Redistancing by Flow of Time Dependent Eikonal Equation

Li-Tien Cheng [*]and Yen-Hsi Tsai [†]

June 11, 2007

Abstract

Construction of signed distance to a given inteface is a topic of special interest to level set methods. There are currently, however, few algorithms that can efficiently produce highly accurate solutions. We introduce an algorithm for constructing an approximate signed distance function through manipulation of values calculated from flow of time dependent eikonal equations. We provide operation counts and experimental results to show that this algorithm can efficiently generate solutions with a high order of accuracy. Comparison with the standard level set reinitialization algorithm shows ours is superior in terms of predictability and local construction, critical, for example, in local level set methods. We further apply the same ideas to extension of values off interfaces. Together, our proposed approaches can be used to advance the level set method for fast and accurate computations of the latest scientific problems.

In a level set method [11], an interface is represented as the zero level set of a continuous real-valued function, called a level set function. Let ϕ denote this function. Then ϕ embeds the interface Γ as its zero level set: $\Gamma = \{x \in \mathbf{R}^m | \phi(x) = 0\}$. This representation retains geometric information of the interface, which can be extracted using formulas involving derivatives and integrals of ϕ. Furthermore, by adding a time variable, the level set function can be used to capture a given dynamics of the interface using a time dependent PDE in ϕ. The location of the interface at time t in this case is the zero level set of ϕ at that time: $\Gamma(t) = \{x \in \mathbf{R}^m | \phi(x, t) = 0\}$.

The level set function ϕ is certainly not unique; for example $\alpha\phi$, $\alpha \neq 0$, is also a level set function for the same interface. Thus one can ask for the best level set function according to some chosen criterion. In applications, even when a physically relevant level set function exists, that function may not satisfy that choice and so should not be used. Instead, the commonly desired criterion for the form of the level set function is one that leads to small errors when numerically solving the time dependent PDE for the dynamics of the interface, or when extracting the interface location. This form is most important to a level set method near the interface of interest.

Analytic solutions to the time dependent PDE for ϕ do not exist in all but the most trivial cases. Thus, one needs to turn to numerical approximations and solutions. The errors of a large class of popular numerical methods that operate on a discretization of the function over a grid, called finite differencing schemes, are typically calculated from Taylor expansions. This implies the derivatives of ϕ have a bearing on the error, with large magnitudes correlated to large errors. In addition to smoothness in ϕ, one would thus desire that ϕ avoid derivatives with large magnitudes, most notably at or near the zero level set of interest.

[*]Department of Mathematics, University of California, San Diego, La Jolla, California 92093-0112

[†]Department of Mathematics, University of Texas, Austin, Austin, TX 78712

On the other hand, for stability of representing the interface by the zero level set, the level set function should not be too flat near its zero level set. For example, in the one dimensional case, linear interpolation to determine the location of a zero of ϕ has error dependent on $1/\phi'(\xi)$, for some ξ. Thus small values of $|f'(x)|$ can lead to large errors. Following this, one would like the magnitude of the gradient of ϕ to be away from zero at or near the zero level set.

The process of reshaping a given level set function into a more desired form for stability and error is called reinitialization. One commonly chosen form is the signed distance function d of ϕ which satisfies the conditions $|\nabla d| = 1$ and $d|_\Gamma = 0$. Though the signed distance function does admit discontinuities in its derivative, called kinks, it is widely used due to its natural geometric interpretation and the existence of fast numerical methods for its construction. Furthermore, kinks only occur near the interface when the interface has large curvatures or kinks, and in this case we may not be able to obtain small errors anyway. Reinitialization of a given level set function to the corresponding signed distance function is also known as redistancing.

Algorithms for redistancing of level set functions began with the works of [18] and [1]. It is possible to categorize the algorithms into two distinct groups: those that work directly on the static boundary value problem and those that manufacture a flow to obtain the signed distance function in the steady state. In the former, solutions can be generated in near optimal times but the ability to obtain high orders of accuracy is suspect. Examples include fast marching [15, 8, 21] and fast sweeping [23, 20] methods. Flow based methods, on the other hand, are more flexible and more easily parallelizable. Furthermore, accurate solutions can be obtained, but only at a sometimes considerable expense to speed due to slow convergence to steady state. Examples include the original work of [18] and [13]. There has been interest in approaches that can achieving higher order accuracy with minimal sacrifice of speed. The work of [14] introduces a locally second order accurate discretization that can achieve subcell resolution. Also, the work of [4] uses bi-cubic interpolation and Newton's iterations to approximate the closest point on the interface to a point of the grid. Research, however, is ongoing in the topic and improvements and alternatives are much needed.

There are two important factors in assessing the quality of the signed distance function constructed by current algorithms. The first considers how stringently the boundary condition $\phi|_\Gamma = 0$ is enforced; in other words, how much the interface is perturbed by numerical procedures. The second considers how closely the differential equation $|\nabla\phi| = 1$ is satisfied and how many derivatives exist in the solution away from kinks. The direct and flow based methods for redistancing also trade off advantages and disadvantages on these two points. Direct methods tend to produce larger errors approximating the differential equation since high order discretization is not convenient to implement in their framework. Flow based methods, on the other hand, tend to excessively perturb the interface location when speed is not entirely sacrificed.

One consideration to note is redistancing is not always needed in all of space. In general, level set methods are following a zero level set surface and thus signed distance forms are only needed a few gridsteps around that location. This is especially true of local level set methods, whose computational domains, for efficiency, are confined to be local to the interface of interest, usually in the form of a tube a few gridsteps in radius around the interface. Other applications, however, such as closest point algorithms, may call for signed distance functions in all of space. A good redistancing algorithm should be able to handle both cases adequately.

