

Noname manuscript No.
(will be inserted by the editor)

An Alternating Direction Implicit Method for Mean Curvature Flows

Han Zhou · Shuwang Li · Wenjun Ying

Received: date / Accepted: date

Abstract This paper is concerned with the mean curvature flow, which describes the dynamics of a hypersurface whose normal velocity is determined by local mean curvature. We present a Cartesian grid-based method for solving mean curvature flows in two and three space dimensions. The present method embeds a closed hypersurface into a fixed Cartesian grid and decomposes it into multiple overlapping subsets. For each subset, extra tangential velocities are introduced such that marker points on the hypersurface only moves along grid lines. By utilizing an alternating direction implicit (ADI)-type time integration method, the subsets are evolved alternately by solving scalar parabolic partial differential equations on planar domains. The method removes the stiffness using a semi-implicit scheme and has no high-order stability constraint on time step size. Numerical examples in two and three space dimensions are presented to validate the proposed method.

Keywords Mean curvature flow · Cartesian grid · Overlapping surface decomposition · Geometric flows · ADI method

Mathematics Subject Classification (2020) 35K93 · 53E10 · 65N06 · 65M55

H. Zhou
School of Mathematical Sciences, Shanghai Jiao Tong University, Minhang, Shanghai 200240, PR China
E-mail: zhouhan@sjtu.edu.cn

S. Li
Department of Applied Mathematics, Illinois Institute of Technology, Chicago, USA
E-mail: sli15@iit.edu

W. Ying
Corresponding author. School of Mathematical Sciences, MOE-LSC and Institute of Natural Sciences, Shanghai Jiao Tong University, Minhang, Shanghai 200240, PR China
E-mail: wying@sjtu.edu.cn

1 Introduction

Geometric evolution of interfaces draws many attentions in the last decades due to its wide applications in mathematics [10], materials science [22], biology [23] and, more recently, image processing [19, 33, 20]. In a geometric evolution problem, the dynamics of a hypersurface are described by its geometry. Typically, the normal velocity of the hypersurface is given by a law defined by geometry. In this paper, we are concerned with a representative case of geometric evolution problems, the mean curvature flow, in which the hypersurface evolves such that its normal velocity equals its negative mean curvature. Mean curvature flow was originally proposed by Mullins to model an ideal grain boundary motion [22]. Thereafter, it was also used to model various other physical phenomena [23, 1].

Numerical methods for mean curvature flows can be classified into three categories, based on their different representations of hypersurfaces, which are parametric approaches, level set method [24, 26, 25] and phase field method [8, 9, 27]. Representative parametric approaches include the parametric finite element method [3, 7, 5, 4], graph approach [12, 13] and the front tracking method [18, 31]. For a comprehensive survey on numerical methods for mean curvature flows, the interested reader is referred to the review article by Deckelnick et al. [13].

For general moving interface problems, although the level set method and the phase field method have their advantages in handling topological changes and ease in implementation, parametric approaches provide surprisingly good results, such as accurate computation of curvature and conservation of mass, even with a coarse grid, and are computationally very cheap as well [6, 30]. Despite the benefits, numerical computation of mean curvature flows with parametric approaches also encounters several difficulties, including the deterioration of mesh quality during the computation and the numerical stiffness induced by the mean curvature term. Due to the pure normal motion of the surface, adjacent mesh nodes may become closer and closer, making the computation highly unstable. The problem is even more severe for the mean curvature flow due to its "curve shortening" property by Mullins [22]. In addition, the evolution equation of mean curvature flows has second-order spatial derivatives in the mean curvature term, which induces numerical stiffness such that a small time step is required for explicit time integration schemes [17]. A naive discretization with implicit time integration to remove stiffness leads to a nonlinear system, for which finding a numerical solution is time-consuming, even with advanced iterative solvers. For two space dimensional curves, these difficulties can be very well handled by the small scale decomposition method, initially proposed by Hou et al. [17]. The idea was also extended to three space dimensional cases for some special surfaces [16, 2]. However, for general closed surfaces in three space dimensions, it is still unclear how to apply small scale decomposition for mean curvature flows.

This work proposes a new numerical method for mean curvature flows. The method decomposes a moving hypersurface into multiple overlapping subsets

such that each subset can be viewed as a Monge patch for which the graph approach [12,13] is applicable. The method is based on an overlapping decomposition method for hypersurfaces, initially proposed by Wilson for computing integrals on implicitly defined curves and surfaces [32]. A few years later, the method was extended by the second author to incorporate a kernel-free boundary integral method for solving elliptic partial differential equations on irregular domains [34]. The decomposition strategy has many advantages in simplicity and efficiency. By representing each subset with its intersection points with grid lines, it is natural to keep marker points quasi-equidistant and maintain mesh quality. By reformulating the evolution equation, which is a nonlinear system, into a sequence of scalar PDEs on overlapping subsets, one can evolve the subsets alternately in the spirit of the alternating direction implicit (ADI) method [28,14,15]. The resulting algorithm is efficient and has no high-order stability constraint.

The remainder of the paper is organized as follows. In Section 2, we describe the governing equation of mean curvature flow and its hybrid formulation based on an overlapping surface decomposition method. The numerical methods for solving mean curvature flows are described in Section 3. In Section 4, the numerical algorithm of the proposed method is briefly summarized. Multiple numerical examples are presented to validate the present method in Section 5. In the final Section 6, we briefly discuss the present method and some further work.

2 Mathematical formulation

Let $\Gamma(t) \subset \mathbb{R}^d, d = 2, 3$ be a closed moving hypersurface. Consider the mean curvature flow problem that, for any point \mathbf{x} on Γ , the evolution is given by,

$$\mathbf{x}_t = V\mathbf{n}, \quad V = -\kappa, \quad \mathbf{x} \in \Gamma, \quad (1)$$

where κ is the (mean) curvature and \mathbf{n} the unit outward normal. Here, a circle/sphere has positive curvature. By applying the transport theorem of evolving hypersurfaces, it can be shown that the mean curvature flow has length-decreasing and area-decreasing properties in 2D and 3D, respectively,

$$\frac{d}{dt}|\Gamma(t)| = \frac{d}{dt} \int_{\Gamma(t)} 1 \, ds = - \int_{\Gamma(t)} \kappa^2 \, ds < 0, \quad \text{if } d = 2, \quad (2)$$

$$\frac{d}{dt}|\Gamma(t)| = \frac{d}{dt} \int_{\Gamma(t)} 1 \, dA = - \int_{\Gamma(t)} \kappa^2 \, dA < 0, \quad \text{if } d = 3, \quad (3)$$

where $|\Gamma(t)|$ is the length of $\Gamma(t)$ in 2D and the area of $\Gamma(t)$ in 3D. Specially, for the case $d = 2$, it holds that

$$\frac{d}{dt}A(t) = \int_{\Gamma(t)} \frac{d\mathbf{x}}{dt} \cdot \mathbf{n} \, ds = - \int_{\Gamma(t)} \kappa \, ds = -2\pi, \quad (4)$$

where $A(t)$ is the area enclosed by $\Gamma(t)$. It suggests that the enclosed area of a 2D mean curvature flow always decreases at a constant rate [11]. The result no longer holds for the enclosed volume of 3D surfaces since the surface integral of mean curvature varies from case to case.

If Γ is closed, and there is no boundary condition, then the solution of mean curvature flow is determined by its initial configuration. For some certain initial configurations, solutions of mean curvature flows may develop singularities, such as pinch-off and topological changes, in finite time before shrinking to a point. In this paper, we seek well-defined solutions of mean curvature flows, *i.e.*, solutions before singularities happen.

In order to tackle the mean curvature flow problem, the evolution equation (1) is divided into multiple subproblems with an overlapping surface decomposition of Γ . For $r = 1, \dots, d$, let \mathbf{e}_r be the r^{th} unit vector in \mathbb{R}^d and $\alpha \in (\cos^{-1}(1/\sqrt{d}), \pi/2)$ be a fixed angle. The set

$$\Gamma_r = \{\mathbf{x} \in \Gamma : |\mathbf{n} \cdot \mathbf{e}_r|(\mathbf{x}) > \cos \alpha\}, \quad (5)$$

is an open subset of the surface Γ for each $r = 1, 2, \dots, d$. The union of the sets $\{\Gamma_r\}_{r=1}^d$ forms an overlapping surface decomposition of Γ . Then the surface Γ is represented by the overlapping subsets Γ_r with a partition of unity.

2.1 Divided problems

Note that the evolution of a hypersurface is only determined by its normal velocity V . One is allowed to add arbitrary tangential velocity to the evolution of Γ without altering its shape. The tangent velocity only changes in the frame for the parametrization of the surface. For arbitrary tangential velocities T , T_1 and T_2 , the evolution governed by (1) is equivalent to

$$\mathbf{x}_t = -V\mathbf{n} + T\boldsymbol{\tau}, \quad \mathbf{x} \in \Gamma, \quad (6)$$

in two space dimensions and

$$\mathbf{x}_t = -V\mathbf{n} + T_1\boldsymbol{\tau}_1 + T_2\boldsymbol{\tau}_2, \quad \mathbf{x} \in \Gamma, \quad (7)$$

in three space dimensions. Here, the notations $\boldsymbol{\tau}$, $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ mean tangent vectors.

