COMPUTATIONAL STUDY OF FAST METHODS FOR THE EIKONAL EQUATION

PIERRE A. GREMAUD* AND CHRISTOPHER M. KUSTER†

Abstract. A computational study of the Fast Marching and the Fast Sweeping methods for the Eikonal equation is given. It is shown that even though Fast Sweeping requires asymptotically less operations than Fast Marching, the latter method is faster than the former for the problems under consideration and for realistic grid sizes. Fully second order generalizations of methods of this type for problems with obstacles are proposed.

Key words. Hamilton-Jacobi, Eikonal, viscosity solution, Fast Marching, Fast Sweeping

AMS subject classifications. 65N06, 65Y20, 49L25, 35F30

1. Introduction. Steady state Hamilton-Jacobi equations play a major role in countless applications. While the *discretization* of those equations is approaching maturity (at least for single equations and convex Hamiltonians), the *resolution* of the nonlinear discretized systems is an active topic of research.

In this paper, two methods are compared: the Fast Marching method (FM) [13, 14, 17] and the Fast Sweeping method (FS) [4, 8, 16]. The two-dimensional Eikonal equation is taken as a test problem: given a slowness field F (a positive function), we look for a time of propagation u satisfying

$$|\nabla u| = F \qquad \text{in } \Omega \subset \mathrm{R}^2, \tag{1.1}$$

$$u = 0 \qquad \text{on } \Gamma \subset \mathrm{R}^2. \tag{1.2}$$

The two-dimensional domain Ω is the computational domain; Γ is a source, i.e., a closed subset of $\bar{\Omega}$ in which u is zero. The slowness field F can be unbounded, corresponding to the presence of obstacles.

We refer the reader to [8, 15] for issues related to the application of the above methods to the more general problem

$$H(x, \nabla u) = F. \tag{1.3}$$

While clearly of importance, those considerations are more related to discretization than they are to resolution and are not the focus of this paper. Discretization algorithms are only briefly reviewed in Section 2. In Section 3, three nonlinear solvers are described (Gauss-Jacobi, FM and FS/nonlinear Gauss-Seidel). Section 4 presents an adaption of the Fast Marching and Fast Sweeping methods to problems with obstacles. The relative efficency of Fast Marching and Fast Sweeping is discussed in Section 5. Concluding remarks are offered in Section 6.

2. Discretization. The basic construction principles for numerical Hamiltonians are well known [10] and are easily described on a Cartesian grid. There, the basic scheme takes the form

$$\mathcal{H}(D^x_+ U_{ij}, D^x_- U_{ij}; D^y_+ U_{ij}, D^y_- U_{ij}) = \mathcal{F}_{ij}, \tag{2.1}$$

*Department of Mathematics and Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC 27695-8205, USA (gremaud@unity.ncsu.edu). Partially supported by the National Science Foundation (NSF) through grants DMS-0204578 and DMS-0244488.

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

with the obvious notation. In that case, we recall that $\mathcal{H}$ is consistent if

$$\mathcal{H}(p,p;q,q) = H(p,q) \qquad \forall p,q, \tag{2.2}$$

where H is the Hamiltonian of the problem under consideration (here $H(p,q) = \sqrt{p^2+q^2}$). The numerical Hamiltonian is monotone if nonincreasing in its first and third arguments and nondecreasing in the other two: $\mathcal{H}(\downarrow,\uparrow,\downarrow,\uparrow)$ [6]. Consistency ensures that the correct problem is approximated while monotonicity guarantees convergence to the correct viscosity solution [5, 12].

Many numerical Hamiltonians can be found in the literature; most of them are derived from corresponding numerical fluxes for conservation laws. The main examples are the Godunov flux $\mathcal{H}^G$ and the Lax-Friedrichs flux $\mathcal{H}^{LF}$ which here take the form

$$\begin{aligned}
\mathcal{H}^G(p_+,p_-;q_+,q_-) &= \sqrt{\max\{p_-^+,p_+^-\}^2 + \max\{q_-^+,q_+^-\}^2},\\
\mathcal{H}^{LF}(p_+,p_-;q_+,q_-) &= \sqrt{\left(\frac{p_++p_-}{2}\right)^2 + \left(\frac{q_++q_-}{2}\right)^2}\\
&\quad - \sigma_x\frac{p_+-p_-}{2} - \sigma_y\frac{q_+-q_-}{2},
\end{aligned}$$

where $\sigma_x \geq \max\left|\frac{\partial H}{\partial p}\right|$ and $\sigma_y \geq \max\left|\frac{\partial H}{\partial q}\right|$, where $(\cdot)^+ = \max\{\cdot,0\}$ and $(\cdot)^- = -\min\{\cdot,0\}$. Upwind discretizations such as the Godunov method are especially simple to implement for (1.1) and for convex Hamiltonians in general. However, $\mathcal{H}^G$, for instance, becomes very involved for general Hamiltonians. While centered methods such as Lax-Friedrichs' do not suffer from that problem, they require special treatment at the boundary of the computational domain [8]. The discretization used below is essentially $\mathcal{H}^G$.