We include now some details of the original flow based method of [18], which we call standard level set reinitialization, to describe its flexibilities and drawbacks. The flow proposed in this paper takes the form

$$\psi_t + \sigma(x)(|\nabla \psi| - 1) = 0,$$

with initial condition $\psi(x, 0) = \phi(x)$. Here, $\sigma(x)$ is typically taken to be some regularized signum function of the initial data $\phi_0(x)$ such as

$$\sigma(x) = \text{sgn}_\epsilon(\phi_0(x)) = \frac{\phi_0(x)}{\sqrt{(\phi_0(x))^2 + \epsilon^2}}.$$

Nonzero ϵ is required so that the coefficient of the PDE is continuous and suitable for numerical computations.

If ϵ is chosen to be independent of mesh size, then $\text{sgn}_\epsilon(\phi_0(x))$ is as smooth as the initial data ϕ_0. However, a quick glance at the characteristic equations suggests a source of inefficiency, since the characteristics emanating from the interface that impart signed distance to points in space travel at speed $\sigma(x)$ which is small near the interface since $\sigma(x)$ is zero at the interface. Thus, this choice of ϵ may lead to slow speeds of convergence to steady state, though high accuracy can be obtained.

At the other end of the spectrum, when ϵ is chosen to be zero, then $\sigma(x)$ is far from zero almost almost everywhere but discontinuous exactly at the interface. When numerical schemes are applied to the PDE, the discontinuities in $\sigma(x)$ produce movement of the zero level set of ψ away from its original location at the interface. Thus, in practice, ϵ is usually chosen small but nonzero by letting it be a function of the mesh size Δx. Common choices are $C\Delta x$ [18] and $C|\nabla\phi_0|\Delta x$ [13], however, the convergence of the resulting PDE's under these choices is not known. In fact, the first choice is known to lead to non-convergent schemes when used on regularized Dirac-δ functions for computing line integrals [19]. For these line integrals, the work of [6] is able to introduce a grid dependent scale of regularization parameters to achieve convergence, however, its results do not apply to the redistancing PDE here, though the second choice of ϵ above is similar to choices of regularizations found in [6]. Our numerical experiments on different choices of ϵ indicate that the two above forms lead often to constructed signed distance functions whose values and derivatives have large errors. Presently, it is not clear what the best choice for the regularization parameter is.

We propose here a stable and convergent algorithm that does not require a regularization parameter and allows for predictable behavior and high order accurate approximations. Furthermore, it is fast compared to many of the manifestations of standard level set reinitialization. The equation that is used for obtaining signed distance is the standard time dependent eikonal equation

$$u_t + |\nabla u| = 0.$$

This allows us to apply the standard viscosity solution theory of [5] for proofs of convergence.

In section 1, we derive the proposed algorithm and discuss its complexity. We also propose a generalization of this algorithm to perform extension of values given on the interface into the ambient space in a subsection. In section 2, we present numerical results demonstrating the features of our introduced algorithms. In section 4, we acknowledge contributions to our research from outside sources. Finally, in section 5, we add useful details in background information.

1 The Proposed Algorithm

Suppose Γ is a smooth, compact hypersurface in $\mathbf{R}^m$ with a chosen orientation and non-zero reach. Then there exists a unit normal vector field $n : \Gamma \to \mathbf{R}^m$. Furthermore, $\mathbf{R}^m$ can be partitioned into the disjoint union $\Omega_1 \cup \Gamma \cup \Omega_2$, where $\partial\Omega_1 = \partial\Omega_2 = \Gamma$ and n is the outer normal of $\partial\Omega_1$. We say Ω_1 is the region enclosed by Γ.

Let $u_0(x)$ be such that $u_0(\Gamma) = 0$ and $\nabla u_0(x)/|\nabla u_0(x)| = n(x)$ for $x \in \Gamma$. Consider the viscosity solutions of

$$|\nabla d| = 1, d|_\Gamma = 0$$

and

$$u_t + |\nabla u| = 0, u(x, t = 0) = u_0(x).$$

Our algorithm originates from the following fact, proved in [10]: the T level set of d, which exists in Ω_2, is the zero level set of u at time T, for any $T > 0$. In other words, for $T > 0$,

$$\{x \in \mathbf{R}^m | d(x) = T\} = \{x \in \mathbf{R}^m | u(x, T) = 0\}.$$

Therefore, for $x \in \Omega_2$, our algorithm sets $\phi(x) = T$ for T satisfying $u(x, T) = 0$. For $x \in \Omega_1$, we reverse the orientation of Γ and assign negative values for $\phi(x)$ in the same manner.

We now give details of this algorithm in a standard situation. Let $\phi_0(x)$ be a given level set function whose zero level set is Γ. The goal is to construct the signed distance function of ϕ_0 that is negative where ϕ_0 is negative and positive where ϕ_0 is positive. We may consider $\Omega_2 = \{x|\phi_0(x) > 0\}$ and $\Omega_1 = \{x|\phi_0(x) < 0\}$. Our algorithm then iterates as follows:

1. Construct the flows defined by

$$\begin{cases} u_t + |\nabla u| = 0, \\ u(x, 0) = \phi_0(x). \end{cases}$$

   and

$$\begin{cases} v_t + |\nabla v| = 0, \\ v(x, 0) = -\phi_0(x). \end{cases}$$

2. Set

$$\phi(x) = \begin{cases} T, & \text{if } \phi_0(x) > 0 \text{ and } u(x, T) = 0, \\ -T, & \text{if } \phi_0(x) < 0 \text{ and } v(x, T) = 0. \end{cases}$$

3. Repeat from the first step if desired using $\phi_0(x) = \phi(x)$. This may improve the solution in certain situations, as discussed further in section 1.1.

When the PDE's of the first step are discretized, the accuracy of the numerical schemes used in time integration play an important role in the total accuracy of the algorithm. We discuss the numerical discretizations we use along with the accuracy they provide in the following section.

