BACK AND FORTH ERROR COMPENSATION AND CORRECTION METHODS FOR REMOVING ERRORS INDUCED BY UNEVEN GRADIENTS OF THE LEVEL SET FUNCTION

TODD F. DUPONT* AND YINGJIE LIU†

Abstract. We propose a method that significantly improves the accuracy of the level set method and could be of value for numerical solutions of differential equations in general. Level set methods use a level set function, usually an approximate signed distance function, Φ, to represent the interface as the zero set of Φ. When Φ is advanced to the next time level by an advection equation, its new zero level set will represent the new interface position. But the non zero curvature of the interface will result in uneven gradients of the level set function which induces extra numerical error. Instead of attempting to reduce this error directly, we update the level set function Φ forward in time and then backward to get another copy of the level set function, say Φ_1. Φ_1 and Φ should have been equal if there were no numerical error. Therefore $\Phi - \Phi_1$ provides us the information of error induced by uneven gradients and this information can be used to compensate Φ before updating Φ forward again in time.

Key words. flux corrected transport, front tracking, level set method.

AMS subject classifications. 65M60, 65M12

1. Introduction.

Interface computation is a challenging and rewarding area for scientific research in recent years. There are many different approaches on how to track or capture the interface and how to do it more accurately, more robustly during topological changes of the interface.

Level set method was proposed by Osher and Sethian [11] to compute the interface motion indirectly by use of the zero set of the level set function. Recent developments on improving the level set method can be found for example on [9, 16, 1, 18, 4, 5, 14, 3, 10].

Using level set method for interface tracking enjoys great simplicity by solving the geometric problem with a PDE based method. Following the pioneering work of [11], given a velocity field $\mathbf{v}$, the level set function Φ satisfies

$$\frac{\partial \Phi}{\partial t} + \mathbf{v} \cdot \bigtriangledown \Phi = 0. \tag{1.1}$$

Since the velocity field could create large variation in Φ, there is usually an auxiliary equation to solve until the steady state at each time step (Sussman *et. al.* [16]),

$$\frac{\partial \Phi}{\partial \tau} + \mathrm{sgn}(\Phi)(| \bigtriangledown \Phi| - 1) = 0. \tag{1.2}$$

This procedure is supposed to transform the Φ into a signed distance function without changing its zero level set.

In Zhao *et. al.* [18], Gomes and Faugeras [5], a modified advection equation is introduced in order to keep Φ a signed distance function while still advancing the zero level set along the velocity field.

*Department of Computer Science, University of Chicago, Chicago, Illinois 60637 (`dupont@cs.uchicago.edu`). The work of this author was supported by the ASCI Flash Center at the University of Chicago under DOE contract B532820, and by the MRSEC Program of the National Science Foundation under award DMR-0213745.

†School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332 (`yingjie@math.gatech.edu`).

Even if the level set function is a perfect signed distance function, the curvature of the interface will still create variations in the gradient of the level set function, which will cause uneven dissipative error. Also since the zero level set of Φ is not located at the grid nodes, the discretization of equation (1.2) adds even more error to the interface position. In Enright *et. al.* [3], the excessive dissipation of discretizing (1.1) and (1.2) is addressed by using particles which move along characteristics of (1.1) to help regulate the interface. In Sussman and Fatemi [14], a volume conserving discretization method is introduced for more accurately approximating the solution of (1.2). Besides these techniques, very high order (3rd-5th order) non-oscillatory spatial and the corresponding high order Runge-Kutta temporal differencing are used for discretization of (1.1) and (1.2).

Here we propose another simple, systematic way of solving this problem. Though we use level set function and the level set advection equation (1.1) to describe the method we propose, it may find applications in other areas of numerical solution of ordinary or partial differential equations.

2. Backward Error Compensation and Forward Error Correction. Suppose we have a fixed computational mesh (with maximum cell size h) on a fixed domain $D \subset R^d$. Let Φ^n be the level set function defined at time level t_n, and suppose that $\mathbf{v}$ is a velocity field given in time intervals (t_n, t_{n+1}). Let $\mathcal{L}$ denote the mapping from an initial value of equation (1.1) at time t_n to the solution at time t_{n+1} with some given boundary values on the influx boundary. Let $\phi^{n+1} = \mathcal{L}(\Phi^n)$. We also solve the equation (1.1) numerically using a r-th order non-oscillatory scheme (e.g. upwind scheme, Lax-Friedrichs scheme, ENO [6], WENO [7], *etc*) to obtain a discretized level set function $\tilde{\Phi}^{n+1}$ at time level t_{n+1}, say $\tilde{\Phi}^{n+1} = \mathcal{L}_h \Phi^n$ where $\mathcal{L}_h$ represents the numerical operator. Let $\mathcal{L}_h^{-1}$ denote the numerical operator given by solving the equation (1.1) backward in time from t_{n+1} to t_n using the same scheme. Let $\Phi_1^n = \mathcal{L}_h^{-1}(\tilde{\Phi}^{n+1})$. If there were no numerical error, Φ^n and Φ_1^n would be equal. Therefore $e(x) = \Phi^n - \Phi_1^n$ provides information we may be able to take advantage of. Since the same scheme is used in both the forward and backward process we expect that the error in each will be about the same. This motivates us to add $\frac{1}{2}e(x)$ to Φ^n at x in order to remove the principal components of the error at the advanced time level t_{n+1}. Since $e(x)$ is bounded by the local truncation error of $\mathcal{L}_h$, we can see that the compensated scheme has at least the same order of accuracy as $\mathcal{L}_h$.

Therefore given Φ^n at time level t_n, we propose the following **backward error compensation algorithm.**

Step 1. Solve forward for $\tilde{\Phi}^{n+1}$ using $\tilde{\Phi}^{n+1} = \mathcal{L}_h \Phi^n$.
Step 2. Solve backward for Φ_1^n using $\Phi_1^n = \mathcal{L}_h^{-1}(\tilde{\Phi}^{n+1})$.
Step 3. Let $\Phi_2^n = \Phi^n + \frac{1}{2}(\Phi^n - \Phi_1^n)$.
Step 4. Solve forward for Φ^{n+1} using $\Phi^{n+1} = \mathcal{L}_h \Phi_2^n$.