Consider the evolution of the overlapping subsets Γ_r in three space dimensions. Let $\mathbf{x}^{(r)}$ denote a point on the subset Γ_r . The evolutions of Γ_r are given by

$$\mathbf{x}_t^{(r)} = -V^{(r)}\mathbf{n}, \quad \mathbf{x}^{(r)} \in \Gamma_r, \quad r = 1, \dots, d. \quad (8)$$

where $V^{(r)}$ are the restrictions of V from Γ to Γ_r . By adding tangential velocities $T_1^{(r)}$ and $T_2^{(r)}$, it yields the equivalent evolution equations

$$\mathbf{x}_t^{(r)} = -V^{(r)}\mathbf{n} + T_1^{(r)}\boldsymbol{\tau}_1 + T_2^{(r)}\boldsymbol{\tau}_2, \quad \mathbf{x}^{(r)} \in \Gamma_r, \quad r = 1, \dots, d. \quad (9)$$

Hence, the evolution equation (1) of Γ is divided into a sequence of evolution equations of subsets Γ_r . With the divided formulation (9), for each subset Γ_r , the tangential velocities $T_1^{(r)}$ and $T_2^{(r)}$ can be chosen independently.

Due to the overlapping surface decomposition (5), each subset Γ_r can be easily parameterized with Cartesian coordinates in a planar domain $\Omega_r \subset \mathbb{R}^{d-1}$. For example, in three space dimensions, denote by Ω_3 the projection of Γ_3 onto X - Y plane. Then Γ_3 can be represented by the Monge patch $\mathbf{x}(x, y) = \mathbf{x}(x, y, z(x, y)), (x, y) \in \Omega_3$ in which $z(x, y)$ is a height function. With this understanding, the evolution of Γ_r can be described as a time-dependent height function on the base plane Ω_3 in $d-1$ space dimensions. The height function representation is an Eulerian description of the moving hypersurface. Its numerical approximation is much simpler than that for tracking a moving hypersurface with its Lagrangian motion. With Eulerian description, it is natural to use a Cartesian grid to approximate the height function. With the understanding that the Eulerian description is equivalent to moving marker points of the hypersurface along fixed grid lines, the evolution equations of the equivalent Eulerian motion can be derived by carefully choosing tangent velocities $T^{(r)}$, $T_1^{(r)}$ and $T_2^{(r)}$ such that Γ_r only have one non-zero velocity component in direction \mathbf{e}_r .

2.1.1 Two space dimensional case

Let $\Gamma \subset \mathbb{R}^2$ be a time-dependent Jordan curve which is given by $\mathbf{x}(t) = (x(\theta, t), y(\theta, t))$ where θ parameterizes the curve. Its curvature κ and unit outward normal vector are, respectively, given by

$$\kappa = \frac{x_\theta y_{\theta\theta} - x_{\theta\theta} y_\theta}{(x_\theta^2 + y_\theta^2)^{\frac{3}{2}}}, \quad \mathbf{n} = \frac{1}{(x_\theta^2 + y_\theta^2)^{\frac{1}{2}}} \begin{pmatrix} y_\theta \\ -x_\theta \end{pmatrix}. \quad (10)$$

For each subset $\Gamma_r, r = 1, 2$, it can be parameterized by x or y to be a height function $y = y(x)$ or $x = x(y)$ depending on its orientation. Suppose Γ_r is represented in the form $\eta = \eta(\xi)$ where (ξ, η) coincides with (x, y) or (y, x) . After extra tangential velocity T is added into the original evolution equation (1), the evolution of Γ_r is equivalent to

$$\frac{d}{dt} \begin{pmatrix} \xi \\ \eta \end{pmatrix} = - \frac{\eta_{\xi\xi}}{(\eta_\xi^2 + 1)^2} \begin{pmatrix} \eta_\xi \\ -1 \end{pmatrix} + T \begin{pmatrix} 1 \\ \eta_\xi \end{pmatrix}. \quad (11)$$

It is worth mentioning that equation (11) does not rely on the orientation of the curve due to the cancellation of signs in the curvature and normal vector when one reverses the parameterization, namely, from ξ to $-\xi$.

To determine a specific tangential velocity T such that marker points on Γ_r only have non-zero velocity component in \mathbf{e}_r direction. One needs to set $\xi_t = 0$, *i.e.*

$$\xi_t = - \frac{\eta_\xi \eta_{\xi\xi}}{(\eta_\xi^2 + 1)^2} + T = 0. \quad (12)$$

The expression of T can be easily solved. By substituting the determined T into (11), it yields the evolution law for Γ_r in terms of height function,

$$\eta_t = \frac{\eta_{\xi\xi}}{\eta_{\xi}^2 + 1}, \quad (13)$$

which is a scalar parabolic-type partial differential equation.

2.1.2 Three space dimensional case

The derivation in three space dimensions is similar. For each subset $\Gamma_r, r = 1, 2, 3$, it can be regarded as a Monge patch $\mathbf{x}(u, v) = \mathbf{x}(u, v, w(u, v)), (u, v) \in \Omega_r$ where Ω_r is the projection of Γ_r on its base plane and $w(u, v)$ is the height function. Denote by $\boldsymbol{\tau}_1 = (1, 0, w_u)^T, \boldsymbol{\tau}_2 = (0, 1, w_v)^T$ two tangent vectors of Γ_r . After adding two tangential velocities T_1 and T_2 , the evolution equation (1) in three space dimensions becomes

$$\frac{d}{dt} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \frac{(1 + w_u^2)w_{vv} - 2w_u w_v w_{uv} + (1 + w_v^2)w_{uu}}{2(1 + w_u^2 + w_v^2)^2} \begin{pmatrix} -w_u \\ -w_v \\ 1 \end{pmatrix} + T_1 \begin{pmatrix} 1 \\ 0 \\ w_u \end{pmatrix} + T_2 \begin{pmatrix} 0 \\ 1 \\ w_v \end{pmatrix}. \quad (14)$$

By setting $u_t = v_t = 0$, one can solve for T_1 and T_2 ,

$$T_1 = w_u \frac{(1 + w_u^2)w_{vv} - 2w_u w_v w_{uv} + (1 + w_v^2)w_{uu}}{2(1 + w_u^2 + w_v^2)^2}, \quad (15)$$

$$T_2 = w_v \frac{(1 + w_u^2)w_{vv} - 2w_u w_v w_{uv} + (1 + w_v^2)w_{uu}}{2(1 + w_u^2 + w_v^2)^2}. \quad (16)$$

By substituting (15) and (16) into (14), it gives the evolution of Γ_r in terms of its height functions w ,

$$w_t = \frac{(1 + w_u^2)w_{vv} - 2w_u w_v w_{uv} + (1 + w_v^2)w_{uu}}{2(1 + w_u^2 + w_v^2)}. \quad (17)$$

The equation (17) is also a scalar parabolic-type partial differential equation.

2.2 Matching condition

Until now, the original evolution equation (1) is reformulated as a sequence of scalar partial differential equations (13) and (17). To ensure the well-posedness of the divided problem, we follow the idea of domain decomposition [21] to add an extra matching condition for the solution of the divided problem at the overlapping zone such that the equations (13) and (17) have boundary condition.

A simple choice of the matching condition is to enforce continuity of the global solution at an overlapping zone with a partition of unity,

$$\mathbf{x}^{(r)} = \sum_{j \neq r} \chi_j \mathbf{x}^{(j)}, \quad \mathbf{x}^{(r)} \in \partial \Gamma_r. \quad (18)$$

where $\partial\Gamma_r$ denotes the boundary of Γ_r . Here, the notation χ_j is the partition of unity subordinate to the subset Γ_j , which satisfies

$$\begin{cases} \chi_r(\mathbf{x}) \geq 0, & \mathbf{x} \in \Gamma_r, \\ \chi_r(\mathbf{x}) = 0, & \mathbf{x} \in \Gamma \setminus \Gamma_r, \\ \sum_{r=1}^d \chi_r(\mathbf{x}) = 1, & \mathbf{x} \in \Gamma. \end{cases} \quad (19)$$

The divided problem (9) together with the matching condition (18) forms an equivalent coupled system, which is called hybrid formulation, to equation (1) in three space dimensions,

$$\begin{cases} \mathbf{x}_t^{(r)} = -V^{(r)}\mathbf{n} + T_1^{(r)}\boldsymbol{\tau}_1 + T_2^{(r)}\boldsymbol{\tau}_2, & \mathbf{x}^{(r)} \in \Gamma_r, \\ \mathbf{x}^{(r)} = \sum_{j \neq r} \chi_j \mathbf{x}^{(j)}, & \mathbf{x}^{(j)} \in \partial\Gamma_r. \end{cases} \quad (20)$$

Once the hybrid formulation (20) is solved, the global solution \mathbf{x} can be reconstructed with a partition of unity,

$$\mathbf{x} = \sum_{r=1}^d \chi_r \mathbf{x}^{(r)}. \quad (21)$$

Unlike Ambrose's method [2], which is applicable only for a particular class of surfaces with doubly-periodic boundary conditions, our method can handle more general cases, including closed surfaces, with this hybrid formulation.

