A Characteristics Driven Fast Marching Method for the Eikonal Equation

Emiliano Cristiani and Maurizio Falcone

Abstract We introduce a new Fast Marching method for the eikonal equation. The method is based on the informations driven by characteristics and it is an improvement with respect to the standard Fast Marching method since it accepts more than one node at every iteration using a dynamic condition. We analyze the method and present several tests on fronts evolving in the normal direction with variable velocities including cases where a change in topology occurs.

1 Introduction

The success of the level set method in the analysis and in the simulation of complex interface problems is mainly due to its capability to handle the changes in topology of the interfaces. The price to pay is the fact that we look for a function $u : \mathbb{R}^n \times [0,T] \to \mathbb{R}$ and we locate the front Γ_t at time t just considering the 0-level set of $u(x,t)$, so we add one extra dimension to the original problem which lives in $\mathbb{R}^n$.
The Fast Marching (FM) method has been proposed to cut down the computational complexity of the level set method (see [7, 9] for the origin of the method and [1, 2, 8] for some recent developments). It was developed for the time-independent eikonal equation

$$\begin{cases} c(x)|\nabla T(x)| = 1, & x \in \mathbb{R}^n \backslash \Omega_0 \\ T(x) = 0, & x \in \Gamma_0 = \partial\Omega_0 \end{cases} \tag{1}$$

where Ω_0 is a closed bounded set and $c(x)$ is Lipschitz continuous and strictly positive. The front is recovered as the t-level set of the minimal time function $T(x)$

Emiliano Cristiani
ENSTA, 32, Boulevard Victor, Paris, France, e-mail: emiliano.cristiani@ensta.fr

Maurizio Falcone
Dipartimento di Matematica, SAPIENZA - Università di Roma, e-mail: falcone@mat.uniroma1.it

which is the unique viscosity solution of (1). Note that the solution u of the level set method can be written as $u(x,t) = T(x) - t$, so (1) is related both to the evolution of a front with velocity $c(x)$ in the normal direction and to the minimum time problem (see [3, 6]).

The FM method sets-up the computation in a narrow band near the front and updates the narrow band in order to follow the evolution of the front at every time. In this way it eliminates the extra dimension introduced by the model since at every iteration the computation is performed in a neighborhood of the front, *i.e.* we work on $O(\sqrt{N})$ nodes if the grid has N nodes. Once a node exits the narrow band it is accepted, *i.e.* it is no more computed in the following iterations. The FM method accepts only one node at every iteration (the node with the minimal value T) to guarantee that the evolution of the front is tracked correctly. However, in several situations one has the impression that the FM method can be improved and that it is not needed to accept just one point at every iteration. For example, this is the case when the initial configuration Ω_0 is convex so that in the normal evolution of the front there are neither a merging nor a crossing of characteristics. The main contribution here is to modify the FM method based on the semi-Lagrangian approximation (see [2, 3] for details) developing a new algorithm which accepts several points provided some local conditions are satisfied (see Section 2). In this way we drop the search for the minimum value *at every iteration* via a Min-Heap structure. We note that another Group Marching algorithm has been proposed by Kim in [5] to speed up the standard FM method based on the finite difference scheme. Our algorithm is different in two respects. The first is that we use a different local rule for the computation whereas the second is related to the condition which allows our method to accept more nodes.

2 The Characteristics Fast Marching method

In this section we present the algorithm and some considerations about its computational cost. Note that the basic algorithm is developed in order to deal with the evolution of a *single* front although it works often for several merging fronts. An extension to the case of general merging front is also given. Via the Kružkov tranform $v(x) = 1 - \mathrm{e}^{-T(x)}$ the equation (1) can be rewritten in the fixed point form $v(x) = F[v(x)] := \min_{a\in B(0,1)}\{c(x)a\cdot\nabla v(x)\} + 1$ where $B(0,1)$ is the unit ball centered in 0. Let us just recall the basic semi-Lagrangian scheme we will use as local rule for the computation. The value of v at the node x_i will be denoted by v_i.

$$\begin{cases} v_i = \min\limits_{a\in B(0,1)} \{\mathrm{e}^{-h} v(x_i - hc(x_i)a) + 1 - \mathrm{e}^{-h}, & x_i \in \mathbb{R}^n \backslash \Omega_0 \\ v_i = 0, & x_i \in \partial\Omega_0 \end{cases} \tag{2}$$

We chose a variable (fictitious) time step $h = h_i$ such that $c(x_i)h_i = \Delta x$ and we use a linear interpolation to compute $v(x_i - h_i c(x_i)a)$.