Our strategy includes solving an equation backward in time which bears similarity to methods of time reversal for acoustics, see *e.g.* Prada *et. al.*[12]. It is interesting to see what happens when applying this method to the ordinary differential equation

$$\frac{dy}{dt} = f(t, y). \tag{2.1}$$

We use the forward Euler scheme to approximate its solution

$$\tilde{y}^{n+1} = y^n + \Delta t f^n, \tag{2.2}$$

where $y^n = y(t_n)$, $\Delta t = t_{n+1} - t_n$, $f^n = f(t_n, y^n)$. Solve the equation (2.1) backward in time with the same scheme

$$b^n = \tilde{y}^{n+1} - \Delta t f^{n+1}, \tag{2.3}$$

where $f^{n+1} = f(t_{n+1}, \tilde{y}^{n+1})$, and define

$$c^n = y^n + \frac{1}{2}(y^n - b^n) = y^n + \frac{1}{2}\Delta t\{f^{n+1} - f^n\}. \tag{2.4}$$

Finally solve the equation (2.1) forward again

$$\begin{aligned} y^{n+1} &= c^n + \Delta t f(t_n, c^n) \\ &= y^n + \tfrac{1}{2}\Delta t\{f^{n+1} + 2f(t_n, c^n) - f^n\}. \end{aligned} \tag{2.5}$$

Comparing (2.5) to the second order improved Euler scheme (predictor-corrector method)

$$y^{n+1} = y^n + \frac{1}{2}\Delta t\{f^{n+1} + f^n\}, \tag{2.6}$$

it is easy to see that

$$\{f^{n+1} + 2f(t_n, c^n) - f^n\} - \{f^{n+1} + f^n\} = O(\Delta t^2). \tag{2.7}$$

Therefore the forward Euler scheme with backward error compensation (2.5) has the same order of accuracy as the second order improved Euler scheme.

Can we use backward error compensation to improve the order of accuracy for second order schemes? Let's look at the linear ODE

$$\frac{dy}{dt} = ay, \tag{2.8}$$

where a is a constant. If we use the second order scheme

$$y^{n+1} = y^n + \frac{1}{2}a\Delta t(y^{n+1} + y^n), \tag{2.9}$$

then the Step 2 of the backward error compensation algorithm will return exactly y^n, thus no order improvement can be expected. In general, if an r-th order scheme for equation (2.8) can be formulated as

$$y^{n+1} = \{\sum_{k=0}^{r} \frac{(a\Delta t)^k}{k!} + \alpha\frac{(a\Delta t)^{r+1}}{(r+1)!} + O(\Delta t^{r+2})\}y^n, \text{ as } \Delta t \to 0, \tag{2.10}$$

where α is a constant, we have the following theorem.

THEOREM 1. *The backward error compensation algorithm improves the order of accuracy of Scheme (2.10) to $r+1$ if r is an odd positive integer.*

Proof. Scheme (2.10) can be written as

$$\tilde{y}^{n+1} = \{e^{a\Delta t} + \frac{\alpha - 1}{(r+1)!}(a\Delta t)^{r+1} + O(\Delta t^{r+2})\}y^n.$$

Step 2 of the backward error compensation algorithm gives

$$\begin{aligned} b^n &= \{e^{-a\Delta t} + \tfrac{\alpha-1}{(r+1)!}(-a\Delta t)^{r+1} + O(\Delta t^{r+2})\}\tilde{y}^{n+1} \\ &= \{1 + ((-1)^{r+1}e^{a\Delta t} + e^{-a\Delta t})(\alpha - 1)\tfrac{(a\Delta t)^{r+1}}{(r+1)!} + O(\Delta t^{r+2})\}y^n \\ &= \{1 + 2(\alpha - 1)\tfrac{(a\Delta t)^{r+1}}{(r+1)!} + O(\Delta t^{r+2})\}y^n. \end{aligned}$$

Step 3 of the backward error compensation algorithm gives

$$\begin{array}{rcl} c^n & = & y^n + \frac{1}{2}(y^n - b^n) \\ & = & \{1 - (\alpha - 1)\frac{(a\Delta t)^{r+1}}{(r+1)!} + O(\Delta t^{r+2})\}y^n. \end{array}$$

Step 4 of the backward error compensation algorithm gives

$$\begin{array}{rcl} y^{n+1} & = & \{e^{a\Delta t} + \frac{\alpha-1}{(r+1)!}(a\Delta t)^{r+1} + O(\Delta t^{r+2})\}c^n \\ & = & \{e^{a\Delta t} + O(\Delta t^{r+2})\}y^n \end{array}$$

□

This Theorem indicates that the backward error compensation algorithm can only be expected to remove the dissipative error.

Next let us see what happens if we apply this method to a one-dimensional translation equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0 \tag{2.11}$$

on $[0, 1]$ with periodic boundary condition, and a first order upwind numerical scheme. Let $0 = x_0 < x_1 < \cdots < x_N = 1$ be a uniform partition and denote $u_i^n = u(x_i, t_n)$. Let t_n, t_{n+1} be the two adjacent time levels in time discretization and $\Delta t = t_{n+1} - t_n < \Delta x$, where $\Delta x = 1/N$. The 1st order upwind scheme can be written as

$$\tilde{u}_i^{n+1} = (1 - \lambda)u_i^n + \lambda u_{i-1}^n, \tag{2.12}$$

where $\lambda = \Delta t/\Delta x < 1$ is fixed. When solving the equation (2.11) backward in time with the same scheme, we have

$$\begin{array}{rcl} b_i^n & = & (1 - \lambda)\tilde{u}_i^{n+1} + \lambda\tilde{u}_{i+1}^{n+1} \\ & = & ((1 - \lambda)^2 + \lambda^2)u_i^n + \lambda(1 - \lambda)(u_{i-1}^n + u_{i+1}^n). \end{array} \tag{2.13}$$

In this case it can also be viewed as the down-wind step of the MacCormack scheme [8]. Following the above Step 3, we can define the compensated numerical solution at t_n as

$$\begin{array}{rcl} c_i^n & = & u_i^n + \frac{1}{2}(u_i^n - b_i^n) \\ & = & u_i^n - \frac{1}{2}\lambda(1 - \lambda)(u_{i+1}^n - 2u_i^n + u_{i-1}^n). \end{array} \tag{2.14}$$

If we write $c^n = \mathcal{S}u^n$, then $\mathcal{S}$ may be viewed as a sharpening or anti-diffusive operator. (So this method has some flavor of the flux corrected transport method (Boris and Book [2]) which improves a diffusive scheme by acting on the result with a, carefully controlled, anti-diffusive operator.) Finally applying the upwind scheme forward again we have

$$\begin{array}{rcl} u_i^{n+1} & = & (1 - \lambda)c_i^n + \lambda c_{i-1}^n \\ & = & (-\frac{1}{2}\lambda^2 + \frac{1}{2}\lambda^3)u_{i-2}^n + (\frac{1}{2}\lambda + 2\lambda^2 - \frac{3}{2}\lambda^3)u_{i-1}^n + \\ & & (1 - \frac{5}{2}\lambda^2 + \frac{3}{2}\lambda^3)u_i^n + (-\frac{1}{2}\lambda + \lambda^2 - \frac{1}{2}\lambda^3)u_{i+1}^n. \end{array} \tag{2.15}$$

