Level set equations on surfaces via the Closest Point Method

Colin B. Macdonald[1⋆], **Steven J. Ruuth**[2⋆⋆]

[1] Department of Mathematics, Simon Fraser University, Burnaby, BC, V5A 1S6 Canada. e-mail: cbm@sfu.ca
[2] Department of Mathematics, Simon Fraser University, Burnaby, BC, V5A 1S6 Canada. e-mail: sruuth@sfu.ca

Preprint: 5 February 2008

Abstract Level set methods have been used in a great number of applications in $\mathbb{R}^2$ and $\mathbb{R}^3$ and it is natural to consider extending some of these methods to problems defined on surfaces embedded in $\mathbb{R}^3$ or higher dimensions. In this paper we consider the treatment of level set equations on surfaces via a recent technique for solving partial differential equations (PDEs) on surfaces, the Closest Point Method [18]. Our main modification is to introduce a Weighted Essentially Non-Oscillatory (WENO) interpolation step into the Closest Point Method. This, in combination with standard WENO for Hamilton–Jacobi equations, gives high-order results (up to fifth-order) on a variety of smooth test problems including passive transport, normal flow and redistancing. The algorithms we propose are straightforward modifications of standard codes, are carried out in the embedding space in a well-defined band around the surface and retain the robustness of the level set method with respect to the self-intersection of interfaces. Numerous examples are provided to illustrate the flexibility of the method with respect to geometry.

Key words Closest Point Method – level set methods – partial differential equations – implicit surfaces – WENO schemes – WENO interpolation

⋆ The work of this author was partially supported by a grant from NSERC Canada and a scholarship from the Pacific Institute for the Mathematical Sciences (PIMS).
⋆⋆ The work of this author was partially supported by a grant from NSERC Canada.

1 Introduction

The level set method [15] has been successfully applied to a tremendous variety of problems involving curve evolution in $\mathbb{R}^2$ or surface evolution in $\mathbb{R}^3$. This curve or surface—the *interface*—is represented as the zero contour of a level set function ϕ. A principal strength of the level set method comes from its ability to handle changes in topology of the evolving interface, i.e., interfaces break apart or merge naturally, and without the need for special code or instructions to detect or treat the shape changes as they occur. A second, and also key, benefit that occurs when using level set methods is that the discretization of the underlying *level set equation* (of Hamilton–Jacobi type) can be carried out using well-known, accurate and reliable discretization techniques, such as the weighted essentially non-oscillatory (WENO) methods described in [10, 8, 7]. Taken together, these benefits have contributed to a widespread adoption of level set techniques in different disciplines [14, 21].

Level set methods have primarily been used to treat evolving interfaces in $\mathbb{R}^2$ and $\mathbb{R}^3$. It is natural to want to evolve level set equations on general domains, to give a way of robustly capturing the motion of interfaces on curved surfaces. Such an extension would be extremely interesting since it could open up the possibility of generalizing existing level set applications to curved surfaces. For example, suppose one wished to segment out objects appearing on a surface. By extending level set methods to surfaces, we gain the possibility of solving this problem by simply transferring existing level set methods for segmentation to the case of surfaces. This approach becomes even more compelling if the algorithms for surface flows end up being based on existing codes for standard two- and three-dimensional flows. Indeed, we shall see this is the case with the Closest Point Method.

An interesting method for evolving interfaces on surfaces was proposed by Cheng et al. [2]. In their approach, a level set representation of the underlying surface was taken, with the evolving interface being represented by the intersection of two level set functions. The level set evolution equation for ϕ made use of standard gradients followed by projections to the surface rather than using the surface gradients that would otherwise appear in a surface PDE. Thus, the method evolved a level set PDE in $\mathbb{R}^3$, and, at any time, gave the position of the interface on the surface as the zero contour of ϕ on the surface. See [2] for further details on the method as well as a fascinating selection of examples using the method.

An alternative way of developing a method to evolve interfaces on surfaces is to start from a level set equation defined on a surface, e.g., a Hamilton–Jacobi equation of the form

$$\phi_t + H(t, \boldsymbol{x}, \phi, \nabla_{\mathcal{S}}\phi) = 0, \tag{1a}$$

$$\phi(0, \boldsymbol{x}) = \phi_0(\boldsymbol{x}), \tag{1b}$$

or some curvature-dependent generalization of this, and to solve it with some existing strategy for evolving PDEs on surfaces. For example, one

might apply the method of Bertalmío et al. [1] or Greer [6] to treat the surface PDE. These methods use a level set representation of the surface and replace surface gradients by standard gradients and projection operators in $\mathbb{R}^3$ to get an *embedding PDE* which is defined throughout time and space and agrees with the surface evolution on the surface. This leads to similar or the same PDEs as those appearing in [2], and will therefore be very similar in character to the methods described there.

In this paper, we will evolve the level set equations of Hamilton–Jacobi type (1) according to the recently proposed Closest Point Method [18]. The Closest Point Method has a number of properties that make it quite attractive for solving level set equations on surfaces. First of all, it takes the underlying surface representation to be a closest point representation. This allows it to treat PDEs on surfaces that have boundaries, lack any clearly defined inside/outside or are of arbitrary codimension. Similar to level set based methods, the method uses an embedding PDE defined in the embedding space (e.g., $\mathbb{R}^3$). The meaning and use of the embedding PDE is fundamentally different, however, since it is only valid initially, and therefore requires an *extension step* to ensure its accuracy. A desirable property of the Closest Point Method is that it works on sharply defined bands around the surface of interest without any loss of accuracy whatsoever. Finally, we note that the method, in its explicit form, leads to embedding PDEs which are simply the PDEs of the corresponding flow in the embedding space. This last advantage means that with the insertion of a simple extension step we can reuse highly effective three-dimensional level set codes without any other modifications to obtain the motion of level sets on surfaces. Note that this paper does not consider curvature-driven flows; such motions have been treated successfully using the Closest Point Method with a central difference spatial discretization in [18].

A crucial step in applying the Closest Point Method to solve level set equations on surfaces is to design an appropriate extension step. This paper considers a new extension based on a WENO interpolation which is sixth order in smooth regions and formally fourth order elsewhere. Our approach has all the practical advantages of the Closest Point Method: flexibility when selecting a surface, efficiency with respect to banding and extreme simplicity of implementation. We emphasize that while our new extension procedure will be used here for Hamilton–Jacobi equations, it also could be valuable for treating much more general PDEs if high-order accuracy is desired but the PDE or the underlying surface is somewhere nonsmooth or marginally resolved.

The paper unfolds as follows. Section 2 reviews the Closest Point Method, and includes details on the closest point representation and the method itself. Section 3 gives our new interpolation technique, a technique inspired by previous WENO methods. This section includes details on situations where WENO interpolation will be preferred over standard fixed-stencil Lagrange interpolation. Section 4 provides the numerical experiments. A focus is on numerical convergence, and we carry out a number of studies that show

high-order accurate flows (up to fifth-order) for passive transport, normal flows, and the reinitialization PDE. A normal flow computation is performed on a non-trivial triangulated surface, illustrating that the method is in no way restricted to simple surfaces. We conclude by carrying out a standard level set calculation on a codimension-two hypersurface in 4D, the Klein bottle. This example highlights the method's ability to treat interesting surfaces without any inside/outside property. Finally Section 5 gives the paper's conclusions, and lists some areas for future work.

2 The Closest Point Method

The Closest Point Method [18] is a general technique for solving partial differential equations and other processes on surfaces. This section reviews the method and some of its key features. We begin with a discussion on how the method represents surfaces before describing the algorithm itself.

2.1 Closest Point Representation of Surfaces

The Closest Point Method for evolving PDEs on surfaces relies on a *closest point representation* of the underlying surface. Let the surface be embedded in $\mathbb{R}^n$ then we introduce the closest point operator $\text{cp} : \mathbb{R}^n \to \mathbb{R}^n$ such that given a point $\boldsymbol{x}$, $\text{cp}(\boldsymbol{x})$ is a point on the surface closest in Euclidean distance to $\boldsymbol{x}$. If $\boldsymbol{x}$ is in a sufficiently small neighborhood of a smooth surface, then it will have a unique closest point, however, in general if there are multiple closest points to $\boldsymbol{x}$, then we define $\text{cp}(\boldsymbol{x})$ to return an arbitrarily chosen closest point. For example, if the surface is a circle of radius R centred at the origin embedded in $\mathbb{R}^2$ then $\text{cp}(\langle x, y\rangle) = \left\langle \frac{Rx}{x^2+y^2}, \frac{Ry}{x^2+y^2} \right\rangle$ provided $\langle x, y\rangle \neq \langle 0, 0\rangle$. The origin is closest to any point on the circle so we define $\text{cp}(\langle 0, 0\rangle)$ to be some arbitrary point on the circle.

Because the Closest Point Method uses a closest point representation, a representation which is defined throughout the embedding space, it belongs to the class of embedding methods. Some other embedding methods for surface PDEs (e.g. [2,6]) make use of level set representations of the underlying surface. Note that the closest point representation has the advantage of not requiring a notion of "inside/outside" allowing the straightforward representation of surfaces with boundaries or non-orientable surfaces (e.g., a Möbius strip). Surfaces of codimension-two or higher such as the Klein bottle in 4D (Section 4.6) or a filament in 3D can also be represented without additional complication. Thus, an important feature of the closest point representation is that it does not inherently impose any limitations on the geometry or dimension of surfaces that can be represented.