Many applications call for the use of noncartesian or unstructured meshes, see [1, 7, 9, 14, 15] for a few examples of such methods. Following [14, 15], consider a node X_0 at which the solution U is to be computed and two neighboring nodes X_1 and X_2 at which the values of U (U_ℓ, $\ell = 1,2$) and its derivatives ($\partial_x U_\ell, \partial_y U_\ell$, $\ell = 1,2$) are known or have already been computed. We set

$$N_\ell = \frac{X_0 - X_\ell}{|X_0 - X_\ell|}, \quad \ell = 1,2 \quad \text{and} \quad \mathcal{N} = \left[\frac{N_1}{N_2}\right],$$

where $\mathcal{N}$ is a 2×2 nonsingular matrix, assuming X_0, X_1 and X_2 are not lined up. The directional derivatives in the directions N_1 and N_2 are approximated by

$$D_\ell U = \frac{U_0 - U_\ell}{|X_0 - X_\ell|} \quad \ell = 1,2 \qquad \text{(1st order formula)}, \tag{2.3}$$

$$D_\ell U = 2\frac{U_0 - U_\ell}{|X_0 - X_\ell|} - N_\ell \cdot [\partial_x U_\ell, \partial_y U_\ell] \quad \ell = 1,2 \qquad \text{(2nd order formula)}. \tag{2.4}$$

Those approximate directional derivatives are linked to the gradient by

$$D_\ell U = N_\ell \cdot \nabla U + \mathcal{O}(|X_0 - X_i|^\alpha) \qquad \text{or} \qquad DU = \mathcal{N}\nabla U + \mathcal{O}(h^\alpha), \tag{2.5}$$

where $DU = [D_1U, D_2U]$, $h = \max\{|X_0 - X_1|, |X_0 - X_2|\}$, and $\alpha = 1$ or 2 depending on whether (2.3) or (2.4) is respectively used. Solving for ∇U and plugging into (1.1), one finds the (quadratic) equation defining the unknown U_0

$$DU^t(\mathcal{N}\mathcal{N}^t)^{-1}DU = (F(X_0))^2. \tag{2.6}$$

The above methods are well understood theoretically: convergence to the viscosity solution when the Godunov flux $\mathcal{H}^G$ is used was established in [12], see also [15] for additional results and references.

3. Resolution. The system of coupled quadratic equations corresponding to imposing (2.6) at all nodes has to be solved. If θ denotes the angles between N_1 and N_2, the matrix $\mathcal{N}\mathcal{N}^t$ is symmetric positive definite provided $0 < \theta < \pi$; its condition number is $\frac{1+\cos\theta}{1-\cos\theta}$. Therefore, regardless of the choice of stencil (first or second order) (2.6) has two real solutions. This situation is generic for Hamilton-Jacobi equations.

Let $(U_1, U_2, \ldots, U_N)$ be the unknowns for an arbitrary ordering of the nodes. The resulting system can be written symbolically as

$$f(U) = f(U_1, U_2, \ldots, U_N) = 0.$$

An iterative scheme, essentially a fixed point method or Gauss-Jacobi method, was proposed in [12]. One step of the algorithm is as follows

Gauss-Jacobi

- choose a node and regard the neighboring values as fixed;
- solve (2.6) for the value at the considered node; the largest of the two possible solutions is selected, in agreement with the viscosity criterion;
- repeat until convergence (fixed point).

The corresponding pseudo-code is

```
k = 0
U^0 = (0, ..., 0)
while  U^k ≠ U^{k+1}  or  k ≠ 0
       k = k + 1
       for  ℓ = 1  to  N
            solve f(U_1^{k-1}, ..., U_{ℓ-1}^{k-1}, U_ℓ^k, U_{ℓ+1}^{k-1}, ..., U_N^{k-1}) = 0  for  U_ℓ^k
       end
end
```

This approach is obviously slow as each node has to be revisited several times before the numerical solution settles down. Further, no advantage is taken of the propagation character of the problem.

With marching methods, the nodes are considered in an order consistent with the way the wave fronts propagate, i.e., with the Huygens principle [13, 14, 17]. This leads to (quasi) single pass algorithms. The Fast Marching algorithm, FM, is based on the monotonicity of the solution along the characteristics. The nodes are divided are into three sets: the already computed nodes, the not yet considered nodes and the nodes "on the propagation front".

Fast Marching

- the node with lowest value in the "front" set is removed from it and added to the "computed" set; its not yet computed neighbors are added to the "front";
- the values of all the nodes in the "front" are recomputed using (2.6);
- repeat until all the nodes are "computed".