The local truncation error for the scheme (2.15) is $O(\Delta x^3)$. Therefore it not only improves the temporal order of accuracy by one (as predicted by the previous example), but also improves the spatial order of the scheme (2.12) by one. This is particularly interesting because if we apply the predictor-corrector method (2.6) instead we can only improve the temporal order. While the method has a stencil that is skewed in the upwind direction, it seems that as λ gets very small it is quite close to explicit centered differencing; however, the stability result below indicates that it has different properties. We conducted numerical tests to see the effect of this correction after a large number of steps. We set up the initial data first as a pyramid: $u = 2x$ for $x \in (0, 0.5)$ and $u = 2 - 2x$ for $x \in (0.5, 1)$. With $N = 100$, $\Delta x = 0.01$, $\Delta t = 0.005$, we plot the numerical solutions (with and without backward error compensation) against the exact one at time $t = 10$, *i.e.* after 2000 steps. See Fig. 2.1. Then we set up the initial data as a square wave: $u = 0$ for $x \in (0, 0.3) \cup (0.7, 1)$ and $u = 1$ for $x \in (0.3, 0.7)$. With $N = 100$, $\Delta x = 0.01$, $\Delta t = 0.005$, and plot the numerical solutions (with and without backward error compensation) against the exact one at time $t = 10$. See Fig. 2.2. Without compensation the square wave is essentially averaged to half of its original height by the upwind scheme.

We also have the following simple stability result for this method.

THEOREM 2. *Suppose that $0 < \lambda \leq 1$ and that u^{n+1} is defined by (2.15) with periodic boundary conditions. Then*

$$\|u^{n+1}\|_{l_2} \leq \|u^n\|_{l_2}.$$

Proof. With $\varphi_k(x) = exp(2\pi ikx)$, write

$$u_j^n = \sum_{k=0}^{N-1} c_k \varphi_k(j\Delta x).$$

Then it is straight-forward to see that

$$u_j^{n+1} = \sum_{k=0}^{N-1} m_k c_k \varphi_k(j\Delta x),$$

where, with $s = \lambda exp(-2\pi ik\Delta x)$,

$$m_k = (1-s)[1 + \frac{1}{2}(1 - (1-\overline{s})(1-s))] = (1-s)[1 + \frac{1}{2}(1 - |1-s|^2)].$$

Since $y = |1 - s| \in [0, 2]$, we get that $|m_k| = y|1 + \frac{1}{2}(1 - y^2)|$ is at most one. ◻

Next we study the Burgers' equation

$$\frac{\partial u}{\partial t} + \frac{\partial(\frac{1}{2}u^2)}{\partial x} = 0. \tag{2.16}$$

We compute the numerical solution using the Lax-Friedrichs scheme with backward error compensation. We first conduct a convergence test with initial value [13]

$$u(x, 0) = \frac{1}{2} + sin(2\pi x), \quad x \in (0, 2), \tag{2.17}$$

with periodic boundary conditions. We choose CFL factor 0.5 and mesh cell numbers $N = 50, 100, 200, 400$ and compare the numerical solutions at final time $T = 0.1$ with a very fine solution $N = 40000$ computed with Lax-Friedrichs scheme. The L^∞ errors and order of accuracy are listed in Table 2.1.

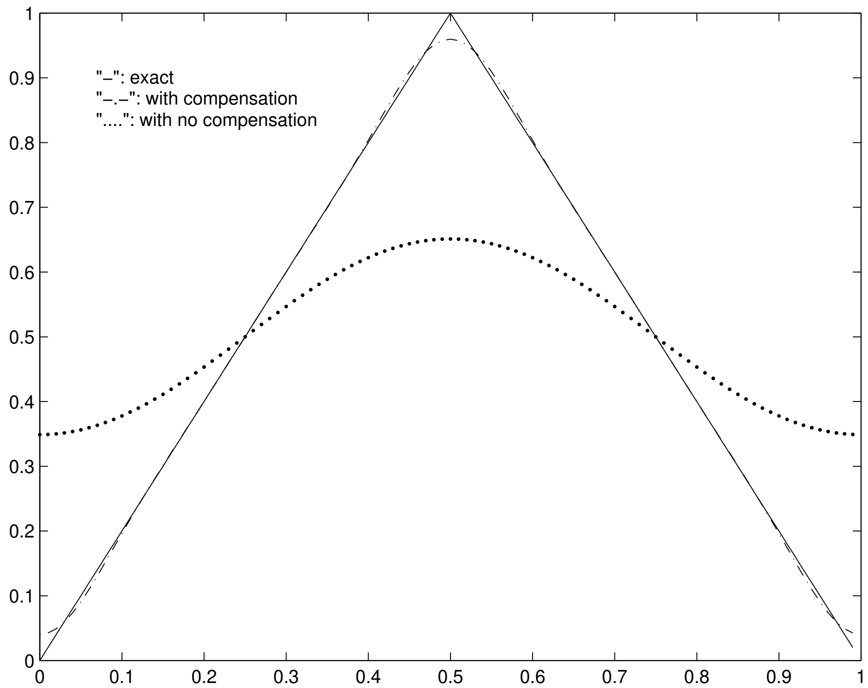

FIG. 2.1. *Pyramid wave after* 2000 *time steps of computation (*$T = 10$*). The linear equation is computed using a 1st order upwind scheme with and without backward error compensation,* $N = 100$.

