A Study of Numerical Methods for the Level Set Approach

Pierre A. Gremaud*, Christopher M. Kuster†, and Zhilin Li‡

October 18, 2005

Abstract

The computation of moving curves by the level set method typically requires reinitializations of the underlying level set function. Two types of reinitialization methods are studied: a high order "PDE" approach and a second order Fast Marching method. Issues related to the efficiency and implementation of both types of methods are discussed, with emphasis on the tube/narrow band implementation and accuracy considerations. The methods are also tested and compared. Fast Marching reinitialization schemes are faster but limited to second order, PDE based reinitialization schemes can easily be made more accurate but are slower, even with a tube/narrow band implementation.

1 Introduction

This paper analyzes numerical issues related to the implementation of the level set method for moving interface problems. Let $\Gamma(t)$, $t > 0$ be a one-parameter family of curves in $\mathbb{R}^2$ resulting from the motion of an initial curve $\Gamma(0)$ under a velocity field $\mathbf{v}$. The idea of the level set method, as devised by Osher and Sethian in 1987 [OS88], see also [OF02, Set99b] for more recent reviews, consists in introducing a function $\varphi(\mathbf{x}, t)$ that represents the curve $\Gamma(t)$ as the zero level set of φ, i.e., $\Gamma(t) = \{\mathbf{x};\ \varphi(\mathbf{x}, t) = 0\}$. If $\{\mathbf{x}(t),\ t \geq 0\}$ denotes the path of a point on the moving curve, then clearly $\varphi(\mathbf{x}(t), t) = 0$ for $t \geq 0$. By taking the time derivative of that last relation, one observes that the function φ is convected by the velocity field $\mathbf{v}$, i.e.

$$\partial_t \varphi + \mathbf{v} \cdot \nabla \varphi = 0. \tag{1}$$

*Department of Mathematics and Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC 27695-8205, USA(`gremaud@ncsu.edu`). Partially supported by the National Science Foundation (NSF) through grants DMS-0244488 and DMS-0410561.

†Department of Mathematics and Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC 27695-8205, USA(`cmkuster@ncsu.edu`). Partially supported by the National Science Foundation (NSF) through grant DMS-0244488.

‡Department of Mathematics and Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC 27695-8205, USA(`zhilin@unity.ncsu.edu`). Partially supported by USA NSF-NIH grant #0201094, a USA-ARO under grant number 43751-MA, and a USA-NSF grant DMS-0412654.

The vector field $\mathbf{v}$ is prescribed on the curve but is arbitrary elsewhere. By decomposing $\mathbf{v}$ into its normal and tangential velocity $\mathbf{v}_n$ and $\mathbf{v}_\tau$, it becomes apparent that only the normal velocity $\mathbf{v}_n = F\frac{\nabla\varphi}{|\nabla\varphi|}$ matters, where F stands for the normal speed. The level set equation (1) becomes

$$\partial_t\varphi + F\,|\nabla\varphi| = 0. \tag{2}$$

Equation (2) is often solved locally near the interface instead of everywhere in space. The advantages of such an approach are obvious in terms of costs and have been discussed at length in the literature, see for instance [AS95, PMO$^+$99]. At each time step, (2) is solved in a tube/narrow band around the current curve. Once the curve, or more precisely the function φ, has been updated, the corresponding tube/narrow band is rebuilt. The process then is repeated. It is convenient to initialize $\varphi(\cdot,0)$ as the signed distance function to $\Gamma(0)$. One way to extend F away from $\Gamma(t)$ is to solve

$$\nabla F_{ext}\cdot\nabla\varphi = 0, \tag{3}$$

$$F_{ext} = F \qquad \text{on } \Gamma(t). \tag{4}$$

If ϕ is then evolved according to the extended normal speed F_{ext}, i.e.

$$\partial_t\varphi + F_{ext}\,|\nabla\varphi| = 0,$$

then it is easy to check that φ remains a distance function [AS99]. After discretization and/or use of a tube/narrow band implementation, this property may be lost. In many cases, this eventually leads to numerical difficulties, see e.g. [SSO94]. A reinitialization of φ has to be applied.

To the authors' knowledge, accuracy issues related to the use of such tube/narrow band techniques have received far less attention than pure efficiency consideration (through complexity analysis for instance). The accuracy in evolving a moving interface is crucial to many applications. For the level set method, the accuracy depends on two main factors, one is the reinitialization process so that the level set function is not too steep or flat. The other one is the evolution process whose discretization is well understood. A number of efficient methods can be used as part of the reinitialization process. Two such methods are Fast Marching [Set99a] and Fast Sweeping [TCOZ03]. The Fast Marching method is used here.