The list operations are handled by the heap sort algorithm [3, 18], which is the key to the speed of FM. This type of sort has the advantage that every tree operation (insertion, removal, update) is of order $\log_2 n$, where n is the number of elements in the tree, i.e., the number of "front" nodes at that step. Note that n is of order $\sqrt{N}$

at every step, where N is the total number of nodes in the mesh. By placing the "front" nodes into this structure, the computational complexity is kept at the order of $N \log_2 N$. The pseudo-code for FM is

```
U = (∞, ..., ∞)
"front" = "source"
while "front" ≠ ∅
      U_min = min U ∈ "front"
       "front" = "front" + neighbors of U_min
      remove U_min from "front"
      for U_ℓ ∈ "front"
           solve f(U_1, ..., U_{ℓ-1}, U_ℓ, U_{ℓ+1}, ..., U_N) = 0 for U_ℓ
      end
end
```

A third type of methods has recently been proposed as the Fast Sweeping method, FS, see among others [4, 8, 16]. FS aims at improving on the first Gauss-Jacobi method by using a Gauss-Seidel type update process. It also avoids the ordering step of FM. Instead of proceeding in a way consistent with the underlying wave propagation, FS considers sweeps in predetermined directions. The directions of sweep can correspond for instance to the direction of the coordinate axes. For two dimensional problems, there are four sweeping directions. More precisely, in a $M \times M$ matrix, we would have: $i = 1 : M, j = 1 : M$ (upper left to lower right, ULLR), $j = 1 : M, i = M : 1$ (lower left to upper right, LLUR), $i = M : 1, j = M : 1$ (lower right to upper left, LRUL), $j = M : 1, i = 1 : M$ (upper right to lower left, URLL).

Fast Sweeping

- choose a direction of sweep and a corresponding ordering;
- loop through of all the nodes in the chosen order and successively solve (2.6) for each of them;
- repeat for the other directions of sweep;
- repeat until convergence.

The corresponding pseudo-code is

```
k = 0
U^0 = (∞, ..., ∞)
order = [ULLR, LLUR, LRUL.URLL]
while U^k ≠ U^{k+1} or k = 0
      k = k + 1
      i = mod(k − 1, 4) + 1
      ordering = order[i]
      for ℓ = 1 to N
           solve f(U^k_1, ..., U^k_{ℓ-1}, U^k_ℓ, U^{k-1}_{ℓ+1}, ..., U^{k-1}_N) = 0 for U^k_ℓ
      end
end
```

The update process for the above three methods is schematically represented in Figure 3.1. FS requires a lot of structure from the mesh. Also, as a result of the absence of an true ordering process (only the way arrays are accessed changes), the complexity of the above algorithm can be shown to be of order N [19], as opposed to $N \log_2 N$ for FM. However, the results from Section 5 show that this rarely translates into any significant computational advantage in practice.

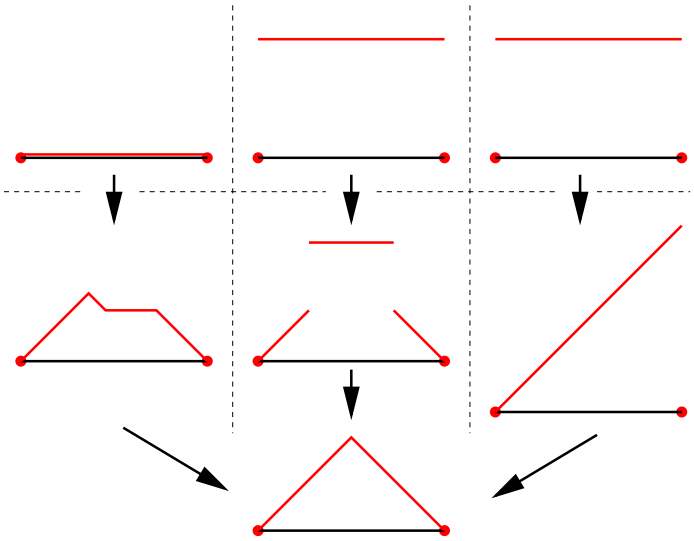

FIG. 3.1. Schematic representation of the update process for Gauss-Jacobi (left), Fast Marching (center) and Fast Sweeping (right), for a one-dimensional Eikonal equation with homogenous boundary conditions.

Unlike what is proposed in [16], we consider FS as a direct method for two reasons. First, as shown in Section 5, the algorithm has to be ran to its completion in all but the simplest examples. Second, the following result gives a natural stopping criterion.

LEMMA 3.1. *Let U^k, $k = 0, \ldots, \infty$, be the iterates generated by the Fast Sweeping method applied to (1.1, 1.2). Then there exists $\bar{k}$ such that $U^k = U^{\bar{k}}$ for all $k \geq \bar{k}$.*

Proof. Consider an arbitrary node i at which the solution has been computed. There exists a path made of successive upwind neighbors from i to Γ. By construction, each successive sweep determines the final value of at least one node on that path, starting on Γ and moving successively to i. This clearly leads to a finite algorithm. The numerical solution at node i only depends on the values at its neighbors. If those values do not change at step $\bar{k}+1$, then $U_i^k = U_i^{\bar{k}}$ for $k \geq \bar{k}$. □

4. Problems with obstacles. An obstacle is a domain in which waves propagate at infinitely slow speed: $F(x, y) = \infty$ if and only if (x, y) belongs to the obstacle. Problems with obstacles require additional care if both accuracy and efficiency, i.e., marching character, of the methods are to be preserved.