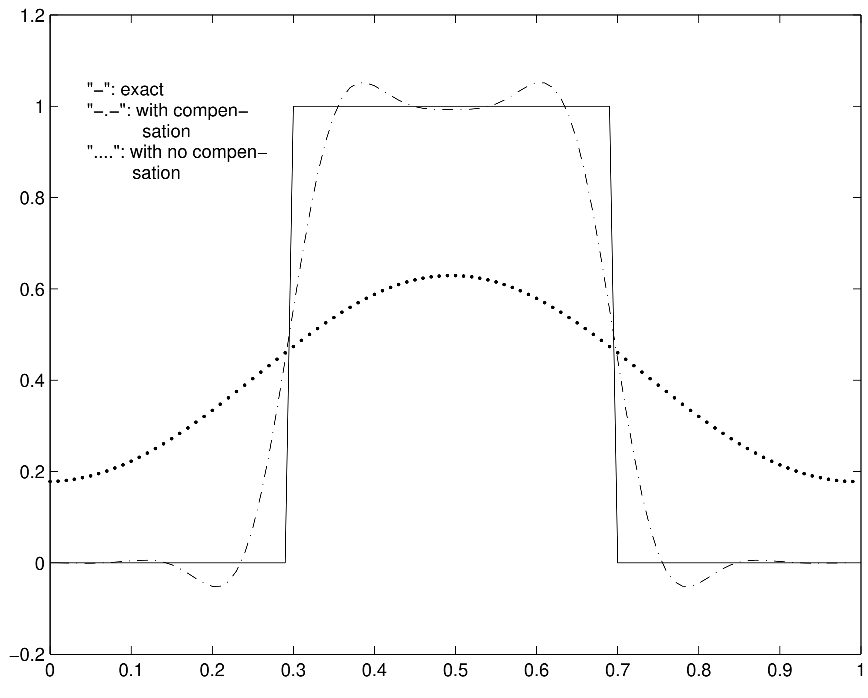

FIG. 2.2. *Square wave after* 2000 *time steps of computation (*$T = 10$*). The linear equation is computed using a 1st order upwind scheme with and without backward error compensation,* $N = 100$.

N	L_∞ error	order
50	0.112	-
100	0.0374	1.58
200	0.00673	2.47
400	0.00154	2.13

TABLE 2.1
L_∞ error for Lax-Friedrichs scheme with backward error compensation for Burgers' equation.

Next we increase the number of modes in the initial value to

$$u(x,0) = \frac{1}{2} + sin(5\pi x), \quad x \in (0,2), \tag{2.18}$$

and with $N = 100$, compute the numerical solutions to $T = 0.1$ using Lax-Friedrichs scheme with and without backward error compensation. The results are shown in Fig. 2.3.

Finally we use a single mode in the initial value

$$u(x,0) = \frac{1}{2} + sin(\pi x), \quad x \in (0,2), \tag{2.19}$$

and with $N = 100$, compute the numerical solutions to $T = 0.5$ using Lax-Friedrichs scheme with backward error compensation. See Fig. 2.4. Notice that there is some overshooting but no oscillations near the shock, the numerical solution gives the correct shock speed and the overshooting stays small. In fact, Lax-Friedrichs scheme with backward error compensation still has global conservation property because the backward compensation step described in the previous Step 3 doesn't change the total conservative quantity (subject to the error at the boundary).

The dual of the above backward error compensation method can be formulated as follows. Let us called it the **forward error correction method.** For given Φ^n at time level t_n,
Step 1. Solve forward for $\tilde{\Phi}^{n+1}$ using $\tilde{\Phi}^{n+1} = \mathcal{L}_h \Phi^n$.
Step 2. Solve backward for Φ_1^n using $\Phi_1^n = \mathcal{L}_h^{-1}(\tilde{\Phi}^{n+1})$.
Step 3. Solve forward for $\tilde{\Phi}_1^{n+1}$ using $\tilde{\Phi}_1^{n+1} = \mathcal{L}_h \Phi_1^n$.
Step 4. Let $\Phi^{n+1} = \tilde{\Phi}^{n+1} + \frac{1}{2}(\tilde{\Phi}^{n+1} - \tilde{\Phi}_1^{n+1})$.
For the one-dimensional translation equation this method is the same as the backward compensation method. We give an example calculation with this method in the next section.

3. Tests On Zalesak's Problem. The two space dimensional Zalesak's Problem [17] can be described as follows: Set a rotational velocity field $(u, v) = (\frac{\pi}{314}(50 - y), \frac{\pi}{314}(x - 50))$ in a domain $(0, 100) \times (0, 100)$. Initially there is a cutout circle centered at $(50, 75)$ with radius 15. The slot being cut out has width 5 and length 25. Every point of this cutout circle is supposed to move along the local velocity field. We set the initial level set function Φ to be a signed distance function which is negative inside the notched disk and positive outside. The computation is up to the final time $t = 628$, allowing a full revolution of the disk. The challenge for computation with level set method is that this disk has corner points, curves, straight lines and a very narrow slot (when the mesh size is 1 or 0.5, the slot width is 5 or 10 mesh cell sizes respectively). In the first test we compute this problem with $N = 100$, $\Delta x = 1$, $\Delta t = 0.707$ (CFL factor ≤ 0.5). The level set advection equation (1.1)

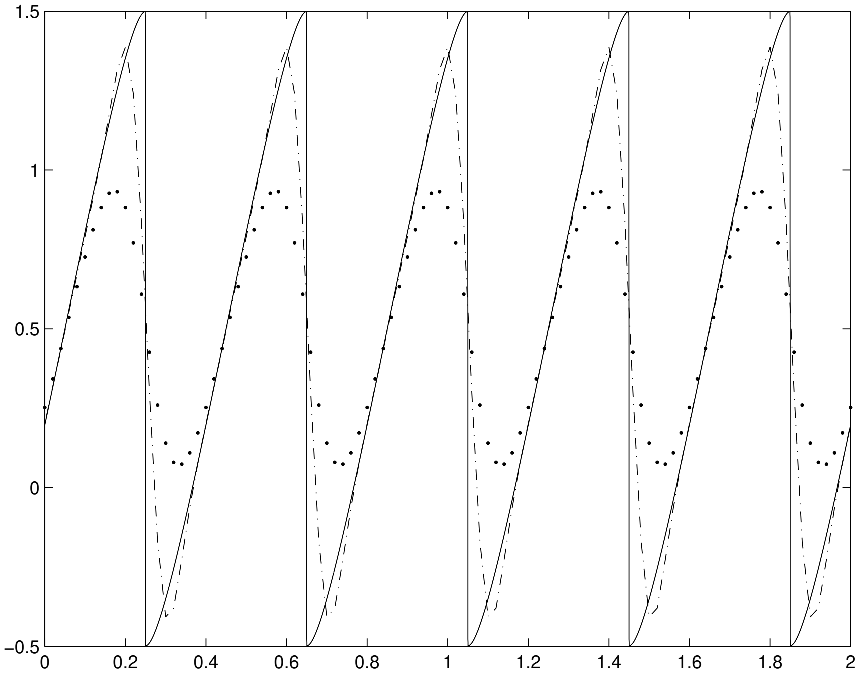

Fig. 2.3. *Comparative numerical solutions for Lax-Friedrichs scheme with and without backward error compensation, $N = 100$, $T = 0.1$. "-": fine solution; "-.-": L-F scheme with compensation; "...": L-F scheme with no compensation.*