1.1 Discretizations and Accuracy

Our numerical method of choice for discretization of the time dependent eikonal equation is a standard high resolution scheme for solving Hamilton-Jacobi equations. We approximate the Hamiltonian by the monotone Godunov Hamiltonian [12] and use fifth order WENO [9] for spatial derivatives. Third order TVD Runge-Kutta [16] or fourth order SSP Runge-Kutta [17] is then applied to the resulting system of ODE's.

Without loss of generality, fix $x \in \Omega_2$; the case $x \in \Omega_1$ is similar. While the time dependent eikonal equation solver iterates in time, the values of u are recorded at each of these times at x. High order interpolation in time of u then allows for the determination of when u becomes zero.

This is the distance away from the interface and our value for $\phi(x)$. The interpolation, however, is complicated by the possibility that u will not be smooth in time. This occurs, for example, when a kink travels through the point x, leaving u only piecewise smooth in time. To illustrate this with an example, let $u_0(x) = \min\{x, -2(x-2)\}$ and consider expressly the location $x = 1$. The viscosity solution is $u(x,t) = \min\{x - t, -2(x-2) - 2t\}$. Thus

$$u(1,t) = \begin{cases} 1 - t, & t \leq 1 \\ 2 - 2t, & t > 1, \end{cases}$$

which has a discontinuity in the time derivative at $t = 1$.

In light of this, in order to achieve an accurate approximation of the zero of u at x, we turn to high order ENO schemes for our interpolation and use Newton's method on the resulting high degree polynomial to determine its root. Though this technique works, we consider it a drawback of our approach that the form of ϕ_0 away from the zero level set affects accuracy. This can be problematic in higher dimensions, where more complicated kink structures are present in functions. Details of ENO interpolation are given in the Appendix in section 5.1.

Thus we have included the third step of our algorithm involving iteration using the latest generated ϕ as initial condition in the flow. Since ϕ is a better approximation of the signed distance function, it may have a better form than that of ϕ_0, leading to fewer kinks in u and a more accurate construction of distance. Determination on when the third step should be used can be made from the smoothness of the time derivatives of u.

In actual implementation, at each point x, we need only store values of u neighboring the location where u becomes zero. The size of this neighborhood depends on the order of the interpolation scheme desired for capturing that zero. Let $U^n(x)$ denote the value computed by our scheme to approximate $u(x, t_n)$. Then at x, we store the values $(U^{n-r}, U^{n-r+1}, \ldots, U^{n+s})$, where n is such that U^n and U^{n+1} are different signs. These stored values are the ones that can be used by the interpolation scheme for finding the root in $[t_n, t_{n+1}]$. For example, for third order ENO interpolation, we may take $r = 2$ and $s = 3$, storing $(U^{n-2}, U^{n-1}, U^n, U^{n+1}, U^{n+2}, U^{n+3})$.

Close to the interface $\{x | \phi_0(x) = 0\}$, there may not be enough values for an interpolation scheme to use. For example, if $n = 0$, then U^0 and U^1 are of different signs and we cannot take any $r > 0$. In order to use high order interpolation, we insert values from the approximation of v. Let $V^n(x)$ denote the value computed by our scheme to approximate $v(x, t_n)$. Then we set $U^{n-j} = -V^{j-n}$ when $n - j < 0$. This makes sense because the level sets of u, initially those of ϕ_0, can be viewed as flowing in their outward normal directions while the level sets of v, also initially those of ϕ_0, can be viewed as flowing in their inward normal directions. Note v, however, does not from running the equation for u backward in time since this generates a different weak solution. This fact does not seem to affect the accuracy of our algorithm in the experiments we conduct in section 2.

1.2 Complexity

Consider a uniform grid placed in space $\mathbf{R}^m$ and time with N points in each dimension. Let Δx denote the stepsize length of the grid in space and Δt that in time. The complexity of our proposed algorithm is directly linked to the length of time it takes characteristics emanating from the zero level set to reach all points of the computational domain. With a stability condition such as $\Delta t = C_0 \Delta x$ used in the numerical scheme, the cost in number of operations is $\mathcal{O}(N \cdot N^m)$ coming from $\mathcal{O}(N)$ time steps of $\mathcal{O}(N^m)$ computations over the grid in $\mathbf{R}^m$. The operations associated to interpolation, $\mathcal{O}(1)$ for each point in the grid, and quadratically convergent Newton's method, $\mathcal{O}(\log\log N)$ for each point in the grid, contribute only to lower order terms in the complexity.

We now compare this to the complexity of the standard level set reinitialization routine under the method of characteristics. Consider the one dimensional case with interface at $x = 0$. Let $\phi_0(x)$ be an increasing level set function with $\phi_0(x) > 0$ for $x > 0$ and $\phi_0(x) < 0$ for $x < 0$. Standard level set reinitialization follows the PDE

$$\psi_t + \frac{\phi_0}{\sqrt{\phi_0^2+\epsilon^2}}(|\nabla\psi| - 1) = 0$$

with $\psi(x, 0) = \phi_0(x)$. Without loss of generality, consider this equation for $x \in [-2, 2]$. We look in detail at a semi-discrete calculation of distance values for $x > 0$.