In this paper, a simple, robust, globally second order accurate method for solving (2) in a tube/narrow band is proposed. The algorithm consists of three parts. First, normal velocity and level set functions are initialized "on" the current curve through a projection method. Two such projections are considered here: a method based on constrained optimization, and the Taylor expansion based method from [HLOZ97, LZG99]. Second, a second order version of the Fast Marching method [SV00, SV03] is used for the extension of the variables at play in the tube/narrow band, away from the current curve. Third, the level set equation is evolved by a third order TVD WENO-RK method [GS98]. The algorithm is described fully in Section 2. Reinitialization can also be achieved efficiently by using a "PDE-approach" that does not require projection, see for instance [SSO94] or (5) below. Numerical examples and results comparing the efficiency and accuracy of the methods are discussed in Section 3. Conclusions are offered in Section 4.

2 Description of the algorithm

For the sake of simplicity, we consider the initial curve $\Gamma(0)$ to be a simple, smooth, closed curve in $\mathbb{R}^2$. An underlying uniform Cartesian grid $\mathcal{T}_h$ of size h is then considered. Assume we want to follow the evolution of $\Gamma(0)$ through (2) from $t = 0$ to $t = T$. Approximations Φ^n to φ at time $t^n = n\Delta t$, $n = 1, \ldots, N$, $\Delta t = T/N$ and at (some of) the nodes of the mesh will be successively computed. For stability purposes, the condition $\Delta t = \mathcal{O}(h)$ is enforced throughout. We denote by $\Gamma^n(k)$, the set of the nodes of the mesh within a distance kh of the zero level set of Φ^n, i.e.

$$\Gamma^n(k) = \{P \in \text{nodes of } \mathcal{T}_h; \text{dist}(P, \{\Phi^n = 0\}) \leq \, k\, h\}.$$

The main step of the algorithm consists in going from Φ^n to Φ^{n+1}

```
n = 0
$\Phi^0$ = nodal values of $\varphi(\cdot, 0)$
while  n < N
       % PROJECTION
       $\forall P \in \Gamma^n(1)$  compute $\text{dist}(P, \Gamma^n) = \tilde{\Phi}^n$ and normal speed $\tilde{F}$
       % REINITIALIZATION
       extend  $\tilde{\Phi}^n$  and  $\tilde{F}$  to  $\Gamma^n(k_2)$
       % EVOLUTION
       solve (2) from $t^n$ to $t^{n+1}$ in $\Gamma^n(k_1)$  to yield $\Phi^{n+1}$
       n = n + 1
end
```

In our implementation, we take $k_1 = 5$ and $k_2 = 10$.

Given a current level set function Φ^n (which is in general not a distance function), the PROJECTION step computes the values of the corresponding corrected reinitialized distance function $\tilde{\Phi}^n$ in a domain that is one-layer thick on each side of the interface. Application dependent techniques have to be used to extend F in that domain. In the present implementation, the REINITIALIZATION step is done through a second order Fast Marching method (see subsection 2.2), or the PDE approach. Finally, the EVOLUTION is done by a classical WENO scheme (see subsection 2.3).

REINITIALIZATION can also be achieved by considering an alternate "PDE-approach" that does not require PROJECTION. In such an approach, the current level set function

Φ^n is considered as an initial value for the problem

$$\partial_{\bar{t}}\varphi + \mathrm{sign}(\Phi^n)(|\nabla\varphi| - 1) = 0, \tag{5}$$

which is to be solved to steady state as the fictitious time $\bar{t} \to \infty$, where sign is a smoothed sign function, see [SSO94] for example. Often, a good approximation of the distance function is good enough and only a few iterations are needed if the computational domain is large enough. This approach has the merit of being simple however, it may not be very efficient (for instance, not taking enough time steps may result in severe losses in accuracy if the computational tube is thin).

2.1 Computing the orthogonal projections

Before we apply the fast marching method for reinitialization process, we need to know a good approximation of the distance function within one grid size of the interface, that is, within $\Gamma^n(1)$. This can be done in various ways.

2.1.1 A direct approach from a quadratic equation.

A simple third order method has been proposed in [HLOZ97, LZG99]. This method is third order accurate and "direct", i.e., no iterations are needed.

