A Fast Marching Method for Hamilton-Jacobi Equations Modeling Monotone Front Propagations

Emiliano Cristiani

A Fast Marching Method for Hamilton-Jacobi Equations Modeling Monotone Front Propagations*

Emiliano Cristiani†

February 17, 2008

Abstract

In this paper we present a generalization of the Fast Marching method introduced by J. A. Sethian in 1996 to solve numerically the eikonal equation. The new method, named Buffered Fast Marching (BFM), is based on a semi-Lagrangian discretization and is suitable for Hamilton-Jacobi equations modeling monotonically advancing fronts, including Hamilton-Jacobi-Bellman and Hamilton-Jacobi-Isaacs equations which arise in the framework of optimal control problems and differential games. We also show the convergence of the algorithm to the viscosity solution. Finally we present several numerical tests proving that the BFM method is accurate and faster than the classical iterative algorithm in which every node of the grid is computed at every iteration.

Keywords Fast Marching methods, front propagation, semi-Lagrangian schemes, Hamilton-Jacobi equations, optimal control problems.

AMS Primary, 65N12; Secondary, 49L20.

*This research was partially supported by the MIUR Project 2006 "Modellistica Numerica per il Calcolo Scientifico ed Applicazioni Avanzate" and by INRIA–Futurs and ENSTA, Paris, France.

†ENSTA, Paris (France). E-mail: emiliano.cristiani@ensta.fr, cristian@mat.uniroma1.it. Corresponding address: via Nazario Sauro 21A, 00012 Villanova di Guidonia (RM), Italy. Tel. +390774527776, Mobile: +393492639591

1 Introduction

The Fast Marching (FM) method is a numerical method for the eikonal equation

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

where Ω_0 is a closed set and $c : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous and strictly positive. This equation appears in front propagation problems in which the interface propagates in normal direction with speed $c(x)$, more precisely the t-level set of its viscosity solution T is the interface at time t. The interface at time $t = 0$ is given by $\Gamma_0 = \partial\Omega_0$. The FM method is officially born with the paper of Sethian [22] in 1996 (see also his book [21]). Before that, Tsitsiklis [26] already proposed a slightly different Dijkstra-like algorithm based on a control-theoretic discretization which contains all the basic ideas of the FM technique. The method is very powerful because it is able to compute the viscosity solution of (1) much faster than any other iterative algorithms in which every node of the grid is computed at every iteration. Its computational cost is $O(N \ln N)$ where N is the total number of grid nodes. Since its first appearance, it was applied in many fields like mesh generation, seismology, geodesic computation, image and video segmentation, image enhancement, dislocation dynamics and so on. The original FM method is based on the following *up-wind* first-order finite difference approximation (we choose $n = 2$ to avoid cumbersome notations)

$$\begin{aligned} (\max\{\max\{D_x^-, 0\}, -\min\{D_x^+, 0\}\})^2 + \\ +(\max\{\max\{D_y^-, 0\}, -\min\{D_y^+, 0\}\})^2 = c_{i,j}^{-2} \end{aligned} \tag{2}$$

where $D_x^- = \frac{T_{i,j} - T_{i-1,j}}{\Delta x}$, $D_x^+ = \frac{T_{i+1,j} - T_{i,j}}{\Delta x}$ (and analogous definition for D_y^+ and D_y^+) and $T_{i,j} = T(i\Delta x, j\Delta y)$ as usual. The FM technique consists in computing the values at the nodes in a special order such that convergence is reached in just one iteration. At a generic step of the algorithm the grid nodes are divided in three sets, *accepted*, *narrow band* and *far* nodes. The accepted nodes are those where the solution has been already computed and where the value can not change in the following iterations. The narrow band nodes are the nodes where the computation actually takes place and their value can still change at the following iterations. Finally, the far nodes are the remaining nodes where an approximate solution has not been computed yet. In physical terms, the far nodes are those in the space region which has not been touched by the front yet, the accepted nodes are those where the front has already passed through and the narrow band nodes are, iteration by iteration, those lying in a neighborhood of the front. The crucial point is how the nodes in the narrow band are chosen to become accepted. This condition must guarantee that the value of those nodes can

