.

1. Introduction

In many engineering fields, moving interface problems still remain a challenge, although much research effort has been spent on solving the problems from a numerical point of view. One of the promising techniques is a level set method, which was first proposed by Osher et al. [1] and has been investigated by many authors in various fields (see, e.g., Refs. [2–8] and the references therein). In level set methods, a moving interface is implicitly captured by a level set function on fixed Eulerian grids, and thus, calculations are stably conducted no matter how complicated the interface evolves (even if the process includes merger and/or split, for example). However, since the interface is not in general positioned on the grid points, one is required to provide a fine mesh, at least in the vicinity of the interface, and sophisticated interpolation functions in order to obtain an accurate numerical solution.

Over a long period of time, many researchers have focused on improving the interpolation techniques for the advection equation, some of which were applied to the level set method (see, e.g., Refs. [9–14]). An important technique, which is the basis for many numerical methods including the preceding references, is the semi-Lagrangian (SL) scheme (see, e.g., Refs. [15–19]). A great advantage of this method is that it removes the constraint of the time step known as the Courant–Friedrichs–Lewy (CFL) condition. This feature is particularly important when multiscale behavior is involved in the analysis, in which case one has to refine a mesh to the extent that the fine-scale variation is sufficiently resolved with an adopted numerical scheme. In an advection problem, the SL method calculates the "start point", where an advected quantity is positioned at Δt before it reaches a grid point, within a fixed mesh by integrating the Lagrangian form of the advection equation. Since the quantity at the start point corresponds to the exact solution at the focused grid point after Δt from the present time, but it is not in general positioned on the other grid nodes, a number of researchers have been made every effort to develop highly accurate (usually high order) interpolation schemes to obtain good approximation at the start point.

In the present paper, we propose a new SL-type approach which can be applied to the level set equation. We will show that using this approach, nearly an exact solution can be obtained without any spatial interpolation when the advected quantity retains certain properties. Furthermore, following the proposed approach, it is possible to straightforwardly calculate the normal vector on the interface (i.e., on a specified contour surface of the level set function). Our method is quite unique and based on hierarchical-gradient truncation and remapping, which we refer to as H-GTaR in brief, of the advection equation expressed with a mapping function. We first present the numerical procedure in detail in Sections 2 and 3, and then give the solutions of one-dimensional (1D) and two-dimensional (2D) test problems in Section 4.

* Correspondence to: Department of Mechanical Information Science and Technology, Kyushu Institute of Technology, 680-4 Kawazu, Iizuka, Fukuoka 820-8502, Japan (The author, H. Kohno, will move to this address from April 1, 2013.).
E-mail addresses: kohno@alum.mit.edu (H. Kohno), jcnave@math.mcgill.ca (J.-C. Nave).

0010-4655/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2013.02.002

2. Derivation of the linear ordinary differential equations

In this section, we present a method for solving multidimensional advection equations with spatially variable velocity. We will see that the approximation $\boldsymbol{\Psi}^{(n+1)} \approx \mathbf{0}$ converts the original equation into (decoupled) linear ordinary differential equation(s) (ODE(s)) with constant coefficients at a specific point.

2.1. Gradient form of the advection equation

First, consider the following N-dimensional advection equation in Eulerian form in $\mathbb{R}^N \times (0, \infty)$:

$$\phi_t + \boldsymbol{u} \cdot \nabla \phi = 0. \tag{1}$$

Here, $\phi : \mathbb{R}^N \times [0, \infty) \to \mathbb{R}$ and $\boldsymbol{u} \in \mathbb{R}^N$. We assume that ϕ and $\boldsymbol{u}$ are sufficiently smooth for now.

Taking the gradient of Eq. (1) yields

$$\boldsymbol{\Psi}_t^{(1)} + \boldsymbol{u} \cdot \nabla \boldsymbol{\Psi}^{(1)} = -\nabla \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(1)}, \tag{2}$$

where $\boldsymbol{\Psi}^{(1)} = \nabla \phi$. In the same way, taking the gradient of Eq. (2) gives

$$\boldsymbol{\Psi}_t^{(2)} + \boldsymbol{u} \cdot \nabla \boldsymbol{\Psi}^{(2)} = -2\nabla \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(2)} - \nabla^{(2)} \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(1)}, \tag{3}$$

where $\boldsymbol{\Psi}^{(2)} = \nabla \boldsymbol{\Psi}^{(1)}$ and $\nabla^{(2)} \boldsymbol{u} = \nabla (\nabla \boldsymbol{u})$. By induction, we reach the following formula:

$$\boldsymbol{\Psi}_t^{(n)} + \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(n+1)} = -\sum_{i=1}^{n} C_i^n \nabla^{(i)} \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(n+1-i)}, \tag{4}$$

where $C_i^n \equiv n!/i!\,(n-i)!$, $\boldsymbol{\Psi}^{(n)} = \nabla \boldsymbol{\Psi}^{(n-1)}$, $\nabla^{(i)} \boldsymbol{u} = \nabla \left(\nabla^{(i-1)} \boldsymbol{u} \right)$ ($\nabla^{(1)} \boldsymbol{u} = \nabla \boldsymbol{u}$), and $n \geq 1$.

2.2. Derivation of the ODE for $\Psi^{(1)}$

Assume that our interest is to solve a problem which has the property of $\left| \boldsymbol{\Psi}^{(\infty)} \right| \to 0$. Let us consider the approximation that $\boldsymbol{\Psi}^{(n+1)} \approx \mathbf{0}$. Then Eq. (4) is approximated at

$$\boldsymbol{\Psi}_t^{(n)} \approx -\sum_{i=1}^{n} C_i^n \nabla^{(i)} \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(n+1-i)}. \tag{5}$$

The goal here is to construct the ODE for $\boldsymbol{\Psi}^{(1)}$ in terms of t at a fixed spatial point. The order n in Eq. (5) can be hierarchically reduced using the following upper-level equations:

$$\boldsymbol{u} \cdot \boldsymbol{\Psi}_{mt}^{(n')} = -\boldsymbol{\Psi}_{(m+1)t}^{(n'-1)} - \sum_{i=1}^{n'-1} C_i^{n'-1} \nabla^{(i)} \boldsymbol{u} \cdot \boldsymbol{\Psi}_{mt}^{(n'-i)}, \tag{6}$$

where $n' = 2, \ldots, n$ and $m = 0, \ldots, n+1-n'$. Here, the coefficient of t in the subscript of $\boldsymbol{\Psi}$, which includes the integer m, corresponds to the number of differentiations (e.g., $\boldsymbol{\Psi}_{0t} = \boldsymbol{\Psi}$ and $\boldsymbol{\Psi}_{2t} = \boldsymbol{\Psi}_{tt}$). Note that the upper limit of m depends on the value of n'.