For future reference, we now introduce three specific test problems used in this paper. In all three cases, we consider (1.1, 1.2) with $\Omega = (0, 6)^2$, $\Gamma = (0, 0)$.

Example 1 $F(x, y) = 1 \quad \forall (x, y) \in \Omega$, no obstacle.

Example 2 $F(x, y) = \begin{cases} \infty & \text{if } (x-3)^2 + (y-3)^2 <= 1, \\ 1 & \text{otherwise,} \end{cases}$ circular obstacle.

Example 3 $F(x, y) = \begin{cases} \infty & \text{if } (x, y) \in \Xi, \\ 1 & \text{otherwise,} \end{cases}$ where Ξ is the "rings" of Figure 5.1.

If no local adaption is made when treating problems with obstacles such as Examples 2 and 3, accuracy is lost. Taking for instance Example 2, a straightforward discretization would consist in applying any of three above methods *outside* the circular obstacle using a simple Cartesian mesh. The left half of Table 4.1 illustrates the

loss accuracy incurred by the first and second order Fast Marching methods respectively, i.e., using (2.3) and (2.4).

first order methods												
	standard method						*modified method*					
M	L^1	rate	L^2	rate	L^∞	rate	L^1	rate	L^2	rate	L^∞	rate
25	2.91(0)		9.00(-1)		6.11(-1)		1.13(0)		3.04(-1)		1.44(-1)	
50	1.71(0)	.77	5.24(-1)	.78	3.52(-1)	.79	7.58(-1)	.58	2.00(-1)	.60	9.18(-2)	.65
100	1.07(0)	.68	3.27(-1)	.68	2.23(-1)	.66	4.41(-1)	.78	1.13(-1)	.82	4.92(-2)	.88
200	6.42(-1)	.74	2.00(-1)	.71	1.46(-1)	.61	2.39(-1)	.88	6.10(-2)	.89	2.65(-2)	.91
400	3.76(-1)	.77	1.20(-1)	.74	9.36(-2)	.64	1.27(-1)	.92	3.24(-2)	.91	2.02(-2)	.39
800	2.21(-1)	.77	7.24(-2)	.73	6.07(-2)	.62	6.38(-2)	.99	1.64(-2)	.99	1.54(-2)	.39
1600	1.31(-1)	.76	4.40(-2)	.72	3.82(-2)	.67	3.07(-2)	1.1	7.91(-3)	1.0	1.14(-2)	.43
3200	7.80(-2)	.75	2.68(-2)	.72	2.44(-2)	.65	1.49(-2)	1.0	3.88(-3)	1.0	6.77(-3)	.75
second order methods												
	standard method						*modified method*					
M	L^1	rate	L^2	rate	L^∞	rate	L^1	rate	L^2	rate	L^∞	rate
25	2.52(0)		1.07(0)		8.99(-1)		1.17(0)		2.84(-1)		1.16(-1)	
50	1.45(0)	.80	5.94(-1)	.85	7.39(-1)	.28	2.83(-1)	2.0	9.59(-2)	1.6	4.54(-2)	1.4
100	6.36(-1)	1.2	2.45(-1)	1.3	3.10(-1)	1.3	1.23(-1)	1.2	4.26(-2)	1.2	1.97(-2)	1.2
200	3.02(-1)	1.1	1.20(-1)	1.0	1.73(-1)	.84	3.29(-2)	1.9	1.14(-2)	1.9	1.22(-2)	.70
400	1.52(-1)	.99	6.16(-2)	.97	8.49(-2)	1.0	1.52(-2)	1.1	5.32(-3)	1.1	5.93(-3)	1.0
800	6.91(-2)	1.1	2.91(-2)	1.1	4.41(-2)	.95	2.06(-3)	2.9	8.22(-4)	2.7	3.11(-3)	.94
1600	3.42(-2)	1.0	1.45(-2)	1.0	2.31(-2)	.93	4.82(-4)	2.1	2.47(-4)	1.7	1.68(-3)	.89
3200	1.48(-2)	1.2	6.36(-3)	1.2	1.12(-2)	1.0	1.15(-4)	2.1	7.64(-5)	1.7	8.83(-4)	.93

TABLE 4.1

Convergence study for formally first and second order methods in the presence of a circular obstacle (Example 2); M measures the number of nodes on one edge of Ω, i.e., total number of nodes $N = \mathcal{O}(M^2)$.

