J Sci Comput
DOI 10.1007/s10915-017-0448-1

An Efficient Boundary Integral Scheme for the MBO Threshold Dynamics Method via the NUFFT

Shidong Jiang[1] · **Dong Wang**[2] · **Xiao-Ping Wang**[2]

Received: 25 January 2017 / Revised: 27 April 2017 / Accepted: 2 May 2017

Abstract The MBO threshold dynamics method consists of two steps. The first step solves a pure initial value problem of the heat equation with the initial data being the indicator function of some bounded domain. In the second step, the new sharp interface is generated via thresholding either by some prescribed solution value or by volume preserving. We propose an efficient boundary integral scheme for simulating the threshold dynamics via the nonuniform fast Fourier transform (NUFFT). The first step is carried out by evaluating a boundary integral via the NUFFT, and the second step is performed applying a root-finding algorithm along the normal directions of a discrete set of points at the interface. Unlike most existing methods where volume discretization is needed for the whole computational domain, our scheme requires the discretization of physical space only in a small neighborhood of the interface and thus is meshfree. The algorithm is spectrally accurate in space for smooth interfaces and has $O(N \log N)$ complexity, where N is the total number of discrete points near the interface when the time step Δt is not too small. The performance of the algorithm is illustrated via several numerical examples in both two and three dimensions.

Keywords Threshold dynamics · Nonuniform FFT · MBO method · Heat equation · Motion by mean curvature

Mathematics Subject Classification 35K05 · 65M70 · 76T99 · 82C24

✉ Shidong Jiang
shidong.jiang@njit.edu

Dong Wang
dwangaf@connect.ust.hk

Xiao-Ping Wang
mawang@ust.hk

[1] Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, NJ 07102, USA

[2] Department of Mathematics, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China

Published online: 16 May 2017

1 Introduction

Interface motion is an important topic in many areas of applied mathematics and scientific computing including fluid mechanics, image processing, and computer vision. Many numerical methods have been developed for simulating interface dynamics including front-tracking methods [21,37], level set methods [9,27,40], and phase field methods [12,39].

In 1992, Merriman, Bence, and Osher (MBO) [24] developed a threshold dynamics method for the motion of the interface driven by the mean curvature, where the governing equation is $V_n(x) = \kappa(x)$ with $V_n(x)$ being the normal velocity of a point x on the interface and $\kappa(x)$ being the mean curvature at x. To be more precise, let $D \in \mathbb{R}^n$ be a domain whose boundary $\Gamma = \partial D$ is to be evolved via motion by mean curvature. For each time step, the MBO method generates the new interface Γ_{new} (or equivalently, D_{new}) via the following two steps:
Step 1 Solve the pure initial value problem of the heat equation to obtain its solution u^* at $t = \Delta t$

$$\begin{aligned} u_t &= \Delta u, \\ u|_{t=0} &= \chi_D. \end{aligned}$$

Here χ_D is the indicator function of the domain D, i.e.,

$$\chi_D(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in D, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Step 2 Obtain the domain D_{new} by

$$D_{\text{new}} = \left\{ \mathbf{x} : u^*(\mathbf{x}, \Delta t) \geq \frac{1}{2} \right\}.$$

The MBO method converges to continuous motion by mean curvature as $\Delta t \to 0$ [1,2,10]. The method has become very popular due to its simplicity and unconditional stability, and thus it has also been extended to handle many other applications. These applications include multi-phase flows with arbitrary surface tensions [7], area- or volume-preserving interface motion [19,35,37], and wetting on solid surfaces [38]. Various algorithms and rigorous error analysis have been performed to refine and extend the original MBO method and related methods for the aforementioned problems (see, for example, [3,8,17,18,23,31–33]). These algorithms all consist of two time March ing steps with the same first step as the MBO method. Only the second thresholding step of obtaining the new sharp interface is modified. Sometimes it is obtained by finding a level surface of a specific value of the solution to the heat flow as in the original MBO method; at other times it is obtained by finding the volume-preserving surface or by minimizing certain functional whose exact form depends on the specific application.

In this paper, we propose an efficient algorithm for the threshold dynamics method with an NUFFT-based heat equation solver. Standard potential theory [11] shows that the solution to the pure initial value problem of the heat equation is given by the convolution of the heat kernel $G(\mathbf{x}, \mathbf{y}; t)$ with the initial data u_0. The convolution integral on the whole domain can be converted to a boundary integral by the divergence theorem or Green's theorem. We first show that an equispaced grid is sufficient to obtain a spectral approximation for the heat kernel at a fixed time $t = \Delta t$. We then apply the NUFFT [4,5,13] to evaluate the boundary integral at a set of target points. For threshold dynamics, the interface moves within a band around previous positions with a bandwidth of $O(\sqrt{\Delta t})$ due to the diffusion process. Thus we may choose a set of pN points as the target points with N being the number of points

on the interface and p being the number of points along each normal direction (p is usually 10–20). Finally, we use a standard root-finding procedure for thresholding. The algorithm is spectrally accurate in space. It is also very efficient when Δt is not too small for three reasons. First, there is no need to discretize the whole computational domain and only pN points near the interface are needed in physical space. Second, the NUFFT has the same $O(N \log N)$ complexity as the standard fast Fourier transform (FFT). Third, the root-finding procedure converges rapidly, usually in 4–6 iterations to full double precision, thanks to the smoothing effect of the diffusion process. Our method is also meshfree since it tracks the motion of a set of points only.

Previous work and our contributions Many excellent numerical methods are available for simulating threshold dynamics. Instead of presenting a comprehensive review of these methods, we would like to discuss other NUFFT-based methods and explain how they differ from our method. In [30,32], efficient algorithms are developed for diffusion-generated motion by mean curvature. The algorithms assume Neumann boundary conditions on a rectangular box, represent the solution to the heat equation with a cosine series, and then use the NUFFT to solve the heat equation on an adaptive grid.

In [14], a spectral approximation of the heat kernel is constructed with uniform absolute error for all $t > \delta$. Due to this much stricter requirement, the spectral approximation needs an adaptive nonuniform node in the Fourier space with dyadic refinement towards the origin. In [22], a fast solver is developed to solve the initial value problem of the heat equation using the NUFFT based on the adaptive nonuniform approximation of the heat kernel in [14]. The paper aims to find the solution for all $t > \delta$ and thus assumes that the initial data is smooth and requires time March ing.

In the threshold dynamics method, due to the thresholding step, we only need to find the solution to the heat equation at $t = \Delta t$ since the diffusion process starts anew for each time step. Thus we only need to approximate the heat kernel at $t = \Delta t$. Our analysis shows that an equispaced grid in the Fourier space is sufficient to approximate the heat kernel with spectral accuracy. That is, we do not need to use adaptive Gaussian nodes in the Fourier space as in [14], nor is it necessary to impose Neumann boundary conditions on an artificial rectangular box in order to be able to use the cosine series to represent the solution as in [32]. The computational advantages are two fold. First, fewer Fourier modes are needed in the approximation of the heat kernel. Second, we only need type-1 and type-2 NUFFTs instead of the much more expensive type-3 NUFFT for evaluating the solution to the heat equation in an arbitrary set of points in physical space.