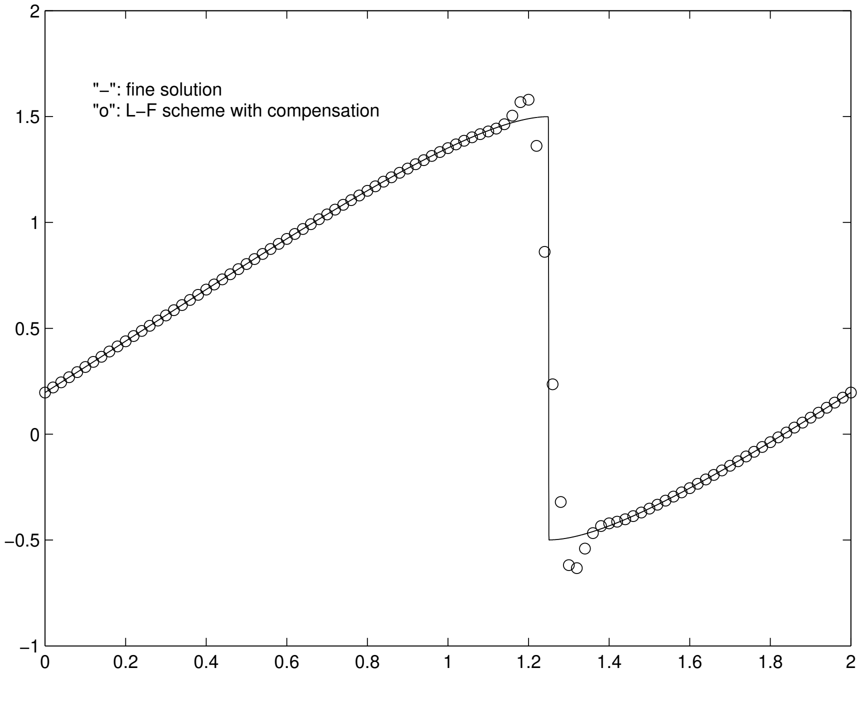

Fig. 2.4. *Numerical solution for Lax-Friedrichs scheme with backward error compensation, $N = 100$, $T = 0.5$.*

is discretized by a first order upwind scheme and the backward error compensation method described in the previous section is applied. There is no re-distancing of the level set function at each time level. In Fig. 3.1 the computed disk is drawn against the exact one after one revolution. Though the error is large due to the large cell size and the 1st order scheme we start from, the essential characteristics like the sizes of the disk and the slot survive, which is particularly surprising because if this test were done without backward error compensation, the whole disk (not just the slot) would disappear before one revolution.

In the second test we basically repeat the first test but with the cell size and time step decreased by half, *i.e.*, $N = 200$, $\Delta x = 0.5$. The final solution is shown in Fig. 3.2. This test was also done without backward error compensation or forward error correction (but with re-distancing (1.2)), and the results are shown in Fig. 3.3. The disk has shrunk very much from its original size and the slot has essentially disappeared.

For comparison a further test without backward error compensation or forward error correction is shown in Fig. 3.4 with $N = 600$, $\Delta x = 1/6$. The level set advection equation (1.1) is still discretized by the first order upwind scheme. In this case we have re-distanced the level set function at each time level using equation (1.2) discretized by the same first order upwind scheme, which improves the numerical results.

We also include a version of the second test using forward error correction described in the previous section, see Fig. 4.3. It is similar to the result in Fig. 3.2 using backward error compensation. The comparison of these different ways of diminishing the error will be the subject of future work.

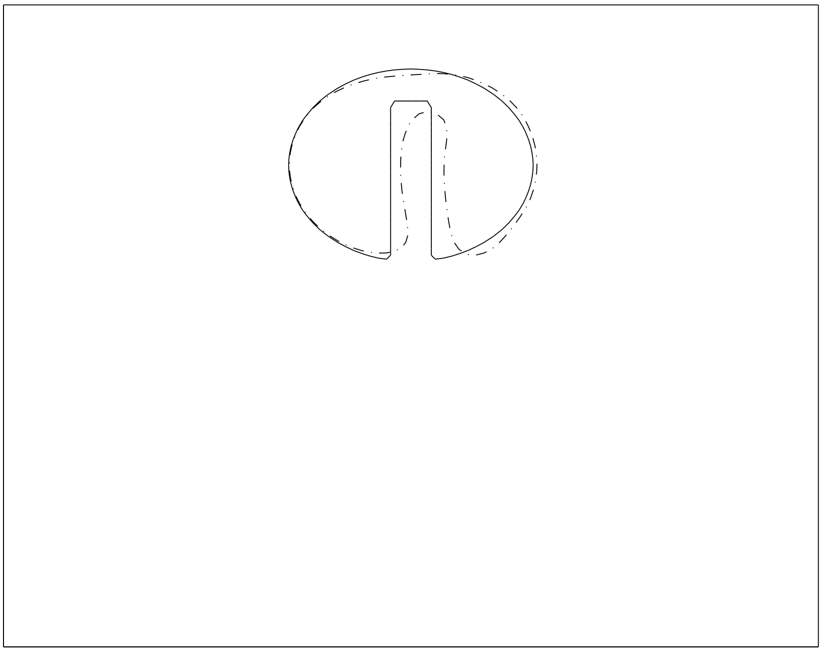

Fig. 3.1. *Zalesak's problem. Comparison of a notched disk that has been rotated one revolution. Level set equation is computed using 1st order upwind scheme with backward error compensation,* $N = 100$, $\Delta x = 1$.

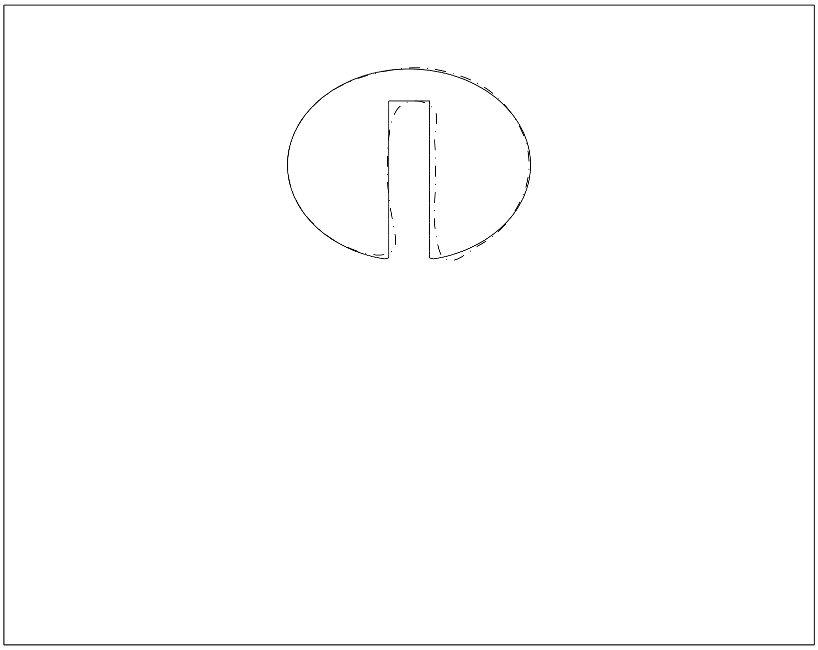

Fig. 3.2. *Zalesak's problem. Comparison of a notched disk that has been rotated one revolution. Level set equation is computed using 1st order upwind scheme with backward error compensation,* $N = 200$, $\Delta x = 0.5$.

