Proceedings of ALGORITMY 2005
pp. 22–31

THE EIKONAL EQUATION: NUMERICAL EFFICIENCY VS. ALGORITHMIC COMPLEXITY ON QUADRILATERAL GRIDS

SHU-REN HYSING* AND STEFAN TUREK †

Abstract. This paper presents a study of the computational efficiency, that is accuracy versus computational effort, for solving the Eikonal equation on quadrilateral grids. The algorithms that are benchmarked against each other for computations of distance functions are the following: the fast marching method, the fast sweeping method, an algebraic Newton method, and also a "brute force" approach. Some comments are also made on the solution of the Eikonal equation via reformulation to a hyperbolic PDE. The results of the benchmark clearly indicate that the fast marching method is the preferred algorithm due to both accuracy and computational speed in our tested context.

Key words. Hamilton-Jacobi equations, Eikonal equations, distance functions

AMS subject classifications. 65M12, 65Y20, 70H20

1. Introduction.

The Eikonal equation, defined by

$$|\nabla\phi(\mathbf{x})| = f(\mathbf{x}),\ \mathbf{x} \in R^n \tag{1}$$

with boundary condition $\phi(\mathbf{x}) = g(\mathbf{x}),\ \mathbf{x} \in \Gamma \subset R^n$, is of general interest in fields such as computational geometry, multiphase flow, path planning etc. In the simplest case where the speed function $f(\mathbf{x})$ is a constant, usually equal to one, then the solution of the Eikonal equation will represent the shortest distance from a point $\mathbf{x}$ to the zero distance curve, usually given by Γ, that is if $g(\mathbf{x}) = 0$.

The simple looking Eikonal equation has generated quite a substantial amount of interest due to both its usefulness and also its inherent nonlinear character making it a challenging task to solve. Many solution schemes have been proposed, and the ones that this investigation is concerned with are the most commonly used ones: the brute force method, the fast sweeping method, the fast marching method, an algebraic Newton method, and finally via reformulation of (1) to a hyperbolic problem.

Up to this date few papers have been published that compares and investigates which solution scheme most efficiently solve the Eikonal equation on computational grids. To mention one study done on strictly Cartesian grids we refer to a paper by Gremaud and Kuster [2], which focuses on solving (1) with inhomogeneous speed functions. Intuitively one is lead to think that the algorithmic efficiency and benefits of $O(N)$ versus $O(N\ log\ N)$ methods should be higher, where N is the number of unknowns or grid points. As shall be seen this is far from the whole truth in a fully general context.

* Institute of Applied Mathematics (LS III), Univeristy of Dortmund, Vogelpothsweg 87, D-44227 Dortmund, Germany (shuren.hysing@math.uni-dortmund.de).
† Institute of Applied Mathematics (LS III), Univeristy of Dortmund, Vogelpothsweg 87, D-44227 Dortmund, Germany (stefan.turek@math.uni-dortmund.de).

This paper is structured as follows. The following section contains brief descriptions of various schemes used to solve equation (1). In section 3 we describe some algorithms needed to implement these solution schemes on quadrilateral grids. Thereafter follows a section describing the test case and the results for the various methods. Finally a summary with conclusions is presented.

2. Algorithms.

In this section we describe the most common algorithms used to solve the Eikonal equation (1) with a speed function $f(\mathbf{x})$ equal to unity. The solution methods are presented in increasing order of algorithmic complexity.

2.1. The fast sweeping method.

The fast sweeping method described in the papers by Tsai, Osher et al. [12, 13] essentially consists of applying upwind type difference formulas, see section 3.1 later, while using Gauss-Seidel type of iterations, to update the distance function field. The key to fast sweeping is to update the points in a certain order that tries to follow the characteristics of the solution, that is the sweeping direction should ideally correspond to the real propagation of information. The basic method is described as follows

- Select a sweeping direction and calculate the corresponding grid point order.
- Loop over all points in accordance with the above sequence and update the distance for each point if the newly calculated distance is smaller than the previous one.
- Repeat this procedure for all sweeping directions.
- Repeat until convergence for all grid points has been achieved.