3 Numerical Methods

In this section, the numerical methods for mean curvature flow are described, including the discrete representation of a moving hypersurface, numerical discretizations of the partial differential equations (13) and (17) as well as the matching condition (18).

3.1 Hypersurface representation

Let Γ be a smooth closed hypersurface. It is separately represented by the height functions of its overlapping subsets $\Gamma_r, r = 1, \dots, d$, which are approximated by nodal values at Cartesian grid nodes in the base domain Ω_r . Equivalently, the nodal values are, in fact, the intersection points of Γ_r and grid lines, which are aligned with \mathbf{e}_r . At the implementation level, this is done by selecting from all intersection points \mathbf{p} of Γ and grid lines for certain ones which satisfy the decomposition rule:

$$\mathbf{p} \in \Gamma_r \text{ and } \mathbf{n}(\mathbf{p}) \cdot \mathbf{e}_r > \cos \alpha. \quad (22)$$

where $\mathbf{n}(\mathbf{p})$ is the unit outward normal at \mathbf{p} and α is a given threshold angle. Those points which satisfy (22) are named as control points for Γ_r and the point set is denoted by Γ_r^h . We also denote all control points on Γ by $\Gamma^h = \cup_{r=1}^d \Gamma_r^h$. The point set Γ^h is used to represent Γ in terms of its overlapping subsets. We remark that, although points in Γ^h are not quasi-equidistant, local interpolation stencil only involves points in each subset Γ_r^h , which are quasi-equidistant. Figure 1 and 2 show the distribution of control points in two and three space dimensions, respectively.

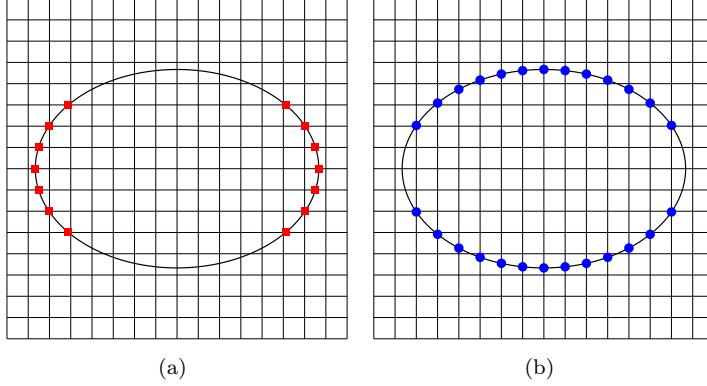


Fig. 1 Control points for the representation of an ellipse: (a) control points on Γ_1 (red rectangle markers); (b) control points on Γ_2 (blue circle markers).

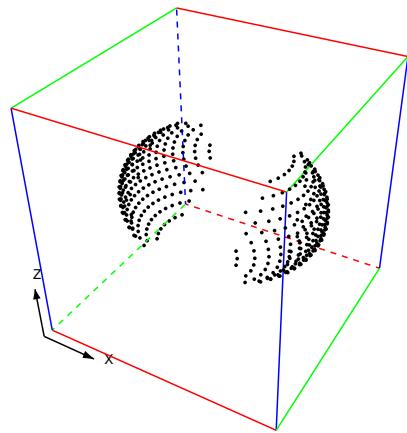


Fig. 2 Control points for the representation of an ellipsoid on Γ_1 .

The advantage of this representation of hypersurface is evident. One can easily find out that the projections of control points in Γ_r^h coincide with Cartesian grid nodes in Ω_r . Instead of tracking Γ by marker points whose velocities are in d dimensions, one needs to solve the evolution for the height functions, *i.e.* marker points moving along grid lines, which only change values in one dimension.

3.1.1 Solving PDEs on hypersurfaces

Generally, for a closed smooth hypersurface Γ , the subset Γ_r consists of several isolated components which are denoted by $\Gamma_{r,l}$, $l = 1, 2, \dots$. For example, suppose $\Gamma : \mathbf{x} = \mathbf{x}(\theta), \theta \in [0, 2\pi]$ is a circle, then the curve segments $\Gamma_{1,1} : \mathbf{x} = \mathbf{x}(\theta), \theta \in (2\pi - \alpha, 2\pi) \cup [0, \alpha]$ and $\Gamma_{1,2} : \mathbf{x} = \mathbf{x}(\theta), \theta \in (\pi - \alpha, \pi + \alpha)$ are both subsets of Γ_1 . Let $\Omega_{r,l}$ denote the projection of $\Gamma_{r,l}$ on the base plane. If $\Omega_{r,l}$ overlap with each other, then Γ_r is a multi-valued function on Ω_r , which induces ambiguity.

The correct understanding is to separate $\Gamma_{r,l}$ from each other, and on which PDEs are solved independently. Once the components are separated from each other, the ambiguity is removed, since $\Gamma_{r,l}$ is a single-valued function on $\Omega_{r,l}$ (see Figure 3). Only points on the same component are involved in a local stencil for solving PDEs. In the implementation, one can check their distance and normals to determine if two points are on the same component $\Gamma_{r,l}$. We identify two points \mathbf{p} and \mathbf{q} on the same component if they satisfy

$$\|\mathbf{p} - \mathbf{q}\| < D_0 \text{ and } \mathbf{n}(\mathbf{p}) \cdot \mathbf{n}(\mathbf{q}) < \cos(\theta_0), \quad (23)$$

where $\|\cdot\|$ is the Euclidean distance, D_0 and θ_0 are threshold values given in advance. In this work, we set $D_0 = 5h$ and $\theta_0 = \pi/6$. The separation procedure is meant to find the correct finite difference stencil point from possible intersection points that lie on the same grid line. The implementation based on the criteria (23) can naturally handle cases with multiple (more than 2) components in each Γ_r , for example, oscillating curves or surfaces, as long as the grid is fine enough to resolve them.

3.1.2 Interpolation on hypersurface

In the discrete representation Γ^h , geometric quantities and functions on the hypersurface are evaluated by local interpolation. Given a point $\mathbf{p} \in \Gamma$, to interpolate the function value at \mathbf{p} using function values on Γ_r^h , one needs to find its projection point $\mathbf{p}^* \in \Omega$ in which the selection of r depends on the direction of $\mathbf{n}(\mathbf{p})$. A quadratic polynomial is locally constructed for interpolation,

$$P_2(u, v) = c_1 + c_2u + c_3v + c_4u^2 + c_5v^2 + c_6uv, \quad (24)$$

where (u, v) is the local coordinate near \mathbf{p}^* in the base plane. With the help of a Cartesian grid, finding interpolation stencils on Γ^h is very simple. One can attach control points to their closest grid nodes and find stencil points by searching nearby grid nodes with their indices in the Cartesian grid.

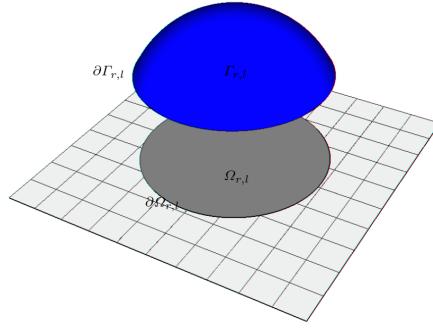


Fig. 3 A surface component $\Gamma_{r,l}$ and its projection on the base plane $\Omega_{r,l}$.

3.1.3 Evolving the hypersurface

After the PDEs (13) and (17) are solved for a time step, control points are moved to different positions and form a new hypersurface. Old points can no longer represent the new hypersurface since they do not satisfy the decomposition rule mentioned before. To represent the new hypersurface, new control points must be added, and some old ones must be deleted. This is achieved by finding out all intersection points by local interpolation and select for new control points. Even in three space dimensions, intersection points can be computed using one-dimensional polynomial interpolations since both stencil points and new intersection points are on the same plane.

Take the surface component $\Gamma_{3,l}$ as an example. The component is discretized by intersection points with coordinates $(x_i, y_j, \eta_{i,j})$ where $(x_i, y_j) \in \Omega_{3,l}^h$. In order to find new intersection points of the component with the grid line $\{(x, y, z) | y = y_j, z = z_k\}$, one first needs to identify the intersection interval (x_{i_0}, x_{i_1}) by check the side of grid nodes, and then choose three intersection points $(x_{i_0-1}, y_j, \eta_{i_0-1,j})$, $(x_{i_0}, y_j, \eta_{i_0,j})$, and $(x_{i_0+1}, y_j, \eta_{i_0+1,j})$ to locally construct a 1D quadratic polynomial $z = P_2(x)$. By solving the equation $P_2(x^*) = z_k$ with either the Newton method or the bisection method, one can obtain the new intersection point (x^*, y_j, z_k) .