not change in following iterations. In the classical FM method the criterion is *picking the node (only one at a time) with the minimal value.*
In the last decade many authors tried to improve the FM method in both velocity and accuracy. Other papers were devoted to the extension of the FM method to more general equation. This is probably the most difficult task since the FM method strictly relies on some particular properties of the eikonal equation as we will see in section 2.2. Kimmel and Sethian [17] extended the FM method to triangulated domains on manifolds preserving the same computational complexity (see also [24]). Again Kimmel and Sethian [18] extended the FM method to an equation of the form (1) in which c depends on x and T itself. They apply their result to the solution of the Shape from Shading problem.
Sethian and Vladimirsky [23] extended the FM method to equation of the form (1) in which c depends on x and $\nabla T/|\nabla T|$ on unstructured grids. This equation includes the case of the anisotropic front propagation problem. The authors explain in detail the limitations of the classical FM technique and how they can be overcome. Unfortunately they did not perform many numerical tests and did not present CPU times needed for computations.
Prados [19] proposed an interesting generalization of the FM method to solve Hamilton-Jacobi-Bellman equations. The new method changes the way a node is accepted, it is no more the node with the minimal value T in the narrow band but it is the node with the minimal value $T - \phi$ where ϕ is a viscosity subsolution of the equation. Of course this can not be considered a real numerical method because a subsolution must be known, nevertheless this procedure can be a useful suggestion for further developments.
The papers [15, 14] made a comparative study of FM method and other existing methods for the eikonal equation, in particular with the Fast Sweeping method (see [25, 20] and references therein) which can overcome FM method in some situations. Carlini et al. [3, 4] extended the FM method to eikonal-type equations modeling non-monotone front propagation problems in which the velocity c of the front can change sign in space and/or in time. Moreover the authors apply their results to the simulation of the dislocation dynamics in which the velocity c does not depend locally on a point x but is given in an integral form.
Vladimirsky [27] deals with equations of the form (1) in which c depends on x, $\nabla T(x)$ and T. He presents a rigorous analysis and some numerical experiments.
The author and Falcone [8] introduced the Characteristic FM method for the eikonal equation which, similarly to Kim [16], accepts more than one node at the same time and it is faster than the FM method in most cases.
Again the author and Falcone [7] (see also [5]) introduced the FM method based on the semi-Lagrangian discretization in the framework of control

theory and minimum time problem. The semi-Lagrangian scheme is proved to be more accurate than the finite difference scheme classically used in FM method although they are both first order schemes. The semi-Lagrangian scheme will be detailed in section 2.1 and will be used also for the new method proposed in this paper. In [7] it is also shown that the finite difference discretization (2) must be implemented carefully. If the numerical scheme is solved explicitly after evaluating the min's and max's using all the available values on the grid then the resulting second order equation in $T_{i,j}$ can have imaginary solutions. This can be avoided using for computation only already accepted nodes, changing the scheme when the case occurs as in [18], or choosing

$$\Delta x \leq (\sqrt{2}-1)\frac{c_{min}}{L_c}$$

where $c_{min} = \min_{x \in \mathbb{R}^n \setminus \Omega_0} c(x)$ and L_c is the Lipschitz constant of c as proved in [7].

In this paper we introduce a new FM method based on a semi-Lagrangian discretization which is able to compute an approximate solution of Hamilton-Jacobi equations modeling front propagation problems in which the front does nor pass more than one time on the same point. This class on equations includes Hamilton-Jacobi-Bellman and Hamilton-Jacobi-Isaacs equations which arise in the framework of optimal control problems and differential games. The new method, named *Buffered Fast Marching*, uses a forth set (the buffer) in addition to the sets accepted, narrow band and far to menage the nodes. The buffer is in the middle between the narrow band and the accepted zone and gathers the nodes until they can be accepted once and for all. The size of the buffer depends on the anisotropy of the problem and shrinks to zero for the eikonal equation (1).