It is easy to see that the complexity of this algorithm is of order $O(N)$, and has the potential to be very fast if the number of sweeps can be kept to a minimum. The sweeping orders may be chosen as follows for a two dimensional tensorproduct mesh

$$\begin{array}{ll} 1.\ i = 1:I, j = 1:J, & 2.\ i = I:1, j = 1:J \\ 3.\ i = I:1, j = J:1, & 4.\ i = 1:I, j = J:1 \end{array}$$

where i is the node position in the x-direction and j is the corresponding position in the y-direction. I and J are the maximum number of nodes in the x- and y-directions respectively.

2.2. Algebraic Newton method.

This method utilizes an existing approximate distance field ψ by solving for each grid point $\mathbf{x}_0$ the following equations for the unknown point $\mathbf{x}$

$$L(\mathbf{x}) = \begin{bmatrix} \psi(\mathbf{x}) \\ \nabla\psi(\mathbf{x}) \times (\mathbf{x} - \mathbf{x}_0) \end{bmatrix} = 0. \tag{2}$$

The operator $L(\mathbf{x})$ specifies that the point $\mathbf{x}$ should lie on the zero distance curve and, additionally, that the vector pointing from the original point $\mathbf{x}_0$ to $\mathbf{x}$ should be parallel to the normal of the zero distance curve. Since each grid point $\mathbf{x}_0$ is only visited once the algorithmic complexity is of order $O(N)$.

The system (2) is solved by way of a Newton scheme, as described in Persson and Strang [4]; also a variant as a two step Newton scheme is presented in a paper by Chopp [1] where the second order derivatives of the Jacobian has been omitted, this approach was tried but eventually dropped due to slow convergence.

For our purposes the two dimensional version of the operator and Jacobian will look like:

$$L(\mathbf{x}) = \left[\begin{array}{c} \psi(x,y) \\ (x-x_0)\psi_y - (y-y_0)\psi_x \end{array} \right]$$

$$J(\mathbf{x}) = \frac{\partial L}{\partial \mathbf{x}} = \left[\begin{array}{cc} \psi_x & \psi_y + (x-x_0)\psi_{xy} - (y-y_0)\psi_{xx} \\ \psi_y & -\psi_x - (y-y_0)\psi_{xy} + (x-x_0)\psi_{yy} \end{array} \right]^T$$

The typical iteration employed is $\mathbf{x}^{k+1} = \mathbf{x}^k - \delta J^{-1}(\mathbf{x}^k)L(\mathbf{x}^k)$, where δ is a relaxation parameter which can be adaptively adjusted to reduce the stepsize and to keep the updates from diverging. After each iteration convergence is checked by taking the residual norm of the operator $L(\mathbf{x})$, if convergence has been achieved the new distance is given by $\phi(\mathbf{x}_0) = |\mathbf{x} - \mathbf{x}_0|$.

The convergence properties of the algebraic Newton method depends on the smoothness of the given approximate distance field ψ. Should this field be non-smooth then the method will fail to find the exact distance in regions where the gradient is undefined (such as the corners of a square). Thus the algebraic Newton method can not be seen as a truly general method to compute distance functions for arbitrary distance fields, but the distance fields must belong to class C^2, that is have continuous second derivatives.

2.3. The fast marching method. The fast marching method originally devised by Sethian [6, 7] takes into account the characteristics of the solution, knowing that information will only propagate outward from the zero distance curve. Starting from there, each grid point is updated in order of increasing distance in an upwind fashion. Initialization of the fast marching algorithm is done with the following steps

- Tag all points on the cells intersecting the zero distance curve as *Accepted*, and calculate exact distance values to these points.
- Tag all grid points that lie in the neighboring cells to the boundary points as *Trial*, also compute an initial approximate distance value to these.
- All other points lie in the *Unknown* set and should be given distance values that are bigger than the largest possible distance value.