After an intersection point is found, following (22), one can determine whether it is kept or deleted by checking the normal vector, which is also evaluated by locally constructing a parabola. It is worth mentioning that if a new component $\Gamma_{r,l}$ is too small and does not have enough control points for interpolation, the whole component should be deleted.

3.2 Discretization of PDEs

3.2.1 Temporal discretization

The equation (13) and (17) are parabolic PDEs and have second-order spatial derivatives. Generally, explicit time integration methods, such as the forward Euler method, for PDEs involving high-order spatial derivatives suffer from high-order stability constraints, and one has to use small time steps to solve the equations. These problems are known as stiff problems. The stiffness for mean curvature flow is rather severe. Take two space dimensional cases as an example. If one discretizes Γ by uniformly partitioning the parameter θ . An explicit time integration suffers from a second order stability constraint $\Delta t \leq C(\min_\theta s_\theta h)^2$ where C is a constant, s is arclength, and h is the grid spacing in θ . Since 2-D mean curvature flow is also known as the "curve shortening flow", $s_\theta h$ decreases with time and results in an even worse situation. Although implicit methods are unconditionally stable for stiff problems, there is another difficulty in implicit time integration for (13) and (17) since it results in a nonlinear system in each time step, for which finding a solution is highly inefficient.

Notice that equations (13) and (17) are quasi-linear equations. The source of stiffness comes from the highest-order terms, which are linear in equations (13) and (17). The stiffness can be removed by only treating the highest order terms implicitly with lower order terms treated explicitly. The resulting time integration scheme is semi-implicit in time. It only requires solving linear systems, which are much more acceptable than nonlinear systems.

Suppose the time interval $[0, T]$ is uniformly partitioned into $0 = t^0 < t^1 < \dots < t^n < \dots < t^N = T$ with $t^{n+1} - t^n = \Delta t$. The semi-implicit schemes for equations (13) and (17), respectively, are give by

$$\frac{\eta^{n+1} - \eta^n}{\Delta t} = \frac{\eta_{\xi\xi}^{n+1}}{(\eta_\xi^n)^2 + 1}, \quad (25)$$

and

$$\frac{w^{n+1} - w^n}{\Delta t} = \frac{(1 + (w_u^n)^2)w_{vv}^{n+1} - 2w_u^n w_v^n w_{uv}^{n+1} + (1 + (w_v^n)^2)w_{uu}^{n+1}}{2(1 + (w_u^n)^2 + (w_v^n)^2)}. \quad (26)$$

The semi-implicit schemes (25) and (26) are only in semi-discrete forms.

3.2.2 Spatial discretization

Generally, physical properties should be encoded into the discretization of spatial derivatives. In the case of mean curvature flow, the evolution of a hypersurface is driven by surface tension, which is diffusive in nature. Since diffusion comes from all directions, it is preferred to adopt central differences for approximating the spatial derivatives in the mean curvature term. Though

tangential velocities terms may introduce the convection effect, which commonly should be discretized with methods for hyperbolic PDEs, we remark that the convection effect is expected to be small compared to the diffusion effect. Hence, for simplicity, we approximated all the spatial derivatives in (13) and (17) with central differences.

Suppose $\Gamma \in \mathbb{R}^d, d = 2, 3$ is embedded into a bounding box \mathcal{B} which is the tensor product of one dimensional intervals $\mathcal{I}_i, i = 1, 2, \dots, d$. If \mathcal{I}_i is uniformly partitioned into a Cartesian grid \mathcal{G}_i for each i , then the tensor product of \mathcal{G}_i forms a natural uniform partition of \mathcal{B} , which is also a Cartesian grid and is denoted by \mathcal{G} . Without loss of generality, we assume equal bounding intervals $\mathcal{I}_i = [a, b], i = 1, 2, \dots, d$, and they are uniformly partitioned into N intervals. Denote by $h = (b - a)/N$ the mesh parameter.

Let Δ_ξ, δ_ξ^2 denote central difference quotients,

$$\Delta_\xi u_i = \frac{u_{i+1} - u_{i-1}}{2h}, \quad \delta_\xi^2 u_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}.$$

The fully discrete form of equation (13) is given by

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\delta_\xi^2 u_i^{n+1}}{(\Delta_\xi u_i^n)^2 + 1}. \quad (27)$$

Similarly, introduce the central difference quotients,

$$\begin{aligned} \Delta_u w_{ij} &= \frac{w_{i+1,j} - w_{i-1,j}}{2h}, & \Delta_v w_{ij} &= \frac{w_{i,j+1} - w_{i,j-1}}{2h}, \\ \delta_{uu}^2 w_{ij} &= \frac{w_{i+1,j} + w_{i-1,j} - 2w_{ij}}{h^2}, & \delta_{vv}^2 w_{ij} &= \frac{w_{i,j+1} + w_{i,j-1} - 2w_{ij}}{h^2}. \end{aligned}$$

The fully discrete form of equation (17) is given by

$$\frac{w_{ij}^{n+1} - w_{ij}^n}{\Delta t} = C_{ij,1}^n \delta_{vv}^2 w_{ij}^{n+1} + C_{ij,2}^n \Delta_u \Delta_v w_{ij}^{n+1} + C_{ij,3}^n \delta_{uu}^2 w_{ij}^{n+1}, \quad (28)$$

where the coefficients $C_{ij,1}$, $C_{ij,2}$ and $C_{ij,3}$ are specified by

$$\begin{aligned} C_{ij,1}^n &= \frac{1 + (\Delta_u w_{ij}^n)^2}{2(1 + (\Delta_u w_{ij}^n)^2 + (\Delta_u w_{ij}^n)^2)}, \\ C_{ij,2}^n &= \frac{-\Delta_u w_{ij}^n \Delta_v w_{ij}^n}{1 + (\Delta_u w_{ij}^n)^2 + (\Delta_u w_{ij}^n)^2}, \\ C_{ij,3}^n &= \frac{1 + (\Delta_v w_{ij}^n)^2}{2(1 + (\Delta_u w_{ij}^n)^2 + (\Delta_u w_{ij}^n)^2)}. \end{aligned}$$

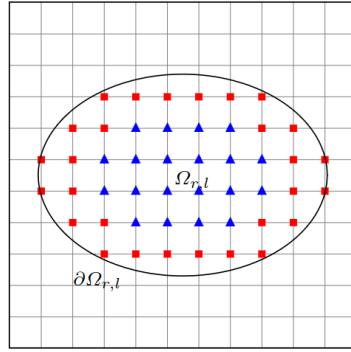


Fig. 4 A schematic of node identification of the nine-point scheme (28) on a base domain: boundary nodes $\partial\Omega_{r,l}^h$ (marked by red rectangles) and interior nodes $\Omega_{r,l}^h \setminus \partial\Omega_{r,l}^h$ (marked by blue triangles).

3.3 Boundary condition

Note that the finite difference schemes (27) and (28) are three-point and nine-point schemes, respectively. Denote by $\Omega_{r,l}^h$ all grid nodes in $\Omega_{r,l}$. A grid node is identified as a boundary node if all of its stencil nodes belong to $\Omega_{r,l}^h$. Otherwise, it is identified as an interior node, see Figure 4 for an example. The set of boundary nodes is denoted as $\partial\Omega_{r,l}^h$. Further, a control point is identified as a boundary control point if its projection on the base plane belongs to $\partial\Omega_{r,l}^h$. Denote by $\partial\Gamma_{r,l}^h$ the set of boundary control points. It is worthwhile to mention that $\partial\Gamma_{r,l}^h$ works as the numerical boundary of $\Gamma_{r,l}$ and is not necessarily a subset of $\partial\Omega_{r,l}^h$. At boundary nodes, the match condition (18) is utilized to pose Dirichlet boundary conditions for (13) and (17). Numerically, the partition of unity χ_r is taken as

$$\chi_r(\mathbf{x}) = \begin{cases} 1, & \text{if } |\mathbf{n}(\mathbf{x}) \cdot \mathbf{e}_r| > |\mathbf{n}(\mathbf{x}) \cdot \mathbf{e}_j|, \forall j \neq r, \\ 0, & \text{otherwise.} \end{cases} \quad (29)$$

This simple choice of partition of unity is sufficient for accuracy, though it is not a smooth one. The partition of unity can be understood as evaluating values at boundary nodes on Γ_r by interpolation from interior nodes on other subsets $\Gamma_j, j \neq r$. Since subsets overlap with each other, interpolation stencils always exist.

In the following, we introduce the approaches to discretize the matching condition (18).