2.2 Equivalence of Gradients

Besides the obvious flexibility they give for representing general surfaces, closest point representations have the advantage of giving us a means to extend quantities ϕ defined on the surface to the rest of space via $\phi(\text{cp}(\boldsymbol{x}))$. Such *closest point extensions* result in functions which are constant in the direction normal to the surface, at least within a neighborhood of the surface. This is important because it leads to simplified derivative calculations in the embedding space [18]. To proceed, let $\nabla_{\mathcal{S}}$ denote the gradient intrinsic to the surface $\mathcal{S}$. Then, at the surface, $\nabla\phi(\text{cp}(\boldsymbol{x})) = \nabla_{\mathcal{S}}\phi$ since the function $\phi(\text{cp}(\boldsymbol{x}))$ is constant in the normal direction and therefore only varies along the surface. In other words, at points on the surface, *surface gradients* are the same as *standard Cartesian gradients* of $\phi(\text{cp}(\boldsymbol{x}))$. This will be all that we need to derive the embedding PDEs used in the examples appearing in this paper.

A second principle also holds [18]. For this, let $\nabla_{\mathcal{S}}\cdot$ denote the divergence operator intrinsic to the surface $\mathcal{S}$ and let $\boldsymbol{v}$ be any vector field on $\mathbb{R}^3$ that is tangent at $\mathcal{S}$, and also tangent at all surfaces displaced by a fixed distance from $\mathcal{S}$ (i.e., all surfaces defined as level sets of the distance function to $\mathcal{S}$). Then at the surface $\nabla \cdot \boldsymbol{v} = \nabla_{\mathcal{S}} \cdot \boldsymbol{v}$. Combinations of this and the gradient property may be made, to allow for very general motion laws for second-order operators, including the level set equation for curvature motion and other nonlinear diffusion operators [18]. Indeed, even higher-order and more general derivative replacements may be considered by carrying out multiple closest point extensions [18], although this has not yet been tried in practice.

To help illustrate these ideas we provide two simple examples on a circle of radius R.

Example 1 Consider the surface gradient, expressed in polar coordinates $\nabla_{\mathcal{S}}\phi = 0\boldsymbol{e}_r + \frac{\partial\phi}{\partial s}\boldsymbol{e}_\theta$, where s is the arc length. We have $s = R\theta$, therefore $\frac{\partial\phi}{\partial s} = \frac{1}{R}\phi_\theta$ and

$$\nabla_{\mathcal{S}}\phi = \frac{1}{R}\phi_\theta \boldsymbol{e}_\theta.$$

Now $\nabla\phi(\text{cp}(\boldsymbol{x})) = \phi_r(\text{cp}(\boldsymbol{x}))\boldsymbol{e}_r + \frac{1}{r}\phi_\theta(\text{cp}(\boldsymbol{x}))\boldsymbol{e}_\theta$. As noted earlier, $\phi(\text{cp}(\boldsymbol{x}))$ is constant in the direction normal to the surface which in this case is the radial direction, so $\phi_r(\text{cp}(\boldsymbol{x})) = 0$. Thus, for points $\boldsymbol{x}$ on the surface

$$\nabla\phi(\text{cp}(\boldsymbol{x})) = \frac{1}{R}\phi_\theta(\text{cp}(\boldsymbol{x}))\boldsymbol{e}_\theta = \frac{1}{R}\phi_\theta(\boldsymbol{x})\boldsymbol{e}_\theta = \nabla_{\mathcal{S}}\phi(\boldsymbol{x}).$$

Example 2 Consider, in polar coordinates, the surface Laplace-Beltrami operator $\nabla^2_{\mathcal{S}}\phi = \phi_{ss} = \frac{1}{R^2}\phi_{\theta\theta}$ and the Laplacian operator $\nabla^2\phi(\text{cp}(\boldsymbol{x})) = \phi_{rr}(\text{cp}(\boldsymbol{x})) + \frac{1}{r}\phi_r(\text{cp}(\boldsymbol{x})) + \frac{1}{r^2}\phi_{\theta\theta}(\text{cp}(\boldsymbol{x}))$. Again, $\phi(\text{cp}(\boldsymbol{x}))$ is constant in the radial r-direction so both radial derivatives are zero, and for $\boldsymbol{x}$ on the surface

$$\nabla^2\phi(\text{cp}(\boldsymbol{x})) = \frac{1}{R^2}\phi_{\theta\theta}(\boldsymbol{x}) = \nabla^2_{\mathcal{S}}\phi(\boldsymbol{x}).$$

2.3 The Algorithm

Having determined a way of evaluating gradients and other derivatives in the embedding space, we are now in a position to give the Closest Point Method. Central to this task is to determine a PDE, defined in the embedding space, to generate the flow corresponding to the surface PDE. Suppose that our surface PDE takes the form

$$\phi_t = F\left(t, \boldsymbol{x}, \phi, \nabla_{\mathcal{S}}\phi, \nabla_{\mathcal{S}} \cdot \left(\frac{\nabla_{\mathcal{S}}\phi}{|\nabla_{\mathcal{S}}\phi|}\right)\right), \tag{2a}$$

$$\phi(0, \boldsymbol{x}) = \phi_0(\boldsymbol{x}). \tag{2b}$$

More general PDEs can be treated directly by the Closest Point Method, but this form includes many of the second-order flows that arise in geometric interface motion [21,14]. Based on the principles described in our previous subsection, and originally given in [18], we may replace the gradients and divergence operators by the standard Cartesian derivatives in the embedding space, according to

$$\phi_t = F\left(t, \mathrm{cp}(\boldsymbol{x}), \phi(\mathrm{cp}(\boldsymbol{x})), \nabla\phi(\mathrm{cp}(\boldsymbol{x})), \nabla \cdot \left(\frac{\nabla\phi(\mathrm{cp}(\boldsymbol{x}))}{|\nabla\phi(\mathrm{cp}(\boldsymbol{x}))|}\right)\right), \tag{3a}$$

$$\phi(0, \boldsymbol{x}) = \phi_0(\mathrm{cp}(\boldsymbol{x})), \tag{3b}$$

and the solutions of (2) and (3) will agree *at the surface*.

Notice that if we start from a $\phi(0, \boldsymbol{x})$ which comes from a closest point extension (as we do here in (3b)), then the right hand side of (3) and the *embedding PDE*

$$\phi_t = F\left(t, \mathrm{cp}(\boldsymbol{x}), \phi, \nabla\phi, \nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|}\right)\right), \tag{4a}$$

$$\phi(0, \boldsymbol{x}) = \phi_0(\boldsymbol{x}), \tag{4b}$$

agree initially (although not for later times). This suggests a way of explicitly treating (3) efficiently: starting from a closest point extension of the solution at time-step t_n, take one forward Euler step (or *stage* of a higher-order explicit Runge–Kutta scheme) of (4) to advance in time to $\tilde{\phi}^{n+1}$. After this evolution step $\tilde{\phi}^{n+1}$ will not be constant in a direction normal to the surface. To regain this property, we perform a closest point extension of $\tilde{\phi}^{n+1}$ according to $\phi^{n+1}(\boldsymbol{x}) = \tilde{\phi}^{n+1}(\mathrm{cp}(\boldsymbol{x}))$. This gives us an update which is constant in the direction normal to the surface, ensuring once again that (4) will initially agree with (3) and hence with the original surface PDE (2). We can then repeat the process of alternating between time-stepping (4) and performing closest point extensions to obtain the solution at the desired time.

The semi-discrete (discrete in time, continuous in space) Closest Point Method with forward Euler time-stepping is thus:

1. Find the embedding PDE (4) corresponding to the surface PDE (2).

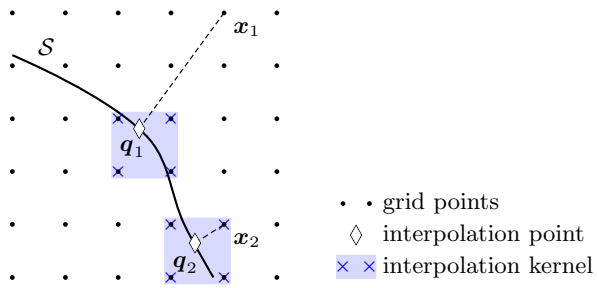


Fig. 1: Closest point extensions for grid points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ where $\boldsymbol{q}_1 = \text{cp}(\boldsymbol{x}_1)$ and $\boldsymbol{q}_2 = \text{cp}(\boldsymbol{x}_2)$. In this case, low-order bilinear interpolation is used to estimate a value for $\phi(\boldsymbol{q}_i) = \phi(\text{cp}(\boldsymbol{x}_i))$ from a stencil of the four neighboring grid points to $\boldsymbol{q}_i$.

2. Extend the initial conditions off the surface out into the surrounding domain to each point $\boldsymbol{x}$, i.e., $\phi^0(\boldsymbol{x}) = \phi^0(\text{cp}(\boldsymbol{x}))$.