Using the method of characteristics, consider location x and time t and the characteristic satisfying $\gamma(t) = x$. Then $\psi(x, t)$ is $\psi(\gamma(0), 0)$ plus the distance traveled by the characteristic from time 0 to t. Therefore, if a characteristic emanating from the interface reaches x, then at that time, and thereafter, $\psi(x, t)$ will have the value of distance away from the interface. Technically, however, the characteristic never reaches x since at the interface, the characteristic has vanishing velocity. We may consider instead the characteristic emanating from a point $\delta > 0$ close to 0 and look at obtaining distance values for $x \in (\delta, 2]$. This characteristic γ satisfies

$$\frac{d\gamma}{dt} = \frac{\phi_0(\gamma)}{\sqrt{(\phi_0(\gamma))^2+\epsilon^2}},$$

with $\gamma(0) = \delta$. Taking $\delta = \Delta x^p$, for example, will preserve pth order accuracy of the approximation $\phi_0(\delta) \approx \delta$, since δ is the exact distance from the interface and

$$\begin{aligned}\phi_0(\delta) &= \phi_0(0) + \delta\phi_0'(\xi)\\ &= \delta + \delta(\phi_0'(\xi) - 1)\\ &= \delta + \mathcal{O}(\Delta x^p),\end{aligned}$$

for some $\xi \in (0, \delta)$.

Note

$$\frac{\phi_0}{\sqrt{\phi_0^2+\epsilon^2}} \le 1 \le x$$

for $x \ge 1$. For $x \in [0, 1)$, we know

$$\frac{d}{dx}\left(\frac{\phi_0}{\sqrt{\phi_0^2+\epsilon^2}}\right) = \frac{\epsilon^2\phi_0'}{(\phi_0^2+\epsilon^2)^{\frac{3}{2}}} \le \frac{|\phi_0'|}{\epsilon},$$

so

$$\frac{\phi_0}{\sqrt{\phi_0^2+\epsilon^2}} \le \frac{K}{\epsilon}x,$$

where $K = \sup_{y\in[0,1)} |\phi_0'(y)| > 0$. Thus,

$$\frac{\phi_0}{\sqrt{\phi_0^2+\epsilon^2}} \le L(x) = \begin{cases} Mx, & \text{for } x \in \left[0, \frac{1}{M}\right) \\ 1, & \text{for } x \in \left[\frac{1}{M}, 2\right] \end{cases}$$

where $M = \max\{K/\epsilon, 1\}$, and so

$$\frac{d\gamma}{dt} \leq L(\gamma).$$

Thus a lower bound for the time it takes for γ to reach $1/M$ can be calculated from

$$\delta e^{Mt} = \frac{1}{M},$$

which gives the lower bound as $\log(1/(\delta M))/M$, or $\log(1/(\delta M))/(M\Delta t)$ time steps. Thereafter, the characteristic travels from $1/M$ to 2 requiring a time greater than or equal to $2 - 1/M$, or $(2M-1)/(M\Delta t)$ time steps. So the characteristic starting from δ reaches the end of the domain at 2 in greater than or equal to

$$\frac{\log(1/(\delta M)) + 2M - 1}{M\Delta t}$$

time steps. For the case where ϵ is small, such as when $\epsilon \to 0$ as $\Delta x \to 0$, $M = K/\epsilon$ and this number becomes

$$\frac{\epsilon \log(\epsilon/(\delta K)) + 2K - \epsilon}{K\Delta t}.$$

With the stability requirement $\Delta t = C_0 \Delta x$ and $\delta = \Delta x^p$, the number of time steps is $\mathcal{O}(N \log N)$ for $\epsilon = 1$ and $\mathcal{O}(N)$ for $\epsilon = \sqrt{\Delta x}$ and Δx. Thus, the maximum total efficiency of standard level set reinitialization ranges from $\mathcal{O}(N \cdot N \log N)$ to $\mathcal{O}(N \cdot N)$ in one spatial dimension and $\mathcal{O}(N \cdot N^m \log N)$ to $\mathcal{O}(N \cdot N^m)$ in m spatial dimensions. Our proposed algorithm is of equal or slightly better efficiency.

However, these numbers alone do not tell the whole story of efficiency. To generate distance values in the shrinking tube $(0, C\Delta x]$, say at $x = C\Delta x$ for C constant, requires at least $\mathcal{O}(N \log N)$ time steps for $\epsilon = 1$, $\mathcal{O}(\sqrt{N} \log N)$ time steps for $\epsilon = \sqrt{\Delta x}$, and $\mathcal{O}(\log N)$ time steps for $\epsilon = \Delta x$. This means standard level set reinitialization spends a lot of effort in constructing distance close to the interface, sometimes equal to the amount of effort spent in the rest of the domain. In comparison, our proposed algorithm just requires $\mathcal{O}(1)$ time steps in this case. Thus, our approach can be much faster local to the interface of interest, and even though the case $\epsilon = \Delta x$ is nearly as fast, it is known to produce larger errors, especially in derivatives of distance values.

In one last comparison, we note that our algorithm is able to determine when the constructed signed distance value at a location x is accurately approximated, namely shortly after the values of u or v become zero there. Standard level set reinitialization, on the other hand, is not able to provide a good estimate because its convergence speed can vary greatly. Furthermore, if we iterate our algorithm a few time steps past a chosen time T, we know we have accurately approximated signed distance at all x of distance T away from the interface. Standard level set reintialization has no such ability to gauge its progress. These issues become evident when we compare the errors of our proposed algorithm with that of standard level set reinitialization in section 2.4.

1.3 Extension of Values

It is possible in our proposed algorithm to add the ability to extend values that exist on the interface off into ambient space. Let ϕ_0 denote a level set function representing the interface of interest and let $f_0(x)$ be another function given in ambient space whose values at the interface are of interest. In other words, f_0 is implicitly defining values on the interface. Just as we desire a better form for

ϕ_0, choosing the signed distance function d, we may also desire a better form for f_0, choosing for it to be constant on the integral curves of ∇d, i.e., using instead f satisfying

$$\nabla d \cdot \nabla f = 0.$$

Though discontinuous at kinks of d, this is a natural form for extended values, describing extension off the interface constant in normal directions. In addition, it imparts stability under perturbations to the implicit representation of values on the interface. Note these are the same advantages that signed distance functions have for representing interfaces. In fluids, the values of the function may refer to the amount of surfactants at a fluid-fluid interface [22]. In level set methods, the values may refer to calculated velocities on the interface [2, 3].