Assume that $\mathbf{x}$ is a grid point near the interface. Let $\mathbf{X}^*$ be the orthogonal projection of $\mathbf{x}$ on the interface. Considering the Taylor expansion of φ at $\mathbf{x}$, we have

$$\begin{aligned} 0 &= \varphi(\mathbf{X}^*) \\ &= \varphi(\mathbf{x}) + \nabla\varphi(\mathbf{x})\cdot(\mathbf{X}^* - \mathbf{x}) + \frac{1}{2}(\mathbf{X}^* - \mathbf{x})^T \mathrm{He}(\varphi(\mathbf{x}))(\mathbf{X}^* - \mathbf{x}) + \mathcal{O}(|\,\mathbf{X}^* - \mathbf{x}\,|^3), \end{aligned}$$

where He is the Hessian matrix of φ. For $\mathbf{x}$ close to the interface,

$$\mathbf{X}^* - \mathbf{x} \approx \alpha\nabla\varphi(\mathbf{x}), \text{ for some } \alpha \in \mathbb{R}.$$

This leads to the following quadratic equation for α

$$\begin{aligned} 0 &= \varphi + \alpha|\nabla\varphi|^2 + \frac{\alpha^2}{2}\nabla\varphi^T \mathrm{He}(\varphi)\nabla\varphi, \\ &= \varphi + \alpha(\varphi_x^2 + \varphi_y^2) + \frac{\alpha^2}{2}(\varphi_{xx}\varphi_x^2 + 2\varphi_{xy}\varphi_x\varphi_y + \varphi_{yy}\varphi_y^2), \end{aligned}$$

where the above functions are all evaluated at $\mathbf{x}$. If the derivatives are approximated by standard five-point finite difference formulas, then the projection method is third order accurate, i.e.

$$|\,|\mathbf{X}^* - \mathbf{x}| - \mathrm{dist}(\mathbf{x}, \Gamma)\,| = \mathcal{O}(h^3).$$

2.1.2 A constrained minimization method

We also propose a projection method based a constrained minimization problem. This method has the same order of accuracy as the direct method mentioned above, gives slightly better results (that is, with smaller error constant) and has about the same cost. For any node $P = (x_0, y_0) \in \Gamma^n(1)$ where the distance function is to be evaluated, we construct the unique polynomial q_P in $Q_2 = \text{span}\{1, x, y, xy, x^2, y^2, x^2y, xy^2, x^2y^2\}$ agreeing with Φ^n at the 9 point centered at P, see Appendix A for details. For the present purpose, $\Gamma^n(1)$ is taken as to contain nodes admitting neighbors of opposite sign, but also neighboring nodes of values less than h^2. The square of the Cartesian distance, $\|x - x_0\|^2$ is then minimized subject to the constraint $q_P = 0$. The Lagrangian for this problem is

$$\mathcal{L}(x, y, \lambda) = (x - x_0)^2 + (y - y_0)^2 + \lambda\, q_P(x, y). \tag{6}$$

To find the point (x, y), we solve the system

$$\nabla\mathcal{L}(x, y, \lambda) = 0, \tag{7}$$

i.e.,

$$\begin{aligned} 2(x - x_0) + \lambda \frac{\partial q_P(x, y)}{\partial x} &= 0 \\ 2(y - y_0) + \lambda \frac{\partial q_P(x, y)}{\partial y} &= 0 \\ q_P(x, y) &= 0. \end{aligned} \tag{8}$$

Newton's method is used to solve this nonlinear system; then the value of Φ at P is set to be the distance between that point and the solution. The sign of the distance function at each point is taken to be the same as the sign of the current level set. Finally, the gradient at each point near the interface is computed using centered differences on the level set and then normalizing the result so that the gradient has magnitude 1.

2.1.3 Other approaches

A method suggested in [LFO95] involves starting at a mesh point $\mathbf{x}_{ij}$ and repeatedly taking a step, in the direction of the (negative) gradient, of length given by φ at the current point. The loop is terminated when $|\varphi|$ is small enough. It is noted [LFO95] that this process does not always converge: if the onset of divergence is detected ($|\varphi|$ larger than some predetermined magnitude) or if too many iterations are taken, a different method is used (see [LFO95] for details). Another method, used by Chopp [Cho01], constructs local bi-cubic approximations. Newton's method is then used to find points on the zero level set of those functions. Note that for both of those methods, what is minimized is an approximate function φ rather than a genuine distance.

More sophisticated methods have also been proposed such as semi-Lagrangian particle level set methods, see e.g. [ELF05]. The present Fast Marching reinitialization method is simpler than the one discussed in [ELF05], has a higher asymptotic rate of convergence and has comparable conservation properties, see Section 3.

2.2 Reinitialization and Fast Marching

The Fast Marching algorithm, first introduced by Tsitsiklis [Tsi95] and then popularized by Sethian and his coworkers, see e.g. [Set99a], is a single-pass Dijkstra-like algorithm designed to solve graph traversal problems. This algorithm calculates the nodes in a smallest first ordering by using a Heap Sort to keep track of the current smallest valued node. The Heap Sort data structure is order $\log(M)$ for insertions, deletions, and updates where M is the total number of nodes in the domain. The use of this data structure, along with the single-pass nature of the algorithm, means that the algorithm as a whole is order $M\log(M)$.