At time t_n perform the following steps to advance to time t_{n+1}:

3. Perform a forward Euler time-step

$$\tilde{\phi}^{n+1} = \phi^n + \Delta t F\left(t_n, \text{cp}(\boldsymbol{x}), \phi^n, \nabla\phi^n, \nabla\cdot\left(\frac{\nabla\phi^n}{|\nabla\phi^n|}\right)\right).$$

4. Perform a closest point extension for each point $\boldsymbol{x}$

$$\phi^{n+1}(\boldsymbol{x}) = \tilde{\phi}^{n+1}(\text{cp}(\boldsymbol{x})).$$

For higher-order explicit Runge–Kutta methods, we employ a closest point extension following each *stage* of the Runge–Kutta scheme.

There is great flexibility available in choosing the spatial discretization. Similar to [18] we will select a finite difference method on a banded, but regular, Cartesian mesh. We will henceforth assume that the computation is carried out on a regular Cartesian mesh, noting that spectral methods have also been used to solve the embedding PDE; see [12] for details.

The closest point extension is an interpolation step because, although $\boldsymbol{x}$ is a grid point in a regular Cartesian grid, $\text{cp}(\boldsymbol{x})$ likely will not be. This is illustrated in Figure 1 where bilinear interpolation is used to estimate a value for $\phi(\text{cp}(\boldsymbol{x}))$ from a stencil of the four neighboring grid points. Clearly one could use a larger interpolation stencil to increase the accuracy of the interpolation, and this is precisely the focus of Section 3 where we construct a smooth high-order interpolation using dynamic stencils.

We conclude this section with a few observations about the Closest Point Method:

– The evolution step of the Closest Point Method (step 3 above) is done using standard methods in the embedding space. Moreover the embedding

PDE does not involve any projections or other surface-specific modifications; it is simply the standard PDE in the embedding space. This is very convenient since it implies that standard, well-understood algorithms in $\mathbb{R}^3$ can be straightforwardly modified to accommodate surface flows. Indeed, this is exactly the approach we take in Section 4 where we reuse standard Hamilton–Jacobi algorithms.
- The Closest Point Method uses a closest point representation for the surface. As discussed earlier in this section, this has the advantage of allowing the treatment of arbitrary surfaces of varying codimension, with boundaries, or even without orientation.
- The method does not introduce any artificial boundaries at the edge of the computational band to carry out banded calculations near the surface. The method merely computes on a band which is wide enough so that all values in the interpolation stencil have been accurately evolved (see Section 4.1). As a general principle, artificial boundaries should be avoided since they can lead to a degradation of accuracy even in simple diffusive problems [6].
- Finally, we remark that the use and meaning of the embedding PDE is fundamentally different for the level set methods for PDEs on surfaces and Closest Point Method. In a level set approach the embedding PDE gives the solution at the surface for all times. In the Closest Point Method the embedding PDE only gives a valid evolution initially and for one explicit time-step (or stage in a Runge–Kutta method). Thus the extension step is necessary to ensure the consistency of the algorithm.

For further details on the Closest Point Method and on how it contrasts with other methods and, in particular, other embedding methods we refer to [18].

3 WENO Interpolation

Weighted Essentially Non-Oscillatory (WENO) spatial discretizations [10,8, 7,9] are well-studied and commonly used for the evolution of PDEs with discontinuous or non-smooth solutions. We will make use of standard WENO discretizations in Section 4 for the evolution step of the Closest Point Method. However, the second part of the Closest Point Method—the extension step—requires an interpolation scheme to approximate $\phi(\text{cp}(\boldsymbol{x}))$ from grid points near the surface. We expect that fixed-stencil Lagrange interpolation may encounter difficulties from either non-smooth ϕ or from non-smooth or poorly resolved surfaces. In this Section we derive a WENO-based interpolation in one dimension and in multiple dimensions. This is followed by some numerical experiments which illustrate that WENO interpolation can give very good results even when fixed-stencil Lagrange interpolation fails or gives undesirable results.

WENO interpolation was considered in [20] to interpolate between subdomains of a multidomain WENO finite difference calculation for hyperbolic

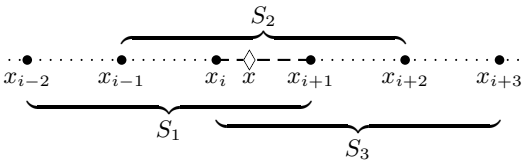


Fig. 2: The one dimensional interpolation grid, where $x \in [x_i, x_{i+1})$ and three candidate stencils S_1, S_2 and S_3.

conservation laws. However, in [20] one of the candidate stencils corresponds to an extrapolation rather than an interpolation (see Fig. 2 of [20]). In this work we derive and study new WENO interpolation schemes in which all candidate polynomials are interpolants. The question of whether improved results can be obtained by allowing some extrapolation in the candidate polynomials will be addressed in future studies.

3.1 One-dimensional WENO interpolation

Consider the 1D interpolation problem (Figure 2): given the six points x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}, x_{i+3}, corresponding data f_{i-2}, f_{i-1}, f_i, f_{i+1}, f_{i+2}, f_{i+3} and a value of $x \in [x_i, x_{i+1})$, we want to estimate $f(x)$.

We being with three candidate interpolants

$$\begin{aligned}
p_1(x) &= f_{i-2} + \tfrac{f_{i-1}-f_{i-2}}{\Delta x}(x - x_{i-2}) + \tfrac{f_i - 2f_{i-1}+f_{i-2}}{2\Delta x^2}(x - x_{i-2})(x - x_{i-1}) \\
&\quad + \tfrac{f_{i+1}-3f_i+3f_{i-1}-f_{i-2}}{6\Delta x^3}(x - x_{i-2})(x - x_{i-1})(x - x_i), \\
p_2(x) &= f_{i-1} + \tfrac{f_i - f_{i-1}}{\Delta x}(x - x_{i-1}) + \tfrac{f_{i+1}-2f_i+f_{i-1}}{2\Delta x^2}(x - x_{i-1})(x - x_i) \\
&\quad + \tfrac{f_{i+2}-3f_{i+1}+3f_i-f_{i-1}}{6\Delta x^3}(x - x_{i-1})(x - x_i)(x - x_{i+1}), \\
p_3(x) &= f_i + \tfrac{f_{i+1}-f_i}{\Delta x}(x - x_i) + \tfrac{f_{i+2}-2f_{i+1}+f_i}{2\Delta x^2}(x - x_i)(x - x_{i+1}) \\
&\quad + \tfrac{f_{i+3}-3f_{i+2}+3f_{i+1}-f_i}{6\Delta x^3}(x - x_i)(x - x_{i+1})(x - x_{i+2}),
\end{aligned}$$

where each interpolant corresponds to the cubic polynomial fit to the data given on one of the three candidate stencils $S_1 = \{x_{i-2}, \ldots, x_{i+1}\}$, $S_2 = \{x_{i-1}, \ldots, x_{i+2}\}$, and $S_3 = \{x_i, \ldots, x_{i+3}\}$ (see Figure 2). These interpolants will be combined to give the WENO interpolant

$$I_{\text{WENO6}}(x) = w_1(x)p_1(x) + w_2(x)p_2(x) + w_3(x)p_3(x),$$

where $w_i(x), i = 1, 2, 3$ are the required weights (still to be determined). In a smooth problem, all the point data should be used to obtain an interpolation which is as high order as possible, i.e., that agrees with the degree five interpolating polynomial through all six points. These "ideal" weights C_i,

$i = 1, 2, 3$ are given by

$$C_1(x) = \frac{(x_{i+2} - x)(x_{i+3} - x)}{20\Delta x^2},$$
$$C_2(x) = \frac{(x_{i+3} - x)(x - x_{i-2})}{10\Delta x^2},$$
$$C_3(x) = \frac{(x - x_{i-2})(x - x_{i-1})}{20\Delta x^2}.$$

Note that unlike WENO for hyperbolic conservation laws [10,8] and WENO for Hamilton–Jacobi problems [7], here the interpolation point x is not fixed and the values of the ideal weights depend on x. Still, these $C_i(x)$ are completely analogous to the well-known "$\frac{1}{10}$, $\frac{6}{10}$, $\frac{3}{10}$" weights in those works.