Another difference is that [22] assumes that the initial data is smooth, while we assume that it is discontinuous piecewise constant. Thus directly applying the method in [22] would decrease the order of accuracy sharply because the Fourier transform of the initial data decays slowly. We use the divergence theorem or Green's theorem to reduce the volume or area integral to a boundary integral, which reduces the dimension of the problem by one without sacrificing the high-order feature of the algorithm for smooth or piecewise-smooth boundaries.

We perform the computation either on or near the interface in both the solving stage and the thresholding stage. This makes our algorithm more efficient than the methods in [30,32] and enables it to achieve high order in space more easily. These advantages increase the appeal of our algorithm for certain applications in computational fluid dynamics. Our scheme may be extended to handle topological changes and multiphase flows such as wetting problems as in [30,32], but that is outside the scope of this paper.

Organization of the paper In Sect. 2, we present an NUFFT-based solver for the initial value problem of the heat equation with piecewise constant initial data in both two and

three dimensions. In Sect. 3, we discuss the thresholding step involving the root-finding procedure. Section 4 contains several numerical examples demonstrating the performance of the proposed numerical algorithm. Finally, we conclude the paper with further discussions and extensions.

2 An NUFFT-Based Solver for the Heat Equation in Free Space

In this section, we consider solving the pure initial value problem of the heat equation:

$$\begin{aligned} u_t &= \Delta u, \\ u(\mathbf{x}, 0) &= u_0(\mathbf{x}) \end{aligned} \tag{2}$$

with the initial data u_0 being the indicator function of some bounded domain $D \in \mathbb{R}^d$ ($d = 2, 3$). We assume that the boundary ∂D of D is a smooth manifold which is homomorphic to S^{d-1} (i.e., the unit circle in $\mathbb{R}^2$ and unit sphere in $\mathbb{R}^3$). We are interested in calculating $u(\mathbf{x}, \Delta t)$ for a given set of target points $\mathbf{x}$ (for example, in the neighborhood of ∂D) at some $\Delta t < 1$. Green's function (or the fundamental solution) of the heat equation in $\mathbb{R}^d$ is given by the formula

$$G_d(\mathbf{x}, \mathbf{y}; t) = \frac{1}{(4\pi t)^{d/2}} e^{-\|\mathbf{x}-\mathbf{y}\|^2/4t}, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^d. \tag{3}$$

$u(\mathbf{x}, \Delta t)$ is given by the formula

$$u(\mathbf{x}, \Delta t) = \int \cdots \int_{\mathbb{R}^d} G_d(\mathbf{x}, \mathbf{y}; \Delta t) u_0(\mathbf{y}) d\mathbf{y} = \int \cdots \int_{D} G_d(\mathbf{x}, \mathbf{y}; \Delta t) d\mathbf{y}, \tag{4}$$

where the second equality follows from (1).

2.1 Fourier Spectral Approximation of the Heat Kernel for a Fixed Time

It is well known that G_d admits the following Fourier representation:

$$G_d(\mathbf{x}, \mathbf{y}; t) = \frac{1}{(2\pi)^d} \int \cdots \int_{\mathbb{R}^d} e^{-\|\mathbf{k}\|^2 t + i\mathbf{k}\cdot(\mathbf{x}-\mathbf{y})} d\mathbf{k}, \quad \mathbf{k} \in \mathbb{R}^d. \tag{5}$$

We will now try to approximate the integral in (5) via a discrete summation. We first consider the approximation of the one-dimensional heat kernel

$$G_1(x, \Delta t) = \frac{1}{\sqrt{4\pi \Delta t}} e^{-x^2/4\Delta t} = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-k^2 \Delta t + ikx} dk. \tag{6}$$

Theorem 1 (Fourier Spectral Approximation of the One-Dimensional Heat Kernel for a Fixed Time) *Suppose that $\epsilon < 1/2$ is the prescribed accuracy. Let $h = \min\left(\frac{\pi}{R}, \frac{\pi}{2\sqrt{\Delta t |\ln \epsilon|}}\right)$ and $M = \frac{1}{h}\sqrt{\frac{\ln(\sqrt{\pi}\epsilon/2h\sqrt{\Delta t})}{\Delta t}}$. Then for all $|x| \le R$,*

$$\left| G_1(x, \Delta t) - \frac{h}{2\pi} \sum_{m=-M}^{M-1} e^{-m^2 h^2 \Delta t + imhx} \right| \le \frac{4\epsilon}{\sqrt{4\pi \Delta t}}. \tag{7}$$

Proof Let $\hat{f}$ be the Fourier transform of f, that is,

$$\hat{f}(k) = \int_{\mathbb{R}} e^{-ikx} f(x) dx, \qquad f(x) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{ikx} \hat{f}(k) dk. \tag{8}$$

Then (6) shows that the Fourier transform $\hat{G}_1$ of the one-dimensional heat kernel is

$$\hat{G}_1(k, \Delta t) = e^{-k^2 \Delta t}. \tag{9}$$

The Poisson summation formula (see, for example, [6]) states that

$$\sum_{n=-\infty}^{\infty} f\left(x + \frac{2\pi n}{h}\right) = \frac{h}{2\pi} \sum_{m=-\infty}^{\infty} \hat{f}(mh) e^{imhx}. \tag{10}$$

Rearranging terms yields

$$f(x) - \frac{h}{2\pi} \sum_{m=-\infty}^{\infty} \hat{f}(mh) e^{imhx} = - \sum_{n=-\infty, n\neq 0}^{\infty} f\left(x + \frac{2\pi n}{h}\right). \tag{11}$$

Applying (11) to G_1 and knowing that the heat kernel is positive, we obtain

$$\left| G_1(x, \Delta t) - \frac{h}{2\pi} \sum_{m=-\infty}^{\infty} e^{-m^2 h^2 \Delta t + imhx} \right| \leq \sum_{n=-\infty, n\neq 0}^{\infty} G_1\left(x + \frac{2\pi n}{h}, \Delta t\right). \tag{12}$$

We would like to bound the right-hand side of (12) for all $|x| \leq R$. To do so, we first require that h satisfy the condition $\frac{2\pi}{h} - R \geq R$, i.e., $h \leq \frac{\pi}{R}$. Note that with this condition, we also have $\frac{2\pi}{h} - R \geq \frac{\pi}{h}$. Assume for now that $0 \leq x \leq R$. Then

$$\left| x + \frac{2\pi n}{h} \right| = x + \frac{2\pi n}{h} > \frac{2\pi n}{h}, \quad \text{for } n > 1, \tag{13}$$

and

$$\left| x + \frac{2\pi n}{h} \right| = \frac{2\pi |n|}{h} - x > \frac{\pi(2|n| - 1)}{h}, \quad \text{for } n < -1. \tag{14}$$

Similar results hold for $x \in [-R, 0]$. To summarize, if $h \leq \frac{\pi}{R}$, then

$$\sum_{n=-\infty, n\neq 0}^{\infty} G_1\left(x + \frac{2\pi n}{h}, \Delta t\right) \leq \sum_{n=1}^{\infty} G_1\left(\frac{\pi n}{h}, \Delta t\right) \quad \text{for } |x| \leq R. \tag{15}$$