The paper is organized as follows. In section 2 we recall the FM method for the eikonal equation based on the semi-Lagrangian discretization introduced in [7]. We show why the FM technique does not work for more general equations. In section 3 we introduce the Buffered Fast Marching (BFM) method, detailing the algorithm and its properties. We also specify the equations the BFM works with. Finally in section 4 we present some numerical tests on a series of benchmarks commenting the general behavior of the solutions and CPU times.

2 The semi-Lagrangian Fast Marching method and its limitations

In this section we recall the FM method based on the semi-Lagrangian discretization introduced in [7]. It will be the foundation of the Buffered FM method proposed here. We also explain in details the limitations of the FM method.

2.1 The semi-Lagrangian Fast Marching method

In [7] it was introduced the FM method based on the semi-Lagrangian discretization and it was proved that the new algorithm is slightly slower than the original FM method but much more accurate.
As shown in [12], by the change of variable (Kružkov transform)

$$v(x) = 1 - e^{-T(x)} \tag{3}$$

we can transform (1) in the following equation

$$\begin{cases} v(x) + \max\limits_{a \in B(0,1)} \{-c(x)a \cdot \nabla v(x)\} = 1 & \text{for } x \in \mathbb{R}^n \backslash \Omega_0 \\ v(x) = 0 & \text{for } x \in \partial\Omega_0 \end{cases} \tag{4}$$

where $B(0,1)$ is the unit ball centered in 0. Note that v is always in the interval $[0,1]$ while T is in general unbounded. We will consider for simplicity a structured grid G denoting its nodes by $x_i\,,\ i = 1 \ldots, N$, *i.e.* $G = \{x_i,\ i = 1, \ldots, N\}$. It was proved in [2] that the numerical scheme stems from a discrete version of Dynamic Programming Principle (see f.e. [1]), this leads to the equation

$$\begin{cases} w(x_i) = \min\limits_{a \in B(0,1)} \{\beta w(x_i - hc(x_i)a)\} + 1 - \beta & \text{for } x_i \in G \backslash \Omega_0 \\ w(x_i) = 0 & \text{for } x_i \in G \cap \Omega_0 \end{cases} \tag{5}$$

where w is an approximation of v, $\beta = e^{-h}$, h is a discretization step (it can be interpreted as a time step used to integrate the equation along characteristics) and we defined $w = 0$ also in the internal nodes of Γ_0. We use a linear interpolation to approximate the value $w(x_i - hc(x_i)a)$ using the values at the three nearest grid nodes. It has been shown in [10] that equation (5) has a unique solution w in the class of piecewise linear functions defined on the grid. It can be computed by a fixed point technique, iterating until convergence

$$w^{(k+1)} = \Lambda(w^{(k)}) \qquad k = 0, 1, 2, \ldots \tag{6}$$

where $\Lambda(w)_i = \min_{a \in B(0,1)} \{\beta w(x_i - hc(x_i)a)\} + 1 - \beta$ and $w^{(0)}$ is equal to 0 in $G \cap \Omega_0$ and 1 elsewhere. The idea which is behind the semi-Lagrangian

FM method is rather simple: we follow the initialization and all the steps of the classical FM method but the step where the value at the node x_i is actually computed where we use the semi-Lagrangian scheme instead of the finite difference scheme. The main difference is in the stencil. The finite difference scheme needs the four nearest nodes to x_i (N,S,E,W directions) to compute $w(x_i)$, while the semi-Lagrangian scheme needs the eight nodes around x_i provided h is chosen such that

$$h = h(x_i) = \Delta x / c(x_i). \tag{7}$$

This difference implies that the narrow band must include the diagonal neighbors of the accepted region so that the narrow band is slightly larger using the semi-Lagrangian scheme.