In nonsmooth regions, at least one of the interpolations $p_i(x), i = 1, 2, 3$ will be superior to an interpolation with the "ideal" values because of the problems associated with fitting high-order polynomials to nonsmooth data—namely highly oscillatory results. To decide which stencils to use, we compute a *smoothness indicator* for each interpolant. We take the smoothness indicator IS_i for interpolant p_i as "a sum of squares of scaled L^2 norms of all the derivatives of the [interpolant p_i] over the interval [of interpolation]" [22]. Specifically

$$IS_i = \sum_{j=1}^{3} \int_{x_i}^{x_{i+1}} (\Delta x)^{2j-1} \left(\frac{\mathrm{d}^j p_i(x)}{\mathrm{d}x^j} \right)^2 \mathrm{d}x. \tag{7}$$

If a particular interpolant exhibits rapid change on the interval (x_i, x_{i+1}) compared to the other two interpolants, then it will have larger-in-magnitude derivatives on that interval, which in turn increases the corresponding smoothness indicator (7). Smooth interpolants—those that are desirable for use in our interpolation—will exhibit less drastic changes in their derivatives and thus minimize (7). If all three candidate interpolants are smooth, then all three smoothness indicators will have similar (small) values. For completeness, (7) can be worked out as

$$\begin{aligned}
IS_1 = \big(&814f_{i+1}^2 + 4326f_i^2 + 2976f_{i-1}^2 + 244f_{i-2}^2 - 3579f_if_{i+1} - 6927f_if_{i-1} \\
&+ 1854f_if_{i-2} + 2634f_{i+1}f_{i-1} - 683f_{i+1}f_{i-2} - 1659f_{i-1}f_{i-2}\big)/180, \\
IS_2 = \big(&1986f_{i+1}^2 + 1986f_i^2 + 244f_{i-1}^2 + 244f_{i+2}^2 + 1074f_if_{i+2} - 3777f_if_{i+1} \\
&- 1269f_if_{i-1} + 1074f_{i+1}f_{i-1} - 1269f_{i+2}f_{i+1} - 293f_{i+2}f_{i-1}\big)/180, \\
IS_3 = \big(&814f_i^2 + 4326f_{i+1}^2 + 2976f_{i+2}^2 + 244f_{i+3}^2 - 683f_if_{i+3} + 2634f_if_{i+2} \\
&- 3579f_if_{i+1} - 6927f_{i+1}f_{i+2} + 1854f_{i+1}f_{i+3} - 1659f_{i+2}f_{i+3}\big)/180.
\end{aligned}$$

We note as expected that the smoothness indicators do not depend on the particular point of interpolation x because they measure a property of the interpolant candidates themselves.

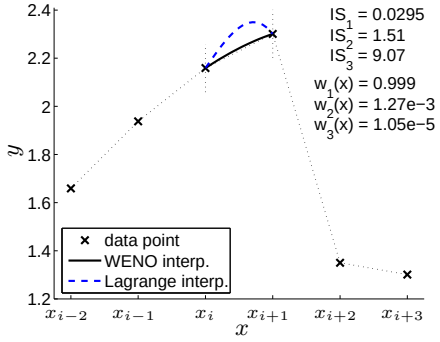


Fig. 3 One-dimensional interpolation example contrasting Lagrange and WENO interpolation near a discontinuity.

The computation of the weights is carried out using the smoothness indicators as in the standard WENO procedure by first calculating

$$\alpha_i(x) = \frac{C_i(x)}{(\varepsilon + IS_i)^2}, \quad i = 1, 2, 3,$$

where ε is a small parameter to prevent division-by-zero in the case when all $IS_i \approx 0$; we use $\varepsilon = 1 \times 10^{-6}$ in all our calculations. Finally, the weights are

$$w_i(x) = \frac{\alpha_i(x)}{\alpha_1(x) + \alpha_2(x) + \alpha_3(x)}, \quad i = 1, 2, 3.$$

Appendix A constructs a formally fourth-order WENO interpolation from two candidate quadratic interpolants using a similar approach to above. Both of these WENO interpolation routines are constructed so that wherever the ideal weights are chosen (i.e., $w_i(x) = C_i(x)$), the results are identical to using fixed-stencil interpolating polynomials through all candidate points. Therefore, in this case, they are also identical to the fixed-stencil Lagrange interpolation procedure used in previous Closest Point Method works [18,12]. In non-smooth problems, however, stencils corresponding to smooth regions are automatically selected. Figure 3 shows one such example, where a one-dimensional function is reconstructed using interpolation in a cell adjacent to a discontinuity. Lagrange interpolation based on the six data points gives a spurious oscillation in the cell of interest, since it interpolates across the discontinuity. On the other hand, with WENO interpolation the smoothness indicators IS_2 and IS_3 are very large resulting in small weights w_2 and w_3 and thus the data from the rightmost two data points give a negligible contribution to the interpolation. This leads to the desired non-oscillatory result.

Another type of problem for which WENO based interpolation is expected to give superior results arises when the underlying surface is marginally resolved by the grid or is nonsmooth. We investigate the former possibility in Section 3.3.

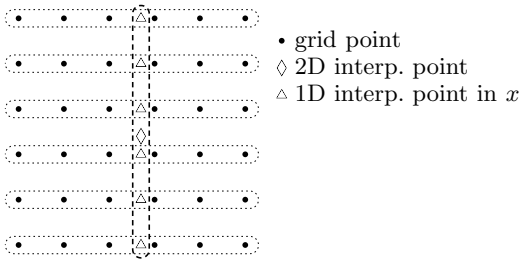


Fig. 4: Using a 1D interpolation scheme to build a 2D interpolation routine. Suppose we want to find a value at the point ◊ by interpolating the values at the grid points •. We begin by performing multiple 1D interpolations in the horizontal direction to obtain the values at the points △. We then do a final 1D interpolation of the values at points △ to obtain a value at point ◊.

3.2 Higher-Dimensional WENO Interpolation

Higher-dimensional interpolation is built in the standard fashion from one-dimensional interpolations. For example, two-dimensional sixth-order WENO interpolation is carried out by first interpolating six times in the x-direction to obtain six values which have x-coordinate values that agree with the interpolation point. One-dimensional interpolation is then carried out on these six points to get the desired interpolated value. See Figure 4 for an illustration. Three and higher dimensions are treated in a similar dimension-by-dimension manner.

3.3 WENO Interpolation and Marginally Resolved Surfaces

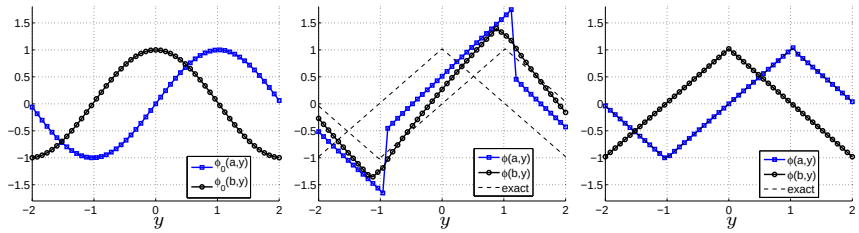
Even when the solution of the PDE is smooth, WENO interpolation can offer superior results over fixed-stencil Lagrange interpolation when the underlying surface is marginally resolved or otherwise nonsmooth. We demonstrate the former situation by considering the reinitialization equation (see Section 4.4) with sinusoidal initial conditions on the lines $x = a$ and $x = b$, i.e.,

$$\phi_t + \operatorname{sgn}(\phi_0)\left(|\nabla_{\mathcal{S}}\phi| - 1\right) = 0,$$

with

$$\begin{aligned}\phi_0(a, y) &= \sin(\pi y),\\ \phi_0(b, y) &= \cos(\pi y).\end{aligned}$$

Thinking of the two lines as representing different parts of a surface embedded in $\mathbb{R}^2$, we may apply the Closest Point Method to compute approximations of the solution. Clearly, the underlying PDE and initial conditions give a smooth flow. The problem becomes computationally interesting when the two lines are separated by only a few cells. Such problems illustrate how the



(a) Initial conditions (b) Lagrange interpolation (c) WENO interpolation

Fig. 5: A comparison of Lagrange interpolation versus WENO interpolation on closely spaced surfaces with $b - a = 4\Delta x$. This is a signed distance computation (Section 4.4) with $t_f = 4$ and periodic boundary conditions.

method treats a smooth PDE on a smooth but marginally resolved surface and gives us insight into whether WENO interpolation can give superior results to the standard Lagrange interpolation approach on marginally resolved surfaces.

Figure 5 shows the results when the lines are separated by four grid nodes (in this example, $b-a = 4\Delta x$). Using standard WENO for Hamilton–Jacobi equations in 2D for the evolution step (see Section 4) and Lagrange interpolation (based on degree five polynomials) for the closest point extension, leads to an obviously incorrect solution with discontinuities and substantial movement of the zero-contour. This error is due to the close proximity of the lines resulting in the stencil using data from both lines, an approach which is clearly nonlocal and incorrect. WENO interpolation, however, chooses stencils based on smoothness and hence avoids using data from the more distant line. This leads to a non-oscillatory numerical approximation that is in excellent agreement with the exact result. It turns out via a straightforward examination of the two interpolation stencils and the evolution stencil that to avoid nonlocal interactions, there must be eight grid points between the lines in the Lagrange interpolation case but only four grid points in the WENO interpolation case.

We conclude that it is safer to use WENO for the interpolation whenever the PDE or the underlying surface is nonsmooth or marginally resolved. We also recommend WENO-based Hamilton–Jacobi methods in the evolution step. Such methods have been widely used to treat standard Hamilton–Jacobi equations in $\mathbb{R}^2$ and $\mathbb{R}^3$ with good results and will be relatively safe when nonsmooth or marginally resolved problems arise.