Extension of values can be viewed as directly linked to construction of signed distance, with the static PDE replace by $\nabla d \cdot \nabla f = 0$. Even standard level set reinitialization can be translated as iteration to steady state of

$$q_t + \operatorname{sgn}(\phi_0)\nabla d \cdot \nabla q = 0,$$

with $q(x, 0) = f_0(x)$, as described in [3]. Thus, our proposed algorithm for constructing signed distance can also be modified to extend values off the interface. We use the fact that for $x \in \Omega_2$, the solution $g(x, t)$ of

$$g_t + \nabla d \cdot \nabla g = 0$$

with $g(x, 0) = f_0(x)$ satisfies $f(x) = g(x, T)$, where T is such that $u(x, T) = 0$. To fit this into the steps we had for generating u, we note that since u satisfies $\{x|u(x, t) = 0\} = \{x|d(x) = t\}$, we have $\nabla u(x, t)/|\nabla u(x, t)| = \nabla d(x)$ for x at the zero level set of u at time t.

Thus we interweave the following steps for value extension into our proposed algorithm:

1. Construct the flows defined by

$$\begin{cases} g_t + \frac{\nabla u}{|\nabla u|} \cdot \nabla g = 0, \\ g(x, 0) = f_0(x). \end{cases}$$

   and

$$\begin{cases} h_t + \frac{\nabla v}{|\nabla v|} \cdot \nabla h = 0, \\ h(x, 0) = f_0(x). \end{cases}$$

   Solve them at the same time as the PDE's for the signed distance function.

2. Set

$$f(x) = \begin{cases} h(x, T), & \text{if } \phi_0(x) > 0 \text{ and } u(x, T) = 0, \\ g(x, T), & \text{if } \phi_0(x) < 0 \text{ and } v(x, T) = 0. \end{cases}$$

   Use the T computed for construction of the signed distance function.

3. Repeat from the first step using $\phi_0(x) = \phi(x)$ and $f_0(x) = f(x)$ if desired. This may improve the solution.

Note ENO interpolation is extremely important in this application since the solution is generally discontinuous at kinks of the signed distance function. We show numerical results of this approach alongside results for the signed distance function in the next section.

grid size	dist error	order	extend error	order
50	0.00014897		0.000977022	
100	$4.64561 \cdot 10^{-6}$	5.0030	$6.20035 \cdot 10^{-5}$	3.9780
200	$1.09534 \cdot 10^{-7}$	5.4064	$4.67794 \cdot 10^{-6}$	3.7284
400	$1.64001 \cdot 10^{-9}$	6.0615	$1.77818 \cdot 10^{-7}$	4.7174
800	$8.04658 \cdot 10^{-11}$	4.3492	$9.59511 \cdot 10^{-9}$	4.2120

Table 1: Errors and orders of accuracy for the constructed signed distance function and extended values.

grid size	1st deriv error	order	2nd deriv error	order
50	0.00567636		0.12875	
100	0.00169254	1.7458	0.0255799	2.3315
200	0.000447995	1.9176	0.00286245	3.1597
400	0.000115987	1.9495	0.000729368	1.9725
800	$2.99124 \cdot 10^{-5}$	1.9551	0.000188086	1.9553

Table 2: Errors and orders of accuracy for the first and second derivatives of the constructed signed distance function.

2 Numerical Study

2.1 Simple Case

Let

$$\phi_0(x, y) = e^{x+y}\left(x^2 + y^2 - \frac{1}{4}\right)$$

so that the interface of interest is a circle centered at the origin with radius 1/2. We use our proposed algorithm to construct the values of the signed distance function of magnitude less than or equal to 0.301636 so that the kink in the solution at the origin is avoided. In addition, we attempt to extend the values of

$$f_0(x) = e^{x+y}$$

off the interface constant in normal directions. We expect to be able to achieve a high order of accuracy in the approximation using SSP-RK of fourth order in time, WENO of fifth order in space, and fourth order ENO interpolation [7]. Table 1 collects our results, showing orders of accuracy of roughly four in the infinity norms of absolute errors in the constructed signed distance function and extended values. Table 2 shows the absolute errors and of the first and second derivatives of the constructed signed distance function. These derivatives are computed using second order central differencing. Orders of accuracy of roughly two are seen in these results.

2.2 Additional Iterations

Let

$$\phi_0(x, y) = (\sin(4\pi x)\sin(4y) + 2)(e^{x^2+y^2-1/4} - 1)$$

grid size	dist error	order	extend error	order
50	0.000631518		0.00266801	
100	$1.17956 \cdot 10^{-5}$	5.7425	0.000150751	4.1455
200	$5.82679 \cdot 10^{-7}$	4.3394	$1.06848 \cdot 10^{-5}$	3.8185
400	$4.02468 \cdot 10^{-8}$	3.8558	$7.44637 \cdot 10^{-7}$	3.8429
800	$2.7407 \cdot 10^{-9}$	3.8763	$6.3016 \cdot 10^{-8}$	3.5627

Table 3: Errors and orders of accuracy for the constructed signed distance function and extended values with an additional iteration.

so that the interface of interest is again a circle centered at the origin with radius 1/2. We use our proposed algorithm to construct the values of the signed distance function of magnitude less than or equal to 0.301636. In addition, we attempt to extend the values of

$$f_0(x) = e^{x+y}$$

off the interface constant in normal directions. In this case, the constructed signed distance function is unable to achieve an order of accuracy of four with the same numerical schemes as the previous example. In fact, the errors form the vector

$$(0.00972792, 0.00160612, 0.000319585, 4.52933 \cdot 10^{-5}, 7.2604 \cdot 10^{-6})$$

with orders of accuracy

$$(2.5986, 2.3293, 2.8188, 2.6412)$$

for a grid starting with 50 points in each dimension and double this for each following entry.