After these initial steps the algorithm proceeds as follows

1. Find the point with the smallest distance value in the set of *Trial* points.
2. Remove this point from the *Trial* set and add it to the *Accepted* set.
3. Add all points of neighboring cells to the newly accepted point, that do not belong to the *Accepted* set to the *Trial* set. Now compute new distance values of all *Trial* points that are neighbors to the newly accepted point.
4. Repeat the procedure until the *Trial* set is empty.

The key to efficiency of fast marching is the realization of a heap structure for the *Trial* set. This enables the sorting and finding of the minimum distance within the *Trial* set (step 1.) to be executed on average in $O(log\ N)$ operations [5]. Thus the algorithmic complexity of the fast marching method is of order $O(N\ log\ N)$ [6]. If we compare the fast marching and fast sweeping methods we see that fast marching costs $O(log\ N)$ operations more to sort out the real propagation order, while fast sweeping assumes this is known in advance thus escaping the added cost.

2.4. Brute force redistancing. The simple brute force method consists of subdividing or approximating the zero distance curve with linear line segments or just sampling points for an even easier version. Then the distance for each grid point to all approximated zero distance segments is computed from which the minimum is taken as the new distance. This algorithm is obviously of order $O(N \cdot M)$ with N being the number of grid points and M the number of zero distance segments. Thus if we have many interface segments the algorithm could approach quadratic costs while on the other hand for very few interface segments the algorithm is close to linear.

2.5. Hyperbolic PDE. The commonly used PDE based redistancing scheme uses the following time dependent PDE for redistancing the given approximate distance function ϕ_0 [10]:

$$\frac{\partial \phi}{\partial t} = S(\phi_0)(1 - |\nabla \phi|) \tag{3}$$

This equation is solved to the stationary limit together with homogeneous Neumann boundary conditions and initial condition $\phi(\mathbf{x}, 0) = \phi_0(\mathbf{x})$.

$S(d)$ is a sign function introduced to propagate the information out from the zero distance curve. Usually not the pointwise sign function is chosen but a smoothed version, such as $S(d) = d^2/\sqrt{d^2 + \epsilon^2}$ where ϵ is proportional to the grid size or smoothing distance. This also leads to a problem that the zero distance curve will eventually wander off its initial position. Additional constraints can be introduced to minimize this effect such as done in a paper by Sussman and Fatemi [11].

Equation (3) can also be rewritten as

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = S(\phi_0), \quad \mathbf{u} = S(\phi_0) \frac{\nabla \phi}{|\nabla \phi|}$$

taking on the guise of a normal, but nonlinear, convection and diffusion transport problem. Since there are a vast number of methods and discretization schemes to solve this problem (FDM, FEM, FVM, coupled with ENO methods, etc.), all resulting in different accuracies and computation timings, no results for this method are presented. Redistancing by this hyperbolic PDE could probably be a benchmark all by itself.

3. Algorithmic components. In this section we describe some algorithms that are needed for some of the solution schemes in a general context.

3.1. Unstructured difference update. Both the fast marching method and the fast sweeping method depend on the ability to solve the Eikonal equation for a given grid point by using the distance values from the surrounding points. While this is rather straightforward given a strict Cartesian grid, this is not so if we have a perturbed or even unstructured grid. To manage this we follow the general approach taken by Sethian and Vladimirsky [8] which allows for both first and second order updates. Another more geometrically oriented approach is taken by Kimmel et al. [3, 9], which however only allows for updates of first order accuracy.