Here, the Fast Marching method is used to compute the distance function and the extension of the normal speed F_{ext}, see (3), in a tube/narrow band around the interface by solving the Eikonal equation

$$\begin{aligned} |\nabla\varphi| &= 1, && (9)\\ \varphi &= 0 \qquad \text{on } \Gamma(t), && (10)\end{aligned}$$

for φ and solving (3,4) for F_{ext}. A second order discretization is used for solving (9,10) as described below. The system (3,4) is discretized at first order. Following the method described in [SV00, SV03] on a Cartesian grid, consider a node X_0 at which the solution φ is to be computed and two neighboring nodes X_1 and X_2 at which the values of φ (φ_ℓ, $\ell = 1,2$) and its derivatives ($\partial_x\varphi_\ell, \partial_y\varphi_\ell$, $\ell = 1,2$) are known or have already been computed. We define

$$N_\ell = \frac{X_0 - X_\ell}{|X_0 - X_\ell|}, \quad \ell = 1,2 \quad \text{and} \quad \mathcal{N} = \begin{bmatrix} N_1 \\ N_2 \end{bmatrix},$$

The directional derivatives in the directions N_1 and N_2 are approximated to second order by

$$D_\ell\varphi = 2\frac{\varphi_0 - \varphi_\ell}{|X_0 - X_\ell|} - N_\ell \cdot [\partial_x\varphi_\ell, \partial_y\varphi_\ell] \quad \ell = 1,2.$$

Those approximate directional derivatives are linked to the gradient by

$$D_\ell\varphi = N_\ell \cdot \nabla\varphi + \mathcal{O}(|X_0 - X_i|^2) \qquad \text{or} \qquad D\varphi = \mathcal{N}\nabla\varphi + \mathcal{O}(h^2), \tag{11}$$

where $D\varphi = [D_1\varphi, D_2\varphi]$. Using the fact that $\mathcal{N}^t\mathcal{N} = I$ on the present Cartesian grid and plugging into (9), the following quadratic equation for φ_0 is obtained

$$D\varphi^t D\varphi = 1. \tag{12}$$

If the value of φ is only known at a single neighboring node, X_1, then the gradient is assumed to lie entirely along the line connecting X_1 and X_0. If this is the case, then the following relation is used

$$\varphi_0 = \varphi_1 + h \tag{13}$$

2.3 Evolution

A third order total variation decreasing (TVD) Runge-Kutta scheme [GS98] coupled with a third order WENO scheme for the spatial derivative [JP00]. The method solves numerically the differential equation

$$\partial_t \varphi = L(\varphi). \tag{14}$$

by

$$\begin{aligned} \varphi^{(1)} &= \varphi^n + \Delta t L(\varphi^n), \\ \varphi^{(2)} &= \frac{3}{4}\varphi^n + \frac{1}{4}\varphi^{(1)} + \frac{1}{4}\Delta t L(\varphi^{(1)}), \\ \varphi^{n+1} &= \frac{1}{3}\varphi^n + \frac{2}{3}\varphi^{(2)} + \frac{2}{3}\Delta t L(\varphi^{(2)}). \end{aligned} \tag{15}$$

where $L(\varphi)$ is the WENO approximation of $\mathbf{v} \cdot \nabla\varphi$. A time step of $\Delta t = \Delta/(2\, s_{max})$ where s_{max} is the absolute maximum normal speed in the tube/narrow band was used.

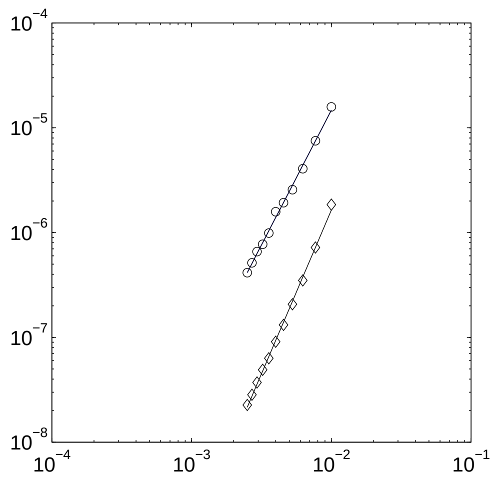

Figure 1: *Constant expansion using Fast Marching with projection from 2.1.2: L^1 (diamond) and L^∞ (circles) rates of convergence in the tube/narrow band under grid refinement: L^1-rate ≈ 3.14, L^∞-rate ≈ 2.58.*