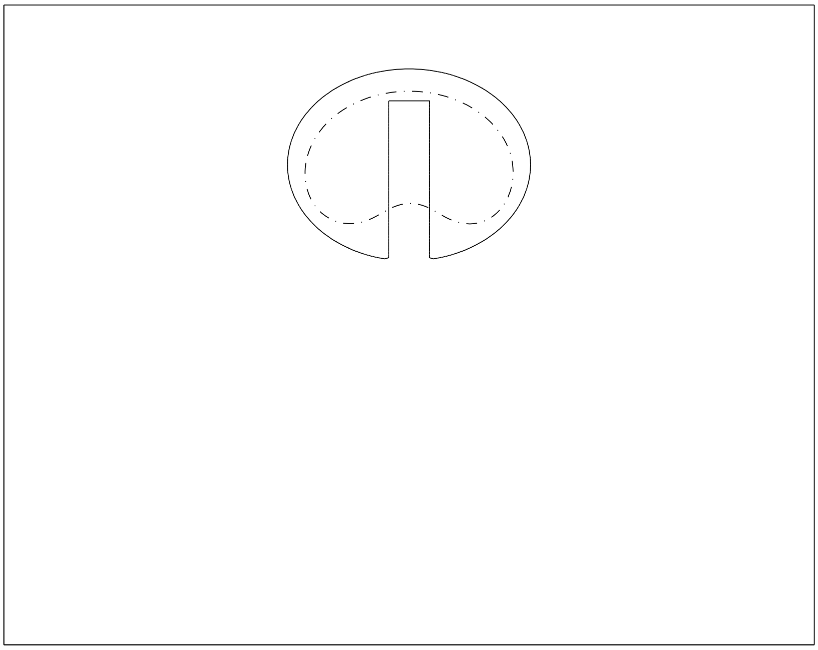

FIG. 3.3. *Zalesak's problem. Comparison of a notched disk that has been rotated one revolution. Level set equation is computed using 1st order upwind scheme without backward error compensation,* $N = 200$, $\Delta x = 0.5$.

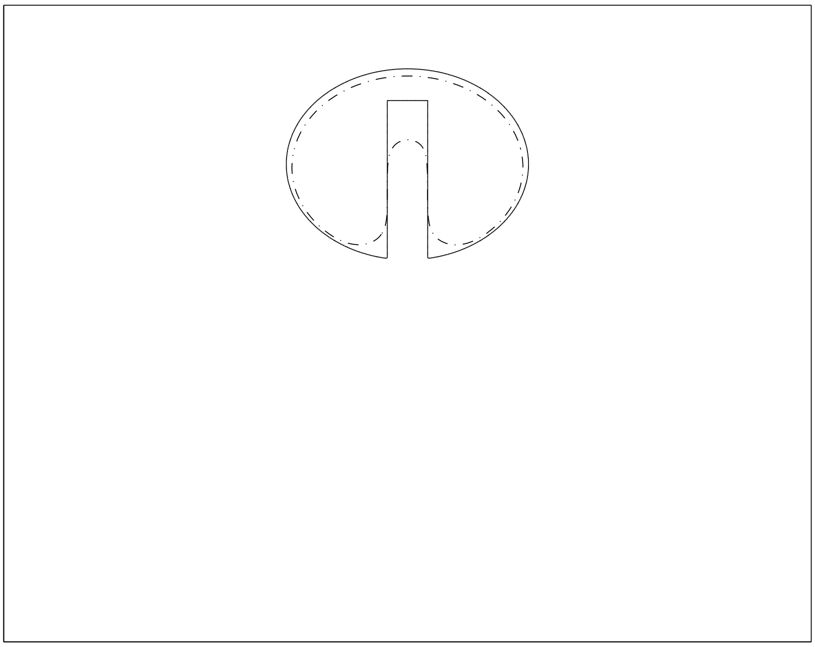

FIG. 3.4. *Zalesak's problem. Comparison of a notched disk that has been rotated one revolution. Level set equation is computed using 1st order upwind scheme without backward error compensation,* $N = 600$, $\Delta x = 1/6$.

4. Simple Redistancing of the Level Set Function. Following [16], we can redistance the level set function at each time step by solving (1.2) so that the level set function stays close to being a signed distance function. This step also helps clean the error pollution coming from the "skeleton", *i.e.*, the non-smooth area of the level set function. The novelty here is that at each time step we update the level set function Φ using (1.2) only at places where $|\Phi| > \Delta x$. This allows us to use a simple low cost first order scheme to discretize equation (1.2) without generating large diffusion. As in [16], equation (1.2) can be written as

$$\Phi_\tau + W \cdot \bigtriangledown \Phi = S(\Phi_0), \tag{4.1}$$

where $W = S(\Phi_0) \bigtriangledown \Phi / | \bigtriangledown \Phi|$ and $S(\Phi_0) = \Phi_0 / \sqrt{\Phi_0^2 + \Delta x^2}$ is the approximated sign function of Φ_0 which is the current level set function obtained by solving equation (1.1) and also serves as the initial value for 4.1. W can be discretized by the centered finite difference and equation (4.1) then can be solved by the simple first order upwind scheme. After solving equation (1.1) at each time step, the equation (4.1) is solved for a few time steps to update Φ wherever $|\Phi| > \Delta x$. We recompute the test examples in Fig. 3.1, 3.2 by using two iterations of the above redistancing procedure with $\Delta\tau = 0.25\Delta x$ after each time step of advancing equation (1.1). The results can be found in Fig. 4.2 and 4.2 and the average distances (defined as in [14]) between the exact and computed interfaces are shown in Table 4.1. The relative error of the computed disk area A is plotted against time for three meshes: 100×100, 200×200 and 400×400. See Fig. 4.4.