To construct an approximation to the Eikonal equation for a grid point $\mathbf{x}$ using the surrounding points $\mathbf{x}_i, \cdots, \mathbf{x}_n$, first define the unit directional vector pointing from point $\mathbf{x}_i$ to point $\mathbf{x}$ as $\mathbf{P}_i = (\mathbf{x} - \mathbf{x}_i)/||\mathbf{x} - \mathbf{x}_i||$. Then let $v_i(\mathbf{x})$ be the value of the approximate directional derivative in direction $\mathbf{P}_i$ and point $\mathbf{x}$. Thus we can write that $v_i(\mathbf{x}) = \mathbf{P}_i \cdot \nabla\phi(\mathbf{x})$, taking all directions i, substituting this into the Eikonal equation (1), and squaring gives the following

$$\mathbf{v}(\mathbf{x})^T(\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{v}(\mathbf{x}) = f(\mathbf{x})^2 \tag{4}$$

where the matrix $\mathbf{P}$ is constructed by placing the directional vectors $\mathbf{P}_i$ as its rows and $\mathbf{v}(\mathbf{x})$ is simply a column vector with elements $v_i(\mathbf{x})$. Since $\mathbf{P}$ has to be nonsingular the updates are restricted to come from simplices, that is triangles in 2D and tetrahedra in 3D.

To solve equation (4) we need to be able to write the directional derivatives $v_i(\mathbf{x})$ as functions of the unknown distance $\phi(\mathbf{x})$. First and second order difference approximations are given as follows:

$$v_i(\mathbf{x})^{O(1)} = \frac{\phi(\mathbf{x})}{||\mathbf{x}-\mathbf{x}_i||} - \frac{\phi(\mathbf{x}_i)}{||\mathbf{x}-\mathbf{x}_i||}$$

$$v_i(\mathbf{x})^{O(2)} = \frac{2\phi(\mathbf{x})}{||\mathbf{x}-\mathbf{x}_i||} - \frac{2\phi(\mathbf{x}_i)}{||\mathbf{x}-\mathbf{x}_i||} - \mathbf{P}_i \cdot \nabla\phi(\mathbf{x}_i)$$

Both of these formulas permit the rewriting of $v_i(\mathbf{x})$ into the form $v_i(\mathbf{x}) = a_i\phi(\mathbf{x}) + b_i$. Thus the Eikonal equation is finally reduced to a quadratic equation for the distance $\phi(\mathbf{x})$ in the form

$$(\mathbf{a}^T\mathbf{Q}\mathbf{a})\phi(\mathbf{x})^2 + (2\mathbf{a}^T\mathbf{Q}\mathbf{b})\phi(\mathbf{x}) + (\mathbf{b}^T\mathbf{Q}\mathbf{b}) = f(\mathbf{x})^2$$

where $\mathbf{Q}$ is defined as $\mathbf{Q} = (\mathbf{P}\mathbf{P}^T)^{-1}$ and the vectors $\mathbf{a}$ and $\mathbf{b}$ by their respective constituents a_i and b_i. This equation is then solved to find two roots of which the largest one is taken as a possible new distance for $\phi(\mathbf{x})$. This new distance can only be accepted if the update comes from within the polygon spanned by $\mathbf{x}, \mathbf{x}_i, \cdots, \mathbf{x}_n$ and if the calculated distance value is smaller than the old one. Thus we only update if the vector components $\mathbf{Q}\mathbf{v}(\mathbf{x}) \approx \mathbf{Q}(\mathbf{a}\phi(\mathbf{x})+\mathbf{b})$ are all positive and if $\phi(\mathbf{x}) < \phi^{old}(\mathbf{x})$.

3.2. Nodal point update and simplex construction. Given a point, A, to update with the unstructured update in a quadrilateral grid we proceed in the following way.

- Loop over all quadrilaterals that include grid point A as one of the defining vertices.
- Construct virtual triangles from grid point A and the other grid points of each quadrilateral. For an arbitrary quadrilateral with node numbering A to D then the virtual triangles will have node combinations $A-B-C$, $A-C-D$, and $A-B-D$.
- Apply the unstructured difference update to grid point A for each virtual triangle in turn.

This splitting of the quadrilateral is quite effective to handle the imposed stability constraint of the unstructured update, that is to only update from acute triangles [3]. It is easy to see that the possibility to update point A from an acute angle is always present. This possibly eliminates or at least reduces the need for splitting and unfolding of obtuse triangles described in references [8, 3]. The quadrilateral splitting procedure is illustrated in Figure 1.