3.3.1 Coupled matching condition

The simplest way to discretize (18) is to enforce it at every time level in a discrete sense.

$$\mathbf{x}^{(r),n+1} = \sum_{j \neq r} \chi_j \mathbf{x}^{(j),n+1}, \quad \mathbf{x}^{(r),n+1} \in \partial \Gamma_r^{h,n+1}. \quad (30)$$

This leads to a nonlinear system that couples the solutions on all subsets Γ_r in each time step. Let \mathbf{u}^i and \mathbf{u}^b denote the vectors of solutions at the interior and boundary nodes, respectively. The system, which needs to be solved in each time step, is written as

$$\begin{aligned} \mathbf{A}\mathbf{u}^i + \mathbf{Q}\mathbf{u}^b &= \mathbf{f}, \\ \mathbf{u}^b &= \mathbf{\Pi}\mathbf{u}^i, \end{aligned} \quad (31)$$

where \mathbf{A}, \mathbf{Q} are matrices, $\mathbf{\Pi}$ is the interpolation operator and \mathbf{f} is the vector containing solutions at previous time levels. Typically, in the system (31), the first equation approximates PDEs, and the second approximates the matching condition. Here, the operator $\mathbf{\Pi}$ is essentially nonlinear since discretizing (30) involves the root-finding of polynomials. Note that the matrix \mathbf{A} is block-wise diagonal and is invertible. The nonlinear system (31) can be solved, in spirit, with the technique of Schur complement. One first needs to solve the lower dimensional system

$$\mathbf{\Pi}\mathbf{A}^{-1}(\mathbf{f} - \mathbf{Q}\mathbf{u}^b) - \mathbf{u}^b = 0, \quad (32)$$

for \mathbf{u}^b and then obtains \mathbf{u}^i by solving

$$\mathbf{A}\mathbf{u}^i = \mathbf{f} - \mathbf{Q}\mathbf{u}^b. \quad (33)$$

The system (32), which looks like a Schur complement system but is nonlinear, can be solved with the method widely used in domain decomposition methods, the Schwarz alternating method, which is a block-wise Gauss-Seidel type iteration method [21, 29]. The main idea of the method is to solve problems alternately on each subdomain and to provide boundary conditions for other subdomains. Generally, the Schwarz alternating method converges geometrically within a few iterations. The blocks in matrix \mathbf{A} are the approximations of elliptic differential operators, which can also be inverted by an iterative method, such as the successive over-relaxation (SOR) method. In particular, in two space dimensions, \mathbf{A} is block-wise tri-diagonal, and the Thomas algorithm is applicable.

3.3.2 ADI method

Instead of directly enforcing the matching condition (18), we can also follow the idea of the alternating direction implicit (ADI) method, which is used to solve time-dependent PDEs in multiple space dimensions and discretize the matching condition with a time splitting technique. Note that there is no

need to enforce (18) accurately since numerical discretization of the PDEs has already introduced numerical errors. One only needs to approximate it with an error on the order of $\mathcal{O}(\tau^p)$ where τ is the time step, and p is the approximation order. We evolve Γ_r alternately and compute boundary conditions with the newest solutions, which is also an accurate approximation to the matching condition, by

$$\mathbf{x}^{(r),n+1} = \sum_{j \neq r} \chi_j \mathbf{x}^{(j),n^*}, \quad \mathbf{x}^{(r),n+1} \in \partial \Gamma_r^{h,n+1}. \quad (34)$$

where $\mathbf{x}^{(j),n^*}$ is the newest solution on Γ_j . For example, boundary conditions for Γ_1 are interpolated from the newest solution on Γ_2 and Γ_3 ; then, update the solution on Γ_1 and compute boundary conditions for Γ_2 using the newest solution on Γ_1 and Γ_3 , etc. This approach is non-iterative in the sense that the solutions on subsets Γ_r are not coupled, and no Schur complement system needs to be solved in each time step. This ADI method is also a time-splitting strategy with a formal splitting error on the order of $\mathcal{O}(\tau)$.

The two approaches only differ in the computation of boundary conditions. Figure 5 presents the numerical solutions obtained by these two approaches. One can see that the solutions obtained by these two approaches only have subtle differences at several nodes in the overlapping region.

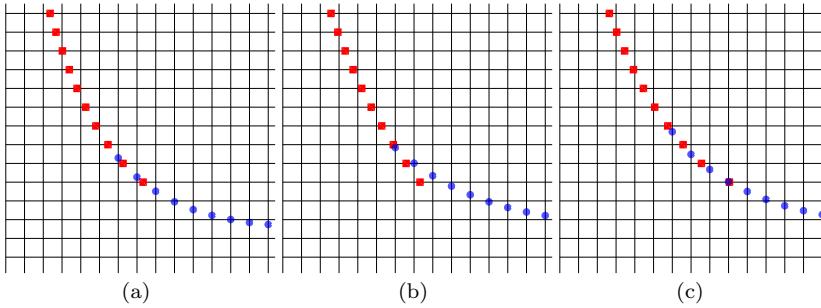


Fig. 5 Zoom-in snapshots of numerical solutions. (a) initial solution; (b) solution after a time step by the ADI method; (c) solution after a time step by coupled matching condition. Control points on horizontal lines are marked as red rectangles, and those on vertical lines are marked as blue circles.

In fact, the ADI method and the Schwarz alternating method are closely related to this problem. The ADI method is only a strategy to provide Dirichlet boundary conditions for PDEs (13) and (17). One can also repeatedly use the ADI method to compute new boundary conditions and update the solution at t^{n+1} , which exactly leads to the Schwarz alternating method. Therefore, the Schwarz alternating method reduces to the ADI method if only one iteration is performed in each time step.

4 Algorithm summary

In this section, the algorithm for solving mean curvature flows (1) with the proposed ADI method is summarized as follows:

Algorithm. The ADI method for mean curvature flows:

Step 1. Given the initial hypersurface by its parametric form or a level set function, embed it into a bounding box that is uniformly partitioned into a Cartesian grid and find the control points with overlapping decomposition strategy described in Subsection 3.1.

Step 2. In each time step, evolve the overlapping subsets alternately. For each subset, do the following three procedures:

- (a) identify all the nodes in each isolated component of the subset with the breadth-first search method.
- (b) compute the Dirichlet-type boundary condition for boundary nodes with the discrete matching condition (34);
- (c) evolve the subset to the next time level by solving (27) or (28);

Step 4. Update control points such that they satisfy the decomposition strategy described in Subsection 3.1.

Step 5. Repeat steps 2-4 until the final computational time.

Remark 1 Procedure (a) in Step 2 is only for matrix assembly such that direct methods, such as the Thomas algorithm, are applicable. Suppose the finite difference equations (27) and (28) are solved with iterative methods which only require the matrix-vector product. In that case, one can find the local stencil points on-the-fly instead of finding all the nodes in the components in advance.

5 Numerical results

This section presents numerical examples in two and three space dimensions to validate the proposed method. Initial hypersurfaces are given in parametric forms or the zero level set of level set functions, which will be prescribed in each example. In all the numerical examples, the bounding box \mathcal{B} is uniformly partitioned into a Cartesian grid \mathcal{G} with N intervals in each direction. For problems with exact solutions, we estimate the numerical error at a surface point by finding its projection on the hypersurface of the exact solution and computing the distance. We take the solution on a fine grid for problems without exact solutions as a reference “exact” solution. Then, we estimate the numerical error at a surface point by finding the closest surface point on the reference solution and computing the distance. The numerical errors in the maximum norm and l_2 norm are computed by

$$\|\mathbf{e}_h\|_\infty = \max_{\mathbf{x}_i \in \Gamma^h} \left\{ \|\mathbf{x}_i - \mathbf{x}_i^{ref}\| \right\}, \quad \|\mathbf{e}_h\|_2 = \sqrt{\frac{1}{N_\Gamma} \sum_{\mathbf{x}_i \in \Gamma^h} \|\mathbf{x}_i - \mathbf{x}_i^{ref}\|^2}, \quad (35)$$

where \mathbf{x}_i^{ref} is the exact solution or reference solution on a fine grid associated with \mathbf{x}_i and N_F is the total point number.

The following numerical experiments are performed on a personal computer with a 3.80 GHz Intel Core i7 processor. The codes for conducting the numerical experiments are written in C++ computer language.

5.1 Two space dimensional examples

First, we solve the mean curvature flow for a simple case and compare the numerical solution with the exact solution to verify the convergence of the proposed method. The initial shape is chosen such that the curve is a circle whose radius $r(t)$ satisfies

$$r(t) = \sqrt{1 - 2t}. \quad (36)$$

The bounding box is taken as $[-1.2, 1.2]^2$. Note that the curve will eventually shrink to a point at $T_{end} = 0.5$. We chose to estimate the numerical errors at $T = 0.2$ to ensure that the coarsest grid $N = 64$ can fully resolve the curve during the computation. On finer grids, the computation can last longer than T . Time step size is chosen to be $\Delta t = 0.1\Delta x$ where Δx is the spatial grid size. Numerical results are summarized in Table 1.

Table 1 Numerical error and convergence order of the 2D MCF for a circle-shaped initial curve.