A closer investigation shows that a kink develops close to the zero level set of u at a certain point and time. This kink diminishes the accuracy of the approximation schemes and lead to large infinity norm errors. The remedy for this is the third step of our algorithm, which repeats computations using the latest constructed forms for initial conditions. Tables 3 and 4 show the errors of the signed distance function, its derivatives, and extended values off the interface after one additional iteration. The progression of error at a grid with 50 points in each dimension under more iterations follows the vector

$$(0.00972792, 0.000631518, 0.000331136, 0.000291222, 0.000298418, 0.000301849)$$

showing iterations one through six. In the course of 200 iterations, the error oscillates slightly but remains mostly stable, giving an error of 0.000277127 in the end.

2.3 Kinks

Let

$$\phi_0(x, y) = \min\{(x + 0.2)^2 + y^2 - (0.3)^2, (x - 0.2)^2 + y^2 - (0.3)^2\}$$

so that the interface of interest is a circle of radius 0.3 centered at $(-0.2, 0)$ in the left half plane and a circle of radius 0.3 centered at $(0.2, 0)$ in the right half plane. Thus there is a kink in the interface at $x = 0$. We use our proposed algorithm to construct the values of the signed distance function of magnitude less than or equal to 0.301636. In this case, we expect only first order accuracy using

grid size	1st deriv error	order	2nd deriv error	order
50	0.0120466		0.310691	
100	0.000150751	2.7492	0.0353025	3.1360
200	0.000468071	1.9366	0.00618495	2.5129
400	0.000116647	2.0046	0.00179208	1.7871
800	$2.99378 \cdot 10^{-5}$	1.9621	0.000435015	2.0425

Table 4: Errors and orders of accuracy for the first and second derivatives of the constructed signed distance function with an additional iteration.

the infinity norm due to the kink in the interface and the signed distance function. The errors of the constructed signed distance function starting with a grid with 50 points in each dimension, then repeatedly doubling up to 800 points, are

$$(0.0060961, 0.00322084, 0.00168399, 0.00085892, 0.00043338),$$

showing first order in the orders of accuracy

$$(0.9204, 0.9356, 0.9713, 0.9869).$$

2.4 Comparison

The results of standard level set reinitialization varies with the choice of regularization for its signum function. In general, our experiments show that $\epsilon = \sqrt{\Delta x}$ produces better error than $\epsilon = \Delta x$ and $\epsilon = |\nabla\phi_0|\Delta x$. However, the form of ϕ_0 still greatly influences the speed of convergence of the PDE

$$\psi_t + \mathrm{sgn}_\epsilon(\phi_0)(|\nabla\psi| - 1) = 0.$$

As mentioned in section 1.2, it is difficult to estimate how many time steps are actually needed to generate results of a certain order of accuracy. This property affects all the experiments we present when comparing results of standard level set reinitialization with those of our algorithm.

One example of this is the case considered in section 2.1. Using the same number of total time steps as in our algorithm on the same problem, the results of standard level set reinitialization with $\epsilon = \sqrt{\Delta x}$ are shown in Table 5. Errors in the constructed signed distance function show eventual slow convergence and compares poorly to our results in Table 1. Errors of the derivative and second derivative are even slower in their convergence and are again poor compared to our results in Table 2. In fact, it takes 97 time steps for a grid with 50 points in each dimension before standard level set reinitialization achieves the same size error in signed distance function as our algorithm does using 24 total time steps. Then, even doubling this number everytime the number of points in the grid in each dimension is doubled, the error of standard level set reinitialization subsequently becomes worse than that of ours by an order of magnitude. The errors are

$$(0.000141764, 1.72627 \cdot 10^{-5}, 3.87304 \cdot 10^{-7}, 3.43358 \cdot 10^{-8})$$

starting with the grid with 50 points in each dimension and ending with the grid with 400 points in each dimension. The corresponding orders of accuracy are

$$(3.0378, 5.4780, 3.4957).$$

grid size	dist error	order	1st deriv error	order	2nd deriv error	order
50	0.0324433		0.103063		0.924852	
100	0.02128	0.5995	0.0811287	0.3452	0.767022	0.2700
200	0.0118589	0.8435	0.0569891	0.5095	0.585	0.3908
400	0.00511372	1.2135	0.0325503	0.8080	0.342561	0.7721
800	0.00165395	1.6285	0.0140648	1.2106	0.162482	1.0761

Table 5: Errors and orders of accuracy for results of standard level set reinitialization. Results can be compared to those of Table 1 and 2.

grid size	dist error	order	1st deriv error	order	2nd deriv error	order
50	0.00112127		0.0233927		1.20496	
100	0.000184584	2.6028	0.00504454	2.2133	0.569893	1.0802
200	$1.21541 \cdot 10^{-5}$	3.9248	0.000750778	2.7483	0.140806	2.0170
400	$4.02073 \cdot 10^{-7}$	4.9178	0.000123405	2.6050	0.0211957	2.7319
800	$1.81624 \cdot 10^{-8}$	4.4684	$2.99856 \cdot 10^{-5}$	2.0411	0.00375	2.4988

Table 6: Errors and orders of accuracy for results of standard level set reinitialization. Results can be compared to those of Table 3 and 4.

However, now considering the case in section 2.2 and once again using the same number of time steps as our algorithm for the same problem, standard level set reinitialization performs as well as ours, with errors

$$(0.00188958, 0.000899057, 0.000333704, 5.58918 \cdot 10^{-5}, 5.33776 \cdot 10^{-6})$$

and orders of accuracy

$$(1.0716, 1.4298, 2.5779, 3.3883).$$

In our approach, we resorted to performing an additional iteration to achieve the order of accuracy we expect to get. When standard level set reinitialization uses time steps equivalent to the new total, including those of the additional iteration, results are also comparable and are shown in Table 6. Specifically, the errors in constructed distance and second derivatives are roughly an order of magnitude better and first derivatives are very similar.