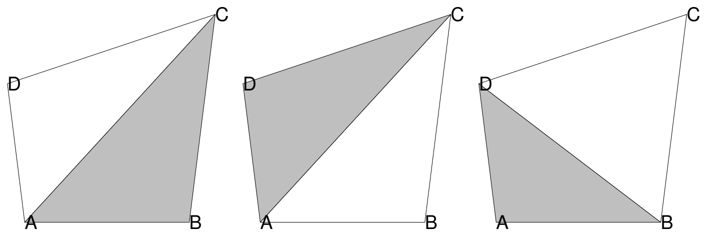

FIG. 1. *Simplex construction from a quadrilateral, the updating of point A comes from the gray triangle*

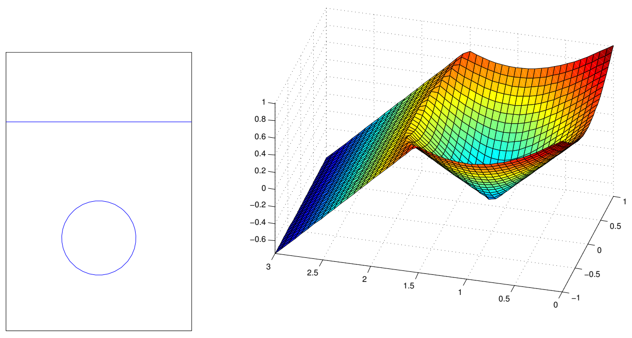

FIG. 2. *Zero distance curve and distance function field for the test case*

4. Numerical experiments. This section describes the test benchmark and results for the various solution schemes. For the interested reader we can say that the benchmarks were entirely written in Fortran 77, compiled with the Portland Group PGI F77 Compiler, and run on a 1.8 GHz AMD Opteron machine. In the following tables the column *CPU* denotes the actual required number of CPU seconds for a given method to converge.

4.1. Test case. For the computational experiments a non trivial test case was chosen that both reflects typical features of an engineering computation, and also allows for an analytical solution. The test case is given by the following exact distance function field

$$\phi(\mathbf{x}) = min(2.25 - y, \sqrt{x^2 + (y-1)^2} - 0.4.$$

The computational domain is a rectangle spanned by $x = [-1,\ 1],\ y = [0,\ 3]$. Both the zero distance curve and distance function field can be seen in Figure 2. The distance field contains both singularities, where the gradient is essentially undefined, and also parts where the solution is smooth, giving us an interesting test case.

The basic coarse Cartesian grid consists of 24 conforming quadrilaterals with cell edge length $h = 0.5$, which correspond to level 1. Successive mesh levels are created by uniform subdivision, that is simply by splitting each cell into four new ones via its centroid coordinate. To test the unstructured capabilities of the solution algorithms presented in section 2, stochastic perturbations were introduced corresponding to 10% and 25% of the approximate mean cell edge length $h = 1/(\sqrt{N} - 1)$.

4.2. Error estimation. The computed solution vectors for all gridpoints were measured against the exact solutions in the following relative error norms

$$l_1 \; error \;:\; ||u||_1 = \frac{1}{N}\sum_{i=1}^{N} \frac{|\phi_{i,exact} - \phi_{i,computed}|}{|\phi_{i,exact}|}$$

$$l_2 \; error \;:\; ||u||_2 = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(\frac{|\phi_{i,exact} - \phi_{i,computed}|}{|\phi_{i,exact}|}\right)^2}$$

$$l_\infty \; error \;:\; ||u||_\infty = \max_i \frac{|\phi_{i,exact} - \phi_{i,computed}|}{|\phi_{i,exact}|}$$

The error norms were only computed for points that did not belong to the cells intersecting the zero distance curve, or to points with a distance greater than a prescribed maximum, thus precluding the influence of singularities. Also convergence rates for the various methods in the difference error norms were established as

$$ROC = log_{10}(\frac{||\mathbf{u}^{l-1}||}{||\mathbf{u}^{l}||})/log_{10}(\frac{h^{l-1}}{h^{l}})$$

where l is the level of refinement and h the mean cell edge length.