As can be seen from those results, the standard first order method converges with an average order of only about .75 (in the L^1 norm) while the standard formally second order method loses a full order (the average rate in the L^1 norm is about 1.0). The results in the L^2-norm are comparable, while the L^∞ rates are significantly lower. This is not surprising since the underlying solution presents a "shock" at the back of the obstacle (discontinuity of the first derivatives).

In [2], we started the study of a generalization of the Fast Marching method from [14, 15] to problems with obstacles. The nodes are defined on a uniform Cartesian mesh away from the obstacle. Near the obstacle, additional nodes corresponding to the intersection of the obstacle's boundary with the mesh lines are added, see Figure 4.1. The detailed solving process is as follows for the modified Fast Marching algorithm.

Let X_0 be a node to be updated. Let X_1, X_2 be a pair of primary neighbors of X_0. Following [15], an upwinding criterion is considered: the characteristic direction should point into the simplex defined by X_0, X_1 and X_2. This is equivalent to requiring the approximate gradient defined by (2.5) to have positive components when expressed in terms of the unit vectors $N_1 = \frac{X_0 - X_1}{|X_0 - X_1|}$ and $N_2 = \frac{X_0 - X_2}{|X_0 - X_2|}$, in other words

$$(\mathcal{N}\mathcal{N}^t)^{-1} DU > 0. \tag{4.1}$$

The value U_0 at X_0 is then determined as follows

$$U_0 = \min \begin{cases} \text{solution of (2.6) with 2nd order stencil (2.4) if (4.1) is satisfied,} \\ \text{solution of (2.6) with 1st order stencil (2.3) if (4.1) is satisfied,} \\ \min_{i=1,2}\{U_i + |X_0 - X_i| F(X_0)\}. \end{cases} \tag{4.2}$$

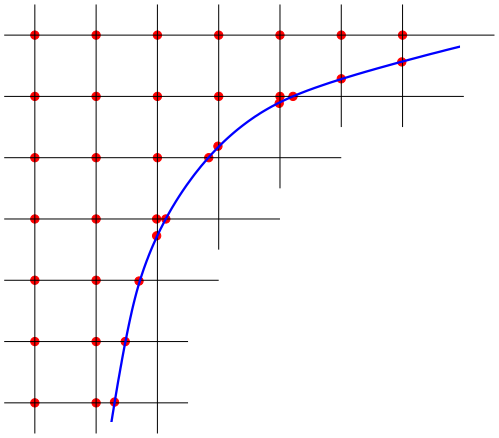

FIG. 4.1. Mesh structure around the obstacle.

Regardless of the choice of stencil (first or second order), (2.6) has two real solutions out of which we always consider the larger one only, to be consistent with causality. In the first two cases, the values of the partial derivatives of U at X_0 are updated using $\nabla U = \mathcal{N}^{-1} DU$. In the third case, we set $\nabla U = N_i F(X_0)$, where i is the index corresponding to the case of lowest value for U_0. This process is repeated for all pairs of primary neighbors X_1, X_2 and the lowest resulting value of U_0 is kept. A similar approach could be used for generalizing FS, however, due to the loss of structure of the modified mesh, not all nodes could be updated in a natural way at all sweeps, slowing down the convergence.

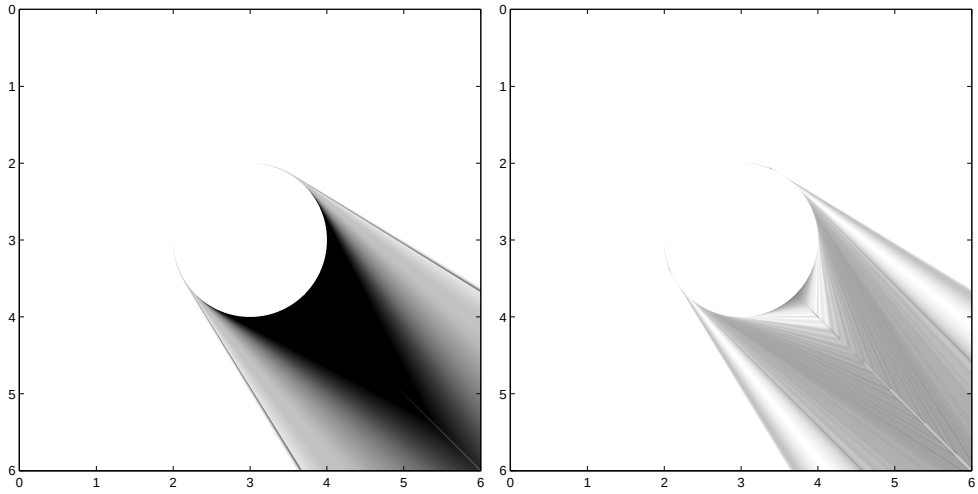

FIG. 4.2. Error propagation downstream of the obstacle for the standard (left) and modified (right) second order Fast Marching method (Example 2, $M = 1600$, $N = \mathcal{O}(M^2)$). The source is located at $(0, 0)$.