2.2 Limitations of the FM technique

As we said in the introduction, in the last ten years the extensions of the FM method to more general equations were quite timid and limited to equations very similar to (1). This is due to the fact that the method strictly relies on its physical interpretation based on the isotropic front propagation problem. From the mathematical point of view, it appears that labeling as accepted the node in the narrow band with the minimal value is suitable only in the case the characteristics curves of the equation coincide with the gradient lines of its solution. This is due to the fact that accepting the minimal value in the narrow band means to compute v (or T) in the ascending order and then to maintain the right up-winding only in the case the optimal control $a^*(x) := \arg \max_{a \in B(0,1)} \{-c(x)a \cdot \nabla v(x)\}$ satisfies

$$a^*(x) = -\frac{\nabla v(x)}{|\nabla v(x)|}.$$

This is the case for the eikonal equation but it is not the case if we substitute the velocity field $c(x)a$ with a function $\hat{c}(x,a)a$ (anisotropic front propagation) or a generic function $f(x,a)$. For a generic function $f(x,a)$ it was proved in [27] that if ω is the angle between the characteristics and the gradient lines at point x then

$$\cos(\omega) \geq \frac{1}{\Upsilon(x)}, \quad \text{where} \quad \Upsilon(x) = \frac{\max_a f(x,a)}{\min_a f(x,a)}.$$

In [23] the authors consider the eikonal equation $|\nabla d(x,y)| = 1$ complemented by Dirichlet boundary condition $d(0,0) = 0$ in a plane $z = \alpha x + \beta y$ for some vector $(\alpha, \beta) \neq (0,0)$. The level sets of d (*i.e.* the front) will be just the circles around the origin in that plane. Although the problem is

set in $\mathbb{R}^3$ it can be reduced to a two-dimensional problem considering the projection of the front onto the underlying $x-y$ plane and then solving a modified front propagation problem in the $x-y$ plane. The speed of the front in the projected problem is given by

$$\hat{c}(x,y,a_1,a_2) = \frac{1}{\sqrt{1+(\alpha a_1+\beta a_2)^2}}\,, \qquad (a_1,a_2)\in B(0,1).$$

The function $T(x,y)$ whose level sets are the front in the projected problem is

$$T(x,y) = \sqrt{(1+\alpha^2)x^2+(1+\beta^2)y^2+2\alpha\beta xy}. \tag{8}$$

In Fig. 3 (section 4) we can see the difference between the exact solution and the solution computed by the FM method on a regular grid (see also [23] where a similar result is obtained). It is easy to see that in this case the characteristics lines are straight lines to the origin so that they clearly not coincide with the gradient lines. As stated in the Criterion 5.1 of [23], it can be seen that the FM method fails exactly where characteristics and gradient lines lie in different simplices but it is able to compute the right solution elsewhere, even if the two directions does not coincide.

3 The Buffered Fast Marching method

In this section we first present the equation the BFM method is designed for and then we present the method in detail. Finally we present a convergence result and some considerations about the computational cost.

3.1 Related equations

A front propagation problem consists in recovering the position of a front $\Gamma_t : \mathbb{R}^+ \to \mathbb{R}^n$ (for example the interface between two layers) at any time t starting from an initial configuration Γ_0. We denote by Ω_t the region inside the front Γ_t. One of the most popular method to face this kind of problem is the level set method [21] in which we look for a function $u : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}$ such that $\Gamma_t = \{x \in \mathbb{R}^n : u(x,t) = 0\}$. It is well known that the function u is solution of the following PDE

$$u_t(x,t) + \phi(x,u(x,t),\nabla u(x,t))\cdot\nabla u(x,t) = 0\,, \quad x\in\mathbb{R}^n\,, t>0 \tag{9}$$

where ϕ is the velocity of the front (note that it can also depend on $u(\cdot)$ all over the domain and on higher order derivatives as well) and the initial condition $u(x,0)$ is chosen as the signed distance function from Γ_0. If the velocity field ϕ is such that the front did not pass for any point x more than one time the evolution is said to be *monotone*, *i.e.*

$$\Omega_{t_1} \subset \Omega_{t_2} \quad \text{for any} \quad t_1 < t_2. \tag{10}$$