3 Numerical examples

All tests were performed using a 1.2 GHz PowerPC G4 processor with 768 MB of DDR SDRAM. The results given below have been computed using the algorithm described above with various projection methods. The L^1 and L^∞ errors are computed in the tube/narrow band (a trapezoidal rule is used for the calculation of the L^1 error). The error is taken here in terms of level set functions, i.e., it is the difference between the exact distance function to the final curve and its numerical approximation (this requires a "final" reinitialization). By the very nature of the present tube/narrow band method, the computational domain shrinks with h. A third order method with respect to the L^1 norm as measured above is in fact second order on a fixed domain.

In the PDE approach, the level set functions are not reinitialized to the signed distance function and the error is measured from the computed orthogonal projections.

3.1 Constant Expansion

The first example is a circle expanding/shrinking along the normal direction, that is $\mathbf{v} = F(t)\mathbf{n}$, where $\mathbf{n}$ is the unit normal direction. The results are shown for $F(t) = 1$. This simple case is in fact representative: results for more general functions $F(t)$ are comparable. For the Fast Marching method with projection from 2.1.2, the L^1 and L^∞ rates of convergence of order are found to be roughly 3 (3.14) and slightly above 2 (2.58), respectively, see Figure 1. Again, this corresponds to a second order method in L^1 on a fixed spatial domain.

$1/h$	PDE	rate	Taylor	rate	Min.	rate
100	.738	–	.418	–	.435	–
160	3.07	3.03	1.08	2.02	1.11	1.99
220	7.78	2.92	2.23	2.28	2.27	2.25
280	16.2	3.04	4.00	2.42	4.16	2.52
340	28.9	2.98	6.46	2.47	6.66	2.42
400	47.1	3.01	10.1	2.75	9.96	2.48

Table 1: *CPU time (in seconds) versus size of the mesh. The above rates refer to α, where time $= \mathcal{O}(h^{-\alpha})$ for the PDE approach (5), the Fast Marching method with projections from 2.1.1 (Taylor) and from 2.1.2 (Min.) respectively.*

Table 1 shows the computational complexity of the PDE approach (5) versus two projection methods: the Taylor expansion based approach of Section 2.1.1 and the constrained minimization of 2.1.2. The PDE approach is done in the entire domain. The errors are against the exact solution. For the Fast Marching approach, the error is measured from the computed level set function in the tube/narrow band $|\varphi| \leq 5h$. The number of nodes in the band is of order $\mathcal{O}(1/h)$. Therefore, at each time step, the PROJECTION and REINITIALIZATION steps cost $\mathcal{O}(1/h \log 1/h)$ by the standard complexity analysis of Fast Marching. As there are $\mathcal{O}(1/h)$ time steps by stability, the total number of operations for the algorithm is of order $\mathcal{O}(1/h^2 \log 1/h)$. This crude analysis is confirmed by the result of the above Table 1. In the PDE approach, we do the reinitialization and solve the level set equation (5) in the entire domain ($\mathcal{O}(1/h^2)$ operations), however, only one reinitialization step is carried out (in $\bar{t}$, see (5)) for each time step in (2). The total number of operations for the PDE approach on the entire domain is thus of order $\mathcal{O}(1/h^3)$, an observation that also confirmed in Table 1.

In Table 2, we show a grid refinement analysis of different methods for the expanding circle. To compute the error, each node in $\Gamma^N(1)$ is numerically projected onto the current curve; the error reported in Table 2 corresponds to the maximum distance between those projected points and the exact curve.

$1/h$	PDE	rate	Taylor	rate	Min.	rate
100	4.09(-6)	–	1.01(-5)	–	7.53(-6)	–
160	1.26(-6)	2.51	2.60(-6)	2.89	2.06(-6)	2.76
220	4.75(-7)	3.06	1.35(-6)	2.06	1.11(-6)	1.94
280	2.40(-7)	2.83	7.10(-7)	2.66	6.06(-7)	2.51
340	1.36(-7)	2.93	4.92(-7)	1.89	3.99(-7)	2.15
400	7.58(-8)	3.60	3.67(-7)	1.80	2.93(-7)	1.90

Table 2: *Constant expansion: L^∞-error of the projection for the PDE approach, the "Taylor" approach from 2.1.1 and the minimization method of 2.1.2.*

In this trivial example, all methods converge with the expected rate, three for the PDE approach (on the full spatial domain) thanks to the use of the third order TVD WENO-RK scheme (15) and two for the Fast Marching based algorithms in spite of the narrow band implementation.