As observed in [11], in the presence of a point source, a singularity of the travel time field is clearly located at the source itself. Unless special care is taken near

that source, a loss of accuracy ensues. The problem can be fixed by defining a mesh independent domain around the source and initializing the values at the nodes there to the corresponding values of the exact solution. This fix was applied to all methods under study here. As illustrated in Table 4.1, the modified scheme is found to preserve first and second order accuracy in the L^1 norm even in the presence of obstacles. For the second order version, the L^2 rates of convergence are slightly below 2, while the method appears to converge with order 1 in the maximum norm. In all cases and all norms, the modified method exhibits much better accuracy than the standard method. Error propagation downstream from the obstacle is illustrated in Figure 4.2 for both standard and modified methods.

It should be noted that second order convergence for the finer meshes was only obtained after rewriting the quadratic formula for solving (2.6) in a form that limits cancellation effects (standard IEC559 double precision was used throughout).

Higher accuracy does not come for free. In Table 4.2, the overhead corresponding to maintaining optimal accuracy is described in terms of computational time and number of operations.

M	Time 1	Ops 1	Time 2	Ops 2	Time 3	Ops 3	Time 4	Ops 4
25	0.01	1.89 (4)	0.02	9.11 (4)	0.05	3.26 (5)	0.05	4.47 (5)
50	0.05	7.73 (4)	0.07	3.65 (5)	0.17	1.22 (6)	0.19	1.72 (6)
100	0.20	3.17 (5)	0.26	1.47 (6)	0.65	4.76 (6)	0.73	6.79 (6)
200	0.79	1.31 (6)	1.00	5.93 (6)	2.60	1.88 (7)	3.01	2.70 (7)
400	3.36	5.37 (6)	4.17	2.39 (7)	10.5	7.47 (7)	11.9	1.16 (8)
800	14.1	2.20 (7)	17.2	9.62 (7)	42.5	2.99 (8)	48.0	4.30 (8)
1600	59.6	9.04 (7)	71.8	3.87 (8)	178	1.20 (9)	197	1.72 (9)

TABLE 4.2
Overhead due to higher accuracy for Example 2, see text for full details.

Table 4.2 compares four different algorithms. Time 1 and Ops 1 correspond to the computational time and number of operations for the standard method where full advantage has been taken of the uniform Cartesian mesh in the implementation (simplification of the quadratic equations (2.6), removal of trivial tests (upwinding)). If none of those simplifications are taken, the corresponding scheme requires more operations; corresponding computational time and number of operations are given by Time 2 and Ops 2. The results for the fully first and second order methods correspond respectively to Time 3 and Ops 3 and Time 4 and Ops 4. For this example, maintaining optimal accuracy more than doubles that computational time. The fully second order method is found to be only marginally slower than the fully first order one.

5. Algorithms comparison. The three examples from Section 4 were implemented. As the implementation on FS on a non-Cartesian grid is awkward and was not attempted, first order FM and FS were compared on Cartesian grids. Therefore in Table 5.1, both methods share the same (in)accuracy. As can be seen from that table, FM is faster in all but the simplest case. Note that each algorithm was run until "convergence". For FM, this means until exhaustion of the nodes in the "front", while for FS it means until a fixed point has been reached (no change after a sweep, see Lemma 3.1).

As argued in [19], FS has an asymptotic computational complexity of $\mathcal{O}(N)$ while FM's is $\mathcal{O}(N \log N)$. Eventually, FS will overcome FM even for nontrivial examples such as Example 3. However, looking at Table 5.1, this happens only on exceedingly

M	*Fast Marching (time)*	*ops*	*Fast Sweeping (time)*	*ops*
Example 1: no obstacle				
25	0.02	2.06(4)	0.01	3.15(4)
50	0.03	8.45(4)	0.04	1.25(5)
100	0.18	3.47(5)	0.14	5.01(5)
200	0.71	1.42(6)	0.61	2.00(6)
400	2.95	5.85(6)	2.46	8.00(6)
800	12.2	2.40(7)	9.89	3.20(7)
1600	52.5	9.86(7)	40.2	1.28(8)
Example 2: circular obstacle				
25	0.01	1.89(4)	0.02	4.22(4)
50	0.05	7.73(4)	0.09	2.82(5)
100	0.20	3.17(5)	0.48	1.36(6)
200	0.79	1.31(6)	1.85	5.46(6)
400	3.36	5.37(6)	7.39	2.19(7)
800	14.1	2.20(7)	29.6	8.75(7)
1600	59.6	9.04(7)	120	3.50(8)
Example 3: "rings"				
25	0.01	1.05(4)	0.02	4.29(4)
50	0.04	6.04(4)	0.16	4.76(5)
100	0.16	2.66(5)	0.78	2.43(6)
200	0.63	1.10(6)	3.26	1.08(7)
400	2.58	4.56(6)	13.4	4.39(7)
800	10.5	1.88(7)	67.1	2.26(8)
1600	44.3	7.72(7)	333	1.10(9)

TABLE 5.1
Runtime and operation count for first order FM and FS on a uniform cartesian mesh for Examples 1, 2 and 3 ($N = \mathcal{O}(M^2)$).

fine meshes, see the remarks at the end of this section. No attempt was made in Table 5.1 to weight the various operations according to their execution time (for instance, a swap is counted as one operation). Runtimes offer thus a more accurate picture of the complexity of each algorithm.