N	$\ \mathbf{e}_h\ _\infty$	order	$\ \mathbf{e}_h\ _2$	order
64	7.99e-03	-	3.99e-03	-
128	3.80e-03	1.07	1.79e-04	1.16
256	1.88e-03	1.02	8.42e-04	1.09
512	9.45e-04	0.99	4.12e-05	1.03
1024	4.72e-04	1.00	2.05e-05	1.01

Next, we change the initial shape to an ellipse which is given by

$$\begin{cases} x = a \cos(\theta), \\ y = b \sin(\theta), \end{cases} \quad \theta \in [0, 2\pi), \quad (37)$$

with $a = 1.0, b = 0.5$. The problem is solved in the bounding box $\mathcal{B} = [-1.2, 1.2]^2$. Time step size is taken as $\Delta t = 0.1\Delta x$. Since there is no exact solution for this configuration, the solution on a fine grid with $N = 2048$ is chosen as a reference solution to estimate numerical errors. The estimated error and the convergence order are summarized in Table 2. It can be observed that the convergence order is a bit larger than 1. This may be due to the inaccurate estimation of numerical error based on the distance between the surface point to its closest point in the reference solution. The evolution history of the curve is presented in Figure 6. The changes in curve length and enclosed area are computed on the grid with $N = 1024$ and shown in Figure

Table 2 Numerical error and convergence order of the 2D MCF for an ellipse-shaped initial curve.

N	$\ e_h\ _\infty$	order	$\ e_h\ _2$	order
128	2.45e-02	-	1.84e-02	-
256	1.04e-02	1.24	7.95e-03	1.21
512	4.31e-03	1.27	3.31e-03	1.26
1024	1.55e-03	1.48	1.15e-03	1.52

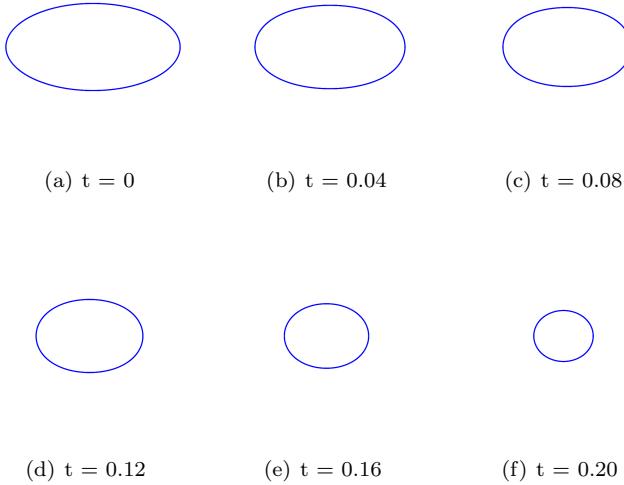


Fig. 6 Time evolution of the MCF for an ellipse-shaped curve.

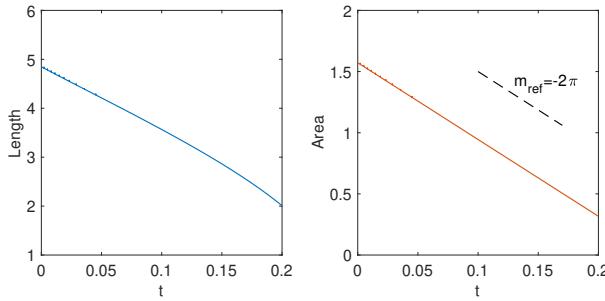


Fig. 7 Time evolution of curve length and enclosed area of the ellipse-shaped curve.

7. It can be observed that the enclosed area loss rate compares favorably with the theoretic result, whose slope is $m_{ref} = -2\pi$.

We also chose a five-fold star-shaped initial curve, which is given by

$$\begin{cases} x = a(\kappa + \eta \sin(m\theta)) \cos(\theta), \\ y = b(\kappa + \eta \sin(m\theta)) \sin(\theta), \end{cases} \quad \theta \in [0, 2\pi), \quad (38)$$

with $a = 1.0, b = 1.0, \kappa = 0.8, \eta = 0.2$ and $m = 5$. The bounding box and time step size are the same as those in the last case. Estimated numerical errors and convergence orders are summarized in Table 3. The evolution history of the curve and changes in curve length and enclosed area are presented in Figure 8 and 7, respectively. The numerical results are also consistent with theoretical results.

Table 3 Numerical error and convergence order of the 2D MCF for a five-fold star-shaped initial curve.

N	$\ e_h\ _\infty$	order	$\ e_h\ _2$	order
128	5.45e-03	-	3.21e-03	-
256	2.60e-03	1.07	1.46e-03	1.14
512	1.21e-03	1.10	6.60e-04	1.15
1024	5.00e-04	1.28	2.26e-05	1.55

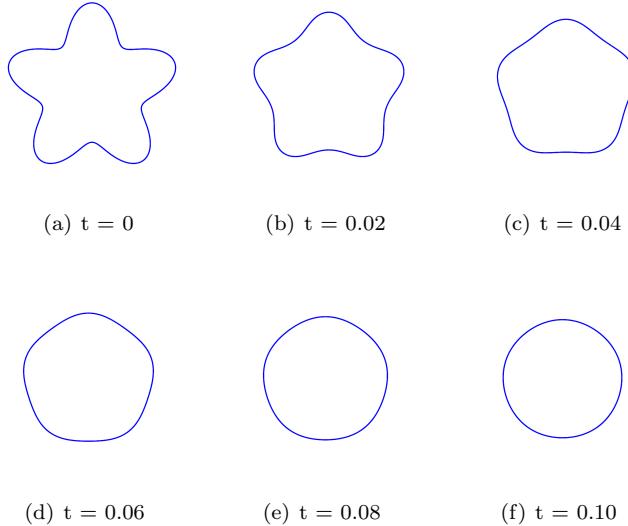


Fig. 8 Time evolution of the MCF for a five-fold star-shaped curve.

For this case, we compare the present method with an explicit time-advancing scheme for the mean curvature flow, which discretizes the equation

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = -\frac{x_\theta y_{\theta\theta} - x_{\theta\theta} y_\theta}{(x_\theta^2 + y_\theta^2)^2} \begin{pmatrix} y_\theta \\ -x_\theta \end{pmatrix}, \quad \theta \in [0, 2\pi), \quad (39)$$

with forward Euler scheme and central differences for temporal and spatial derivatives, respectively. The time step size for the explicit method is chosen

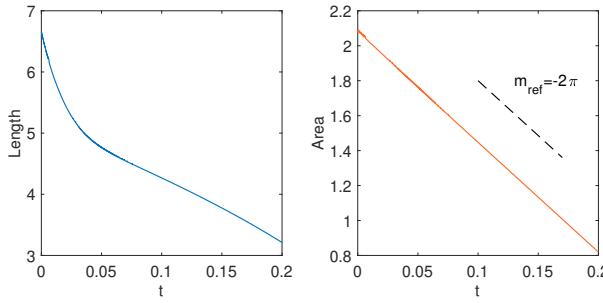


Fig. 9 Time evolution of curve length and enclosed area of the five-fold star-shaped curve.

to ensure numerical stability, using the adaptive time step

$$\Delta t = 0.8(\min_{\theta} \Delta s)^2, \quad (40)$$

where Δs denotes the Euclidean distance between two adjacent grid nodes. We properly optimize the codes for both methods and collect the required CPU times for solving the mean curvature flow to the final time $T = 0.1$. The control point number at time t is denoted by M_t . Numerical results are summarized in Table 4. It can be observed that while the forward Euler method is faster on coarse grids, the present method becomes more efficient than the forward Euler method as the point number increases. It can be explained by the complexities of the two methods. To compute the solution to a fixed final time, since the time step size can be chosen as linearly proportional to the spatial grid size for the present method, the computational complexity is $\mathcal{O}(N^2)$. However, the computational complexity of the forward Euler scheme is $\mathcal{O}(N^3)$ due to high-order constraints on time step size. In fact, the forward Euler method fails for long-time computation since the required time step size quickly decreases to 10^{-6} due to the curve shortening phenomenon and poor mesh quality.

Table 4 CPU time comparison between the present method and an explicit time advancing scheme.

Present method				Forward Euler method	
N	M_0	M_T	CPU times(secs)	$M_0 = M_T$	CPU times(secs)
128	392	224	9.66e-03	224	3.73e-03
256	786	444	3.54e-02	448	1.73e-02
512	1574	890	1.34e-01	896	1.34e-01
1024	3150	1776	5.15e-01	1792	1.02e+00
2048	6300	3548	2.08e+00	3584	8.30e+00

5.2 Three space dimensional examples

For three space dimensional mean curvature flows, we test the convergence rate of the method by considering a simple case, a sphere-shaped initial surface. Similar to two space dimensional case, this configuration has an exact solution: the surface maintains a sphere, and the radius $r(t)$ satisfies

$$r(t) = \sqrt{1 - 2t}. \quad (41)$$

Numerical errors are estimated at $T = 0.2$ and summarized in Table 5.

Table 5 Numerical error and convergence order of the 3D MCF for a sphere-shaped initial surface.

N	$\ e_h\ _\infty$	order	$\ e_h\ _2$	order
64	4.72e-03	-	1.42e-03	-
128	3.46e-03	0.45	7.46e-04	0.93
256	1.45e-03	1.25	3.63e-04	1.04
512	9.18e-04	0.66	1.82e-04	1.00
1024	3.47e-04	1.40	8.81e-05	1.05

We also solve the mean curvature flow in three space dimensions for more examples. In the following numerical examples, the bounding boxes partitioned into a Cartesian grid are all chosen as $\mathcal{B} = [-1.2, 1.2]^3$. The time step is chosen as $\Delta t = 0.05\Delta x$.