4 Numerical Results

We now provide some numerical studies illustrating the behaviour and convergence of the method for a variety of interesting cases: passive transport,

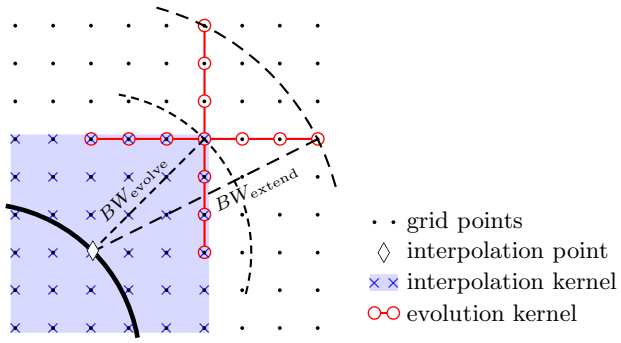


Fig. 6: The minimum bandwidths involved in a Closest Point Method computation for a surface in the vicinity of point ◊.

geometric flow under constant normal flow, and redistancing via the standard reinitialization PDE. We also illustrate the geometric flexibility of the method by treating normal flow on the triangulated surface of a human hand, and flow on a Klein bottle, a codimensional-two object in four dimensions. In all of our examples, our WENO-based interpolation procedure is used to carry out the extension step and standard Hamilton–Jacobi WENO-based techniques are used to treat the embedding PDE. For efficiency, calculations are performed in a narrow band around the surface as is described in Section 4.1.

4.1 Bandwidth

The evolution and extension steps of the Closest Point Method may be performed over the entire embedding space. However, such an implementation is inefficient because only a subset of grid points in the vicinity of the surface have any effect on the numerical solution. A more efficient approach is to perform calculations in a narrow band around the surface. This band must be wide enough to ensure that all nodal values within the interpolation stencil have been accurately evolved. By working in such a band we obtain the same results as we would for a global calculation because the closest point extension extends all values out from the surface after each step.

4.1.1 Bandwidth upper bounds We begin by determining upper bounds on the bandwidth. We consider an $\mathbb{R}^d$ embedding space and for simplicity assume that the grid spacing is Δx in all d dimensions. Our WENO interpolation stencil from Section 3.2 is a d-dimensional hypercube where each side has a width of $5\Delta x$. Considering Figure 6, the *evolution step* can therefore be carried out on a set of grid points which lie within a distance of

$$BW_{\text{evolve}} = \sqrt{3^2 + \cdots + 3^2}\Delta x = 3\sqrt{d}\Delta x, \tag{9}$$

Table 1 Sufficient bandwidth for Closest Point Method evolution and interpolation steps in various embedding dimensions.

dim.	BW_{evolve}	BW_{extend}
2D	$4.2426\Delta x$	$6.7082\Delta x$
3D	$5.1962\Delta x$	$7.3485\Delta x$
4D	$6\Delta x$	$7.9373\Delta x$

from the surface, i.e., the diagonal distance across a d-dimensional hypercube with side widths $3\Delta x$.

To generate accurate values on points inside the interpolation kernel at t_{n+1}, the evolution kernel needs accurate values inside its own finite difference stencil at t_n. Thus the interpolation step at t_n must update all grid points within the evolution stencil applied at every grid point in the interpolation stencil around every point on the surface. In our case, the fifth-order Hamilton–Jacobi WENO finite difference scheme has a stencil consisting of a "hypercross" (see Figure 6) where each arm has width $6\Delta x$. From the furthest corner of the interpolation kernel hypercube, the evolution stencil extends in grid-aligned directions; the extension step should therefore be performed on a bandwidth of

$$BW_{\text{extend}} = \sqrt{3^2(d-1) + (3+3)^2}\Delta x. \tag{10}$$

Values of the bandwidths (9) and (10) for two, three and four dimensions are tabulated in Table 1.

Bandwidth calculations for standard Lagrange interpolation can be carried out using similar considerations; see [18] for further details.

When defining the computational domain based on these upper bounds, we note there may be some points which are inside BW_{evolve} but are outside the union of the interpolation kernels around each closest point on the surface. These points therefore have, at most, a negligible effect[1] on the solution as the calculation proceeds. Evolving on these points thus introduces redundant computation. Likewise, there may be points inside BW_{extend} which are not needed for the evolution; these introduce additional redundant calculations.

To illustrate these ideas, Table 2 gives some results for a Closest Point Method calculation on bands which vary by width. We find that our banded procedures gives results identical to computing over the entire embedding space. Table 2 also illustrates that using smaller bandwidths than those in Table 1 results in a different solution.

[1] In principle, these points could influence the artificial dissipation parameters appearing in schemes such as the (global) Lax–Friedrichs (LF) scheme [4,14] or the local Lax–Friedrichs (LLF) scheme [23,14]. Although we did not observe any such effects in practice, the stencil set approach discussed next avoids this issue altogether.

4.1.2 The stencil set approach As noted above, using the bandwidth upper bounds (9) and (10) to define the computational bands may include unnecessary points thus increasing the computational expense without improving accuracy. An alternate approach is to explicitly construct the sets of points which define the evolution and extension bands. Let S_{evolve} be the set of nodes in the evolution band, and S_{extend} be the set of nodes in the extension band. We proceed as follows:

- Initialize both sets to the empty set, i.e., set $S_{\text{evolve}} = \emptyset$ and $S_{\text{extend}} = \emptyset$.
- Loop over all grid nodes $\boldsymbol{x}$ in the embedding space that are within a distance BW_{extend} of the surface:
  - Loop over all grid nodes $\boldsymbol{y}$ in the interpolation stencil surrounding $\text{cp}(\boldsymbol{x})$:
    - Add node $\boldsymbol{y}$ to the evolution band by setting $S_{\text{evolve}} = \{\boldsymbol{y}\} \cup S_{\text{evolve}}$.
    - Let K be the set of grid nodes appearing in the evolution stencil for $\boldsymbol{y}$. Add this set to the extension band by setting $S_{\text{extend}} = K \cup S_{\text{extend}}$.

After this procedure, S_{evolve} and S_{extend} are the sets over which the evolution and extension steps should be carried out for the Closest Point Method.

Table 2 confirms that for a LLF calculation, the stencil set approach produces identical results to computing on the entire embedding domain. Table 2 also indicates the number of points in each band and we note that in three dimensions the extension band contains 94% of the points used in the bandwidth approach. Likewise, the evolution band contains 85% of the points used in the bandwidth approach. In the Klein bottle computation in 4D (see Section 4.6), these savings are more significant since only 72% and 54% of the points are required in the respective bands. The stencil set approach thus offers computational savings, which, in combination with its simplicity, leads us to use this approach in the calculations appearing throughout this section.

4.2 Passive transport: flow under a specified velocity field

An important case of interface motion is *passive transport* or flow under a specified velocity field. In this case, an interface—represented by the zero-contour of a level set function—is advected via a velocity field which is independent of the interface geometry. On the surface, such a motion corresponds to the equation

$$\phi_t + \boldsymbol{V} \cdot \nabla_{\mathcal{S}} \phi = 0,$$

for some velocity field $\boldsymbol{V}$ specified on the surface. To evolve this surface PDE using the Closest Point Method, we instead treat the embedding PDE

$$\phi_t + \boldsymbol{V}(\text{cp}(\boldsymbol{x})) \cdot \nabla \phi = 0, \tag{11}$$

Table 2: Numerical verification of the bandwidths for a 3D Closest Point Method calculation. The actual values of error are not important here; we note only that the errors are identical (down to the bit-pattern of the double precision floating point numbers) provided the bandwidths are taken larger than the minimums from Table 1. (The computation is the same as Table 4 on a grid of $101 \times 101 \times 101$ with error measured as the maximum absolute value of ϕ along the theoretical interface location with $x \geq 0$ and $z \geq 0$.)

banding strategy		error	points in band	
BW_{evolve}	BW_{extend}		evolution	extension
$10\Delta x$	$10\Delta x$	$3.546949572342186954 \times 10^{-8}$	165335	165335
$5.2\Delta x$	$7.35\Delta x$	$3.546949572342186954 \times 10^{-8}$	82830	119026
$5.15\Delta x$	$7.35\Delta x$	$3.5469495\overline{70771058627} \times 10^{-8}$	82326	119026
$5.2\Delta x$	$7.3\Delta x$	$3.54694957\overline{2069116105} \times 10^{-8}$	82830	117970
stencil set bands		$3.546949572342186954 \times 10^{-8}$	70296	111412
no banding		$3.546949572342186954 \times 10^{-8}$	1030301	1030301

on a uniform 3D grid. Equation (11) is simply the standard equation for passive transport in 3D since $\boldsymbol{V}(\text{cp}(\boldsymbol{x}))$ is well-defined in $\mathbb{R}^3$. It is therefore natural to use standard methods [14], and approximate $\nabla\phi$ using upwinding and WENO approximations in a dimension-by-dimension fashion. Namely, for each point x_j we compute ϕ_x^+ from the values $\{\phi_{j-2}, \ldots, \phi_{j+3}\}$ using the Hamilton–Jacobi WENO procedure. Similarly we compute ϕ_x^- using $\{\phi_{j-3}, \ldots, \phi_{j+2}\}$. If v_x, the first component of $\boldsymbol{V}(\text{cp}(\boldsymbol{x}))$, is positive then we approximate ϕ_x with ϕ_x^-, otherwise we approximate ϕ_x with ϕ_x^+. The same procedure is repeated to approximate ϕ_y and ϕ_z. We then use these values to approximate $\boldsymbol{V}(\text{cp}(\boldsymbol{x})) \cdot \nabla\phi = v_x\phi_x + v_y\phi_y + v_z\phi_z$ which allows us to proceed by the method-of-lines. Time-stepping is done with the three-stage, third-order strong-stability-preserving (SSP) Runge–Kutta scheme [23] (SSPRK(3,3)) with $\Delta t = \frac{1}{2}\Delta x$ and with closest point extensions performed after each stage. We emphasize that apart from the closest point extensions, this is simply a standard procedure used for evolving (11) in three dimensions.