If (10) is satisfied, it is proved in [21] that the front can be recovered by $\Gamma_t = \{x \in \mathbb{R}^n : T(x) = t\}$ where T is the viscosity solution of the following time-independent equation

$$\phi(x, T, \nabla T) \cdot \nabla T(x) = 1\,, \qquad x \in \mathbb{R}^n \backslash \Omega_0. \tag{11}$$

Note that if the direction of the velocity is normal to the interface the function ϕ has the form $\phi(x, T, \nabla T) = c(x)\frac{\nabla T}{|\nabla T|}$, so equation (11) can be written as $c(x)|\nabla T(x)| = 1$ (eikonal equation).
By the Kružkov transform (3), the equation (11) becomes

$$v(x) + \phi(x, v, \nabla v) \cdot \nabla v(x) - 1 = 0\,, \qquad x \in \mathbb{R}^n \backslash \Omega_0. \tag{12}$$

This equation is very general and is found in many applications, for example in the minimum time problem as follows. Let us consider the controlled nonlinear dynamical system

$$\begin{cases} \dot{y}(t) = f(y(t), a(t))\,, & t > 0 \\ y(0) = x \end{cases} \tag{13}$$

where $y(t)$ is the state of the system, $a(\cdot) \in \mathcal{A}$ is the control of the player, $\mathcal{A}$ being the set of admissible controls defined as

$$\mathcal{A} = \{a(\cdot) : [0, +\infty) \to A,\ \text{measurable}\},$$

and A is a given compact set of $\mathbb{R}^m$. Assume hereafter $f : \mathbb{R}^n \times A \to \mathbb{R}^n$ is continuous in both variables and Lipschitz continuous with respect to y uniformly in a. The unique trajectory solution of (13) will be denoted by $y_x(t; a(\cdot))$. In the minimum time problem the final goal is to find an optimal control $a^*(t)$ such that the corresponding trajectory $y_x(t; a^*(\cdot))$ minimizes over all admissible trajectories the time needed by the system to reach a given closed *target* $\mathcal{T} \subset \mathbb{R}^n$. The optimal control $a^*(t)$ can be computed by means of the *value function* defined as

$$T(x) := \begin{cases} \inf\limits_{a(\cdot)\in\mathcal{A}} \min\{t : y_x(t; a(\cdot)) \in \mathcal{T}\} & \text{if } y_x(t; a(\cdot)) \in \mathcal{T} \text{ for some } t \geq 0 \\ +\infty & \text{if } y_x(t; a(\cdot)) \notin \mathcal{T} \text{ for all } t \geq 0. \end{cases} \tag{14}$$

We will refer to T also as the minimum time function and we set $T = 0$ on $\mathcal{T}$. By the Dynamic Programming Principle it can be shown (see f.e. [1]) that $v = 1 - e^{-T}$ is the viscosity solution of

$$\begin{cases} v(x) + \max\limits_{a\in A}\{-f(x, a) \cdot \nabla v(x)\} - 1 = 0 & x \in \mathbb{R}^n \backslash \mathcal{T} \\ v(x) = 0 & x \in \partial T. \end{cases} \tag{15}$$

Equation (15) is known as the Hamilton-Jacobi-Bellman equation for the minimum time problem. Finally note that defining

$$a_*(x, \nabla v(x)) := \arg\max_{a \in A} \{-f(x,a) \cdot \nabla v(x)\},$$

$$\phi(x, \nabla v(x)) := -f(x, a_*(x, \nabla v(x))) \quad \text{and} \quad \Omega_0 := \mathcal{T}$$

the equation (15) takes the general form (12).
A semi-Lagrangian scheme for the equation (15) can be recovered as shown before for the eikonal equation, it reads

$$\begin{cases} w(x_i) = \min_{a \in A} \{\beta w(x_i - h f(x_i, a))\} + 1 - \beta & \text{for } x_i \in G \backslash \mathcal{T} \\ w(x_i) = 0 & \text{for } x_i \in G \cap \mathcal{T}. \end{cases} \tag{16}$$