Thus, if $h = \frac{\pi}{2\sqrt{\Delta t |\ln \epsilon|}}$, then we have $e^{-\pi^2/(4h^2 \Delta t)} = \epsilon$ and

$$\sum_{n=1}^{\infty} G_1\left(\frac{\pi n}{h}, \Delta t\right) < 2 G_1\left(\frac{\pi}{h}, \Delta t\right) = \frac{2\epsilon}{\sqrt{4\pi \Delta t}}. \tag{16}$$

Finally, we would like to truncate the infinite Fourier series. To do so, we choose M such that $\frac{h}{\pi} e^{-M^2 h^2 \Delta t} = \frac{\epsilon}{\sqrt{4\pi \Delta t}}$, i.e., $M = \frac{1}{h} \sqrt{\frac{\ln(\sqrt{\pi}\epsilon / 2h\sqrt{\Delta t})}{\Delta t}}$. This leads to

$$\left| \frac{h}{2\pi} \left(\sum_{m=-M-1}^{-\infty} + \sum_{m=M}^{\infty} \right) e^{-m^2 h^2 \Delta t + imhx} \right| \leq \frac{h}{\pi} \sum_{m=M}^{\infty} e^{-m^2 h^2 \Delta t} \leq \frac{2\epsilon}{\sqrt{4\pi \Delta t}}. \tag{17}$$

Combining (12), (15), (16), and (17) yields (7). □

Note that the trapezoidal rule achieves spectral accuracy for integrals involving infinitely long contours with a rapidly decaying smooth integrand (see, for example, [34] for a comprehensive review on this topic).

Table 1 Number of Fourier nodes needed to approximate the one-dimensional heat kernel $G_1(x, \Delta t)$ for $x \in [-\pi, \pi]$

ϵ	Δt			
	10^{-1}	10^{-2}	10^{-3}	10^{-4}
10^{-3}	8	21	55	136
10^{-6}	11	34	100	296
10^{-9}	14	43	130	396
10^{-12}	17	50	154	475

To get an idea of how many Fourier nodes are needed in practice, we list the values of M in Table 1 for various levels of precision ϵ and time steps Δt for the approximation of $G_1(x, \Delta t)$ for $|x| \leq \pi$.

Since $G_d(\mathbf{x}, \Delta t) = \prod_{i=1}^{d} G_1(x_i, \Delta t)$, we simply use the tensor product to construct the Fourier spectral approximation of the heat kernel in higher dimensions. That is,

$$G_d(\mathbf{x}, \mathbf{y}; \Delta t) \approx \frac{h^d}{(2\pi)^d} \prod_{i=1}^{d} \left(\sum_{m_i=-M}^{M-1} e^{-m_i^2 h^2 \Delta t + i h m_i \cdot (x_i - y_i)} \right), \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^d. \tag{18}$$

Obviously $G_d(\mathbf{x}, \mathbf{y}; \Delta t)$ decays exponentially quickly and is almost negligible when $\|\mathbf{x} - \mathbf{y}\| > 10\sqrt{\Delta t}$ in physical space. Thus, if R is very large or Δt is very small, we can take advantage of the fact that the heat kernel has a sharp peak at the origin and design more efficient algorithms accordingly (for example, using fast Gauss transform [15,16]) .

2.2 Solving the Pure Initial Value Problem

We substitute (5) into (4) and change the order of integration to obtain

$$u(\mathbf{x}, \Delta t) = \frac{1}{(2\pi)^d} \int \cdots \int_{\mathbb{R}^d} e^{-\|\mathbf{k}\|^2 \Delta t + i \mathbf{k} \cdot \mathbf{x}} f(\mathbf{k}) d\mathbf{k}, \tag{19}$$

where $f(\mathbf{k})$ is given by the formula

$$f(\mathbf{k}) = \int \cdots \int_{D} e^{-i \mathbf{k} \cdot \mathbf{y}} d\mathbf{y}. \tag{20}$$

Note that $f(\mathbf{k})$ in (20) is a C^∞ function since D is a bounded domain in $\mathbb{R}^d$. We can also show that $f(\mathbf{k})$ tends to 0 as $\|\mathbf{k}\| \to \infty$ (see, for example, [28]).

For two-dimensional problems, we can convert the double integral in (20) into a line integral using Green's theorem as follows:

$$f(\mathbf{k}) = \iint_D e^{-i \mathbf{k} \cdot \mathbf{y}} d\mathbf{y} = \begin{cases} \dfrac{i}{k_1} \displaystyle\int_{\partial D} e^{-i \mathbf{k} \cdot \mathbf{y}} dy_2, & k_1 \neq 0, \\ \displaystyle\int_{\partial D} y_1 e^{-i k_2 y_2} dy_2, & k_1 = 0. \end{cases} \tag{21}$$

Since the boundary curve ∂D is smooth, the above integrals can be discretized using the trapezoidal rule to achieve spectral accuracy. To be more precise, let $s \in [0, 2\pi]$ be the parameter for the boundary curve and N be the total number of discretization points (equispaced in the parameter space but non-equispaced in physical space) on ∂D. Then we have

$$f(\mathbf{k}) \approx \begin{cases} \dfrac{2\pi i}{Nk_1} \displaystyle\sum_{j=1}^{N} e^{-i\mathbf{k}\cdot\mathbf{y}(s_j)} y_2'(s_j), & k_1 \neq 0, \\ \dfrac{2\pi}{N} \displaystyle\sum_{j=1}^{N} y_1(s_j) e^{-ik_2 y_2(s_j)} y_2'(s_j), & k_1 = 0. \end{cases} \tag{22}$$

Note that for a smooth curve we can compute the derivatives $\mathbf{y}'(s_j)$ and the unit normal vectors $\mathbf{n}_y$ from $\mathbf{y}(s_j)$ $(j = 1, \ldots, N)$ using the FFT with spectral accuracy (see, for example, [36]). Furthermore, the area bounded by ∂D can be computed as follows:

$$A = \iint_D e^{-i\mathbf{k}\cdot\mathbf{y}} d\mathbf{y} = \int_{\partial D} y_1 dy_2 \approx \frac{2\pi}{N} \sum_{j=1}^{N} y_1(s_j) y_2'(s_j). \tag{23}$$

Similarly, for three-dimensional problems, we can convert the volume integral in (20) into a surface integral using the divergence theorem as follows:

$$f(\mathbf{k}) = \iiint_D e^{-i\mathbf{k}\cdot\mathbf{y}} d\mathbf{y} = \begin{cases} \dfrac{i}{k_1} \displaystyle\iint_{\partial D} e^{-i\mathbf{k}\cdot\mathbf{y}} (\hat{i}\cdot\mathbf{n}_y) ds_y, & k_1 \neq 0, \\ \displaystyle\iint_{\partial D} y_1 e^{-ik_2 y_2 - ik_3 y_3} (\hat{i}\cdot\mathbf{n}_y) ds_y, & k_1 = 0, \end{cases} \tag{24}$$

where $\hat{i}$ is the unit vector along the x-axis and $\mathbf{n}_y$ is the unit normal vector at $\mathbf{y}$. Since we have assumed that ∂D is homomorphic to the unit sphere, we can parametrize ∂D using $[u, v] \in [0, \pi] \times [0, 2\pi]$ where u is the polar angle and v is the azimuthal angle. We discretize the integration in v using the trapezoidal rule and the integration in u using the Gauss-Chebyshev quadrature. Hence, the integrals in (24) are approximated by:

$$f(\mathbf{k}) \approx \begin{cases} \dfrac{i}{k_1} \displaystyle\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} e^{-i\mathbf{k}\cdot\mathbf{y}(u_i, v_j)} n_1(u_i, v_j) J(u_i, v_j) w_{ij}, & k_1 \neq 0, \\ \displaystyle\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} y_1(u_i, v_j) e^{-ik_2 y_2(u_i, v_j) - ik_3 y_3(u_i, v_j)} n_1(u_i, v_j) J(u_i, v_j) w_{ij}, & k_1 = 0, \end{cases} \tag{25}$$

where $n_1(u_i, v_j)$ is the x-component of the unit outward normal vector, $J(u_i, v_j) = |\mathbf{y}_u \times \mathbf{y}_v|$ is the Jacobian at the point (u_i, v_j), and w_{ij} is the quadrature weight at that point. Note again that provided we have the coordinates $\mathbf{y}(u_i, v_j)$, all other quantities can be computed efficiently since both $\mathbf{y}_u$ and $\mathbf{y}_v$ can be calculated using the FFT [36]. Furthermore, the volume bounded by ∂D can be computed as follows:

$$\begin{aligned} V &= \iiint_D d\mathbf{y} = \iint_{\partial D} x\hat{i}\cdot\mathbf{n}_y ds_y \\ &\approx \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} x(u_i, v_j) n_1(u_i, v_j) J(u_i, v_j) w_{ij}. \end{aligned} \tag{26}$$

After computing $f(\mathbf{k})$, the solution $u(\mathbf{x}, \Delta t)$ in (19) can be evaluated by approximating the Fourier integral using the truncated trapezoidal rule as in Theorem 1. Thus we have

$$u(\mathbf{x}, \Delta t) \approx \frac{h^d}{(2\pi)^d} \sum_{m_1=-M}^{M-1} \cdots \sum_{m_d=-M}^{M-1} e^{-\|\mathbf{m}\|^2 h^2 \Delta t + ih\mathbf{m}\cdot\mathbf{x}} f(h\mathbf{m}). \tag{27}$$

Recall the definitions of NUFFTs. We follow the convention in [13] to define type-1 and type-2 NUFFTs. The type-1 NUFFT evaluates sums of the form

$$f(\mathbf{k}) = \frac{1}{N} \sum_{j=1}^{N} c_j e^{\pm i \mathbf{k} \cdot \mathbf{x}_j}, \tag{28}$$

for "targets" $\mathbf{k}$ on a regular (equispaced) grid in $\mathbb{R}^d$ ($d = 1, 2, 3$ in our case), given function values c_j prescribed at arbitrary locations $\mathbf{x}_j$ in physical space. Here, N denotes the total number of source points.

The type-2 NUFFT evaluates sums of the form

$$F(\mathbf{x}_n) = \sum_{m_1=-M_1}^{M_1-1} \cdots \sum_{m_d=-M_d}^{M_d-1} f(h\mathbf{m}) e^{\pm i h \mathbf{m} \cdot \mathbf{x}_n}, \tag{29}$$

where the "targets" $\mathbf{x}_n$ are points irregularly located in $\mathbb{R}^d$, given the function values $f(h\mathbf{m})$ on a regular grid in the Fourier space.

Clearly we can apply the type-1 NUFFT to evaluate $f(h\mathbf{m})$ defined in (22) or (25), and then apply the type-2 NUFFT to evaluate $u(\mathbf{x}, \Delta t)$ in (27).

We summarize the algorithm in Algorithm 1. The total cost of Algorithm 1 is $O(N_S + N_T + N_F \log N_F)$, where N_S is the total number of non-equispaced source points on the boundary, N_T is the total number of target points near the boundary, and $N_F = (2M)^d$ is the total number of equispaced points in the Fourier space.

Algorithm 1 An NUFFT-Based Solver for the Initial Value Problem (2)

Given the prescribed accuracy ϵ, the time step size Δt, and the boundary ∂D, compute $u(\mathbf{x}, \Delta t)$ defined in (4) on a set of prescribed target points $\mathbf{x}_i$, $i = 1, \cdots, N_T$.

1: Discretize the boundary ∂D via N_S points in the parameter space (equispaced nodes for curves in 2D; scaled and shifted Chebyshev nodes along the polar direction and equispaced nodes along the azimuthal direction for surfaces in 3D), and compute the source locations $\mathbf{y}_j$ ($j = 1, \ldots, N_s$) via the given parametrization of ∂D and the associated weights w_j.

2: Compute the derivatives $\mathbf{y}'_j$ and the Jacobians using the FFT.

3: Compute the mesh spacing h and M for the Fourier spectral approximation of the heat kernel as in Theorem 1.

4: Use the type-1 NUFFT to evaluate $f(h\mathbf{m})$ defined in (22) or (25) for $m_i = -M, \cdots, M-1$ ($i = 1, \ldots, d$).

5: Use the type-2 NUFFT to evaluate $u(\mathbf{x}_j, \Delta t)$ defined in (27) for $j = 1, \cdots, N_T$.

3 Thresholding

We now discuss how the thresholding is performed in our algorithm. For grid- or mesh-based methods, the accuracy of the thresholding step is generally of the first order. However, due to the smoothing effect of the diffusion process, the interface generally becomes smoother and thus it is possible to improve the accuracy of thresholding substantially. As mentioned in the preceding section, we use a carefully chosen set of points to represent the interface in both two and three dimensions from which all other geometric quantities such as the tangential derivatives, unit normal vectors, and area elements can be computed efficiently. Hence, we only need to keep track of this set of points in the threshold dynamics.

Thresholding begins by finding the level set of a given solution value v to the initial value problem (2). The algorithm is as follows. We start from a set of source points $\mathbf{y}_j$ ($j = 1, \ldots, N_s$). We then apply the FFT to compute the unit normal vector to each source point $\mathbf{y}_j$ and allocate p target points along each normal direction on both sides of the source point. After applying Algorithm 1 to compute the solution to the initial value problem (2), we use a root-finding algorithm (Müller's method in [25]) to find a point whose solution value is equal to v along each normal direction. We summarize the whole scheme in Algorithm 2.

Algorithm 2 Computing the level set at a given function value v

Given the prescribed accuracy ϵ, the time step size Δt, a set of source points $\mathbf{y}_j$ ($j = 1, \ldots, N_S$) describing the interface at the current time, and a specified function value v, compute the level set for the function value v and return a new set of N_S points representing the new interface after diffusion and thresholding, at which the solution value of the initial value problem is equal to v. Also return the area or the volume bounded by the new level set.