Below are a few examples explicitly showing the resulting ODEs for $\boldsymbol{\Psi}^{(1)}$ with different dimensions (i.e., $d' = 1$ and $d' = d > 1$, where d' is the number of dimensions).

Example 1. $d' = 1$ (1D case)

$$\begin{aligned}
n = 2 &: \ \Psi_{tt}^{(1)} + 3u_x \Psi_t^{(1)} + \left(2u_x^2 - u u_{xx} \right) \Psi^{(1)} = 0 \\
n = 3 &: \ \Psi_{ttt}^{(1)} + 6u_x \Psi_{tt}^{(1)} + \left(11u_x^2 - 4u u_{xx} \right) \Psi_t^{(1)} \\
&\quad + \left(6u_x^3 - 6u u_x u_{xx} + u^2 u_{xxx} \right) \Psi^{(1)} = 0
\end{aligned}$$

Example 2. $d' = d$ (multidimensional case)

$$\begin{aligned}
n = 2 &: \ \boldsymbol{\Psi}_{tt}^{(1)} + 3\nabla \boldsymbol{u} \cdot \boldsymbol{\Psi}_t^{(1)} + 2\nabla \boldsymbol{u} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(1)} \\
&\quad - \nabla^{(2)} \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(1)} \cdot \boldsymbol{u} = \mathbf{0} \\
n = 3 &: \ \boldsymbol{\Psi}_{ttt}^{(1)} + 6\nabla \boldsymbol{u} \cdot \boldsymbol{\Psi}_{tt}^{(1)} + 11\nabla \boldsymbol{u} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{\Psi}_t^{(1)} \\
&\quad - 4\nabla^{(2)} \boldsymbol{u} \cdot \boldsymbol{\Psi}_t^{(1)} \cdot \boldsymbol{u} + 6\nabla \boldsymbol{u} \cdot \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(1)} \\
&\quad + \left(-3\nabla^{(2)} \boldsymbol{u} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(1)} - 3\nabla \boldsymbol{u} \cdot \nabla^{(2)} \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(1)} \right. \\
&\quad \left. + \nabla^{(3)} \boldsymbol{u} \cdot \boldsymbol{\Psi}^{(1)} \cdot \boldsymbol{u} \right) \cdot \boldsymbol{u} = \mathbf{0}
\end{aligned}$$

Note that for $d = 1$ two equations in Example 2 reduce to the corresponding equations in Example 1.

2.3. Derivation of the ODE for $\Psi^{\beta(1)}$

Let us consider a multidimensional problem ($d' = d$) with an arbitrary order ($n' = n$). The component equations corresponding to the ODE for $\boldsymbol{\Psi}^{(1)}$ are written as follows:

$$I^j = \Psi_{nt}^{j(1)} + \sum_{l=1}^{n} \sum_{k=1}^{d} \alpha_{j,k,l} \Psi_{(n-l)t}^{k(j)} = 0, \tag{7}$$

where $j = 1, \ldots, d$ and the coefficients $\alpha_{j,k,l}$ are the functions of space. Now, let us consider deriving the ODE for one component $\Psi^{\beta(1)}$ ($1 \leq \beta \leq d$) by using these d equations. For this purpose one needs to delete the following $n\,(d-1)$ variables: $\Psi_{(n-l)t}^{k(1)}$ where $k = 1, \ldots, d$ and $l = 1, \ldots, n$ ($k \neq \beta$). The required $n\,(d-1)+1$ equations are derived using the following $nd\,(d-1)+1$ distinct equations: I_{lt}^j ($j = 1, \ldots, d$, $l = 0, \ldots, n\,(d-1)-1$) and $I_{n(d-1)t}^{\beta}$. Through calculation, one obtains the (nd)-order ODE for $\Psi^{\beta(1)}$. The differential equations for the other components can also be derived in the same way; as a result, the component equations are completely decoupled. Note that this approach is applicable even for $n = 1$ or $d = 1$. The initial condition for each component is calculated using Eq. (4) with the initial distribution of ϕ. As an example, we demonstrate the above procedure for the case where $n = 2$ and $d = 2$ in the Appendix.

2.4. Time-integration algorithm

The ODEs for $\Psi^{k(1)}$ ($k = 1, \ldots, d$) derived in Section 2.3 can be numerically solved using any Runge–Kutta time-integration algorithm. Here, we apply the third-order Runge–Kutta method proposed by Shu and Osher [20]. At each time step, one obtains the intermediate solutions $\Psi^{[0]}$, $\Psi^{[1]}$, and $\Psi^{[2]}$, where the superscript $k\,(1)$ is omitted here. Using these solutions, the advected quantity ϕ is finally computed using the same Runge–Kutta approach as follows:

$$\begin{cases}
\phi^{[1]} = \phi^{[0]} - u \Delta t \Psi^{[0]} \\
\phi^{[2]} = \dfrac{3}{4}\phi^{[0]} + \dfrac{1}{4}\phi^{[1]} - \dfrac{1}{4} u \Delta t \Psi^{[1]} \\
\phi^{[3]} = \dfrac{1}{3}\phi^{[0]} + \dfrac{2}{3}\phi^{[2]} - \dfrac{2}{3} u \Delta t \Psi^{[2]}.
\end{cases} \tag{8}$$

3. Mapping function

Although the proposed scheme so far possesses the desirable feature of being independent of the grid shape and resolution, its range of application is still limited due to the restriction that the advected variable ϕ is required to satisfy the property, $\left| \boldsymbol{\Psi}^{(\infty)} \right| \to 0$ (see Section 2.1). However, this issue can be avoided by introducing a reference map $\boldsymbol{\xi}$. The procedure is summarized as follows.