3.2 Rotations

The second example is the rotation of an off center circle. Initially, the circle is $(x-0.5)^2+y^2=0.25^2$. The velocity field is $\mathbf{v}=(y,-x)$. The errors are measured after one complete rotation.

Table 3 shows the L^∞-error, the corresponding convergence rate and the CPU time obtained from different methods after one complete rotation. It is possible to implement the PDE approach in a tube/narrow band. The method is efficient if the parameters are chosen appropriately. First of all, the tube should not be too narrow, otherwise the error from the boundary may propagate. We choose the tube width as $\delta = 10h$. Secondly, sufficiently

$1/h$	PDE	rate	CPU	PDE-tube	rate	CPU	Taylor	rate	CPU	Min.	rate	CPU
64	4.86(-3)	–	26.6	5.26(-3)	–	23.8	6.04(-3)	–	3.76	5.70(-3)	–	3.87
128	1.90(-4)	4.68	220	1.90(-4)	4.79	115	4.90(-4)	3.62	19.2	4.06(-4)	3.81	19.7
256	9.87(-6)	4.27	1808	1.07(-5)	4.15	675	1.20(-4)	2.03	118	9.07(-5)	2.08	119
512	1.33(-6)	2.89	14555	1.51(-6)	2.82	4167	2.63(-5)	2.19	900	2.16(-5)	2.07	900

Table 3: *Rotation of an off center circle: L^∞-errors, convergence rates and CPU times (in seconds) after one complete rotation.*

many reinitialization steps are needed, usually $int(\delta/h)$ steps so that the reinitialization process reaches the tube boundary. The error is measured from all the orthogonal projections of $\Gamma^n(1)$, i.e., of the first layer of nodes. Note that for the PDE approach, the level set function may not be a good approximation of the signed distance function. But this does not bring inaccuracy to the level set method as long as the level set function remains smooth and well-behaved (not too steep or flat) near the front. The computed orthogonal projections within one grid size of the interface is third order accurate. In Table 3, PDE refers to the PDE approach in the entire domain while PDE-tube corresponds to that approach in a narrow band of width $10h$. For both versions, the PDE approach is again observed to be third order accurate but is significantly slower than the Fast Marching method with either projection.

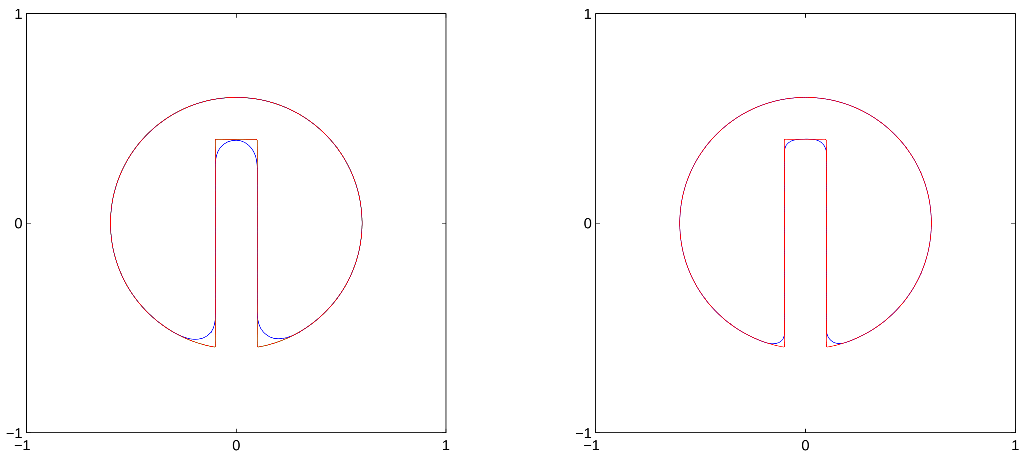

Figure 2: *Comparison after one rotation on a 100×100 grid, left, and a 200×200 grid, right, for the Zalesak's disk using Fast Marching with projection from 2.1.2.*

The third example is a single rotation of the shape (called Zalesak's disk) shown in Figure 2. In this example, the velocity field is taken as fixed, $v = (y, -x)$. No extension of F was taken. This example is similar to the one used in [ELF05] and other publications.