The results from Table 5.1 also show that the more complicated the domain is, the better FM performs with respect to FS. Indeed, while FM continually advances the wavefront, FS has to do another set of sweeps every time the direction of propagation changes. In the case of Example 3 for instance, that direction changes several times, see Figure 5.1. In Table 5.2, the runtime and operation counts for a modified Example 3 are reported: the number of partial rings was increased from zero to a total of five as displayed in Figure 5.1. It is observed that the runtime is roughly constant for FM as the complexity of the domain (number of rings) increases, while it significantly shoots up for FS (in fact, the runtime for FM even goes slightly down due to the decreasing number of nodes at which the solution is computed for that example).

FS as described in [16] for instance is proposed as an iterative method, i.e., the algorithm is stopped before completion according to a stopping criterion of the type $\|u^{k+1} - u^k\|_{L^1} < \delta = 10^{-10}$. Figure 5.2 illustrates the convergence history of the numerical solution towards the exact solution of the *discretized* problem. In each case

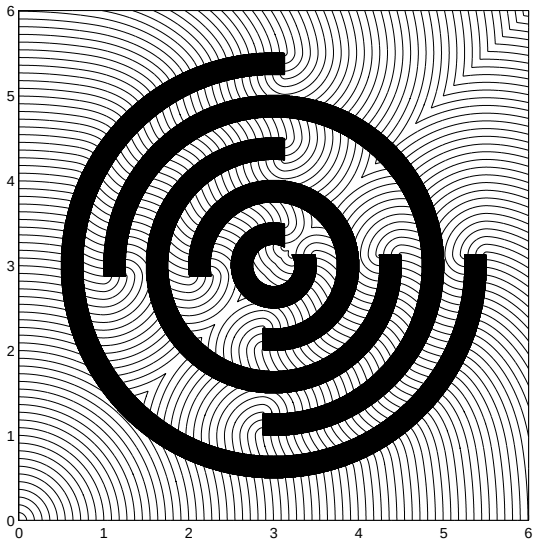

FIG. 5.1. Propagation around an obstacle: Example 3, FM with $M = 1600$. The source is located at $(0, 0)$.

Rings	*Fast Marching (time)*	*ops*	*Fast Sweeping (time)*	*ops*
0	54.8	9.86(7)	41.9	1.28(8)
1	53.8	9.27(7)	132	4.38(8)
2	49.7	8.67(7)	165	5.36(8)
3	46.9	8.18(7)	230	7.50(8)
4	44.9	7.86(7)	264	8.70(8)
5	44.3	7.72(7)	333	1.10(9)

TABLE 5.2
Runtime and operation count w/differing numbers of concentric partial rings ($M = 1600$)

in Figure 5.2, i.e., left: Example 2 and right: Example 3, the discrepancy falls down to zero if one additional step is taken. It is observed that unless δ is chosen quite large, the above stopping criterion will not prevent the algorithm from running until completion. Therefore, running FS until completion appears to be the best strategy and the algorithm has to be considered as a "direct" method.

Since FM has complexity of $\mathcal{O}(N \log_2 N)$ and FS has complexity of $\mathcal{O}(N)$, the latter will be faster on a fine enough mesh. Proceeding to a more detailed operation count, let x be the cost of doing one update of a single node. FM calculates each node one time for each upwind neighbor. Almost every node has two upwind neighbors, so the total operation count for FM is, roughly, at most $2Nx + N \log_2 N$. The $N \log_2 N$ term is an upper bound on the heap sort operations. Let m be the number of sweeps required for FS to complete. FS calculates every node in every sweep. Therefore, the total operation count for FS is mNx.

The number of nodes, N^*, above which FS is faster than FM can be found by

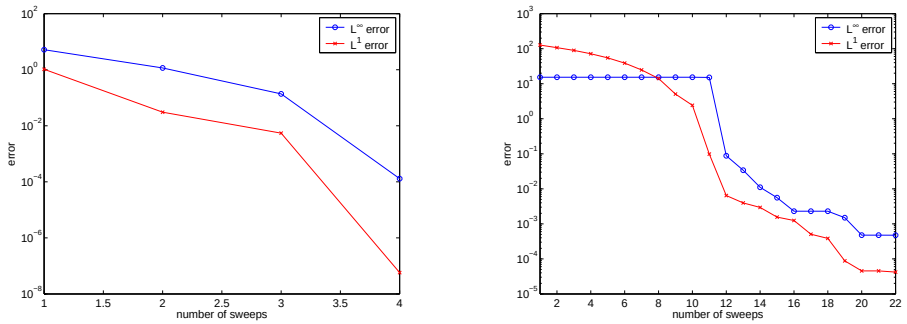

FIG. 5.2. L^1 and L^∞ norm of the difference between iterates after each sweep and the exact solution of the discretized problem with the Fast Sweeping method; left: circular obstacle (Example 2), right: rings (Example 3); $M = 1600$.