3.1. Introduction of the mapping function

The advection equation (1) can be expressed in a different way with a variable $\boldsymbol{\xi}$ as follows:

$$\begin{cases} \boldsymbol{\xi}_t + \boldsymbol{u} \cdot \nabla \boldsymbol{\xi} = \boldsymbol{0} \\ \boldsymbol{\xi}(\boldsymbol{x}, t=0) = \boldsymbol{x} \\ \phi(\boldsymbol{x}, t) = \phi_0(\boldsymbol{\xi}(\boldsymbol{x}, t)) \\ \phi(\boldsymbol{x}, t=0) = \phi_0(\boldsymbol{x}). \end{cases} \tag{9}$$

Here, $\boldsymbol{\xi}$ is called a reference map or a mapping function. An important point is that with this approach, restrictions pertaining to the initial distribution of ϕ are removed; the proposed method can be applied to the advection problem even if the initial value of ϕ has corners in the calculation domain. This is obvious since the gradient $\nabla[\boldsymbol{\xi}(\boldsymbol{x}, t=0)]$ yields a constant tensor. However, when we consider a general problem where the velocity $\boldsymbol{u}$ is arbitrary, we expect that, at some time T_{R}, the approximation $\nabla^{(n+1)}\boldsymbol{\xi} \approx \boldsymbol{0}$ is no longer valid. In such a case, we need to evaluate the function $\phi(\boldsymbol{x}, t=T_{\mathrm{R}}) = \phi_0(\boldsymbol{\xi}(\boldsymbol{x}, t=T_{\mathrm{R}}))$ and reset as $\boldsymbol{\xi}(\boldsymbol{x}, t=T_{\mathrm{R}}) = \boldsymbol{x}$. We call this procedure remapping.

3.2. Remapping

The remapping procedure is based on the idea that the mapping function $\boldsymbol{\xi}$ can be reset to the initial profile whenever the approximation $\nabla^{(n+1)}\boldsymbol{\xi} \approx \boldsymbol{0}$ is not well satisfied, and then reconstruct the one-to-one relationship between ϕ and $\boldsymbol{\xi}$. Let us call $T_{\mathrm{R}}^{[k]}$ the sequence of remapping times, where k is an integer and represent the number of times the remapping is executed. A numerical solution at an arbitrary time $\widetilde{t}$ is obtained after successive remapping at time $T_{\mathrm{R}}^{[k]}$ as follows:

$$\begin{cases} \phi_k\left(\boldsymbol{x}, t=T_{\mathrm{R}}^{[k]}\right) = \phi_{k-1}\left(\boldsymbol{\xi}_{k-1}\left(\boldsymbol{x}, t=T_{\mathrm{R}}^{[k]}\right)\right) \\ \qquad k = 1, \ldots, k_{\max} \\ \boldsymbol{\xi}_l\left(\boldsymbol{x}, t=T_{\mathrm{R}}^{[l]}\right) = \boldsymbol{x} \quad l = 0, \ldots, k_{\max}-1, \end{cases} \tag{10}$$

where $T_{\mathrm{R}}^{[0]} = 0$, and $\phi_{k_{\max}}(\boldsymbol{x})$ is the numerical solution at time $t = \widetilde{t} = T_{\mathrm{R}}^{[k_{\max}]}$. Here, note that only $\phi_0(\boldsymbol{x})$ is given by an equation, if provided, which determines the initial profile of ϕ. Since we obtain the value of ϕ_k only at each node, some interpolation function needs to be used to define $\phi_k(\boldsymbol{x})$ over the calculation domain. For example, if we apply finite element interpolation functions $N_i(\boldsymbol{x})$ for this purpose, an intermediate solution $\phi_k(\boldsymbol{x})$ is expressed as $\phi_k(\boldsymbol{x}) \approx \sum_i N_i(\boldsymbol{x})\,\phi_{k(i)}$ [21], where the subscript i represents the node number of the grids.

3.3. Gradient of the advected function

Using the expression in Eq. (10), it is straightforward to obtain the gradient of ϕ_k ($\boldsymbol{\Psi}_k = \nabla\phi_k$) by employing the chain rule:

$$\boldsymbol{\Psi}_k\left(\boldsymbol{x}, t=T_{\mathrm{R}}^{[k]}\right) = \nabla\boldsymbol{\xi}_{k-1}\left(\boldsymbol{x}, t=T_{\mathrm{R}}^{[k]}\right) \cdot \boldsymbol{\Psi}_{k-1}\left(\boldsymbol{\xi}_{k-1}\left(\boldsymbol{x}, t=T_{\mathrm{R}}^{[k]}\right)\right). \tag{11}$$

Here, recall that $\nabla\boldsymbol{\xi}_{k-1}$ is obtained as a preliminary step for getting $\boldsymbol{\xi}_{k-1}$ by solving the equations corresponding to Eq. (7) with the use of the Runge–Kutta method (see Section 2.4), and $\boldsymbol{\Psi}_{k-1}$ can be calculated in the same procedure as for ϕ_k in Eq. (10) (i.e., we can employ any interpolation function of our choice). Once we obtain $\boldsymbol{\Psi}_k$ over the calculation domain, the normal vectors on the contours of ϕ_k are easily calculated by $\boldsymbol{n} = \boldsymbol{\Psi}_k / |\boldsymbol{\Psi}_k|$.

4. Numerical examples

In this section, we illustrate the performance of the hierarchical-gradient truncation and remapping method using four test problems. For the 2D cases, we represent the initial profiles by a signed distance function, $|\nabla\phi(\boldsymbol{x})| = 1$, and track the zero-contour ($\phi = 0$) as conducted by a level set method. For all test cases we need a mapping function, since the initial profile has corners (Section 4.1), or the time evolution yields large deformation of the initial profiles (Sections 4.2–4.4). Here, we apply the schemes for $n = 2$ (see Examples in Section 2.2 and Appendix).

4.1. Rotation of a Zalesak's circle

First, we consider the rigid body rotation of a Zalesak's circle [22] in a constant vorticity field. On the domain $[0, 1] \times [0, 1]$, we consider initial data that describes a slotted circle, centered at $(x, y) = (0.5, 0.75)$ with a radius 0.15, a slot width of 0.05, and a slot length of 0.25. The velocity field $\boldsymbol{u} = (u, v)$ is given by

$$\begin{cases} u = -\omega(y - 0.5) \\ v = \omega(x - 0.5). \end{cases}$$

Here, the angular velocity ω is fixed at 1, so that the circle completes one revolution in a time interval $0 \le t \le 2\pi$. Due to the rigid body motion, we do not need to apply remapping to this example, since $|\nabla\boldsymbol{\xi}(\boldsymbol{x}, t)|$ is constant at any time. A uniform mesh including 101×101 grid points is used, and the time step is $\Delta t = 0.01$.