1: Use the FFT to compute the derivatives $\mathbf{y}'_j$ and the unit normal vector $\mathbf{n}_j$ for $j = 1, \ldots, N_S$.
2: Compute a length L that is proportional to $\sqrt{\Delta t}$, say, $L = 5\sqrt{\Delta t}$.
3: Compute p Gauss-Legendre nodes on the standard interval $[-1, 1]$.
4: For each point $\mathbf{y}_j$ on the boundary, allocate along the normal direction p scaled Gauss-Legendre nodes on the interval $[-L, L]$ centered at $\mathbf{y}_j$; altogether we obtain pN_S target points $\mathbf{x}_i$ for $i = 1, \cdots, pN_S$.
5: Apply Algorithm 1 to compute the solution u_c to the initial value problem (2) at these target points.
6: For each point $\mathbf{y}_j$ on the boundary along its normal direction, approximate $u(\mathbf{x}(s), \Delta t)$ with a $p-1$th order Legendre polynomial and use Müller's method [25] to find the parameter value $s \in [-L, L]$ at which the function value is equal to the given value v.
7: Calculate the coordinates of the points in the level set by setting $\mathbf{y}_j = \mathbf{y}_j + s \cdot \mathbf{n}_j$.
8: Use (23) or (26) to compute the area or the volume bounded by the new level set.

Remark 1 Several remarks are in order regarding the algorithm. First, the root-finding procedure converges rapidly. It only takes 5–6 iterations to converge to 12-digit accuracy. Second, the total number of points in physical space is pN_s, where p is some number around 10. We set p to 16 in our implementation. Third, the original MBO method [24], i.e., the motion by mean curvature, corresponds to the case when $v = \frac{1}{2}$.

When Algorithm 2 is applied to find the level set whose corresponding function value causes the interface to shrink, as in the original MBO method, numerical instability may occur for prolonged simulations due to the cross-over of the normal vectors. However, one can easily overcome the instability by filtering out the high frequency content in the point set. For curves in two dimensions, the procedure is as follows: (1) compute the Fourier transform $\hat{f}_i$ ($i = 1, \ldots, M$) of the coordinates; (2) multiply $\hat{f}_i$ by a smooth filter, e.g. $h(k) = \frac{1}{2}\mathrm{erfc}\left(\frac{12(|k|-(M/2+M)/2)}{M-M/2}\right)$, to gradually filter out the high frequency part $\hat{f}_i$ for $i \in [M/2, M]$ to zero, where erfc is the complementary error function defined by the formula $\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2} dt$; and (3) apply the inverse Fourier transform to obtain a smoother point set. For surfaces in three dimensions, the procedure is almost identical except that the Fourier transform in step (1) is replaced by the forward spherical harmonic transform and the inverse Fourier transform in step (3) is replaced by the backward spherical harmonic transform.

Many applications require finding the area- or volume-preserving level set. We may simply add another layer of iterations for this. The algorithm is summarized in Algorithm 3. Our numerical experiments show that 4–6 iterations are required to achieve 12-digit accuracy and

only 2–3 iterations are needed to reach single precision. Obviously, the cost of the root-finding procedure in Algorithms 2 and 3 is $O(N_s)$.

Algorithm 3 Calculating the area- or volume-preserving level set

Given the prescribed accuracy ϵ, the time step size Δt, and a set of source points $\mathbf{y}_j$ ($j = 1, \ldots, N_S$) describing the interface at the current time, return a new set of N_S points representing the new interface after diffusion and thresholding, whose area or volume is equal to that bounded by the original set of points.

1: Use (23) or (26) to compute the area or the volume bounded by the original set of source points.
2: Set initial guesses for the solution values v_0, v_1, v_2 randomly but close to 0.5 for example, and apply Algorithm 2 to find the level sets and the areas or the volumes associated with v_i ($i = 0, 1, 2$).
3: Use Müller's method to find the solution value v and associated level set whose enclosed area or volume is equal to that bounded by the original set of points.
4: Return the new set of points and the solution value v.

4 Numerical Results

We implemented the aforementioned algorithms in Fortran and MATLAB. We used the NUFFT library from [20]. We now illustrate the performance of our algorithm via several numerical examples. The timing results were obtained on a laptop with a 2.7 GHz Intel Core i5 processor and 8 GB of RAM.

Example 1 (Accuracy of the NUFFT-Based Heat Solver) We first check the accuracy of the NUFFT-based heat solver in Algorithm 1 in both two and three dimensions. For the two-dimensional solver, we use the hexagram in Example 4 as the boundary and compute the solution to the initial value problem (2)-(1) at $\Delta t = 0.002$. The reference solution is obtained with $N_S = 1024$ points on the boundary. The numerical solutions are evaluated at $N_T = p \cdot N_S = 16 \times 1024$ fixed target points with $p = 16$ points along each normal direction of the source points. Table 2 shows the relative L^2 error of the numerical solution with various numbers of source points on the boundary.

For the 3D solver, we use a sphere of radius 0.5 as the boundary and compute the solution to the heat equation at $\Delta t = 0.001$. The reference solution is obtained with $N_S = 256 \times 256$ points on the boundary. We evaluate the numerical solution at $N_T = p \cdot N_S = 16 \times 256 \times 256$ fixed target points with $p = 16$ points along each normal direction. Table 3 shows the relative L^2 error of the numerical solution with various numbers of source points on the boundary.

Tables 2 and 3 show that our heat solver is spectrally accurate in space for both two- and three-dimensional problems.

Example 2 (Efficiency and Accuracy of the Threshold Dynamics Algorithm) In this example, we consider the motion by mean curvature of a circle with a radius of 0.5 centered at the

Table 2 Relative L^2 error versus number of discretization points on the boundary for the two-dimensional heat solver

N_S	30	60	90	120	150
E	5.738e−2	2.077e−4	2.373e−7	1.211e−10	3.5438e−14

The boundary curve is a smooth hexagram in Example 4

Table 3 Relative L^2 error versus number of discretization points on the boundary for the 3D heat solver

N_S	40×40	60×60	80×80	100×100	120×120
E	5.104e−4	2.519e−7	6.758e−10	2.929e−12	6.229e−15

The boundary is a sphere of radius 0.5

Table 4 Accuracy and timing results of Algorithm 2 and the uniform mesh method

Δt	#(Source points)	Error	Time	#(Fourier modes)
(a) Algorithm 2				
0.002	400	0.28%	0.49 s	78×78
0.001	400	0.14%	0.97 s	112×112

Δt	Δx	Error	Time
(b) The uniform mesh method			
0.002	$\frac{1}{2048}$	0.32%	10.63 s
0.001	$\frac{1}{4096}$	0.15%	62.65 s

Table 5 Relative error and convergence order in time of Algorithm 2 for the motion by mean curvature of a circle

Δt	Relative error	Convergence order
0.004	0.005941	–
0.002	0.002938	1.02
0.001	0.001461	1.01
0.0005	0.000728	1.01

origin. As discussed in Remark 1, motion by mean curvature is equivalent to finding the level set with the solution value equal to 0.5. We March to the final time $T = 0.01$ and compare the area loss with the exact answer $0.01 \times 2\pi$ [26]. Table 4 shows the relative error of the area loss and the total computational time of Algorithm 2 and the method in [38], which is an FFT-based fast solver on a uniform mesh. Since Algorithm 2 only requires discretization near the boundary and the spatial mesh size can be chosen more or less independently of the time step size, fewer discretization points are needed in space to achieve the same accuracy. We can see from the table that Algorithm 2 is significantly more efficient than the uniform mesh method, with the latter taking 20 times longer for the case $\delta t = 0.002$ and 60 times longer for the case $\delta t = 0.001$.