In evaluating these results it is clear that even the very simple redistancing that is done here improves the quality of the solution. It is not easy to make precise comparisons between different schemes even if they are used to solve the same problem, because the difference in the work required per point per step. We did look at the very nice results on this problem in [14], in which they use a clever scheme to compensate for volume change. Our, admittedly crude, estimates of the work involved in their scheme indicate that the method of this section does as well, or close to it, in terms of mass loss and interface position error when normalized for work. The method in [14] clearly does better with a given number of unknowns, but does more work per unknown, even after taking into account the three solves involved here. We note that this apparent good performance of this simple procedure may be an artifact of the test problem, since even the higher order schemes of [14] asymptotically become first order on this problem because of the corners of the level set. We also looked at the results in [15] and tried to make a comparison. There is a factor of 16 difference in the sizes of the computational domains that must be taken into account. The method they use is a coupled level set and volume of fluid method that conserves mass exactly, a very nice feature. For the same size meshes the average distance errors in Table 4.1 are approximately three times those cited in [15] for ELVIRA. Their scheme is more complex than the one of this section, but it seems likely that they are somewhat more accurate for a given level of work.

Another problem is that the level set function could become too flat near the interface. To overcome this problem, we could simply update the level set function Φ using (1.2) only at places, say x_i, where $|\Phi(x_i)| > \Delta x$ or $\Phi(x_i)$ is of the same sign with Φ at the neighboring grid nodes of x_i. Applying it to the test problems here doesn't make much a difference. We should report in the future with more applications.

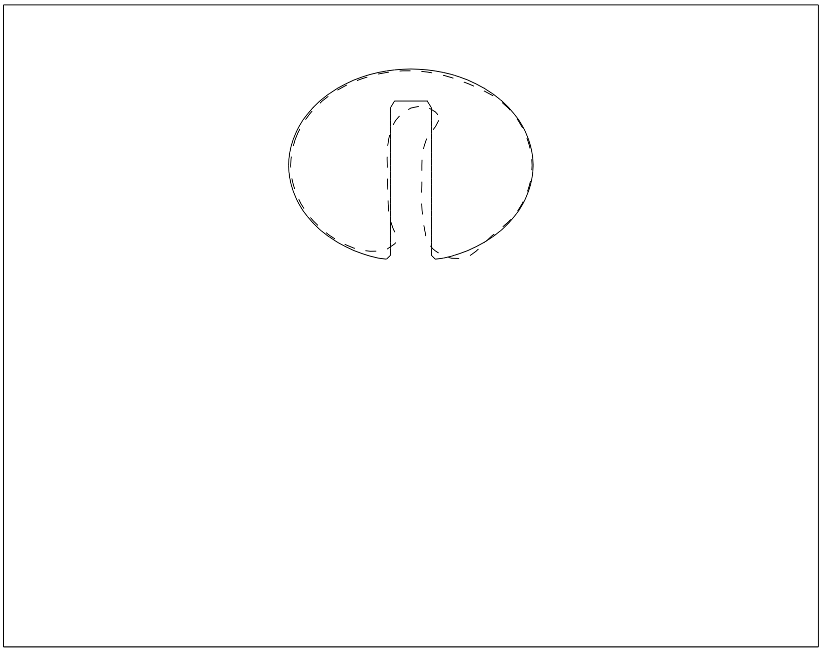

FIG. 4.1. *Zalesak's problem. Comparison of a notched disk that has been rotated one revolution. Level set equation is computed using 1st order upwind scheme with backward error compensation and redistancing,* $N = 100$, $\Delta x = 1$.

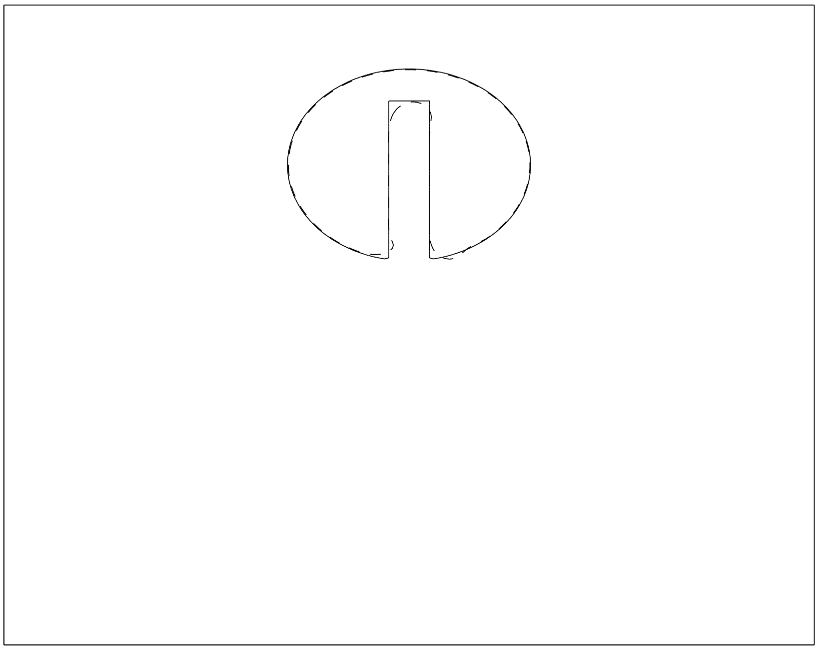

FIG. 4.2. *Zalesak's problem. Comparison of a notched disk that has been rotated one revolution. Level set equation is computed using 1st order upwind scheme with backward error compensation and redistancing,* $N = 200$, $\Delta x = 0.5$.