Fig. 1 shows the zero-contour of the solution at four particular times in one revolution. It is observed that the circle is quite accurately rotated without any deformation. Fig. 2 shows the comparison of the solution after 100 revolutions with the initial shape, i.e., the exact solution (filled with color). One can observe that the numerical result perfectly agrees with the exact solution when the initial profile of ϕ is provided by an equation. In fact, the L^∞-norm defined by $L^\infty \equiv \max_j \left|\hat{\phi}_j - \phi_j\right|$ is calculated at 5.3×10^{-4}, where the subscript j denotes the grid nodes, and $\hat{\phi}_j$, ϕ_j are the exact and numerical solutions, respectively.

As is evident from our derivation of the ODEs in Section 2, this example (or any rigid body motion) can actually be calculated using the scheme for $n = 1$ without loss of accuracy, since $\nabla^{(2)}\boldsymbol{\xi} = \boldsymbol{0}$ at any spatial point and time. The only approximation here is the time-integration described in Section 2.4. Further, although not shown in this paper, but relating to the above fact, our scheme is completely meshless for a rigid body motion as far as the condition $\left|\boldsymbol{\Psi}^{(\infty)}\right| \to 0$ is satisfied, since no spatial interpolation is required in this case.

4.2. 1D deformation field

The capability of the proposed scheme is next assessed by solving the 1D advection problem with deformation. On the domain $[0, 1]$, we consider the initial profile ϕ_0 and the velocity field u given by

$$\phi_0 = \exp\left[-\frac{(x - x_0)^2}{2\sigma^2}\right],$$

$$u = \sin(2\pi x),$$

with $x_0 = 0.5$ and $\sigma = 4$. As is obvious from the direction of velocity, the initial profile ϕ_0 (or the mapping function ξ) deforms with time, and thus, one expects that the remapping technique described in Section 3.2 is inevitable in this example. For the interpolation mentioned in the last sentence in Section 3.2, we apply the piecewise quadratic interpolation functions, which are

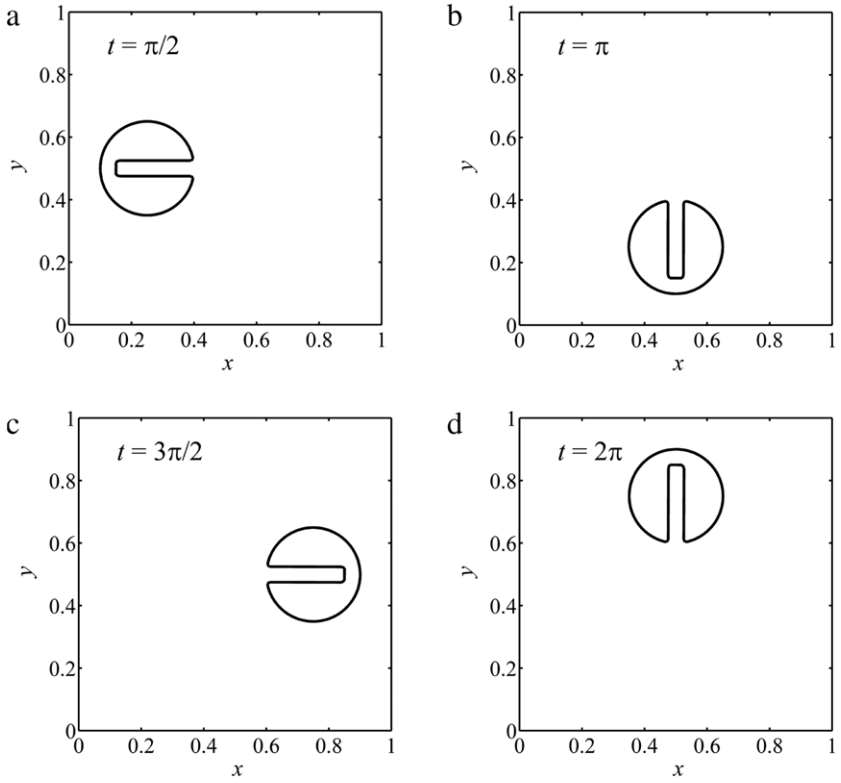

Fig. 1. Profiles of a Zalesak's circle at $t = \pi/2$ (a), $t = \pi$ (b), $t = 3\pi/2$ (c), and $t = 2\pi$ (d).

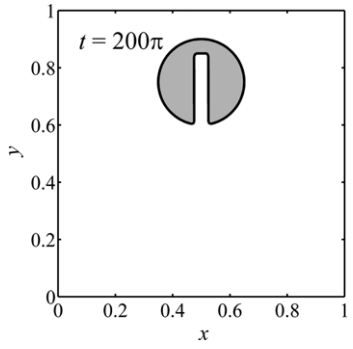

Fig. 2. Comparison of the profile of a Zalesak's circle at $t = 200\pi$ with the exact solution.

defined in each 3-node element with the coordinate value g based on the normalized coordinate system $(-1 \leq g \leq 1)$ as follows:

$$N_\alpha(g) = \frac{g_\alpha g}{2}(1 + g_\alpha g) + \left(1 - g_\alpha^2\right)\left(1 - g^2\right), \tag{12}$$

where the subscript α denotes the local node number, and the values of g_α are $g_{1,2,3} = -1, 1, 0$, respectively.

Fig. 3 shows the comparison of the solutions at $t = 0.4$ obtained by the finite element method (FEM) with piecewise quadratic interpolations, the H-GTaR method with and without remapping. Here, uniform meshes including 401 and 201 grid points are used for the finite element and H-GTaR methods, respectively. The time step is fixed at $\Delta t = 1 \times 10^{-3}$. Due to the temporally constant velocity field, the remapping is executed at the same interval of time; here the number of remapping is set at 7 for the time range $0 \leq t \leq 0.4$, so that the number of time marching for each evolution of ξ is 50. As shown, two numerical solutions obtained by the finite element and H-GTaR methods with remapping are in good agreement. However, in contrast, it is observed that the solution accuracy is largely spoiled if the remapping technique is not applied to this deformation problem.