We now examine the order of accuracy in time. According to [23,24,32], the MBO method is first order in time for smooth two-dimensional problems. To test the order of accuracy in time of our algorithm, we consider the same problem and March to $T = 0.02$ using Algorithm 2, compute the area loss, and compare it with the exact answer -0.04π. Table 5 shows the relative error and convergence order of our algorithm. It can be seen that our algorithm has the predicted first-order accuracy in time. Here we have used 400 points to discretize the boundary.

Finally, we consider the motion by mean curvature of a sphere of radius 0.5. The sphere shrinks without changing its shape since it has constant curvature and the dynamics of the radius of is governed by [17]:

Table 6 Numerical results for the motion by mean curvature of a sphere

Δt	Error	Conv. order	Time (s)	#(Fourier modes)
0.004	1.365e−03	–	25.27	$72 \times 72 \times 72$
0.002	6.741e−04	1.02	56.96	$90 \times 90 \times 90$
0.001	3.350e−04	1.01	149.73	$120 \times 120 \times 120$
0.0005	1.670e−04	1.01	472.57	$160 \times 160 \times 160$

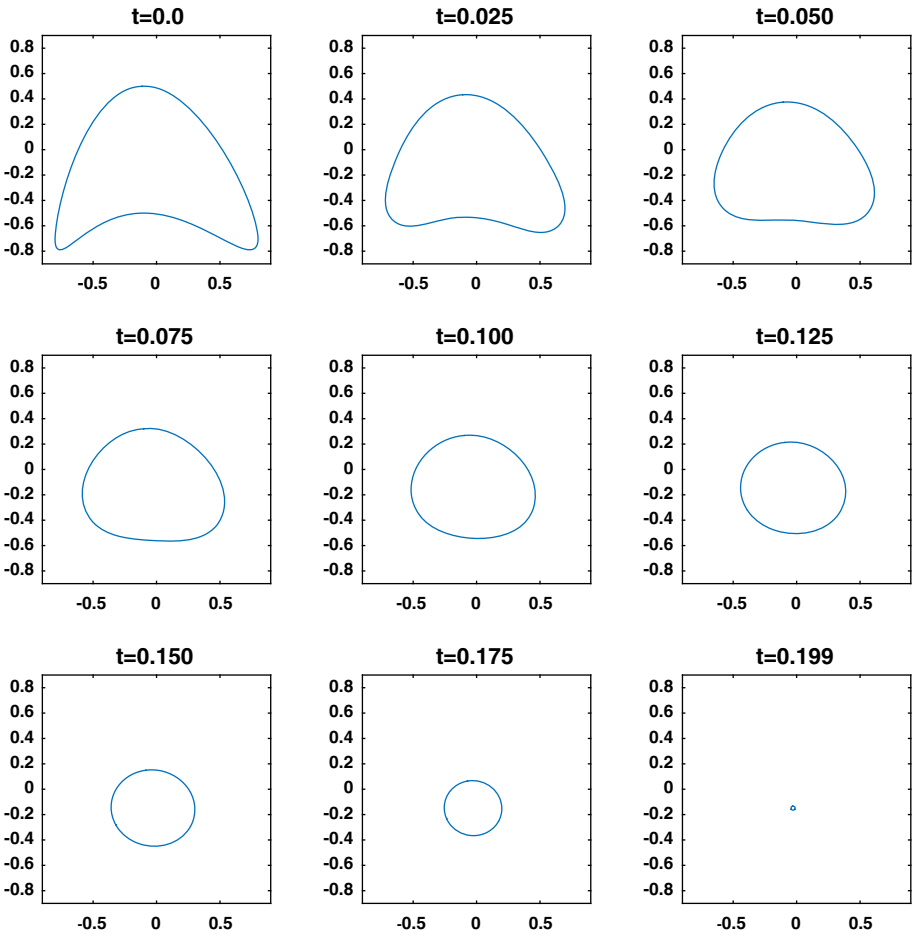

Fig. 1 The motion by mean curvature of a crescent at various times. We set $\Delta t = 0.0005$ and use 400 points to discrete the interface

$$\frac{dr(t)}{dt} = -\frac{2}{r(t)}, \qquad r(0) = r_0, \tag{30}$$

where $r(t)$ is the radius of the sphere at time t. The exact solution to the above ordinary different equation is

$$r(t) = \sqrt{r_0^2 - 4t}. \tag{31}$$

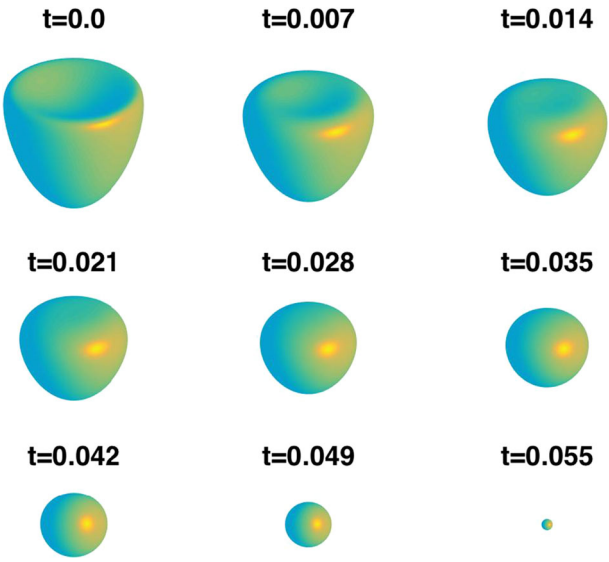

Fig. 2 Snapshots of the motion by mean curvature of a bowl. $\Delta t = 0.001$

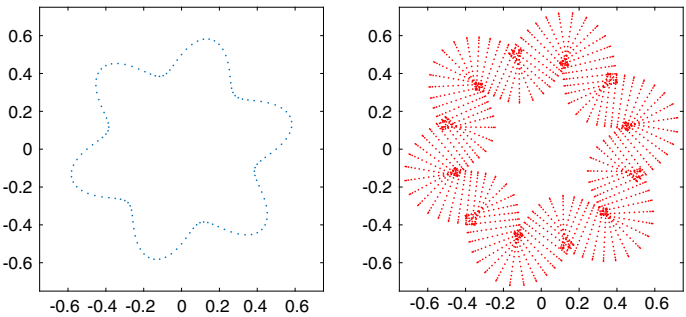

Fig. 3 *Left*: a subset of the source points at $t = 0$. *Right*: a subset of the target points at $t = 0$

We apply Algorithm 2 to March to $T = 0.02$, calculate the volume loss and compare it against the exact value 0.229995 obtained from (31). We use 96×96 points to discretize the sphere. Table 6 shows the relative L^2 error, the estimated convergence order in time, and the total computational time in seconds for various time step sizes Δt. Our algorithm has first-order accuracy in time and is fairly efficient for three-dimensional problems.

Remark 2 While it is spectrally accurate in space, our algorithm is only first-order accurate in time, consistent with the original MBO method. However, many researchers have used extrapolation to achieve high order in time as well. Interested readers can refer to [23,29,30] for these high-order temporal schemes. Our algorithm may be extended to high order in time in a similar fashion.

Example 3 (Interface Motion by Mean Curvature in 2D and 3D) We now show that our algorithm is capable of handling various smooth geometries in both 2D and 3D. We first compute the motion by mean curvature of a crescent in 2D using a step size $\Delta t = 0.0005$.