In the first case, we set the initial shape as an ellipsoid which is given by

$$\Gamma = \left\{ (x, y, z) \left| \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0 \right. \right\}, \quad (42)$$

with $a = 1.0, b = 0.7, c = 0.5$. Numerical error and convergence order estimated at $t = 0.2$ are summarized in Table 6. The time evolution of the surface and its area and enclosed volume are presented in Figure 10 and 11, respectively. One can observe that the major axis of the ellipsoid decreases faster compared with the other two axes and the ellipsoid becomes very close to a sphere. Surface area decreases with time, which is consistent with the theoretical result.

Table 6 Numerical error and convergence order of the 3D MCF for an ellipsoid-shaped initial surface.

N	$\ e_h\ _\infty$	order	$\ e_h\ _2$	order
128	1.58e-03	-	3.78e-04	-
256	1.32e-03	0.26	1.74e-04	1.12
512	3.70e-04	1.83	6.36e-05	1.45

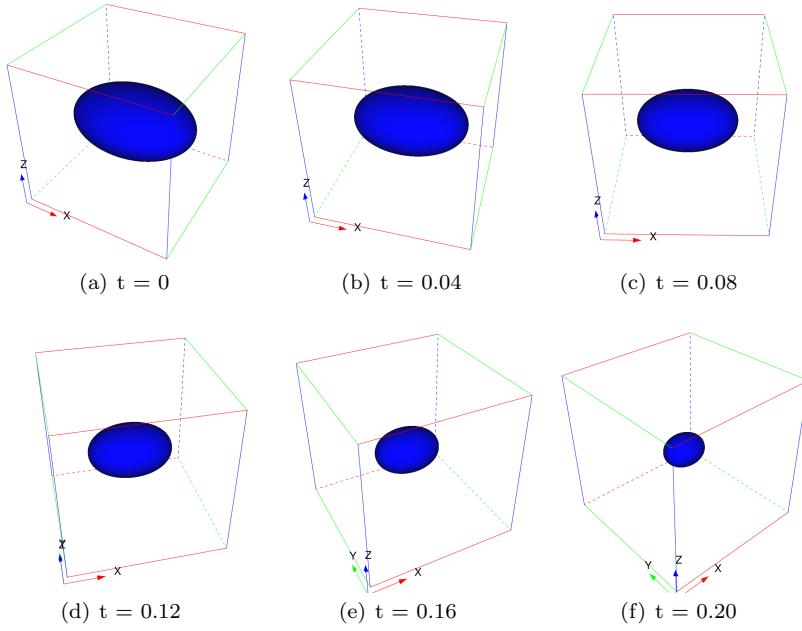


Fig. 10 Time evolution of the MCF for an ellipsoid-shaped surface.

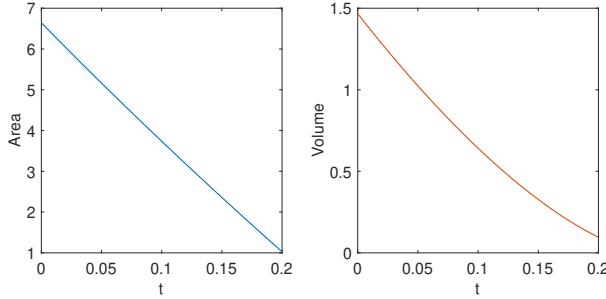


Fig. 11 Time evolution of the surface area and enclosed volume of the ellipsoid-shaped surface.

In the second case, we chose a genus 1 torus-shaped initial surface. The surface is given by

$$\Gamma = \left\{ (x, y, z) \mid \left(c - \sqrt{x^2 + y^2} \right)^2 + z^2 - a^2 = 0 \right\}, \quad (43)$$

with $a = 0.34$, $c = 0.8$. Numerical error and convergence order are summarized in Table 7. The time evolution of the surface and its area and enclosed volume are presented in Figure 12 and 13, respectively. Driven by mean curvature, the torus-shaped surface becomes thinner with time.

Table 7 Numerical error and convergence order of the 3D MCF for a torus-shaped initial surface.

N	$\ e_h\ _\infty$	order	$\ e_h\ _2$	order
128	1.65e-03	-	3.08e-04	-
256	1.29e-03	0.36	1.64e-04	0.91
512	8.26e-04	0.64	5.94e-05	1.47

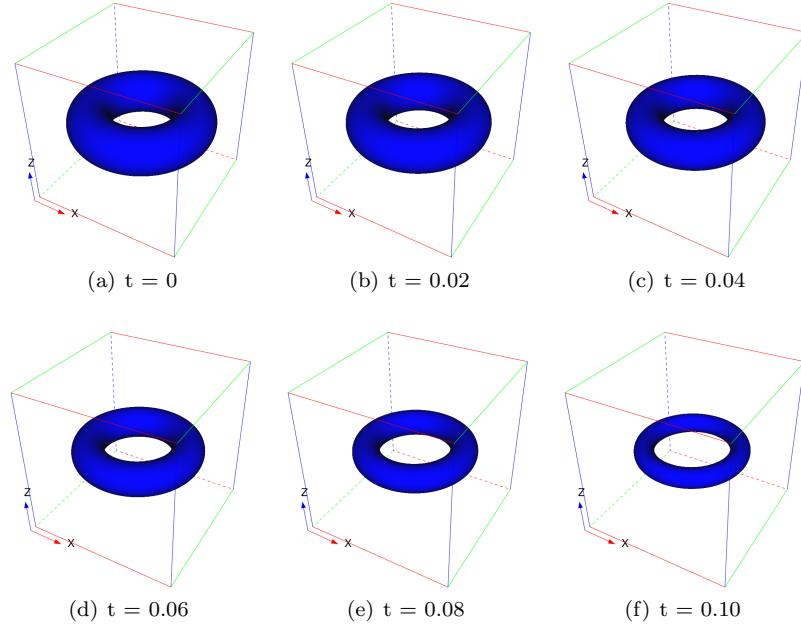


Fig. 12 Time evolution of the MCF for a torus-shaped surface.

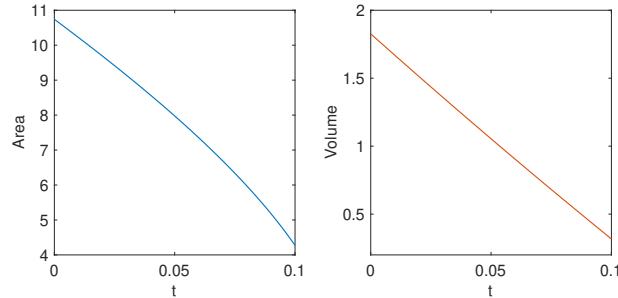


Fig. 13 Time evolution of the surface area and enclosed volume of the torus-shaped surface.

In the final case, the initial surface is a four-atom molecular-shaped surface which is given by

$$\Gamma = \left\{ (x, y, z) \mid c - \sum_{k=1}^4 \exp \left(-\frac{|\mathbf{x} - \mathbf{x}_k|^2}{r^2} \right) = 0 \right\}, \quad (44)$$

with $\mathbf{x}_1 = (\sqrt{3}/3, 0, -\sqrt{6}/12)$, $\mathbf{x}_2 = (-\sqrt{3}/6, 0.5, -\sqrt{6}/12)$, $\mathbf{x}_3 = (-\sqrt{3}/6, -0.5, -\sqrt{6}/12)$, $\mathbf{x}_4 = (0, 0, \sqrt{6}/4)$ and $c = 0.5$, $r = 0.5$. The numerical error and convergence order are summarized in Table 8. The time evolution of the surface and its area and enclosed volume are presented in Figure 14 and 15, respectively.

Table 8 Numerical error and convergence order of the 3D MCF for a molecular-shaped initial surface.

N	$\ e_h\ _\infty$	order	$\ e_h\ _2$	order
128	1.79e-03	-	4.43e-04	-
256	1.24e-03	0.53	2.51e-04	0.82
512	5.19e-04	1.26	9.12e-05	1.46

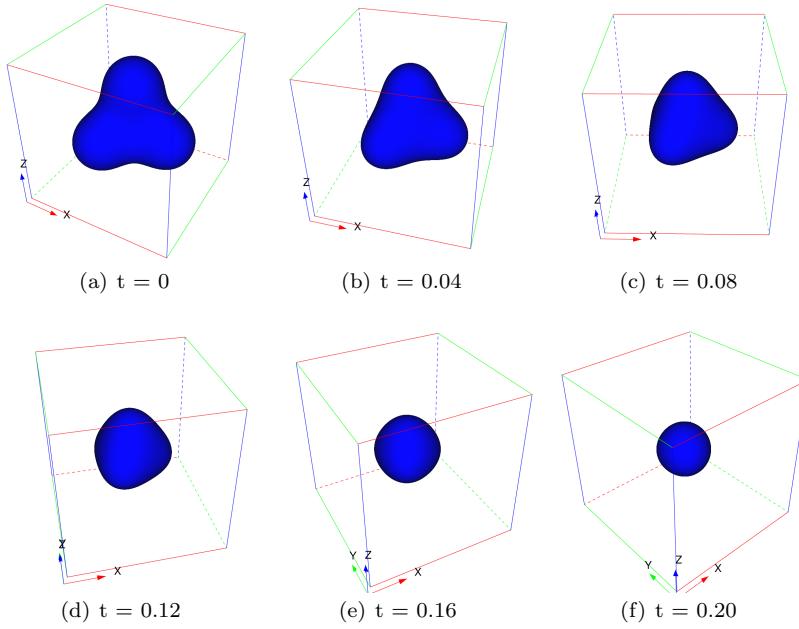


Fig. 14 Time evolution of the MCF for a molecular-shaped surface.