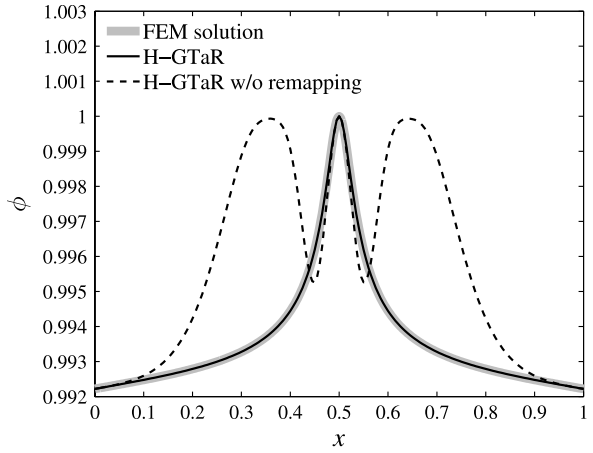

Fig. 3. Comparison of the numerical solutions at $t = 0.4$ obtained by the finite element and H-GTaR methods with and without remapping.

4.3. 2D deformation field for a closed curve

As a 2D deformation example, we consider the swirling flow in a square domain. This test is often times referred to as "vortex in a box" [23,24] (but we neglect the time dependence of the velocity field here). On the domain $[0, 1] \times [0, 1]$, we consider the velocity field

$$\boldsymbol{u}(x, y) = \left(\varphi_y, -\varphi_x\right),$$

given by the stream function

$$\varphi(x, y) = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y).$$

Clearly, this velocity field yields large deformation of the initial profile ϕ_0, and thus, one can assess the robustness of the proposed

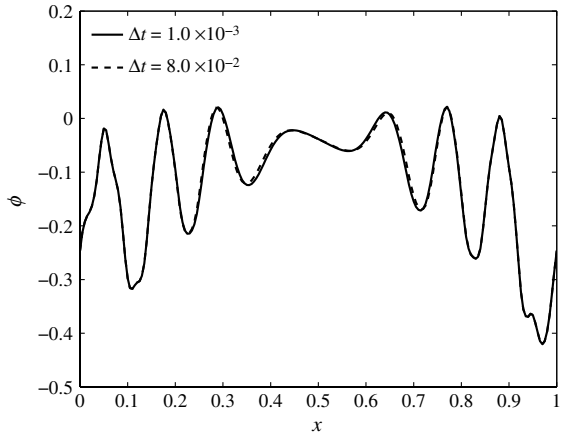

Fig. 4. Comparison of the numerical solutions at $t = 3.2$ obtained with different time steps.

Table 1
Four cases with nearly constant calculation costs.

	Case 1	Case 2	Case 3	Case 4
Grid points	51×51	101×101	201×201	401×401
Time step (Δt)	0.001	0.004	0.016	0.064
Remapping time ($T_{\mathrm{R}}^{[k]}$)	$0.16k$	$0.16k$	$0.16k$	$0.128k$
Calculation time ($t = 3.2$) (s)	2.0	2.9	3.3	5.5

method through this example. The initial condition considered here is given by

$$\phi_0(x, y) = \exp\left(-|\boldsymbol{x} - \boldsymbol{x}_0|^2\right) - \exp\left(-r_0^2\right),$$

with $\boldsymbol{x}_0 = (0.5, 0.75)$ and $r_0 = 0.15$.

Fig. 4 shows the comparison of two solution profiles along the centerline ($y = 0.5$) at $t = 3.2$ obtained by the H-GTaR method using piecewise biquadratic interpolation functions. Here we compare the results using different time steps: $\Delta t = 1 \times 10^{-3}, 8 \times 10^{-2}$. A uniform mesh including 201×201 grid points is used, and the remapping times are chosen at $T_{\mathrm{R}}^{[k]} = 1.6k \times 10^{-1}$. As expected, both profiles are in good agreement, which indicates that the proposed method is rather insensitive to the magnitude of a time step. Hereafter, we show the results obtained by simply applying the piecewise biquadratic interpolation functions into the H-GTaR scheme, but, of course, there are other interpolation methods which should contribute to accuracy improvement of our approach (see the references listed in the introduction).

In closing, we compare the solution accuracy of four numerical results, which are obtained by the H-GTaR method, at $t = 3.2$ with the quasi-exact solution given by the front-tracking approach in Fig. 5. The calculation conditions and corresponding calculation time are tabulated in Table 1. Notice that the four cases require nearly constant calculation costs; a difference in cost is made when a temporal solution is calculated after each remapping time, due to the difference in number of grid points. It is observed that the solution accuracy greatly improves as the number of grid points increases, with only slight increase in calculation time. Here, the calculation time in Table 1 denotes the elapsed real time on a single core of the Loki cluster[1] at the Plasma Science and Fusion Center (PSFC) in MIT.

[1] This is a 75 compute-node cluster with 2 Opteron 2352 processors on each node, each having 8 cores for a total of 600 cores connected by an infiniband network.

4.4. 2D deformation field for an open curve

Finally, we consider a deformation problem for an open curve on the domain $[0, 1] \times [0, 1]$. Here we solve for two different advected functions ϕ_1 and ϕ_2, whose initial profiles are, respectively, given by

$$\phi_{10} = y - 0.5,$$
$$\phi_{20} = -(x - 0.15)(x - 0.5),$$

and the open curve is defined in a manner such that $\phi_1 = 0$ and $\phi_2 \geq 0$. Note that it is not necessary to solve two independent advection equations since both ϕ_1 and ϕ_2 retain a one-to-one relationship with the mapping function $\boldsymbol{\xi}$; consequently, just solving Eq. (9) suffices, and then interpolate ϕ_1 and ϕ_2 independently according to Eq. (10). Initially, the open curve is evolved by the same velocity field used in Section 4.3, and then it is assumed that the velocity is abruptly turned in the opposite direction at $t = T$. Thus, the open curve should return to its initial profile at $t = 2T$ if the numerical scheme solves this problem accurately. A uniform mesh including 401×401 grid points is used, and the time step is $\Delta t = 1 \times 10^{-3}$. The remapping times are chosen at $T_{\mathrm{R}}^{[k]} = 2k \times 10^{-2}$, each of which is smaller than that used in the previous problem considering that we also aim at calculating $\nabla\phi_1$ according to Eq. (11) (note that $\nabla\boldsymbol{\xi}_{k-1}$ is more susceptible to the error caused by the hierarchical-gradient truncation than $\boldsymbol{\xi}_{k-1}$).