To test the numerical convergence of the method we consider a circular interface on a unit sphere, evolving by passive transport. As shown in Figure 7, we take a velocity field that is of unit length and emanates from one pole to the other in a radially symmetric fashion about the axis between the poles (like lines of longitude on a tilted globe). A comparison of the numerical result against the exact solution is provided in Table 3. These results show a clear fifth-order convergence.

For this smooth problem we see that the Closest Point Method gives the full accuracy of the underlying discretization of the embedding PDE. This implies that our closest point extension procedure based on WENO interpolation performs as anticipated and without degrading the fifth-order accurate treatment of the embedding PDE.

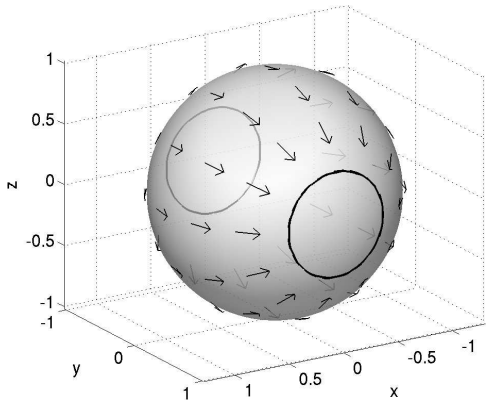


Fig. 7: Passive transport of a circular interface on a sphere. The circle starts initially at $y = -0.9$ on the far side of the sphere. It is advected over the surface of the sphere via the velocity field indicated with arrows to the final solution at $t_f = 2.24$ shown on the front of the sphere. The exact solution—also a circle—is shown but within the tolerances of the plot, it is almost indistinguishable from the numerical solution. A $26 \times 26 \times 26$ computational grid is chosen on the domain $[-2, 2]^3$.

Table 3: Convergence study for a circle moving on a sphere according to a specified velocity field. Error-in-position measures the error in the position of the interface along the surface where $z = 0$. Graphically, the situation is similar to Figure 7, but with the circle beginning initially at $y = -0.25$ and running to $t_f = 1$. The embedding domain is $[-2, 2]^3$.

grid	error-in-position	order
$26 \times 26 \times 26$	1.7263×10^{-4}	
$51 \times 51 \times 51$	2.6721×10^{-6}	6.01
$101 \times 101 \times 101$	7.3215×10^{-8}	5.19
$201 \times 201 \times 201$	2.1474×10^{-9}	5.09
$401 \times 401 \times 401$	6.3800×10^{-11}	5.07
$801 \times 801 \times 801$	1.8378×10^{-12}	5.12

4.3 Normal flow

We next consider the case of *normal flow* where the motion of the interface is governed not by an external velocity field but by the shape of the interface itself. We begin with constant normal flow

$$\phi_t + C|\nabla_{\mathcal{S}}\phi| = 0,$$

where the interface moves in the direction of its normal vector at a constant speed C. If $C = 1$, we refer to this as *unit normal flow* and for this problem,

Table 4: Convergence study for constant normal flow for a circle moving on a unit-radius sphere. Error-in-position measures the maximum error in the position of the zero-contour over the quadrant of the sphere where $x \geq 0$ and $z \geq 0$. The circle begins at $y = -0.25$ and the computation proceeds using LLF to $t_f = 0.5$ with $\Delta t = \frac{1}{2}\Delta x$. The computational domain is $[-2, 2]^3$.

grid	error-in-position	order
$26 \times 26 \times 26$	9.6093×10^{-5}	
$51 \times 51 \times 51$	1.8654×10^{-6}	5.69
$101 \times 101 \times 101$	3.4943×10^{-8}	5.74
$201 \times 201 \times 201$	5.5902×10^{-10}	5.97
$401 \times 401 \times 401$	1.0222×10^{-11}	5.77
$801 \times 801 \times 801$	2.2932×10^{-13}	5.48

the underlying 3D embedding PDE is

$$\phi_t + |\nabla \phi| = 0, \tag{12}$$

which is a Hamilton–Jacobi equation with *Hamiltonian* $H(\nabla\phi) = |\nabla\phi|$.

We discretize the embedding PDE in space using Lax–Friedrichs for Hamilton–Jacobi equations [14,16]. Specifically we use the *numerical Hamiltonian*

$$\begin{aligned} \hat{H} = &\left| \left\langle \tfrac{\phi_x^- + \phi_x^+}{2}, \tfrac{\phi_y^- + \phi_y^+}{2}, \tfrac{\phi_z^- + \phi_z^+}{2} \right\rangle \right| \\ &- \alpha^x \left(\tfrac{\phi_x^+ - \phi_x^-}{2} \right) - \alpha^y \left(\tfrac{\phi_y^+ - \phi_y^-}{2} \right) - \alpha^z \left(\tfrac{\phi_z^+ - \phi_z^-}{2} \right), \end{aligned} \tag{13}$$

where ϕ_x^+, ϕ_y^-, etc. are calculated using Hamilton–Jacobi WENO and the latter three terms provide artificial dissipation. The dissipation coefficients α^x, α^y, and α^z are calculated as the bounds for partial derivatives of the Hamiltonian H over some domain, the choice of which leads to variations of the Lax–Friedrichs scheme. We implement the local Lax–Friedrichs (LLF) and stencil local Lax–Friedrichs (SLLF) variants [14]. After computing the numerical Hamiltonian, we can proceed by the method-of-lines where time-stepping is again done with the SSPRK(3,3) scheme with $\Delta t = \frac{1}{2}\Delta x$ and closest point extensions after each stage.

To test the order of convergence of our method, we compute the motion of a circle on a sphere via unit normal flow. The exact solution is simply the circle moved along the surface of the sphere, similar to the passive transport case in Figure 7. Table 4 shows that the Closest Point Method achieves at least fifth order on this problem, again validating the choice of our WENO interpolation technique.

Of course, non-spherical surfaces may also be treated. Figure 8 shows the motion of an initial interface on a torus, as computed using the SLLF scheme for the embedding PDE. As anticipated, the interface moves from left to right parallel to the y-axis via unit normal flow, separating and recombining as necessary.

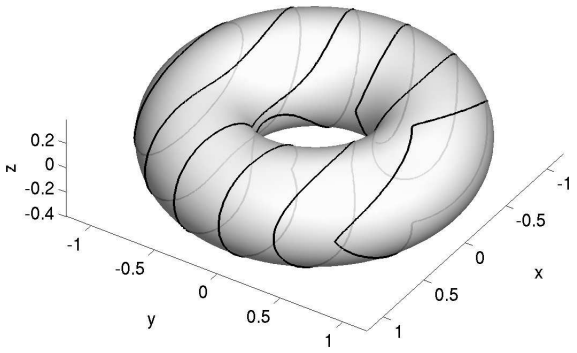


Fig. 8: Unit normal flow on a torus with radii 0.8 and 0.4. The interface begins at $y = -1$ and the computation proceeds using SLLF to $t_f = 2$. The interface is shown at every 0.4 units of time. A $101 \times 101 \times 101$ computational grid is taken on the embedding domain $[-2, 2]^3$.

4.4 Signed Distance / Reinitialization

In practical applications, level set functions may become either too steep or flat during their evolution. *Reinitialization* is often used to take general level set functions closer to signed distance functions, or even to generate accurate approximations of signed distance functions. Given the widespread use of such techniques, it is of interest to see whether the corresponding surface reinitialization PDE

$$\phi_t + \operatorname{sgn}(\phi_0)\left(|\nabla_{\mathcal{S}}\phi| - 1\right) = 0, \tag{14}$$

can be accurately treated using the Closest Point Method. Here we assume that the initial interface is, as usual, specified as the zero-contour of an initial ϕ_0. Starting from ϕ_0, the surface level set equation (14) evolves ϕ so that in the steady state $\phi(\boldsymbol{x})$ gives the signed distance (along the surface) from $\boldsymbol{x}$ to the interface. In practice, this evolution will also move the zero-contour of ϕ; we want this motion to be small, and to vanish as the discretization grid spacings tend to zero.