4.3. The fast marching method. Tables 1 and 2 give the results for the fast marching method on levels 6 through 9 with 1st and 2nd order accuracy respectively. Only points with a smaller distance than 3.4 were included in the error calculation showing the local properties of the method. It can be seen that the convergence rates very accurately approach the expected values with successive grid refinements. Also note that the grid distortion only causes a small reduction in accuracy.

4.4. The Fast Sweeping Method. Fast sweeping should ideally produce identical results to the fast marching method since it uses the same difference update formula. From our calculations we do obtain identical results for non distorted grids. Unfortunately as the grids become more and more distorted the update procedure often fails since we cannot be sure that the sweeping order is correct for an unstructured grid. In our benchmarks we used the same update order for distorted grids as for the purely Cartesian one.

Regarding computational time, the fast sweeping method require more and more sweeps to converge when the grid size decreases, especially for truly unstructured grids. All in all it can be said that even on fully Cartesian grids the fast marching method is quicker if we use the unstructured update.

4.5. Algebraic Newton method. Table 3 presents the results from the algebraic Newton redistancing scheme with an approximate distance function given by a bilinear polynomial on each quadrilateral cell (the standard finite element Q^1 basis function). This field approximation was quite challenging since it is only piecewise differentiable, although it was a representative choice since one can rarely rely on to have a nice analytical approximate distance function in practice.

A convergence criteria of 10^{-9} and a maximum number of nonlinear iterations equal to 200 were used with the Newton method. The stepsize was also restricted not to exceed one coarse mesh cell edge length.

Level	CPU	$\|\|u\|\|_1$	ROC$_1$	$\|\|u\|\|_2$	ROC$_2$	$\|\|u\|\|_\infty$	ROC$_\infty$
No mesh perturbation							
6	0.05	0.211E-02	0.82	0.435E-02	0.82	0.241E-01	0.72
7	0.24	0.113E-02	0.90	0.232E-02	0.91	0.139E-01	0.80
8	1.18	0.582E-03	0.96	0.119E-02	0.96	0.738E-02	0.91
9	6.47	0.297E-03	0.97	0.609E-03	0.97	0.388E-02	0.93
10% mesh perturbation							
6	0.05	0.214E-02	0.85	0.438E-02	0.85	0.242E-01	0.74
7	0.25	0.114E-02	0.91	0.232E-02	0.92	0.139E-01	0.80
8	1.27	0.585E-03	0.96	0.119E-02	0.96	0.738E-02	0.91
9	7.10	0.298E-03	0.97	0.607E-03	0.97	0.387E-02	0.93
25% mesh perturbation							
6	0.05	0.218E-02	0.85	0.432E-02	0.86	0.242E-01	0.79
7	0.26	0.117E-02	0.90	0.230E-02	0.91	0.138E-01	0.81
8	1.46	0.599E-03	0.96	0.118E-02	0.96	0.731E-02	0.92
9	6.45	0.305E-03	0.97	0.601E-03	0.97	0.383E-02	0.93

TABLE 1
Results for redistancing by the fast marching method with 1st order update