Fig. 6 shows the comparison of two solution profiles at $t = T, 2T$ for $T = 3.2$ together with the initial profile ($t = 0$). In each figure, the solid black curve shows the numerical result ($\phi_1 = 0$) obtained by the H-GTaR method, and the gray thick curve corresponds to the exact solution (at $t = 2T$) or the quasi-exact solution (at $t = T$) given by the front-tracking approach. Also, the black arrows indicate the normal vectors on the open curve, which are calculated using the numerical solution of $\nabla\phi_1$ (see Section 3.3; here the normal vectors in Fig. 6(a) are exact). As seen, the calculated open curve profiles are in good agreement with the (quasi-)exact solutions, and the normal vectors point in the correct directions.

5. Conclusions

In this paper we presented a new numerical scheme for the level set equation, which is applicable to any moving interface problem as long as the velocity is sufficiently smooth. The basic strategy of our scheme is as follows. First, we need to assess whether the problem considered possesses nonzero infinite-order spatial derivative values (including spatial discontinuity) and/or time-dependent deformation of the advected function. If that is the case, we convert the original advection equation to an equivalent vector form with a mapping function, in order to ensure that differentiation of the advected function vanishes at a certain finite order at any spatial point. Then we can construct hierarchical equations by taking successive gradients of either the original or vector-form advection equation, and truncate the derivative value of the advected function at some level. By manipulating these hierarchical equations, we obtain a set of decoupled linear ODEs with constant coefficients, which are solved at any spatial point.

The accuracy of our scheme is ensured as far as the truncated derivative value is negligibly small. Whenever this approximation becomes less satisfied in the process of time evolution, we can reconstruct the one-to-one relationship between the original advected quantity and the mapping function. In this remapping procedure we require spatial interpolation, and thus a highly accurate interpolation scheme and fine mesh are desirable. On the other hand, for a rigid body motion in which our truncation approximation is valid all the time without the necessity of

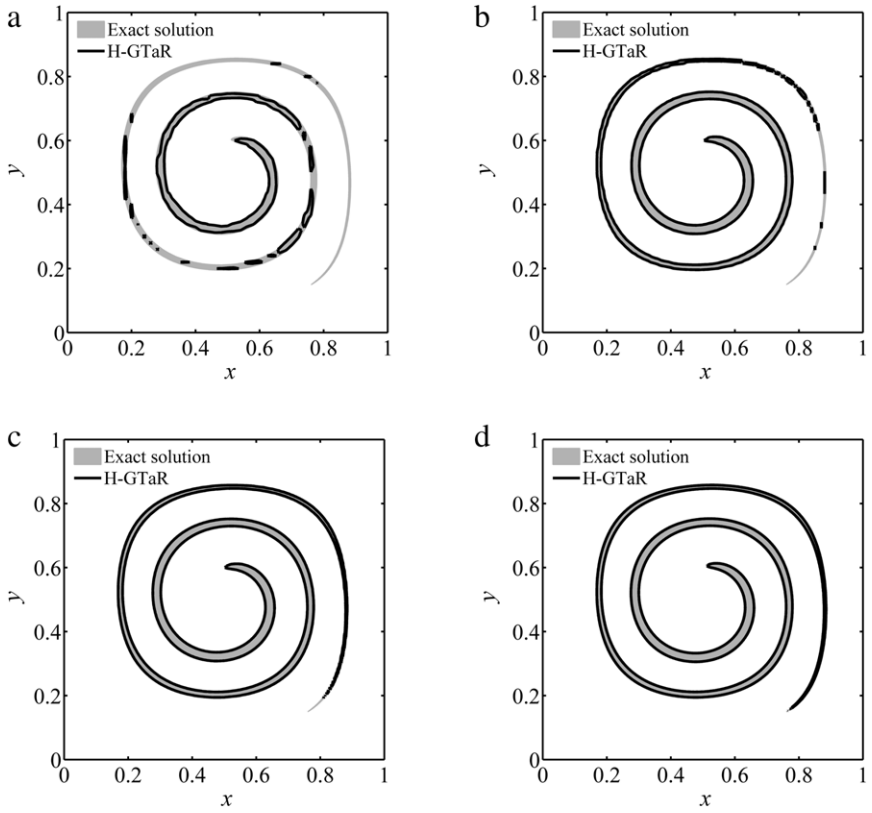

Fig. 5. Comparison of the calculated closed curve profiles with the true solutions at $t = 3.2$ for: (a) 51×51 grid points; (b) 101×101 grid points; (c) 201×201 grid points; and (d) 401×401 grid points.

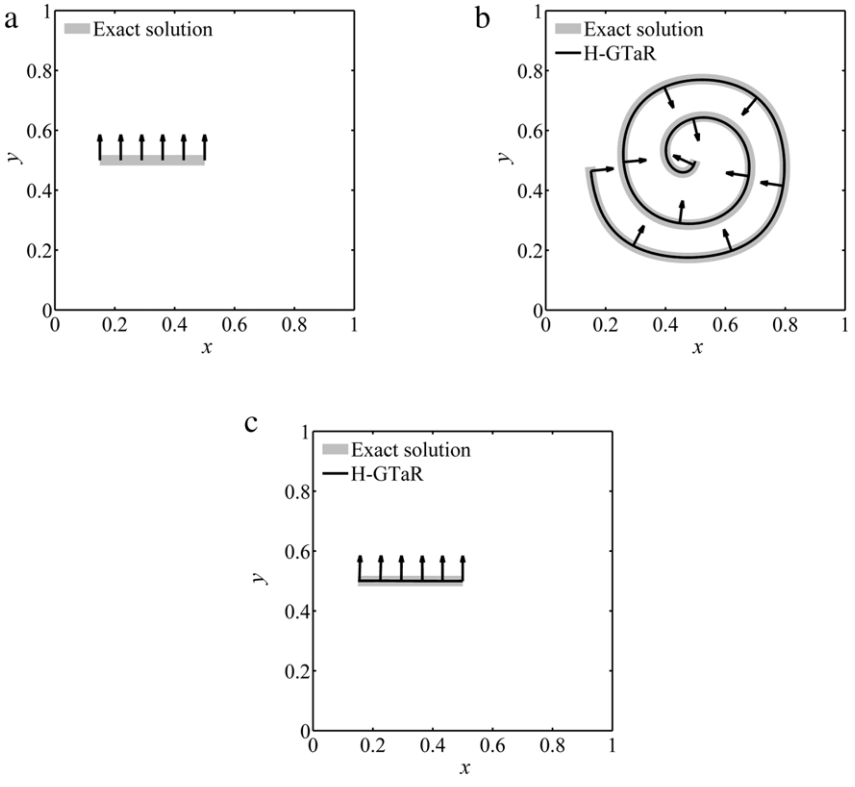

Fig. 6. Comparison of the calculated open curve profiles with the true solutions at: (a) $t = 0$; (b) $t = T$; and (c) $t = 2T$.

remapping, the proposed method yields almost an exact solution with a high order time-integration scheme.