As before, we want to use just the three nearest nodes to x_i to compute $w(x_i - hf(x_i, a))$ by linear interpolation so, similarly to (7), we choose $h = h(x_i, a) = \Delta x / |f(x_i, a)|$. Of course the constant $\beta = e^{-h}$ must be included in the minimum over a's. Note that the dependence of h on a does not produce any kind of oscillation or instability in any case.

The BFM method can also compute an approximate solution of the Hamilton-Jacobi-Isaacs equation which arises in the framework of differential games

$$\begin{cases} v(x) + \min_{b \in B} \max_{a \in A} \big\{ -f(x,a,b) \cdot \nabla v(x) \big\} - 1 = 0 & x \in \mathbb{R}^n \backslash \mathcal{T} \\ v(x) = 0 & x \in \partial \mathcal{T}. \end{cases} \tag{17}$$

Here f is the dynamics for the game, A and B are two compact sets in $\mathbb{R}^m$ representing respectively the control sets for the first player and the second player. The two players can both steer the system, the first wants the system reaches the target $\mathcal{T}$ in the minimum time while the second player wants the system goes away for ever. The value function $T = -\ln(1 - v)$ represents the time to reach the target if both players play optimal nonanticipative strategies (see [1]). It is clear that the accepting-the-minimum rule of the FM technique can not work in this case because the optimal trajectory for the second player goes toward the higher values of the value function. The discrete scheme based on the Discrete Dynamic Programming Principle is

$$\begin{cases} w(x_i) = \max_{b \in B} \min_{a \in A} \big\{\beta w(x_i - h f(x_i, a, b))\big\} + 1 - \beta & x_i \in G \backslash \mathcal{T} \\ w(x_i) = 0 & x_i \in G \cap \mathcal{T}. \end{cases} \tag{18}$$

3.2 Main idea of the BFM method

As we explained in section 2.2 the update procedure of FM method is not suitable for the numerical solution of equations different from the eikonal

equation. On the contrary the BFM method is designed to solve correctly and rapidly any equation of the form (12).
In the proposed algorithm, the node in the narrow band with the minimum value is not accepted as it happens in the FM method but it is moved in a *buffer*. All the nodes in the buffer are recomputed at each step of the algorithm until it is sure that their value can not change any more, this is guaranteed by a local condition which will be introduced below. After that, they are finally labeled as accepted (see Fig. 1). Note that the

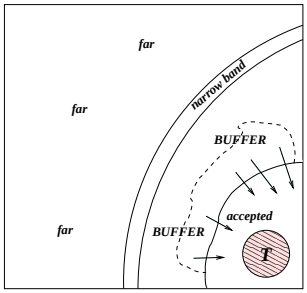

Figure 1: Division of the nodes: target, accepted, part of the buffer which is going to be accepted, buffer, narrow band and far

size of the buffer strictly depends on the anisotropy coefficient and so the computational cost does.
We have now to choose how the nodes go out from the buffer. A possible choice is the following. In a copy of the matrix where the computation is being performed we substitute the value $v = 1$ for all the values in the narrow band. Then we compute until convergence the nodes in the buffer iterating computation (for example (16)). After that, we repeat the procedure substituting the value $v = 0$ for all the values in the narrow band. Finally, we look for the nodes whose values has not changed in the two steps. In other words, we treat the narrow band as part of the boundary of the computation domain, imposing at nodes in it two different boundary conditions. The first one is $v = 1$, that is the maximum value a node can assume and the second is $v = 0$ that is the minimum value a node can assume. Clearly, if a node does not change its value after this kind of modification it means that its value does not depend on the next steps of the algorithm whatever it happens and then it can be labeled as accepted.