2.1 Main idea

In several situations the accept-the-node-with-minimum-value condition of FM method appears to be too restrictive, particularly when $c(x) \equiv c_0$ and Ω_0 is a small ball. In fact, in this case one can accept almost all the nodes in the narrow band at the same time without loosing informations.
In the Characteristics Fast Marching (CFM) method the point of view is reversed with respect to the FM method. Instead of declaring accepted the node with the minimum value in the narrow band (*i.e.* the first node which will be reached by the front in the next iteration), we look for the node in the narrow band with the maximal velocity c_{max}, *i.e.* the node from which we can cover the distance Δx and enter the accepted zone in the minimum time. Once we have c_{max} we can compute the time step $\Delta t = \frac{\Delta x}{c_{max}}$ for that iteration which is the time needed by the fastest node to reach the accepted zone. While *all* the nodes with the maximal velocity c_{max} reach the accepted zone the other nodes in the narrow band come closer to the accepted zone without touching it. In order to take into account this displacement (which is smaller than Δx), we introduce the *local time* t_i^{loc}. The local time is set to 0 when the node i enters the narrow band and it is increased at each iteration by the (variable) increment Δt until the node is accepted. At each iteration we label as accepted all the nodes having a local time t_i^{loc} large enough to safisfy two conditions: they reach the accepted zone moving at speed $c(x_i)$ *and* they are computed by (2) just using nodes in the accepted region.

2.2 The CFM algorithm for a single front

Let us introduce the algorithm. In the following, the set of the nodes belonging to the narrow band will be denoted by *NB*.

Initialization

1. The nodes belonging to the initial front Γ_0 are located and labeled as *accepted*. They form the set $\widetilde{\Gamma}_0$. The value of v of these nodes is set to 0.
2. *NB* is defined as the set of the neighbors of $\widetilde{\Gamma}_0$, external to Γ_0.
3. Set $t_i^{loc} := 0$ for any $i \in NB$.
4. The remaining nodes are labeled as *far*, their value is set to 1 (corresponding to $T = +\infty$).

Main Cycle

1. Compute $c_{max} = \max\{c_i \ : \ i \in NB\}$ and set $\Delta t := \Delta x / c_{max}$.
2. For any $i \in NB$:
   a. Update the local time: $t_i^{loc} := t_i^{loc} + \Delta t$.
   b. If $t_i^{loc} \cdot c_i \geq \Delta x$ then
      i. Compute v_i by the scheme (2).

ii. Check if v_i is computed using only *accepted* nodes. If yes, set `flag`=true, else set `flag`=false.
iii. If `flag`=true, then
- Label i as *accepted* and remove i from *NB*.
- Define *FN* as the set of the *far* neighbors of i. Include *FN* in *NB* and set $t_k^{loc} := 0$ for any $k \in FN$.

3. If not all nodes are *accepted* go back to 1.

2.3 Front Merging

Let us extend the previous algorithm to the more interesting situation when more fronts are merging together. This a delicate point because we risk to accept too many points before they reach the correct value. One of the main differences with respect to the FM method is that the narrow band does not follow exactly the level sets of the solution. For example, if $c(x)$ is constant, the narrow band of a rectangular front evolves without smoothing corners contrary to the real level sets of the exact solution. In computing the evolution of a single front this is not a major difficulty, since the correctness of the solution is guaranteed by the fact that we accept *only* those nodes computed by other already accepted nodes. However, the evolution of m fronts merging together can be difficult to follow in some cases (but not in all cases). In fact, a single node can be reached first by the narrow band corresponding to one on those fronts and it is accepted according to these informations whereas it should be accepted according to the informations driven by another front. In Fig. 1 we can see a front merging where this situation occurs and produces a wrong so-