The proposed scheme was tested for one- and two-dimensional problems, and yielded accurate solutions. As in other semi-Lagrangian methods, our scheme is unconditionally stable; one can choose the grid resolution and time step independently without any restriction of the CFL condition. In addition, the hierarchical-gradient truncation procedure yields not only the solution of the advection equation but also its spatial derivative. The gradient of the original advected function is then calculated in the same procedure as for the original advected function when remapping is necessary. Although we did not show in this paper, the given procedure can also be straightforwardly applied to three-dimensional analyses, and the same performance as in present one- and two-dimensional examples can be expected.

Acknowledgments

We are grateful for the financial support of H. Kohno by the U.S. DOE Contracts DE-FG02-91ER54109 and DE-FC02-01ER54648. Also, this work was supported by NSF grant DMS-0813648.

Appendix. Procedure to formulate ODEs for $n = 2$ and $d = 2$

In this appendix, we demonstrate the derivation of ODEs for $n = 2$ and $d = 2$, which are used in our 2D numerical examples. First, the equation for $n = 2$ shown in Example 2 in Section 2 is rewritten in component form as follows:

$$\Psi_{tt}^{i(1)} + 3\frac{\partial u_j}{\partial x_i}\Psi_t^{j(1)} + \left(2\frac{\partial u_j}{\partial x_i}\frac{\partial u_k}{\partial x_j} - u_j\frac{\partial^2 u_k}{\partial x_i \partial x_j}\right)\Psi^{k(1)} = 0, \tag{A.1}$$

where $i, j, k = 1, \ldots, d$ and the Einstein summation convention is applied. For $d = 2$, Eq. (A.1) can be expressed in the following set of equations:

$$\begin{cases} \Psi_{tt}^{x(1)} + C_1\Psi_t^{x(1)} + C_2\Psi_t^{y(1)} + C_3\Psi^{x(1)} + C_4\Psi^{y(1)} = 0 \\ \Psi_{tt}^{y(1)} + D_1\Psi_t^{x(1)} + D_2\Psi_t^{y(1)} + D_3\Psi^{x(1)} + D_4\Psi^{y(1)} = 0, \end{cases} \tag{A.2}$$

where for $\boldsymbol{\Psi}^{(1)} = \left(\Psi^{x(1)}, \Psi^{y(1)}\right)$ and $\boldsymbol{u} = (u, v)$

$$\begin{aligned} C_1 &= 3u_x, \\ C_2 &= 3v_x, \\ C_3 &= 2\left(u_x^2 + v_x u_y\right) - uu_{xx} - vu_{xy}, \\ C_4 &= 2\left(u_x v_x + v_x v_y\right) - uv_{xx} - vv_{xy}, \\ D_1 &= 3u_y, \\ D_2 &= 3v_y, \\ D_3 &= 2\left(u_x u_y + u_y v_y\right) - uu_{xy} - vu_{yy}, \\ D_4 &= 2\left(v_x u_y + v_{yy}^2\right) - uv_{xy} - vv_{yy}. \end{aligned} \tag{A.3}$$

In order to derive the ODE for $\Psi^{x(1)}$, we additionally require the following three equations:

$$\begin{cases} \Psi_{ttt}^{x(1)} + C_1\Psi_{tt}^{x(1)} + C_2\Psi_{tt}^{y(1)} + C_3\Psi_t^{x(1)} + C_4\Psi_t^{y(1)} = 0 \\ \Psi_{ttt}^{y(1)} + D_1\Psi_{tt}^{x(1)} + D_2\Psi_{tt}^{y(1)} + D_3\Psi_t^{x(1)} + D_4\Psi_t^{y(1)} = 0 \\ \Psi_{tttt}^{x(1)} + C_1\Psi_{ttt}^{x(1)} + C_2\Psi_{ttt}^{y(1)} + C_3\Psi_{tt}^{x(1)} + C_4\Psi_{tt}^{y(1)} = 0, \end{cases} \tag{A.4}$$

which are readily obtained by differentiating Eq. (A.2) with respect to t. By manipulating Eqs. (A.2) and (A.4), we can obtain the ODE with respect to $\Psi^{x(1)}$. The detailed procedure is described as follows.

First, substituting the second equation in Eq. (A.2) into the first equation in Eq. (A.4) yields

$$\Psi_{ttt}^{x(1)} + C_1\Psi_{tt}^{x(1)} + E_1^x\Psi_t^{x(1)} + E_2^x\Psi_t^{y(1)} + E_3^x\Psi^{x(1)} + E_4^x\Psi^{y(1)} = 0, \tag{A.5}$$

where

$$E_1^x = -C_2D_1 + C_3, \quad E_2^x = -C_2D_2 + C_4, \quad E_3^x = -C_2D_3, \quad E_4^x = -C_2D_4.$$

Then $\Psi_t^{y(1)}$ can be eliminated using the first equation in Eq. (A.2) and Eq. (A.5). The result is

$$C_2\Psi_{ttt}^{x(1)} + F_1^x\Psi_{tt}^{x(1)} + F_2^x\Psi_t^{x(1)} + F_3^x\Psi^{x(1)} + F_4^x\Psi^{y(1)} = 0, \tag{A.6}$$

where

$$F_1^x = C_1C_2 - E_2^x, \quad F_2^x = C_2E_1^x - C_1E_2^x, \quad F_3^x = C_2E_3^x - C_3E_2^x, \quad F_4^x = C_2E_4^x - C_4E_2^x.$$

Next, substituting the second equations in Eqs. (A.2) and (A.4) into the third equation in Eq. (A.4) gives

$$\Psi_{tttt}^{x(1)} + C_1\Psi_{ttt}^{x(1)} + E_1^x\Psi_{tt}^{x(1)} + G_1^x\Psi_t^{x(1)} + G_2^x\Psi_t^{y(1)} + G_3^x\Psi^{x(1)} + G_4^x\Psi^{y(1)} = 0, \tag{A.7}$$

where

$$G_1^x = -D_1E_2^x + E_3^x, \quad G_2^x = -D_2E_2^x + E_4^x, \quad G_3^x = -D_3E_2^x, \quad G_4^x = -D_4E_2^x.$$