As for variations of the form we tested, in both the above example cases, the PDE does not quite reach steady state for the choices of $\epsilon = \Delta x$ and $\epsilon = |\nabla \phi_0| \Delta x$ for certain gridsizes. In fact, this is true when $\epsilon = \sqrt{\Delta x}$ as well, though to a lesser extent. Furthermore, the errors in constructed signed distance function and its derivatives when $\epsilon = \Delta x$ or $\epsilon = |\nabla \phi_0| \Delta x$ is used are much larger than when $\epsilon = \sqrt{\Delta x}$. These undesirable characteristics, we believe, are due to perturbation of the interface location during the flow. Thus, speed is sacrificed for accuracy when $\epsilon = \sqrt{\Delta x}$ and accuracy sacrificed for speed when $\epsilon = \Delta x$ and $\epsilon = |\nabla \phi_0| \Delta x$.

Other variations we tested include replacing $\text{sgn}_\epsilon(\phi_0)$ in the PDE by $\text{sgn}_\epsilon(d)$, where d is the exact signed distance function and replacing ϕ_0 with the latest constructed signed distance function after each time step. For these changes, results are often better but surprisingly sometimes worse. Deeper analysis is needed to determine the reasons behind this behavior.

From these and additional experiements run on other examples, we conclude that our algorithm at worst performs just as well as standard level set reinitialization for constructing signed distance functions over a fixed region. Even in this case, however, our errors are usually better, especially on coarser grids. On the other hand, in general, our algorithm usually performs noticeably better. In the case of a shrinking tube such as encountered by local level set methods, our approach is obviously superior to that of standard level set reinitialization. Numerical results to back up the theoretical remarks made in section 1.2 show that standard level set reinitialization does not converge when the number of time steps is fixed as the grid becomes finer and the tube becomes thinner. Our algorithm, on the other hand, displays errors such as

$$(0.00014897, 1.73394 \cdot 10^{-7}, 7.88333 \cdot 10^{-9}, 4.3846 \cdot 10^{-10}, 2.57252 \cdot 10^{-11})$$

with orders of accuracy

$$(9.7468, 4.4592, 4.1683, 4.0912)$$

for the function of section 2.1 in a tube of radius $10\Delta t$ under 12 time steps.

The only advantages of standard level set reinitialization are in flexibility in creating variations, some of which may be better than what is currently available; slightly less memory usage; and the possibility that even with low order time discretization, the results in steady state may achieve high order of accuracy from the spatial discretization. This, if true, may increase the efficiency of standard level set reinitialization and make it more competitive with our approach. Our experiments using third order TVD-RK in time hint that this may be true but for lower order discretizations, the larger errors in time may end up contributing more significantly to the final error.

2.5 Three Dimensions

Our algorithm stays the same in higher dimensions. In three dimensions, we consider an example with

$$\phi_0(x, y, z) = e^{x+y+z}(x^2 + y^2 + z^2).$$

As the number of computations becomes too large for fine grids in three dimensions, we consider smaller grids with 25, 50, and 100 points in each dimension. The errors in the computed signed distance function for values of magnitude less than or equal to 0.301636 are given in the vector

$$(0.000980799, 0.000225319, 1.26161 \cdot 10^{-5})$$

with corresponding orders of accuracy

$$(2.1220, 4.1586).$$

Thus we achieve high order of accuracy in three dimensions as well as the grid is made finer.

3 Conclusion

We propose an algorithm for constructing approximate signed distance functions for a given interface. The algorithm is as efficient as standard level set reinitialization in general, and more efficient for specialized purposes such as when only local computations are of interest. Furthermore, the approximate functions constructed by our algorithm have high order of accuracy and smooth derivatives. With fourth order differencing in time, fifth order in space, and fourth order interpolation, we can construct fourth order approximations with at least second order first and second derivatives. In addition, slight modification of the algorithm gives high order extension of values off the interface constant in normal directions. Applications of our results include providing an initialization for closest point methods, allowing higher order results with level set methods, and tackling problems with values defined on interfaces. We seek to apply our approach to these problems as part of future.

4 Acknowledgements

The work of the first author is supported by NSF DMS-0511766, NSF DMS-0619173, and a Sloan Foundation Fellowship. The work of the second author is supported by NSF DMS-0513394 and a Sloan Foundation Fellowship.

5 Appendix

5.1 ENO Interpolating Polynomial

Consider the function $u(t)$ in $[a, b]$. Let

$$a = t_0 < t_1 < \ldots < t_n = b$$

denote a uniform grid over $[a, b]$ with stepsize Δt. Furthermore, let u_i denote $u(t_i)$ for all i. We consider here the details involved in finding the root of u in $[t_j, t_{j+1}]$ when $u_j u_{j+1} \leq 0$.

For a second order accurate approximation to the location of the root, we may construct the linear interpolant $p_1(t)$ through the points (t_j, u_j) and (t_{j+1}, u_{j+1}), then solve the linear equation corresponding to $u^1 = 0$ for the location of the root.

Similarly, for a third order accurate approximation, we may consider constructing a quadratic interpolant instead; however, there is no natural choice for which points (t_i, u_i) to use for this. For example, it is natural to use (t_j, u_j) and (t_{j+1}, u_{j+1}), since they are close to the root of interest, but the choice of the next closest, (t_{j-1}, u_{j-1}) or (t_{j+2}, u_{j+2}), is not fixed. Use of either one gives a quadratic interpolant that can be used to find the location of the root.