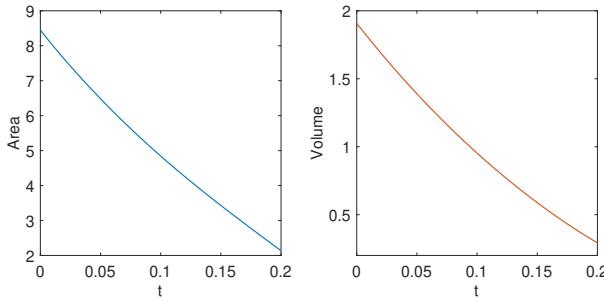


Fig. 15 Time evolution of the surface area and enclosed volume of the molecular-shaped surface.

6 Discussion

This work presents a Cartesian grid-based alternating direction implicit method for solving mean curvature flows in two and three space dimensions. The method decomposes a hypersurface into multiple overlapping subsets for which new evolution equations are derived by adding extra tangential velocities. The new formulations for the moving hypersurface only require solving a sequence of scalar quasi-linear parabolic PDEs on planar domains, which is one dimensional lower than the original formulation. The overlapping subsets of the hypersurface can be represented in terms of height functions of Monge patches which are discretized with Cartesian grids. With this representation of the hypersurface, an ADI-type semi-implicit time integration method is proposed such that the subsets can be evolved alternately.

The convergence of the proposed method is validated by numerical experiments. The results show that the ADI method is efficient compared with an explicit scheme since it does not have high-order stability constraints on time step size. Mean curvature flows for various hypersurfaces in two and three space dimensions are also presented, including one whose initial configuration is a genus 1 surface.

Although the method in this paper is designed for solving mean curvature flows, it is expected to be able to solve more moving interface problems described by geometric evolution laws, such as the anisotropic mean curvature flow, the surface diffusion flow, and the Willmore flow. Further, for problems that involve moving interfaces and bulk PDEs simultaneously, such as the Stefan problem and two-phase Stokes flow, the method can also be applicable if combined with a PDE solver such as the kernel-free boundary integral method [34].

Funding W. Y. is financially supported by the National Key R&D Program of China, Project Number 2020YFA0712000, the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDA25010405), the National Natural Science Foundation of China (Grant No. DMS-11771290) and the Science Challenge Project of China (Grant No. TZ2016002). S. L.

is partially supported by the U.S. National Science Foundation, Division of Mathematical Sciences grants DMS-1720420 and DMS-2309798.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work. There is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

References

1. Mohd Almie Alias and Pascal R. Buerzli. A level-set method for the evolution of cells and tissue during curvature-controlled growth. *International Journal for Numerical Methods in Biomedical Engineering*, 36, 1 2020.
2. David M. Ambrose, Michael Siegel, and Svetlana Tlupova. A small-scale decomposition for 3D boundary integral computations with surface tension. *Journal of Computational Physics*, 247:168–191, 2013.
3. Weizhu Bao and Quan Zhao. A structure-preserving parametric finite element method for surface diffusion. *SIAM Journal on Numerical Analysis*, 59(5):2775–2799, 2021.
4. John W Barrett, Harald Garcke, and Robert Nürnberg. A parametric finite element method for fourth order geometric evolution equations. *Journal of Computational Physics*, 222(1):441–467, 2007.
5. John W Barrett, Harald Garcke, and Robert Nürnberg. On the parametric finite element approximation of evolving hypersurfaces in \mathbb{R}^3 . *Journal of Computational Physics*, 227(9):4281–4307, 2008.
6. John W. Barrett, Harald Garcke, and Robert Nürnberg. Phase field models versus parametric front tracking methods: Are they accurate and computationally efficient? *Communications in Computational Physics*, 15(2):506–555, 2014.
7. John W. Barrett, Harald Garcke, and Robert Nürnberg. Parametric finite element approximations of curvature-driven interface evolutions. *Handbook of Numerical Analysis*, 21:275–423, 2020.
8. G. Bellettini and M. Paolini. Anisotropic motion by mean curvature in the context of finsler geometry. *Hokkaido Mathematical Journal*, 25(3):537–566, 1996.
9. Xinfu Chen. Generation and propagation of interfaces for reaction-diffusion equations. *Journal of Differential Equations*, 96(1):116–141, 1992.
10. Tobias Holck Colding, William P. Minicozzi, and Erik Kjær Pedersen. Mean curvature flow. *Bulletin of the American Mathematical Society*, 52(2):297–333, 2015.
11. Michael C. Dallaston and Scott W. McCue. A curve shortening flow rule for closed embedded plane curves with a prescribed rate of change in enclosed area. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 472(2185), 2016.
12. Klaus Deckelnick and Gerhard Dziuk. Discrete anisotropic curvature flow of graphs. *ESAIM: Mathematical Modelling and Numerical Analysis*, 33(6):1203–1222, 1999.
13. Klaus Deckelnick, Gerhard Dziuk, and Charles M. Elliott. Computation of geometric partial differential equations and mean curvature flow. *Acta Numerica*, 14(2005):139–232, 2005.
14. Jim Douglas and James E. Gunn. Alternating direction methods for parabolic systems in m space variables. *Journal of the ACM (JACM)*, 9(4):450–456, 1962.

15. Jim Douglas and James E. Gunn. A general formulation of alternating direction methods - part i. parabolic and hyperbolic problems. *Numerische Mathematik*, 6(1):428–453, 1964.
16. Thomas Y Hou, Isaac Klapper, and Helen Si. Removing the stiffness of curvature in computing 3-d filaments. *Journal of Computational Physics*, 143(2):628–664, 1998.
17. Thomas Y Hou, John S Lowengrub, and Michael J Shelley. Removing the stiffness from interfacial flows with surface tension. *Journal of Computational Physics*, 114(2):312–338, 1994.
18. Ming Chih Lai, Che Wei Hsu, and Huaxiong Huang. A front-tracking method for motion by mean curvature with surfactant. *Advances in Applied Mathematics and Mechanics*, 1(2):288–300, 2009.
19. R. Malladi and J. A. Sethian. Image processing via level set curvature flow. *Proceedings of the National Academy of Sciences of the United States of America*, 92(15):7046–7050, 1995.
20. Stéphane Mallat and Gabriel Peyré. A review of bandlet methods for geometrical image representation. *Numerical Algorithms*, 44(3):205–234, 2007.
21. Tarek Mathew. *Domain decomposition methods for the numerical solution of partial differential equations*, volume 61. Springer Science & Business Media, 2008.
22. W. W. Mullins. Two-dimensional motion of idealized grain boundaries. *Journal of Applied Physics*, 27(8):900–904, 1956.
23. Celeste M. Nelson, Ronald P. Jean, John L. Tan, Wendy F. Liu, Nathan J. Sniadecki, Alexander A. Spector, Christopher S. Chen, and Robert Langer. Emergent patterns of growth controlled by multicellular form and mechanics. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11594–11599, 2005.
24. Stanley Osher, Ronald Fedkiw, and K Piechor. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.*, 57(3):B15–B15, 2004.
25. Stanley Osher and Nikos Paragios. *Geometric level set methods in imaging, vision, and graphics*. Springer Science & Business Media, 2003.
26. Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.
27. Maurizio Paolini. *An efficient algorithm for computing anisotropic evolution by mean curvature*. Dipartimento di Matematica " F. Enriques", Univ. degli Studi, 1995.
28. Rachford Peaceman, D. W., D W Peaceman, and H H Rachford Jr. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):28–41, 1955.
29. Yousef Saad. Iterative methods for sparse linear systems. *Notes*, 3(2nd Edition):xviii+528, 2003.
30. G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y. J. Jan. A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics*, 169(2):708–759, 2001.
31. Salih Ozen Unverdi and Grétar Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1):25–37, 1992.
32. Jason R. Wilson. *On computing smooth, singular, and nearly singular integrals on implicitly defined surfaces*. PhD thesis, Duke University, 2010.
33. Juntao Ye and Guoliang Xu. Geometric flow approach for region-based image segmentation. *IEEE Transactions on Image Processing*, 21(12):4735–4745, 2012.
34. Wenjun Ying and Wei Cheng Wang. A kernel-free boundary integral method for implicitly defined surfaces. *Journal of Computational Physics*, 252:606–624, 2013.