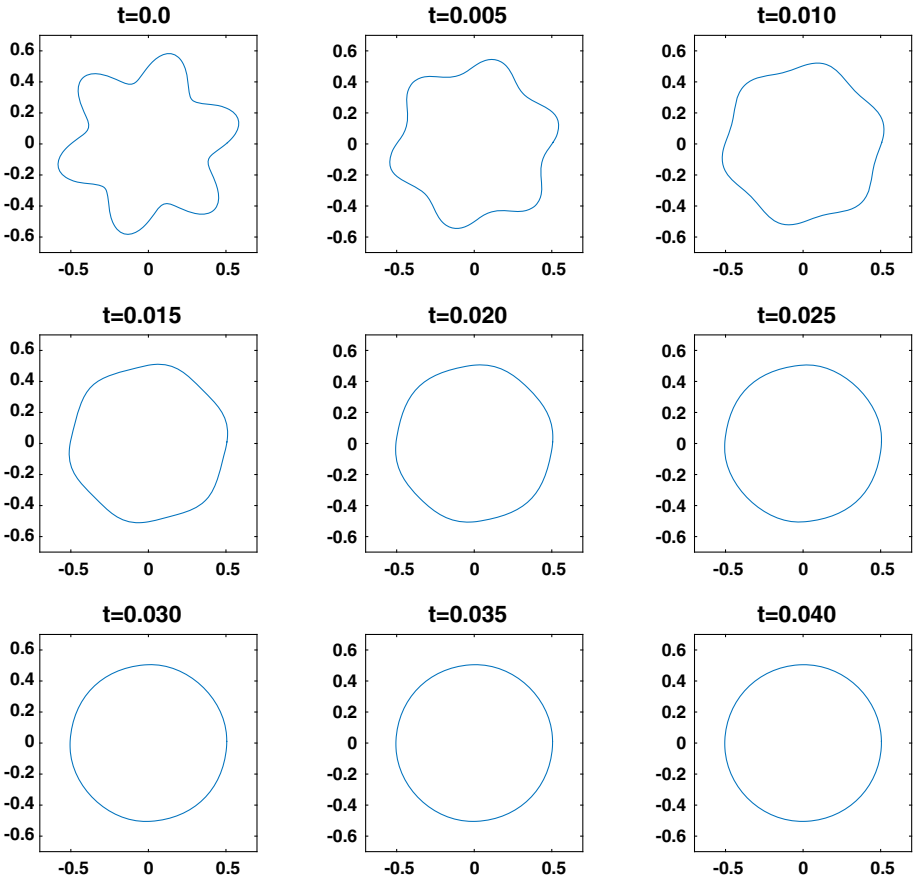

Fig. 4 Snapshots of the motion by mean curvature of a hexagram under the area-preserving constraint. $\Delta t = 0.001$

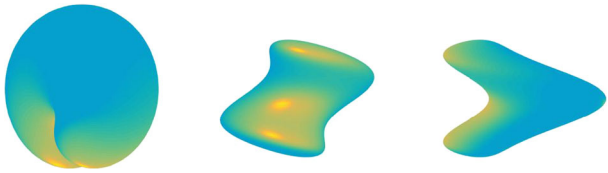

Fig. 5 The *front*, *side*, and *vertical* views of the initial geometric profile in Example 5

The dynamics at a set of time points is shown in Fig. 1. From these graphs, we can see that the topology has not changed and that the boundary eventually becomes a circle as it shrinks.

Next we compute the dynamics of a smooth surface in 3D. Figure 2 shows the motion by mean curvature of a bowl-shaped initial region. We set $\Delta t = 0.001$ and use 96×192 points to discretize the surface. The bowl evolves into a sphere and shrinks as t increases.

Example 4 (Area-Preserving Motion of a Hexagram in 2D) We consider the motion of a hexagram under mean curvature with the area-preserving constraint. We set the time step $\Delta t = 0.001$ and discretize the boundary with 400 points. The number of points along each normal direction is set to $p = 16$. Figure 3 shows subsets of the source and target points for

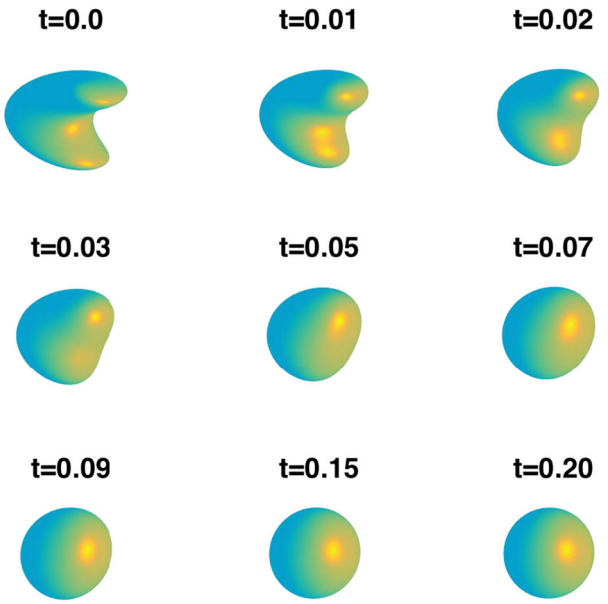

Fig. 6 Snapshots of the volume preserving motion of a wedge. $\Delta t = 0.001$

the initial geometry. Figure 4 shows the motion of the interface at various times. From these graphs, we can see that the shape eventually becomes a circle, which is in agreement with the exact solution at equilibrium. Furthermore, the initial area has numerical value $A = 0.801106$. From this, we may obtain the radius of the equilibrium circle $r = \sqrt{A/\pi} = 0.504975$. We have computed the relative L^∞ error of the shape at $t = 0.04$ by comparing it with the equilibrium circle and found it to be 0.000459. This example shows that our algorithm can simulate motion by mean curvature under the area-preserving constraint reasonably accurately.

Example 5 (Volume-Preserving Motion of a Wedge in 3D) To show the capability and efficiency of our method in simulating the dynamics of a complicated surface in 3D, we compute the volume-preserving motion of a wedge in 3D. Figure 5 shows the initial shape of the wedge from various angles. We set $\Delta t = 0.001$ and use 96×96 points to discretize the surface. Figure 6 displays snapshots at different times of the volume-preserving motion of the wedge. Again, the wedge becomes a sphere when it reaches equilibrium. The simulation takes approximately 2068 seconds only. For such 3D simulations, the uniform mesh method with a similar accuracy would require at least $1024 \times 1024 \times 1024$ grid points and would take roughly 200 hours to reach equilibrium.

5 Conclusions and Further Discussions

We have presented an algorithm for the threshold dynamics of interface motion. The algorithm discretizes physical space only in a neighborhood of the interface and applies the NUFFT to solve the initial value problem of the heat equation. Unlike many grid-based methods where the spatial mesh size is required to be of the same order as the time step size, our

numerical experiments show that the spatial mesh size can be chosen based upon the accuracy consideration only and thus is more or less independent of the time step size for our algorithm. Hence, our algorithm is efficient and robust for smooth interfaces if the time step is not too small. When the time step is very small, the number of Fourier modes in the spectral approximation of the heat kernel becomes excessively large and the accuracy of our algorithm decreases, especially for three-dimensional problems. However, other fast algorithms such as the fast Gauss transform or even asymptotic analysis can be used to obtain the solution to the heat equation for the diffusion stage in this regime.