Treating this problem using the Closest Point Method is straightforward; we discretize the corresponding three dimensional redistancing embedding PDE

$$\phi_t + \operatorname{sgn}(\phi_0)\left(|\nabla\phi| - 1\right) = 0, \tag{15a}$$

and as is standard practice [14,5,13], we replace the signum function with a smoother version

$$\operatorname{sgn}(\phi_0) \approx S(\phi_o) = \frac{\phi_0}{\sqrt{\phi_0^2 + \epsilon^2}}. \tag{15b}$$

Typically, ϵ is set equal to Δx but in this work we use $\epsilon = \sqrt{\Delta x}$, as suggested in [3]. This latter choice of ϵ gave considerably better convergence results

Table 5: Convergence study for signed distance at $t = 5$ where ϕ_0 is a signed *half*-distance function (i.e., $\phi_0 = d/2$ where d is the signed distance function) to a circular interface at $y = -0.25$ on the surface of a unit sphere. Error-in-position measures the maximum error in the position of the contour in the quadrant of the sphere where $x \geq 0$ and $z \geq 0$.

grid	zero-contour		0.15-contour	
	error-in-position	order	error-in-position	order
$26 \times 26 \times 26$	4.42×10^{-4}		6.06×10^{-4}	
$51 \times 51 \times 51$	1.08×10^{-5}	5.36	1.71×10^{-5}	5.15
$101 \times 101 \times 101$	4.08×10^{-7}	4.72	5.39×10^{-7}	4.99
$201 \times 201 \times 201$	2.57×10^{-8}	3.99	2.27×10^{-8}	4.57
$401 \times 401 \times 401$	1.12×10^{-9}	4.52	9.14×10^{-10}	4.64

than the former. The approach of [17] could also be considered although we have not done so here.

Following [5,13], we implement a modified Godunov scheme for (15). Specifically, at each grid point $\boldsymbol{x}_j$ we compute ϕ_x^+ and ϕ_x^- using Hamilton–Jacobi WENO. We then select an approximation Φ_x to ϕ_x: if $S(\phi_0)\phi_x^+ \geq 0$ and $S(\phi_0)\phi_x^- \geq 0$ then choose $\Phi_x = \phi_x^-$; if $S(\phi_0)\phi_x^+ \leq 0$ and $S(\phi_0)\phi_x^- \leq 0$ then choose $\Phi_x = \phi_x^+$; if $S(\phi_0)\phi_x^+ > 0$ and $S(\phi_0)\phi_x^- < 0$ then choose $\Phi_x = 0$; finally if $S(\phi_0)\phi_x^+ < 0$ and $S(\phi_0)\phi_x^- > 0$ then we compute $s = S(\phi_0)\frac{|\phi_x^+| - |\phi_x^-|}{\phi_x^+ - \phi_x^-}$, and if $s \geq 0$, choose $\Phi_x = \phi_x^-$ or if $s < 0$, then choose $\Phi_x = \phi_x^+$. Having repeated this procedure in the y and z directions at $\boldsymbol{x}_j$, we can approximate

$$S(\phi_0)\left(|\nabla\phi| - 1\right) \approx S(\phi_0)\left(\sqrt{\Phi_x^2 + \Phi_y^2 + \Phi_z^2} - 1\right).$$

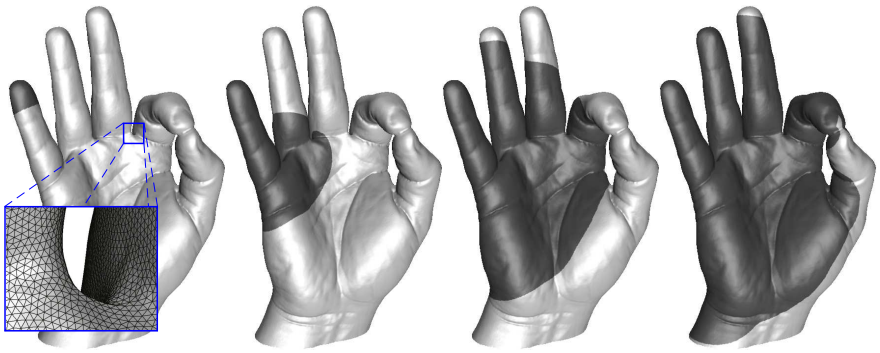
The Closest Point Method then proceeds as a method-of-lines computation with closest point extensions following each stage of the SSPRK(3,3) scheme with $\Delta t = \frac{1}{2}\Delta x$.

Table 5 shows between fourth- and fifth-order convergence for the signed distance problem. We remark that in both columns of the table, errors in contour position are used to determine convergence rates.

4.5 Triangulated Surfaces

Our examples thus far have involved fairly simple surfaces. Complex surfaces may also be treated by the Closest Point Method so long as a closest point representation of the underlying surface is available or can be computed.

Extensive and freely available collections of complex shapes exist, and many of the available surfaces are in a triangulated form. Naturally, we wish to be able to compute flows on such surfaces, a task that, for the Closest Point Method, requires the construction of a closest point representation from a surface triangulation.



(a) $t = 0$ (b) $t = 0.665$ (c) $t = 1.065$ (d) $t = 1.315$

Fig. 9: Unit normal flow on "Laurent's Hand" shown at various times. The interface—visualized here as a transition from dark to light—begins at the tip of the little finger. The computation uses LLF on the computational domain $[-1, 1]^3$ with a $401 \times 401 \times 401$ grid.

A straightforward method to convert triangulated surfaces into closest point representations is to loop over the list of triangles to directly determine the triangle closest to each grid node in the embedding space. Then the closest point on the surface is given by the closest point on the corresponding closest triangle. Naïvely implemented, this approach is computationally expensive (and inefficient) since each triangle must be examined for each node.

A much faster construction of the closest point function is obtained by taking into account that the method works on a narrow computational band [11]. First, for each triangle, we determine all nodes that are within the bandwidth, BW_{extend}, of the triangle (nodes that are further away cannot be influenced by that part of the surface). For each node, this gives a list of triangles. This list is sufficiently small that it is normally quite practical to directly examine each member to determine the closest triangle (and hence the closest point on the surface). See [11] for full details on this initialization procedure.

Notice that the triangulation is used only in the initial computation of the closest point representation and for plotting purposes; otherwise the Closest Point Method calculation proceeds exactly as described in Section 2 and our previous examples. For example, Figure 9 gives a computation for unit normal flow on the surface of "Laurent's Hand" [19] (the hand consists of 351048 vertices comprising 701543 triangles). The flow itself was carried out using the same code as was used in Section 4.3; as anticipated, the only modifications appear in the initial computation of the closest point representation and in the plotting procedure.

4.6 Klein Bottle

The Klein bottle is a famous hypersurface embedded in 4D which is closed but has no inside and outside. Although the Klein bottle appears self-intersecting when drawn in 3D (see Figure 10), the complete hypersurface in 4D is not. We consider a parameterization [24] in terms of $(u, v) \in [0, 2\pi)^2$

$$x = \begin{cases} \frac{3}{7}\cos u\,(1+\sin u) + \frac{2}{7}r\left(1-\frac{\cos u}{2}\right)\cos u\cos v, & \text{if } u \le \pi, \\ \frac{3}{7}\cos u\,(1+\sin u) - \frac{2}{7}r\left(1-\frac{\cos u}{2}\right)\cos v, & \text{otherwise,} \end{cases} \tag{16a}$$

$$y = \begin{cases} \frac{8}{7}\sin u + \frac{2}{7}r\left(1-\frac{\cos u}{2}\right)\sin u\cos v - \frac{1}{7}, & \text{if } u \le \pi, \\ \frac{8}{7}\sin u - \frac{1}{7}, & \text{otherwise,} \end{cases} \tag{16b}$$

$$z = \frac{2}{7}r\left(1-\frac{\cos u}{2}\right)\sin v, \tag{16c}$$

$$w = -\frac{8}{7}\cos u, \tag{16d}$$

where r controls the radius of the bottle (we use $r = 1$). We define a function $\boldsymbol{f}_{\text{Klein}} : \mathbb{R}^2 \to \mathbb{R}^4$ such that $\boldsymbol{x} = \langle x, y, z, w\rangle = \boldsymbol{f}_{\text{Klein}}(u, v)$ as in (16).

This surface is unlikely have a simple closest point function. However we have the parameterization (16), and thus for a given grid point $\boldsymbol{x}_0 = \langle x_0, y_0, z_0, w_0\rangle$, we can compute the closest point by minimizing

$$d^2(u, v) = \|\langle x_0, y_0, z_0, w_0\rangle - \boldsymbol{f}_{\text{Klein}}(u, v)\|_2^2,$$

over (u, v) (using, for example, MATLAB's fminsearch) to find $(u_{\min}, v_{\min})$. The closest point to $\boldsymbol{x}_0$ is then

$$\text{cp}(\boldsymbol{x}_0) = \boldsymbol{f}_{\text{Klein}}(u_{\min}, v_{\min}).$$

Once we have performed this straightforward—albeit time consuming—series of optimizations, we store the results which can then be reused for any further Closest Point Method calculations on the same grid.

Figure 10 shows the results of the reinitialization equation calculation (Section 4.4) on the surface of the Klein bottle. This example illustrates that the Closest Point Method can treat both non-orientable surfaces and surfaces of codimension-two. We stress this computation requires no special changes to the Closest Point Method illustrating that the method can handle very general surfaces without any particular modifications. We also note that although the computational grid was $51 \times 51 \times 51 \times 51$, only 340057 points or about 5% of that grid is contained in the band used for computation. We anticipate that problems of high codimension would benefit in terms of memory requirements from a more flexible storage scheme then the simple 4D array used here.