Level	CPU	$\|\|u\|\|_1$	ROC$_1$	$\|\|u\|\|_2$	ROC$_2$	$\|\|u\|\|_\infty$	ROC$_\infty$
No mesh perturbation							
6	0.05	0.531E-04	1.84	0.104E-03	1.86	0.111E-02	1.45
7	0.26	0.141E-04	1.92	0.271E-04	1.93	0.354E-03	1.64
8	1.36	0.360E-05	1.97	0.686E-05	1.98	0.901E-04	1.98
9	7.54	0.911E-06	1.98	0.173E-05	1.99	0.232E-04	1.96
10% mesh perturbation							
6	0.07	0.544E-04	1.88	0.105E-03	1.87	0.119E-02	1.46
7	0.29	0.145E-04	1.91	0.275E-04	1.94	0.361E-03	1.72
8	1.46	0.374E-05	1.96	0.700E-05	1.97	0.915E-04	1.98
9	8.09	0.957E-06	1.96	0.177E-05	1.98	0.234E-04	1.97
25% mesh perturbation							
6	0.06	0.594E-04	1.84	0.112E-03	1.85	0.130E-02	1.55
7	0.27	0.162E-04	1.87	0.296E-04	1.91	0.376E-03	1.79
8	1.66	0.423E-05	1.94	0.764E-05	1.95	0.944E-04	1.99
9	8.14	0.109E-05	1.96	0.195E-05	1.97	0.240E-04	1.98

TABLE 2
Results for redistancing by the fast marching method with 2nd order update

From the table we can see that the overall accuracy were almost unaffected by grid distortion. Also the accuracy and convergence rates seem to be restricted for the Q^1 construction of the approximate distance field.

Level	CPU	$\|\|u\|\|_1$	ROC$_1$	$\|\|u\|\|_2$	ROC$_2$	$\|\|u\|\|_\infty$	ROC$_\infty$
No mesh perturbation							
6	0.51	0.247E-03	1.68	0.492E-03	1.51	0.355E-02	0.86
7	2.55	0.752E-04	1.72	0.174E-03	1.50	0.176E-02	1.01
8	13.44	0.221E-04	1.76	0.614E-04	1.50	0.881E-03	1.00
9	81.02	0.638E-05	1.79	0.217E-04	1.50	0.442E-03	1.00
10% mesh perturbation							
6	0.55	0.253E-03	1.70	0.501E-03	1.51	0.396E-02	0.88
7	2.80	0.765E-04	1.72	0.178E-03	1.49	0.221E-02	0.84
8	14.91	0.225E-04	1.77	0.626E-04	1.51	0.102E-02	1.11
9	88.99	0.645E-05	1.80	0.220E-04	1.51	0.519E-03	0.98
25% mesh perturbation							
6	0.56	0.267E-03	1.69	0.539E-03	1.51	0.547E-02	0.72
7	2.73	0.804E-04	1.73	0.192E-03	1.49	0.304E-02	0.85
8	14.44	0.237E-04	1.77	0.675E-04	1.51	0.210E-02	0.53
9	90.36	0.680E-05	1.80	0.237E-04	1.51	0.716E-03	1.55

TABLE 3
Results for the algebraic Newton method with tolerance 10^{-9}

Level	CPU	$\|\|u\|\|_1$	ROC$_1$	$\|\|u\|\|_2$	ROC$_2$	$\|\|u\|\|_\infty$	ROC$_\infty$
No mesh perturbation							
6	0.42	0.258E-03	1.69	0.570E-03	1.50	0.500E-02	0.80
7	3.36	0.759E-04	1.76	0.193E-03	1.56	0.249E-02	1.00
8	26.69	0.211E-04	1.85	0.648E-04	1.58	0.127E-02	0.98
9	215.62	0.609E-05	1.80	0.229E-04	1.50	0.654E-03	0.96
10% mesh perturbation							
6	0.48	0.266E-03	1.71	0.588E-03	1.52	0.613E-02	0.77
7	3.77	0.777E-04	1.78	0.199E-03	1.56	0.296E-02	1.05
8	30.22	0.214E-04	1.86	0.658E-04	1.59	0.149E-02	0.99
9	243.15	0.617E-05	1.79	0.232E-04	1.50	0.729E-03	1.04
25% mesh perturbation							
6	0.48	0.283E-03	1.68	0.632E-03	1.52	0.815E-02	0.82
7	3.78	0.823E-04	1.78	0.215E-03	1.55	0.399E-02	1.03
8	30.13	0.224E-04	1.87	0.704E-04	1.61	0.198E-02	1.01
9	240.69	0.650E-05	1.79	0.248E-04	1.50	0.942E-03	1.07

TABLE 4
Results for the brute force method with two line segment interface approximation

4.6. Brute force method. In Table 4 one can see the results for brute force redistancing where each quadrilateral has been subdivided into two triangles. From there straight line segments were constructed by taking the zero distance points on the triangle edges. Also here as expected the accuracy is virtually independent of grid distortion. As for the increase in time it is obvious that this method scales quite unfavorably.