For two-dimensional problems, our algorithm can be easily extended to treat piecewise smooth boundaries such as polygons, or multiphase flows such as wetting problems on a solid surface with a given contact angle. For three-dimensional problems, it is straightforward to modify our algorithm to handle the wetting problems on a flat solid plane. Our algorithm can also be extended to treat cases involving topological changes. We are currently working on these extensions.

Acknowledgements S. Jiang was supported by the National Science Foundation under Grant DMS-1418918. X. P. Wang was supported in part by the Hong Kong RGC-GRF Grants 605513 and 16302715, RGC-CRF Grant C6004-14G, and NSFC-RGC joint research Grant N-HKUST620/15.

References

1. Barles, G., Georgelin, C.: A simple proof of convergence for an approximation scheme for computing motions by mean curvature. SIAM J. Numer. Anal. **32**(2), 484–500 (1995)
2. Chambolle, A., Novaga, M.: Convergence of an algorithm for the anisotropic and crystalline mean curvature flow. SIAM J. Math. Anal. **37**(6), 1978–1987 (2006)
3. Deckelnick, K., Dziuk, G., Elliott, C.M.: Computation of geometric partial differential equations and mean curvature flow. Acta Numerica **14**, 139–232 (2005)
4. Dutt, A., Rokhlin, V.: Fast Fourier transforms for nonequispaced data. SIAM J. Sci. Comput. **14**, 1368–1393 (1993)
5. Dutt, A., Rokhlin, V.: Fast Fourier transforms for nonequispaced data. II. Appl. Comput. Harmon. Anal. **2**, 85–100 (1995)
6. Dym, H., McKean, H.P.: Fourier Series and Integrals. Academic Press, Cambridge (1972)
7. Esedoglu, S., Otto, F.: Threshold dynamics for networks with arbitrary surface tensions. Comm. Pure Appl. Math. **68**, 808–864 (2015)
8. Esedoglu, S., Ruuth, S., Tsai, R.: Threshold dynamics for high order geometric motions. Interf. Free Bound. **10**(3), 263–282 (2008)
9. Esodoglu, S., Smereka, P.: A variational formulation for a level set representation of multiphase flow and area preserving curvature flow. Comm. Math. Sci. **6**(1), 125–148 (2008)
10. Evans, L.C.: Convergence of an algorithm for mean curvature motion. Indiana Math. J. **42**(2), 533–557 (1993)
11. Evans, L.C.: Partial Differential Equations, 2nd edn. American Mathematical Society, Providence (2010)
12. Gao, M., Wang, X.: A gradient stable scheme for a phase field model for the moving contact line problem. J. Comput. Phys. **231**, 1372–386 (2012)
13. Greengard, L., Lee, J.Y.: Accelerating the nonuniform fast Fourier transform. SIAM Rev. **46**(3), 443–454 (2004)
14. Greengard, L., Lin, P.: Spectral approximation of the free-space heat kernel. Appl. Comput. Harmon. Anal. **9**(1), 83–97 (2000)
15. Greengard, L., Strain, J.: The fast Gauss transform. SIAM J. Sci. Statist. Comput. **12**(1), 79–94 (1991)
16. Greengard, L., Sun, X.: A new version of the fast Gauss transform. Proc. Int. Cong. Math. **III**, 575–584 (1998)
17. Ilmanen, T.: Lectures on Mean Curvature Flow and Related Equations, Lecture Notes. ICTP, Trieste (1995). http://www.math.ethz.ch/~ilmanen/papers/pub.html
18. Ishii, K.: Optimal rate of convergence of the Bence–Merriman–Osher algorithm for motion by mean curvature. SIAM J. Math. Anal. **37**(3), 841–866 (2005)

19. Kublik, C., Esedoglu, S., Fessler, J.A.: Algorithms for area preserving flows. SIAM J. Sci. Comput. **33**(5), 2382–2401 (2011)
20. Lee, J.Y., Greengard, L., Gimbutas, Z.: NUFFT Version 1.3.2 Software Release. http://www.cims.nyu.edu/cmcl/nufft/nufft.html (2009)
21. Leung, S., Zhao, H.: A grid based particle method for moving interface problems. J. Comput. Phys. **228**(8), 2993–3024 (2009)
22. Li, J.R., Greengard, L.: On the numerical solution of the heat equation. I. Fast solvers in free space. J. Comput. Phys. **226**(2), 1891–1901 (2007)
23. Mascarenhas, P.: Diffusion Generated Motion by Mean Curvature. Department of Mathematics. University of California, Los Angeles, Los Angeles (1992)
24. Merriman, B., Bence, J.K., Osher, S.: Diffusion generated motion by mean curvature. UCLA CAM Report 92-18 (1992)
25. Müller, D.E.: A method for solving algebraic equations using an automatic computer. Math. Tables Aids Comput. **10**, 208–215 (1956)
26. Mullins, W.W.: Two-dimensional motion of idealized grain boundaries. J. Appl. Phys. **27**(8), 900–904 (1956)
27. Osher, S., Sethian, J.: Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations. J. Comput. Phys. **79**(1), 12–49 (1988)
28. Randol, B.: On the Fourier transform of the indicator function of a planar set. Trans. Amer. Math. Soc. **139**, 271–278 (1969)
29. Ruuth, S.: An algorithm for generating motion by mean curvature. ICAOS'96 pp. 82–91 (1996)
30. Ruuth, S.J.: Efficient algorithms for diffusion-generated motion by mean curvature. Ph.D. thesis, University of British Columbia, Vancouver, Canada (1996)
31. Ruuth, S.J.: A diffusion-generated approach to multiphase motion. J. Comput. Phys. **145**(1), 166–192 (1998)
32. Ruuth, S.J.: Efficient algorithms for diffusion-generated motion by mean curvature. J. Comput. Phys. **144**(2), 603–625 (1998)
33. Ruuth, S.J., Wetton, B.T.: A simple scheme for volume-preserving motion by mean curvature. J. Sci. Comput. **19**(1–3), 373–384 (2003)
34. Stenger, F.: Numerical methods based on Whittaker cardinal, or sinc functions. SIAM Rev. **23**(2), 165–224 (1981)
35. Svadlenka, K., Ginder, E., Omata, S.: A variational method for multiphase volume-preserving interface motions. J. Comput. Appl. Math. **257**, 157–179 (2014)
36. Trefethen, L.N.: Spectral Methods in MATLAB. SIAM, Philadelphia (2000)
37. Womble, D.E.: A front-tracking method for multiphase free boundary problems. SIAM J. Numer. Anal. **26**(2), 380–396 (1989)
38. Xu, X., Wang, D., Wang, X.P.: An efficient threshold dynamics method for wetting on rough surfaces. J. Comput. Phys. **330**, 510–528 (2017)
39. Yue, P., Feng, J., Liu, C., Shen, J.: A diffuse-interface method for simulating two-phase flows of complex fluids. J. Fluid Mech. **515**, 293–317 (2004)
40. Zhao, H., Chan, T., Merriman, B., Osher, S.J.: A variational level set approach to multiphase motion. J. Comput. Phys. **127**(1), 179–195 (1996)