equating these two formulas. This results in

$$N^* = 2^{(m-2)x}.$$

In our implementation, solving the Eikonal equation requires approximately $x = 40$ floating point operations for the calculation of a single node. From this, we see that if FS requires only 2 sweeps to complete, it will be faster than FM for any sized mesh. However, if FS requires 3 or more sweeps to complete then $N^* > 2^{40}$. A mesh of this size being unreasonable, FM will be the faster of the two methods. It should be noted that any problem where the wave-front propagates into multiple quadrants will require at least 3 sweeps to complete.

6. Conclusion. We have brought to the fore three main points. First, ordered methods for the Eikonal equation such as Fast Marching and Fast Sweeping can be modified to retain optimal convergence rates (first or second order) even on low regularity problems. Second, the considered examples show that both methods should be considered as direct and not iterative algorithms. While obvious for Fast Marching, this is less clear for Fast Sweeping. In that case, a natural stopping criterion is proposed and justified. Finally, despite having a lower order of complexity, Fast Sweeping is observed to be slower than Fast Marching for the strongly nonuniform Eikonal problems considered here on all realistic grids.

Both methods can be relatively easily extended to more general convex Hamiltonians [15, 16]. Behaviors similar to those reported here are expected. The case of nonconvex Hamiltonians is more complex, even with respect to the construction of appropriate numerical fluxes. Very few schemes have been proposed (see for instance [8] where a Lax-Friedrichs sweeping method is introduced); more needs to be done to develop fast and accurate methods for those problems.

REFERENCES

[1] R. ABGRALL, *Numerical discretization of the first order Hamilton-Jacobi equations on triangular meshes*, Comm. Pure Appl. Math., 49 (1996), pp. 1339–1377.

[2] S.A. AHMED, R. BUCKINGHAM, P.A. GREMAUD, C.D. HAUCK, C.M. KUSTER, M. PRODANOVIC, T.A. ROYAL, V. SILANTYEV, *Volume determination for bulk materials in bunkers*, submitted to Int. J. for Num. Meth. in Eng., Center for Research in Scientific Computation, NCSU, Technical Report CRSC-TR03-24.

[3] J. Bentley, *Thanks, Heaps*, Com. ACM, 28 (1985), pp. 245–250.
[4] M. Boué, and P. Dupuis, *Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control*, SIAM J. Numer. Anal, 36 (1999), pp. 667–695.
[5] M. Crandall, L. Evans, and P.-L. Lions, *Some properties of viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 282 (1984), pp, 487–502.
[6] M. Crandall, and P.-L. Lions, *Two approximations of solutions of Hamilton-Jacobi equations*, Math. Comp., 43 (1984), pp. 1–19.
[7] C. Hu, and C.-W. Shu, *A discontinous Galerkin method for Hamilton-Jacobi equations*, SIAM J. Sci. Comput., 21 (1999), pp. 666-690.
[8] C.-Y. Kao, S. Osher, and J. Qian, *Lax-Friedrichs sweeping scheme for static Hamilton-Jacobi equations*, Preprint, Department of Mathematics, UCLA (2003).
[9] G. Kossioris, C. Makridakis, and P. Souganidis, *Finite volume schemes for Hamilton-Jacobi equations*, Numer. Math., 83 (1999), pp. 427–442.
[10] S. Osher, and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton-Jacobi*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.
[11] J. Qian, and W.W Symes, *An adaptive finite difference method for traveltimes and amplitudes*, Geophysics, 67 (2002), pp. 167–176.
[12] E. Rouy, and A. Tourin, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal., 29 (1992), pp. 867–884.
[13] J.A Sethian, *Fast marching methods*, SIAM Review, 41 (1999), pp. 199–235.
[14] 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), pp. 5699–5703.
[15] J.A. Sethian, and A. Vladimirsky, *Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms*, SIAM J. Numer. Anal., 41 (2003), pp. 325–363.
[16] Y.H.R. Tsai, L.T. Cheng, S. Osher, and H.K. Zhao, *Fast sweeping algorithms for a class of Hamilton-Jacobi problems*, SIAM J. Numer. Anal., 41 (2003), pp. 659-672.
[17] J.N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control, 40 (1995), pp. 1528–1538.
[18] J.W.J Williams, *Heapsort*, Com. ACM, 7 (1964), pp. 347–348.
[19] H.K. Zhao, *Fast Sweeping Method for Eikonal Equations*, to appear in Math. Comp.