5. Conclusions.

From the tables it can clearly be seen that the fast marching method outperforms all other methods with respect to speed, accuracy and robustness. It is also notable that the second order method is only marginally more expensive than the first order method and actually produces the most accurate results of all. The only drawback of the second order fast marching method is that we have to compute and store arrays of the first derivatives causing some increase in memory consumption.

The fast sweeping method proved both slower than the fast marching method and also quite unstable for highly perturbed grids. It is possible that these deficiencies could be overcome by more precise definitions of the sweeping order, but as it stands now this method is not competitive in a general context.

The algebraic Newton method is limited by the fact that it requires an approximate distance function for it to work. It is not possible to use the algorithm with a simple step function as initial distance field. The choice of an approximate distance function therefore influences the end results and requires careful tuning of the Newton scheme. For the considered benchmark the timings were a factor of ten slower than for the fast marching method, but this could vary somewhat depending on convergence criteria, the maximum number of allowed iterations and the choice of relaxation parameter.

Finally, redistancing via brute force does produce consistently good results for general grids but the timings scale unfavorably for this method to be generally interesting. The method could possibly prove useful to redistance a limited number of cells, such as the cells intersecting the initial zero distance curve. Brute force redistancing also shares a deficiency with the algebraic Newton method, that it can only produce the Euclidean distance, which will not be correct unless the domain is simply connected.

REFERENCES

[1] D. L. Chopp, *Some improvements of the fast marching method*, SIAM J. Sci. Computing. Vol. 23 (2001), No. 1, pp. 230-244.

[2] P. A. Gremaud and C. M. Kuster, *Computational study of fast methods for the Eikonal equation*, NCSU-CRSC Tech Report CRSC-TR04-15, submitted to SIAM J. Sc. Comp.

[3] R. Kimmel and J. A. Sethian, *Computing geodesic paths on manifolds*, Proceedings of National Academy of Sciences, Vol. 95 (1998), No. 15, pp. 8431-8435.

[4] P.-O. Persson and G. Strang, *A simple mesh generator in Matlab*, SIAM Review Vol. 46 (2004), No. 2, pp. 329-345.

[5] R. Sedgewick, *Algorithms, 2nd Ed.*, Addison-Wesley, 1988.

[6] J. A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proceedings of National Academy of Sciences, Vol. 93 (1996), No. 4, pp. 1591-1595.

[7] J. A. Sethian, *Level set methods and fast marching methods*, Cambridge University Press, 1999.

[8] J. A. Sethian and A. Vladimirsky, *Fast methods for the Eikonal and related Hamilton-Jacobi equations on unstructured meshes*, Proceedings of National Academy of Sciences, Vol. 97 (2000), No. 11, pp. 5699-5703.

[9] A. Spira and R. Kimmel, *An efficient solution to the Eikonal equation on parametric manifolds*, Interfaces and Free Boundaries, Volume 6, Issue 3, 2004, pp: 315-327.

[10] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solutions to incompressible two-phase flow* J. Comp. Phys., 94, pp. 146-159 (1994).

[11] 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., Vol. 20, No. 4, pp. 1165-1191 (1999).

[12] Y. R. Tsai, *Rapid and accurate computation of distance function using grids*, UCLA CAM Report 00-36, 2000.

[13] Y. R. Tsai, H.-K. Zhao, and S. Osher, *Fast sweeping algorithms for a class of Hamilton-Jacobi equations*, UCLA CAM Report 01-27, 2001.