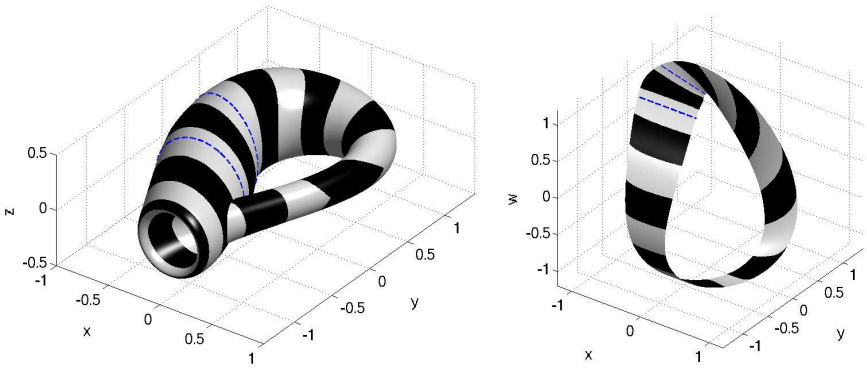


Fig. 10: Reinitialization / Signed Distance on a Klein bottle from initial conditions $\phi_0 = 1.1 - w$. Left: (x, y, z)-projection, right: (x, y, w)-projection. Each transition from dark-to-light or light-to-dark represents contours of equal distance from the closer of the two initial interfaces indicated with dashed lines. Note that some shaded bands appear narrower than others in the 3D projections of the Klein bottle. The grid is $51 \times 51 \times 51 \times 51$ on the domain $[-2, 2]^4$.

5 Conclusions

The Closest Point Method is a recent technique for the solution of PDEs or other motions constrained to surfaces. This paper applies the Closest Point Method to level set equations to obtain a robust technique for evolving interfaces on very general surfaces. Note, in particular, that the method retains the advantages of the level set method itself—i.e., it automatically handles self-intersecting interfaces and it makes use of standard high-order WENO methods in the embedding space (typically $\mathbb{R}^3$) to evolve the level set equations themselves.

New to the Closest Point Method is our derivation of a Weighted Essentially Non-Oscillatory (WENO) based interpolation scheme suitable for use in the closest point extension step. The Closest Point Method, using this interpolation scheme together with standard Hamilton–Jacobi WENO discretizations for the evolution of the embedding level set equation, achieved fourth- and fifth-order results on convergence test problems for passive transport, normal flow, and the reinitialization equation. It is noteworthy that these are the first results that indicate the Closest Point Method is capable of computing surface flows with a high order of accuracy. While our study here focused on the level set equation, WENO interpolation is likely to find use in many other settings such as Hyperbolic Conservation Laws; the main requirement being the desire to use a high-order accurate method for a PDE or surface which is somewhere nonsmooth or marginally resolved.

We computed flows defined on a triangulation of the surface of a human hand and on the surface of the non-orientable, codimension-two Klein bottle, illustrating that the Closest Point Method is very flexible with respect to the geometry and dimension of the surface.

Our examples show that level set equations on surfaces can be treated to high order in a straightforward manner using the Closest Point Method. We are investigating the application of the Closest Point Method to other classes of PDEs, including high-order PDEs which should be treated with implicit time-stepping. We are also interested in applications of PDEs on surfaces such as might arise in image inpainting or segmentation on surfaces.

Acknowledgements The authors thank Richard Tsai for his suggestions regarding convergence for the reinitialization equation.

A Fourth-order WENO interpolation

In Section 3 we derived a formally sixth-order WENO interpolation scheme using three candidate interpolants. We can also construct a fourth-order (in smooth regions) WENO interpolation scheme based on two quadratic interpolant candidates. In this case, we have the four points $x_{i-1}, x_i, x_{i+1}, x_{i+2}$ and corresponding data $f_{i-1}, f_i, f_{i+1}, f_{i+2}$ and again want to estimate $f(x)$ for $x \in [x_i, x_{i+1})$. The two candidate interpolants are

$$p_1(x) = f_i + \tfrac{f_{i+1}-f_{i-1}}{2\Delta x}(x-x_i) + \tfrac{f_{i+1}-2f_i+f_{i-1}}{2\Delta x^2}(x-x_i)^2,$$
$$p_2(x) = f_i + \tfrac{-f_{i+2}+4f_{i+1}-3f_i}{2\Delta x}(x-x_i) + \tfrac{f_{i+2}-2f_{i+1}+f_i}{2\Delta x^2}(x-x_i)^2,$$

with ideal weights $C_1(x) = \frac{x_{i+2}-x}{3\Delta x}$ and $C_2(x) = \frac{x-x_{i-1}}{3\Delta x}$, and smoothness indicators

$$IS_1 = \left(26f_{i+1}f_{i-1} - 52f_if_{i-1} - 76f_{i+1}f_i + 25f_{i+1}^2 + 64f_i^2 + 13f_{i-1}^2\right)/12,$$
$$IS_2 = \left(26f_{i+2}f_i - 52f_{i+2}f_{i+1} - 76f_{i+1}f_i + 25f_i^2 + 64f_{i+1}^2 + 13f_{i+2}^2\right)/12.$$

The fourth-order WENO interpolant is thus

$$I_{\text{WENO4}}(x) = w_1(x)p_1(x) + w_2(x)p_2(x),$$

where $w_1(x)$ and $w_2(x)$ are calculated from the smoothness indicators by $\alpha_i(x) = \frac{C_i(x)}{(\varepsilon+IS_i)^2}$ and $w_i(x) = \frac{\alpha_i(x)}{\alpha_1(x)+\alpha_2(x)}$, $i = 1, 2$.

References

1. Marcelo Bertalmío, Li-Tien Cheng, Stanley Osher, and Guillermo Sapiro. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, 174(2):759–780, 2001.
2. Li-Tien Cheng, Paul Burchard, Barry Merriman, and Stanley Osher. Motion of curves constrained on surfaces using a level-set approach. *J. Comput. Phys.*, 175(2):604–644, 2002.
3. Li-Tien Cheng and Richard Tsai. Redistancing by flow of the time dependent Eikonal equation. 2008. Under Review.

4. M. G. Crandall and P.-L. Lions. Two approximations of solutions of Hamilton–Jacobi equations. *Math. Comp.*, 43(167):1–19, 1984.
5. Ronald P. Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152(2):457–492, 1999.
6. John B. Greer. An improvement of a recent Eulerian method for solving PDEs on general geometries. *J. Sci. Comput.*, 29(3):321–352, 2006.
7. Guang-Shan Jiang and Danping Peng. Weighted ENO schemes for Hamilton–Jacobi equations. *SIAM J. Sci. Comput.*, 21(6):2126–2143, 2000.
8. Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted ENO schemes. *J. Comput. Phys.*, 126(1):202–228, 1996.
9. Culbert B. Laney. *Computational gasdynamics*. Cambridge University Press, Cambridge, 1998.
10. Xu-Dong Liu, Stanley Osher, and Tony Chan. Weighted essentially non-oscillatory schemes. *J. Comput. Phys.*, 115(1):200–212, 1994.
11. Barry Merriman and Steven J. Ruuth. Embedding methods for the numerical solution of PDEs on manifolds. In preparation.
12. Barry Merriman and Steven J. Ruuth. Diffusion generated motion of curves on surfaces. *J. Comput. Phys.*, 225(2):2267–2282, 2007.
13. Ian Mitchell. A toolbox of level set methods. Technical Report TR-2004-09, University of British Columbia Department of Computer Science, July 2004. `http://www.cs.ubc.ca/~mitchell/ToolboxLS/Papers/Toolbox/toolboxLS-1.0.pdf`.
14. Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2003.
15. Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.
16. Stanley Osher and Chi-Wang Shu. High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations. *SIAM J. Numer. Anal.*, 28(4):907–922, 1991.
17. Giovanni Russo and Peter Smereka. A remark on computing distance functions. *J. Comput. Phys.*, 163(1):51–67, 2000.
18. Steven J. Ruuth and Barry Merriman. A simple embedding method for solving partial differential equations on surfaces. *J. Comput. Phys.*, 227(3):1943–1961, 2008.
19. L. Saboret, M. Attene, and P. Alliez. "Laurent's Hand", the AIM@SHAPE shape repository. `http://shapes.aimatshape.net`, 2007.
20. Kurt Sebastian and Chi-Wang Shu. Multidomain WENO finite difference method with interpolation at subdomain interfaces. *J. Sci. Comput.*, 19(1-3):405–438, 2003.
21. J. A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, second edition, 1999.
22. Chi-Wang Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. Technical Report NASA CR-97-206253 ICASE Report No. 97-65, Institute for Computer Applications in Science and Engineering, November 1997.

23. Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially nonoscillatory shock-capturing schemes. *J. Comput. Phys.*, 77(2):439–471, 1988.
24. Wikipedia contributors. Klein bottle. Wikipedia, the free encyclopedia, `http://en.wikipedia.org/w/index.php?title=Klein_bottle&oldid=133679151`, May 2007. Accessed 2007-05-29.