In fact, either choice leads to a third order accurate approximation of the root when u is smooth. When u is not smooth, however, it is better to pick the one that avoids discontinuous derivatives in its interval. The ENO idea carries this through by building up the interpolating polynomial in Newton's form and using the Newton divided difference corresponding to the highest degree term in the polynomial as a smoothness indicator.

Thus the ENO scheme builds its quadratic interpolating polynomial $p_2(t)$ by starting with the linear interpolating polynomial

$$p_1(t) = u[t_j] + u[t_j, t_{j+1}](t - t_j)$$

and picking out the smoother option by choosing

$$p_2(t) = p_1(t) + (t - t_j)(t - t_{j+1}) \cdot \begin{cases} u[t_{j-1}, t_j, t_{j+1}], & \text{if } |u[t_{j-1}, t_j, t_{j+1}]| \leq |u[t_j, t_{j+1}, t_{j+2}]| \\ u[t_j, t_{j+1}, t_{j+2}], & \text{if } |u[t_{j-1}, t_j, t_{j+1}]| > |u[t_j, t_{j+1}, t_{j+2}]| \end{cases}$$

The notation used here is the usual one involving Newton divided differences found in Newton's form for an interpolating polynomial. Higher degree ENO interpolating polynomials are built by continuing this process.

Classical root finding methods can then be used to extract the roots of these polynomials. We use Newton's method with initial guesses of either $t = t_j$ or $t = t_{j+1}$ or the root generated from linear interpolation and iterate until successively generated approximations differ by close to machine tolerance. This usually only takes a couple of iterations.

References

[1] D. Adalsteinsson and J.A. Sethian. A Fast Level Set Method for Propagating Interfaces. *J. Comput. Phys.*, 118:269–277, 1995.

[2] D. Adalsteinsson and J.A. Sethian. The Fast Construction of Extension Velocities in Level Set Methods. *J. Comput. Phys.*, 148:2–22, 1999.

[3] S. Chen, B. Merriman, S. Osher, and P. Smereka. A Simple Level Set Method for Solving Stefan Problems. *J. Comput. Phys.*, 135:8–29, 1997.

[4] D.L. Chopp. Some Improvements of the Fast Marching Method. *SIAM J. Sci. Comput.*, 23(1):230–244, 2001.

[5] M.G. Crandall and P. Lions. Viscosity Solutions of Hamilton-Jacobi Equations. *Trans. Amer. Math. Soc.*, 277(1):1–42, 1983.

[6] B. Engquist, A.-K. Tornberg, and Y.-H. Tsai. Discretization of Dirac Delta Functions in Level Set Methods. *J. Comput. Phys.*, 207(1):28–51, 2005.

[7] A. Harten, B. Engquist, S. Osher, and S.R. Chakravarthy. Uniformly High-Order Accurate Essentially Nonoscillatory Schemes. III. *J. Comput. Phys.*, 71(2):231–303, 1987.

[8] J. Helmsen, E. Puckett, P. Colella, and M. Dorr. Two New Methods for Simulating Photolithography Development in 3D. *Proc. SPIE*, 2726:253–261, 1996.

[9] G.S. Jiang and D. Peng. Weighted ENO Schemes for Hamilton Jacobi Equations. *SIAM J. Scient. Comput.*, 21(6):2126–2143, 2000.

[10] S. Osher. A Level Set Formulation for the Solution of the Dirichlet Problem for Hamilton-Jacobi Equations. *SIAM J. Math. Anal.*, 24(5):1145–1152, 1993.

[11] S. Osher and J.A. Sethian. Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *J. Comput. Phys.*, 169(1):12–49, 1988.

[12] S. Osher and C.-W. Shu. High order essentially non-oscillatory schemes for Hamilton-Jacobi equations. *SINUM*, 28:907–922, 1991.

[13] D. Peng, B. Merriman, S. Osher, H.K. Zhao, and M. Kang. A PDE-Based Fast Local Level Set Method. *J. Comput. Phys.*, 155(2):410–438, 1999.

[14] G. Russo and P. Smereka. A Remark on Computing Distance Functions. *J. Comput. Phys.*, 163(1):51–67, 2000.

[15] J.A. Sethian. Fast Marching Level Set Methods for Three Dimensional Photolithography Development. *Proc. SPIE*, 2726:261–272, 1996.

[16] C.-W. Shu and S. Osher. Efficient Implementation of Essentially Nonoscillatory Shock-Capturing Schemes. *J. Comput. Phys.*, 77(2):439–471, 1988.

[17] R.J. Spiteri and S.J. Ruuth. A New Class of Optimal High-Order Strong-Stability-Preserving Time Discretization Methods. *SIAM J. Numer. Anal.*, 40(2):469–491, 2002.

[18] M. Sussman, P. Smereka, and S. Osher. A Level Set Method for Computing Solutions to Incompressible Two-Phase Flow. *J. Comput. Phys.*, 114:146–159, 1994.

[19] A.-K. Tornberg and B. Engquist. The Segment Projection Method for Interface Tracking. *Comm. Pure Appl. Math.*, 56(1):47–79, 2003.

[20] Y.-H. Tsai, L.-T. Cheng, S. Osher, and H.K. Zhao. Fast Sweeping Algorithms for a Class of Hamilton-Jacobi Equations. *SIAM J. Numer. Anal.*, 41(2):673–694,, 2003.

[21] J.N. Tsitsiklis. Efficient Algorithms for Globally Optimal Trajectories. *IEEE Transactions on Automatic Control*, 50:1528–1538, 1995.

[22] J.-J. Xu, Z. Li, J. Lowengrub, and H.K. Zhao. A Level-Set Method for Interfacial Flows with Surfactant. *J. Comput. Phys.*, 212(2):590–616, 2006.

[23] H.K. Zhao. A Fast Sweeping Method for Eikonal Equations. *Math. Comp.*, 74(250):603–627, 2005.