After eliminating $\Psi_t^{y(1)}$ with the first equation in Eq. (A.2) and Eq. (A.7), one gets

$$C_2\Psi_{tttt}^{x(1)} + H_1^x\Psi_{ttt}^{x(1)} + H_2^x\Psi_{tt}^{x(1)} + H_3^x\Psi_t^{x(1)} + H_4^x\Psi^{x(1)} + H_5^x\Psi^{y(1)} = 0, \tag{A.8}$$

where

$$H_1^x = C_1C_2, \quad H_2^x = C_2E_1^x - G_2^x, \quad H_3^x = C_2G_1^x - C_1G_2^x, \quad H_4^x = C_2G_3^x - C_3G_2^x, \quad H_5^x = C_2G_4^x - C_4G_2^x.$$

Finally, the desired ODE with respect to $\Psi^{x(1)}$ is derived from Eqs. (A.6) and (A.8) by eliminating $\Psi^{y(1)}$ as follows:

$$\sum_{i=1}^{5} I_i^x\Psi_{(5-i)t}^{x(1)} = 0, \tag{A.9}$$

where

$$I_1^x = C_2F_4^x, \quad I_2^x = F_4^xH_1^x - C_2H_5^x, \quad I_3^x = F_4^xH_2^x - F_1^xH_5^x, \quad I_4^x = F_4^xH_3^x - F_2^xH_5^x, \quad I_5^x = F_4^xH_4^x - F_3^xH_5^x.$$

Here, note that $I_1^x, \ldots, I_5^x$ are constants at each grid node. The ODE with respect to $\Psi^{y(1)}$ (i.e., the coefficients $I_i^{y(1)}$) can also be derived in the same way.

When we apply the remapping technique described in Section 3, the variables $\Psi^{x(1)}$ and $\Psi^{y(1)}$ are replaced with the derivatives of the components of $\boldsymbol{\xi} = (\xi, \eta)$ so that we need to solve the following four ODEs at each grid node:

$$\begin{aligned} &\sum_{i=1}^{5} I_i^x\Xi_{(5-i)t}^{x(1)} = 0, \quad \sum_{i=1}^{5} I_i^y\Xi_{(5-i)t}^{y(1)} = 0, \\ &\sum_{i=1}^{5} I_i^x\Gamma_{(5-i)t}^{x(1)} = 0, \quad \sum_{i=1}^{5} I_i^y\Gamma_{(5-i)t}^{y(1)} = 0, \end{aligned} \tag{A.10}$$

where $\nabla\xi = \left(\Xi^{x(1)}, \Xi^{y(1)}\right)$ and $\nabla\eta = \left(\Gamma^{x(1)}, \Gamma^{y(1)}\right)$.

References

[1] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[2] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1994) 146–159.
[3] J.A. Sethian, A fast marching level set method for monotonically advancing fronts, Proc. Natl. Acad Sci. USA 93 (1996) 1591–1595.
[4] M. Sussman, E. Fatemi, An efficient, interface preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow, SIAM J. Sci. Comput. 20 (1999) 1165–1191.
[5] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (2002) 83–116.
[6] H. Kohno, T. Tanahashi, Numerical analysis of moving interfaces using a level set method coupled with adaptive mesh refinement, Internat. J. Numer. Methods Fluids 45 (2004) 921–944.
[7] F. Gibou, L. Chen, D. Nguyen, S. Banerjee, A level set based sharp interface method for the multiphase incompressible Navier–Stokes equations with phase change, J. Comput. Phys. 222 (2007) 536–555.
[8] C.H. Min, F. Gibou, A second order accurate level set method on non-graded adaptive cartesian grids, J. Comput. Phys. 225 (2007) 300–321.
[9] H. Takewaki, A. Nishiguchi, T. Yabe, Cubic interpolated pseudo-particle method (CIP) for solving hyperbolic-type equations, J. Comput. Phys. 61 (1985) 261–268.
[10] X.D. Liu, S. Osher, T. Chan, Weighted essentially non-oscillatory schemes, J. Comput. Phys. 115 (1994) 200–212.
[11] T. Yabe, F. Xiao, T. Utsumi, The constrained interpolation profile method for multiphase analysis, J. Comput. Phys. 169 (2001) 556–593.
[12] T. Himeno, T. Watanabe, A. Konno, Numerical analysis for propellant management in rocket tanks, AIAA J. Propulsion Power 21 (2005) 76–86.
[13] T. Aoki, Interpolated differential operator (IDO) scheme for solving partial differential equations, Comput. Phys. Comm. 102 (1997) 132–146.
[14] J.C. Nave, R.R. Rosales, B. Seibold, A gradient-augmented level set method with an optimally local, coherent advection scheme, J. Comput. Phys. 229 (2010) 3802–3827.
[15] C.Z. Cheng, G. Knorr, The integration of the vlasov equation in configuration space, J. Comput. Phys. 22 (1976) 330–351.
[16] A. Staniforth, J. Côté, Semi-Lagrangian integration schemes for atmospheric models — a review, Mon. Weather Rev. 119 (1991) 2206–2223.
[17] E. Sonnendrücker, J. Roche, P. Bertrand, A. Ghizzo, The semi-Lagrangian method for the numerical resolution of the vlasov equation, J. Comput. Phys. 149 (1999) 201–220.
[18] R.D. Nair, J.S. Scroggs, F.H.M. Semazzi, A forward-trajectory global semi-Lagrangian transport scheme, J. Comput. Phys. 190 (2003) 275–294.
[19] J.M. Qiu, A. Christlieb, A conservative high order semi-Lagrangian WENO method for the Vlasov equation, J. Comput. Phys. 229 (2010) 1130–1149.
[20] C.W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (1988) 439–471.
[21] K.J. Bathe, Finite Element Procedures, Prentice-Hall, 1996.
[22] S. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys. 31 (1979) 335–362.
[23] J. Bell, P. Colella, H. Glaz, A second-order projection method for the incompressible Navier–Stokes equations, J. Comput. Phys. 85 (1989) 257–283.
[24] R. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM J. Numer. Anal. 33 (1996) 627–665.