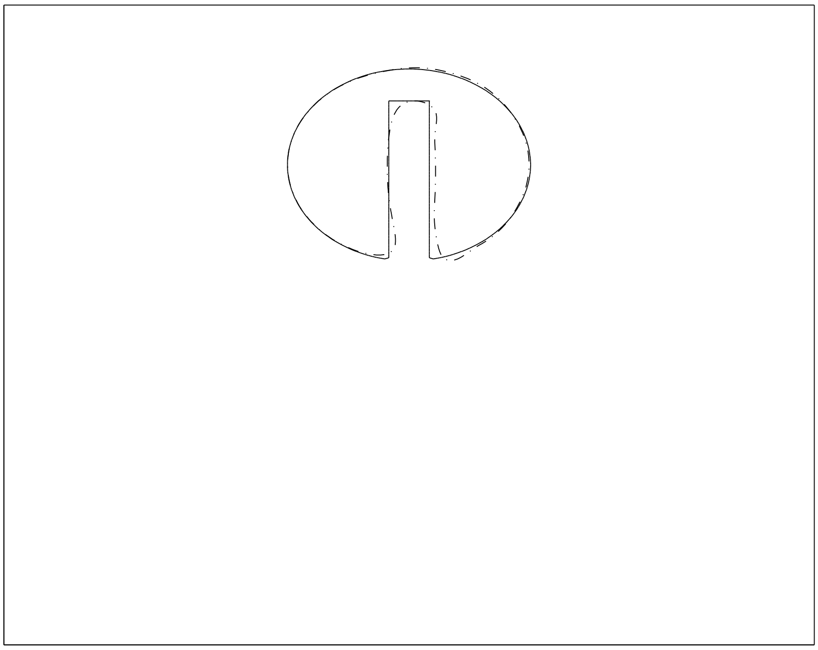

Fig. 4.3. *Zalesak's problem. Comparison of a notched disk that has been rotated one revolution. Level set equation is computed using 1st order upwind scheme with forward error correction,* $N = 200$, $\Delta x = 0.5$.

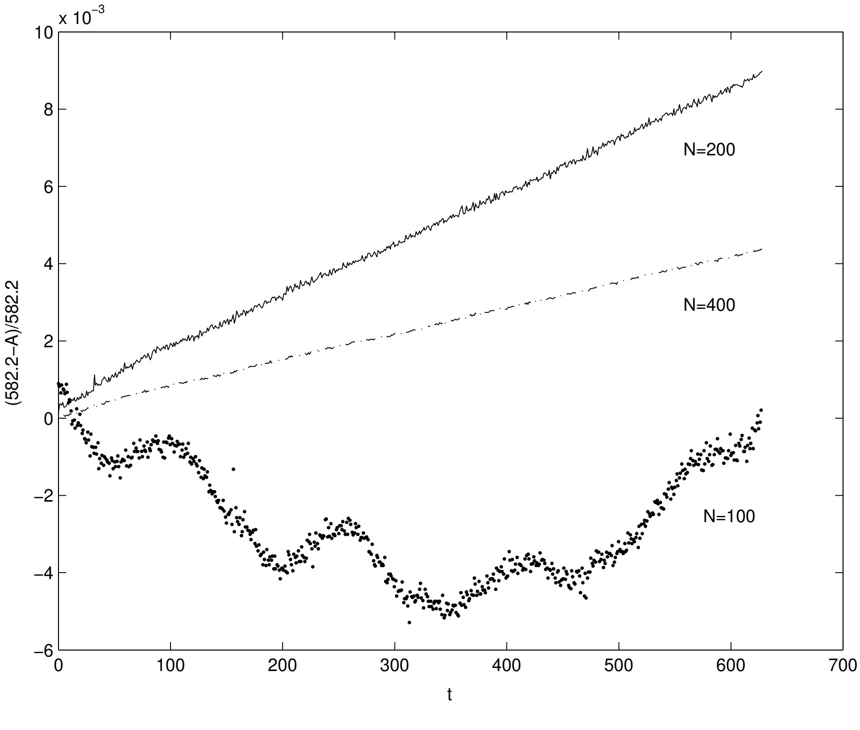

Fig. 4.4. *Zalesak's problem. Relative area loss of the notched disk as a function of time. Level set equation is computed using 1st order upwind scheme with backward error compensation and redistancing.*

N	average distance	order
100	0.412	-
200	0.130	1.66
400	0.0616	1.08

TABLE 4.1
Rotating slotted disk: average distance between the exact interface and the one computed using 1st order upwind scheme with backward error compensation and redistancing.

5. Acknowledgments. The authors thank Dr. Hongkai Zhao for the helpful discussion on the curvature induced error for level set method, and thank Dr. Luca Dieci for helping us compare this method to some numerical methods for ODE.

REFERENCES

[1] D. Adalsteinsson and J. A. Sethian, *A fast level set method for propagating interfaces*, J. Comput. Phys., 118 (1993), pp. 269–277.
[2] J. P. Boris and D. L. Book, *Flux-corrected transport i. shasta, a fluid transport algorithm that works*, J. Comput. Phys., 11 (1973), pp. 38–69.
[3] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, *A hybrid particle level set method for improved interface capturing*, J. Comput. Phys., 183 (2002), pp. 83–116.
[4] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, *A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method)*, J. Comput. Phys., 152 (1999), pp. 457–492.
[5] J. Gomes and O. Faugeras, *Advancing and keeping a signed distance function*, Tech. Report 3666, INRIA, 1999.
[6] A. Harten, S. Osher, B. Engquist, and S. Chakravarthy, *Uniformly high order accurate essentially non-oscillatory schemes, III*, J. Comput. Phys., 71 (1987), pp. 231–303.
[7] X. D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.
[8] R. W. MacCormack, AIAA Paper 69-354, (1969).
[9] B. Merriman, J. Bence, and S. Osher, *Motion of multiple functions: a level set approach*, J. Comput. Phys., 112 (1994), p. 334.
[10] D. Nguyen, F. Gibou, and R. Fedkiw, *A fully conservative ghost fluid method & stiff detonation waves*, 12th International Detonation Symposium, San Diego, CA, (2002).
[11] S. Osher and J. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi equations*, J. Comput. Phys, 79 (1988), pp. 12–49.
[12] C. Prada, F. Wu, and M. Fink, *The iterative time reversal mirror: A solution to self-focusing in the pulse echo mode*, J. Acoust. Soc. Amer., 90 (1991), pp. 1119–1129.
[13] J. Qiu and C.-W. Shu, *Finite difference weno schemes with lax-wendroff type time discretization*, SIAM J. Sci. Comput., (to appear).
[14] M. Sussman and E. Fatemi, *An efficient, interface preserving level set re-distancing algorithm and its application to interfacial incompressible fluid flow*, SIAM J. Sci. Comput., 20 (1999), pp. 1165–1191.
[15] M. Sussman and E. G. Puckett, *A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows.*, J. Comp. Phys, 162 (2000), pp. 301–337.
[16] M. Sussman, P. Smereka, and S. Osher, *A level set method for computing solutions to imcompressible two-phase flow*, J. Comput. Phys, 119 (1994), pp. 146–159.
[17] S. T. Zalesak, *Fully multidimensional flux-corrected transport*, J. Comput. Phys., 31 (1979), pp. 335–362.
[18] H.-K. Zhao, T. Chan, B. Merriman, and S. Osher, *A variational level set approach to multiphase motion*, J. Comput. Phys., 127 (1996), pp. 179–195.