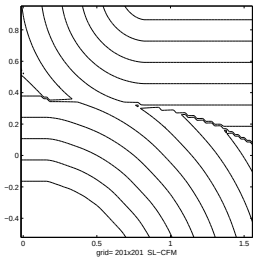

Fig. 1 The merging zone in the case the CFM method fails

lution in the merging zone. Note that this difficulty can not be solved reducing the space step Δx and it seems that there is no way to solve the problem without making some important changes to the algorithm.
In order to deal with m merging fronts we start considering we can compute by CFM method the m value functions $v^{(k)}$, $k = 1,\dots,m$ corresponding to the evolutions of the m fronts on m copies of the domain and than take the minimum $v := \min\{v^{(1)},\dots,v^{(m)}\}$ as final solution. Of course this choice leads to multiply

by m the time required for computation (we do not consider here problems related to memory) and then it is not efficient. To overcome this problem we use the following strategy. When the k-th CFM method accepts the node i with value $v_i^{(k)}$, it checks if another CFM method has already computed a value $\tilde{v}_i < v_i^{(k)}$ for the same node. If this is the case, the k-th CFM method avoids to enlarge the narrow band from the node i. This procedure leads to a very small overlapping zone between fronts and a CPU time comparable with the CFM method for a single front (see Fig. 2).

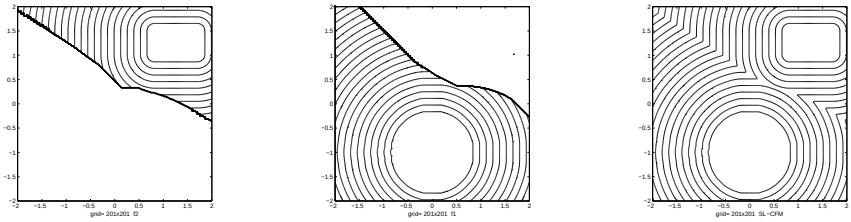

Fig. 2 Merging via two applications of CFM method

Computational cost
The computational cost of the CFM method is more complicated to determine with respect to the FM method. This is mainly due to the fact that it is very difficult to say *a priori* how many times the `flag` remains fixed to false for each node. By the experiments it seems that this happens when the velocity field $c(x)$ increases along characteristics.
Searching for the maximal velocity c_{max} in the narrow band costs $O(\ln N_{nb})$ where N_{nb} is the number of nodes in the narrow band (bounded by N but expected to be of order $\sqrt{N}$).
When the velocity is constant all the nodes of the narrow band are accepted at the same time, then the number of times we need to search for c_{max} in the narrow band is divided by a factor N_{nb} with respect to the FM method and this is surely the most powerful feature of the CFM method.
In conclusion, for a constant velocity $c(x) \equiv c_0$ we expect the computational cost to be of order $O((N \ln N_{nb})/N_{nb}) = O(\sqrt{N} \ln \sqrt{N})$.

3 Numerical results

In this section we present some numerical tests in order to compare the CFM method with the standard iterative semi-Lagrangian (SL) scheme and the FM method based on the semi-Lagrangian scheme (FM-SL) introduced in [2]. The domain of compu-

tation is $[-2,2]^2$. We used MATLAB 7.0 on a Processor Intel dual core 2x2.80 GHz with 1 GB RAM.

Test 1: constant velocity. $\Gamma_0 = (0,0)$. $c(x,y) \equiv 1$. Solution: $T(x,y) = \sqrt{x^2+y^2}$.

Test 2: velocity depending on x. $\Gamma_0 = (0,0)$. $\quad c(x,y) = x+3$.

Test 3: velocity depending on x and y. $\Gamma_0 = (0,0)$. $\quad c(x,y) = |x+y|$.

Test 4: merging of two fronts. $\Gamma_0 =$ a rectangle and a circle. $\quad c(x,y) \equiv 1$.