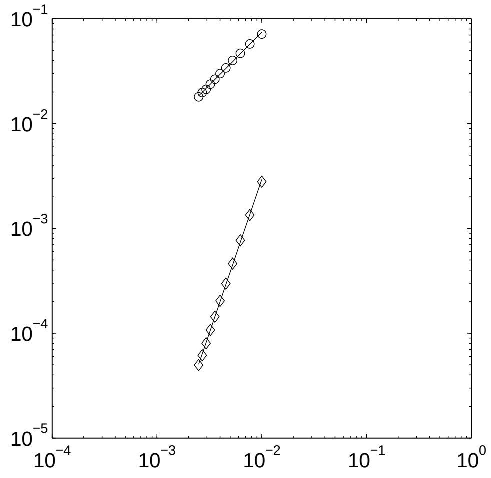

Figure 3: *Rotation of Zalesak's disk with Fast Marching with projection from 2.1.2:* L^1 *(diamond) and* L^∞ *(circles) rates of convergence in the tube/narrow band under grid refinement:* L^1*-rate* ≈ 2.93*,* L^∞*-rate* ≈ 1.01*.*

For Fast Marching with projection from 2.1.2, the L^1 and L^∞ rates of convergence of order in the tube/narrow band are found to be roughly 3 (2.93) and 1 (1.01), respectively, see Figure 3. The order 1 convergence in the maximum norm is consistent with the presence of "shocks" (discontinuities in the first derivatives) for the present example.

As is well known, the numerical diffusion inherent to the discretization process may prevent area conservation in the present example. In order to minimize this type of effect, a semi-Lagrangian method involving not only the evolution of a level set function but also of particles was tested in [ELF05]. The area conservation properties along with the L^1 error are reported for the Fast Marching method with projection from 2.1.2 in Table 4, see also Figure 2. It can be noted by comparison with the results from [ELF05] (see Tables 2 and 3 there) that even though the present method is much simpler conceptually, it preserves the area in a very similar as the semi-Lagrangian particle method proposed there or the HJ-WENO method. Further, the present algorithm has a higher convergence rate (order 2 in fixed domains) than the method from [ELF05] (order 1) .

	Grid Cells	Area	% Area loss	L-1 Error	Order	CPU
	Exact	0.933				
1 revolution	100	0.927	0.652	2.79(-3)	N/A	12.3
	200	0.931	0.222	3.95(-4)	2.82	73.5
2 revolutions	100	0.925	0.801	3.89(-3)	N/A	24.1
	200	0.930	0.258	5.14(-4)	2.92	146

Table 4: *Area loss, L^1 errors and rates and CPU time (in seconds) for the Zalesak's disk rotation problem for Fast Marching with projection from 2.1.2.*

4 Conclusions

Fully second order numerical methods for the reinitialization and evolution processes of a level set function are investigated. The methods result from the combination of (i) an accurate process to find the orthogonal projections at grid points directly next to the interface, (ii) a second order Fast Marching method and (iii) a third order TVD WENO-RK scheme. For computational efficiency, all calculations are made only in the neighborhood of the current curve (tube/narrow band) for this approach.

A qualitative analysis and comparison between a high order PDE approach and the above Fast Marching based methods are given. Both types of methods are found to work well, the latter ones being faster, better suited to tube/narrow band implementation but limited to second order.

References

[AS95] David Adalsteinsson and James A. Sethian, *A fast level set method for propagating interfaces*, J. Comput. Phys. **118** (1995), no. 2, 269–277. MR MR1329634 (96a:65154)

[AS99] D. Adalsteinsson and J. A. Sethian, *The fast construction of extension velocities in level set methods*, J. Comput. Phys. **148** (1999), no. 1, 2–22. MR MR1665209 (99j:65189)

[Cho01] David L. Chopp, *Some improvements of the fast marching method*, SIAM J. Sci. Comput. **23** (2001), no. 1, 230–244.

[ELF05] Douglas Enright, Frank Losasso, and Ronald Fedkiw, *A fast and accurate semi-lagrangian particle level set method*, Computers and Structures **83** (2005), 479–490.

[GS98] Sigal Gottlieb and Chi-Wang Shu, *Total variation diminishing Runge-Kutta schemes*, Math. Comp. **67** (1998), no. 221, 73–85. MR MR1443118 (98c:65122)

[HLOZ97] T. Hou, Z. Li, S. Osher, and H. Zhao, *A hybrid method for moving interface problems with application to the Hele-Shaw flow*, Journal of Computational Physics **134** (1997), 236–252.

[JP00] Guang-Shan Jiang and Danping Peng, *Weighted ENO schemes for Hamilton-Jacobi equations*, SIAM J. Sci. Comput. **21** (2000), no. 6, 2126–2143 (electronic). MR MR1762034 (2001e:65124)

[LFO95] Frank Losasso, Ronald Fedkiw, and Stanley Osher, *Spatially adaptive techniques for level set methods and incompressible flow*, Computers and Fluids (1995), In Press.