3.3 The algorithm

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

Definition 3.1 (neighboring nodes for the semi-Lagrangian scheme). Let the dimension n be 2 and let $x_{i,j}$ be a node. We define the set of neighboring nodes to $x_{i,j}$ as

$$N_{SL}(x_{i,j}) := \{x_{i\pm1,j}, x_{i,j\pm1}, x_{i+1,j+1}, x_{i+1,j-1}, x_{i-1,j+1}, x_{i-1,j-1}\}.$$

The nodes in $N_{SL}(x_{i,j})$ are the nodes that appear in the stencil of the first order semi-Lagrangian discretization. The above definition can be easily extended to the n-dimensional case.

The BFM algorithm
Initialization

1. Locate the nodes belonging to the initial front Γ_0 and label them as accepted. They form the set $\widetilde{\Gamma}_0$. Impose $v(\widetilde{\Gamma}_0) = 0$ (corresponding to $T(\widetilde{\Gamma}_0) = 0$).
2. Define *NB* as the set of the nodes belonging to $N_{SL}(\widetilde{\Gamma}_0)$, external to Γ_0.
3. Iterate the computation on all the nodes in *NB* until convergence (as in the classical fixed point method).
4. Label the remaining nodes as far, setting their values to $v = 1$ (corresponding to $T = +\infty$).

Main Cycle

1. Let A be the node with the minimum value among all the nodes in *NB*. Find A, remove it from *NB* and label it as buffer. Move the far nodes of $N_{SL}(A)$ into *NB* and (re)compute the nodes in $N_{SL}(A)$ which are not accepted.
2. Compute all the nodes in the buffer once.
3. In a copy of the matrix where the computation is being performed, substitute $v = 1$ for the value of the nodes in *NB*. Then iterate the computation on all the nodes in the buffer until convergence (as in the classical fixed point method).
4. Again in the copy of the matrix, substitute $v = 0$ for the value of the nodes in *NB*. Iterate again the computation on all the nodes in the buffer until convergence.

5. Remove from the buffer and label as accepted the nodes whose value is not changed in the two previous steps.

6. If *NB* is not empty go back to step 1, otherwise iterate the computation on all the nodes in the buffer until convergence and stop computation.

Real implementation
Unfortunately, the described algorithm can be slower than the classical iterative scheme in many situations. Nevertheless, some tricks can speed up the computation. Most of them do not modify the final solution so they can not affect possible theoretical results for the BFM method. On the contrary, other modifications change the final solution although it is reasonable to expect that the new solution is close to that of the ideal algorithm. The modifications are:

M1. Assuming that the solution is increasing along characteristics (this is the typical situation in the minimum time problem, for example) we can use the current minimal value v_{min} in the narrow band instead of $v = 0$ in step 4. In fact the values of not-yet-accepted nodes will be greater than v_{min} in the following iterations.

M2. The step 3 can be completely skipped because the far zone already plays the role of a moving boundary condition with values $v = 1$.

M3. It is not really needed to store a second matrix to perform intermediate computations described in steps 3 and 4. Storing the values of the node in the same dynamic list which contains the indexes of the nodes in the narrow band and in the buffer, we can modify the full matrix and then restore the right values by the values stocked in the list. This leads to a faster algorithm and a gain in memory requirement.

M4. In step 5 it is possible to accept the nodes such that their variation is smaller than a given quantity ε. This modification produces an error in the solution with respect to the ideal algorithm which is expected to vanish as ε tends to zero.

M5. It is possible to do steps 2-5 every $p > 1$ steps. It seems that $p = p(N) = \sqrt{N}/2$ is a good choice in most cases (N being the total number of nodes and the dimension of the problem being 2). This leads to a larger buffer but to a faster algorithm.