Table 1 Errors and CPU time for Test 1

method	Δx	L^∞ error	L^1 error	CPU time (sec)
CFM	0.08	0.0329	0.3757	0.27
FM-SL	0.08	0.0329	0.3757	0.58
SL (46 it)	0.08	0.0329	0.3757	9.7
CFM	0.04	0.0204	0.2340	1.14
FM-SL	0.04	0.0204	0.2340	2.44
SL (86 it)	0.04	0.0204	0.2340	70.95
CFM	0.02	0.0122	0.1406	4.9
FM-SL	0.02	0.0122	0.1406	10.56
SL (162 it)	0.02	0.0122	0.1406	530.56

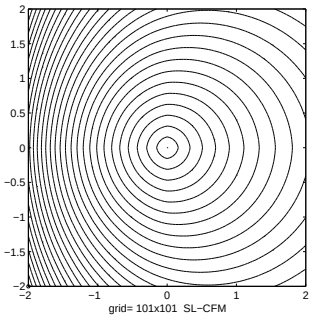

Fig. 3 Numerical result for Test 2

Comments

By Test 1 we can see that the semi-Lagrangian iterative method is much slower than both Fast Marching methods although all methods compute exactly the same approximation of the viscosity solution of equation (1).

Table 2 CPU time for Test 2

method	Δx	CPU time (sec)
CFM	0.04	1.19
FM-SL	0.04	2.34
CFM	0.02	5.20
FM-SL	0.02	10.44

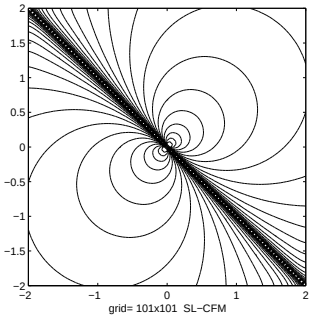

Fig. 4 Numerical result for Test 3

Table 3 CPU time for Test 3

method	Δx	CPU time (sec)
CFM	0.04	3.5
FM-SL	0.04	2.5
CFM	0.02	16.73
FM-SL	0.02	11.59

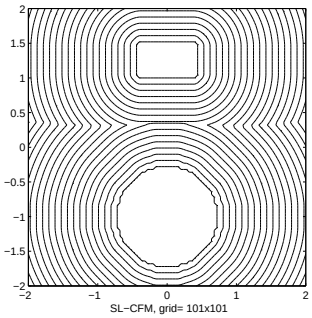

Fig. 5 Numerical result for Test 4

Table 4 CPU time for Test 4

method	Δx	CPU time (sec)
CFM	0.04	1.05
FM-SL	0.04	2.12
CFM	0.02	5.08
FM-SL	0.02	9.53

As expected, the CPU time for CFM method is smaller than that of FM method when the velocity field is constant or have relatively small variations in the domain (Test 1 and 2).
When the function $c(x)$ is increasing along characteristics (Test 3) the CFM method is slower than the FM method, this is due to the fact that the `flag` is often false after the step (ii) of the algorithm so that a lot of nodes in the narrow band are computed but only few of them are accepted. Note that in the FM method not all the nodes in the narrow band are computed at each iterations but only the new entries.
In Test 4 the evolution of two merging fronts is computed without any modification of the basic algorithm and again the CPU time for the CFM method is about the half of the time needed by the FM method.

References

1. Cristiani, E.: Fast Marching and semi-Lagrangian methods for Hamilton-Jacobi equations with applications. Ph.D. thesis, Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate, SAPIENZA - Università di Roma, Rome, Italy, February 2007
2. Cristiani, E., Falcone, M.: Fast semi-Lagrangian schemes for the Eikonal equation and applications. SIAM J. Numer. Anal., **45** (2007), pp. 1979–2011
3. Falcone, M.: The minimum time problem and its applications to front propagation. In A. Visintin e G. Buttazzo (eds.), "Motion by mean curvature and related topics", De Gruyter Verlag, Berlino, 1994
4. Falcone, M., Giorgi, T., Loreti, P.: Level sets of viscosity solution: some applications to fronts and rendez-vous problems. SIAM J. Appl. Math., **54** (1994), pp. 1335–1354
5. Kim, S.: An $O(N)$ level set method for eikonal equations. SIAM J. Sci. Comput., **22** (2001), pp. 2178–2193
6. Sethian, J. A.: Level set methods and fast marching methods. Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. Cambridge University Press, 1999
7. Sethian, J. A.: A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci. USA, **93** (1996), pp. 1591–1595
8. Sethian, J. A., Vladimirsky, A.: Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms. SIAM J. Numer. Anal., **41** (2003), pp. 325–363
9. Tsitsiklis, J. N.: Efficient algorithms for globally optimal trajectories. IEEE Tran. Automatic. Control, **40** (1995), pp. 1528–1538