[LZG99] Z. Li, H. Zhao, and H. Gao, *A numerical study of electro-migration voiding by evolving level set functions on a fixed cartesian grid*, Journal of Computational Physics **152** (1999), 281–304.

[OF02] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces*, Springer, 2002.

[OS88] Stanley Osher and James A. Sethian, *Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys. **79** (1988), no. 1, 12–49. MR MR965860 (89h:80012)

[PMO$^+$99] Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang, *A PDE-based fast local level set method*, J. Comput. Phys. **155** (1999), no. 2, 410–438. MR MR1723321 (2000j:65104)

[Set99a] J. A. Sethian, *Fast marching methods*, SIAM Review **41** (1999), 199–235.

[Set99b] ———, *Level set methods and fast marching methods*, second ed., Cambridge Monographs on Applied and Computational Mathematics, vol. 3, Cambridge University Press, Cambridge, 1999, Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. MR MR1700751 (2000c:65015)

[SSO94] Mark Sussman, Peter Smereka, and Stanley Osher, *A level set approach for computing solutions to incompressible two-phase flow*, J. Comput. Phys. **114** (1994), 146–159.

[SV00] J. A. Sethian and A. Vladimirsky, *Fast methods for the eikonal and related Hamilton-Jacobi equations on unstructured meshes*, Proc. Natl. Acad. Sci. USA **97** (2000), no. 11, 5699–5703 (electronic). MR MR1761547 (2001b:65100)

[SV03] James A. Sethian and Alexander Vladimirsky, *Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms*, SIAM J. Numer. Anal. **41** (2003), no. 1, 325–363 (electronic). MR MR1974505 (2004f:65161)

[TCOZ03] Y. R. Tsai, L-T. Cheng, S. Osher, and H. Zhao, *Fast sweeping algorithms for a class of Hamilton-Jacobi equations*, SIAM Journal of Numerical Analysis **41** (2003), 673–694.

[Tsi95] John N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control **40** (1995), no. 9, 1528–1538. MR MR1347833 (96d:49039)

A Q_2 Coefficients

Given a Cartesian grid discretization, a 9 point stencil around the point x_{ij} can be used to approximate $\varphi(x)$ as a unique Q_2 function in the vicinity of x_{ij}. This is done by solving the linear system

$$a_0 + a_1 x_k + a_2 y_l + a_3 x_k y_l + a_4 x_k^2 + a_5 y_l^2 + a_6 x_k^2 y_l + a_7 x_k y_l^2 + a_8 x_k^2 y_l^2 = \varphi(x_k, y_l) \quad (16)$$

where $k = i-1 \dots i+1$ and $j = j-1 \dots j+1$. This is a set of 9 linear equations with 9 unknowns. If the local discretization is translated and scaled such that $x_{ij} = (0,0)$, and the grid size is, $h = 1$, then one finds

$$\begin{aligned}
a_0 &= \varphi_{ij} \\
a_1 &= (\varphi_{(i+1)j} - \varphi_{(i-1)j})/2 \\
a_2 &= (\varphi_{i(j+1)} - \varphi_{i(j-1)})/2 \\
a_3 &= ((\varphi_{(i+1)(j+1)} - \varphi_{(i-1)(j+1)}) - (\varphi_{(i+1)(j-1)} - \varphi_{(i-1)(j-1)}))/4 \\
a_4 &= (\varphi_{(i+1)j} - 2\varphi_{ij} + \varphi_{(i-1)j})/2 \\
a_5 &= (\varphi_{i(j+1)} - 2\varphi_{ij} + \varphi_{i(j-1)})/2 \\
a_6 &= ((\varphi_{(i+1)(j+1)} - \varphi_{(i+1)(j-1)}) - 2(\varphi_{i(j+1)} - \varphi_{i(j-1)}) \\
&\quad + (\varphi_{(i-1)(j+1)} - \varphi_{(i-1)(j-1)}))/4 \\
a_7 &= ((\varphi_{(i+1)(j+1)} - \varphi_{(i-1)(j+1)}) - 2(\varphi_{(i+1)j} - \varphi_{(i-1)j}) \\
&\quad + (\varphi_{(i+1)(j-1)} - \varphi_{(i-1)(j-1)}))/4 \\
a_8 &= ((\varphi_{(i+1)(j+1)} - 2\varphi_{(i+1)j} + \varphi_{(i+1)(j-1)}) - 2(\varphi_{i(j+1)} - 2\varphi_{ij} + \varphi_{i(j-1)}) \\
&\quad + (\varphi_{(i-1)(j+1)} - 2\varphi_{(i-1)j} + \varphi_{(i-1)(j-1)}))/4.
\end{aligned} \quad (17)$$