M6. In step 3 and 4 it is not needed to be very accurate, we are just interested if the values of the nodes in the buffer change or not. So

we iterate the computation for each node $x_{i,j}$ until

$$|w^{(k+1)}(x_{i,j}) - w^{(k)}(x_{i,j})| < \varepsilon', \quad k = 0, 1, \dots$$

The values we chose for all the constant involved in the real implementation of the algorithm will be detailed in the last section dedicated to numerical tests.

Remark **3.1** In the case the front propagation problem is isotropic, the buffer's size is never greater than one.

3.4 Properties of the BFM

To prove the convergence of the algorithm to the viscosity solution of equation (12) we adopt the following strategy. We show that the BFM computes the same solution of the classical iterative algorithm and then we recover convergence and *a priori* estimates by the results already available for that scheme (see f.e. [1] for Hamilton-Jacobi-Bellman equations).

Proposition 3.1 (Convergence to the viscosity solution) Assume equation (12) has a unique viscosity solution. Let V_i, $i = 1, \dots, N$ be its discrete solution computed by the classical iterative (fixed point) scheme based on a convergent numerical scheme and $\widetilde{V}_i$, $i = 1, \dots, N$ be the discrete solution of the same equation computed by the BFM method based on the same scheme. Then $V = \widetilde{V}$.

Proof. By induction on the steps of the algorithm. We want to show that the accepted nodes are *correct* in the sense that their values will not change in the following iterations even if we compute on all over the grid for any number of times. At the beginning we consider as accepted those nodes lying on the initial front so their correctness comes from the boundary condition on Γ_0. At a generic step of the algorithm, by (10) we know that the nodes in the buffer are between the accepted zone and the narrow band and this will be true also in the following iterations because the narrow band can not come back at nodes already visited. Let $\Omega_B^{(s)}$ be the buffer zone at the s-th step of the algorithm. Its boundary $\Gamma^{(s)} = \partial\Omega_B^{(s)}$ is divided in two regions. The first, denoted by $\Gamma_{ACC}^{(s)}$ is adjacent to the accepted zone while the second, denoted by $\Gamma_{NB}^{(s)}$ is adjacent to the narrow band zone. We have $\Gamma^{(s)} = \Gamma_{ACC}^{(s)} \cup \Gamma_{NB}^{(s)}$ (see Fig. 2). The values of the nodes in $\Gamma_{ACC}^{(s)}$ and $\Gamma_{NB}^{(s)}$ play the role of two boundary conditions for the buffer. The algorithm

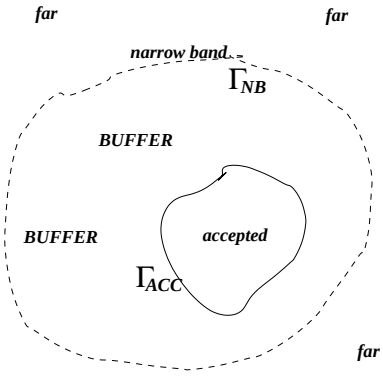

Figure 2: The buffer zone and its boundaries.

produces three solutions in $\Omega_B^{(s)}$:

$$\begin{array}{ll} v^{(s)} & \text{with } v(\Gamma_{NB}^{(s)}) \text{ and } v(\Gamma_{ACC}^{(s)}) \text{ unchanged} \\ v_1^{(s)} & \text{with } v(\Gamma_{NB}^{(s)}) = 1 \text{ and } v(\Gamma_{ACC}^{(s)}) \text{ unchanged} \\ v_0^{(s)} & \text{with } v(\Gamma_{NB}^{(s)}) = 0 \text{ and } v(\Gamma_{ACC}^{(s)}) \text{ unchanged} \end{array}$$

The set

$$ACC^{(s+1)} = ACC^{(s)} \cup \{x_{i,j} : v^{(s)}(x_{i,j}) = v_1^{(s)}(x_{i,j}) = v_0^{(s)}(x_{i,j})\}$$

defines the new accepted nodes. For these nodes, the boundary condition on $\Gamma_{ACC}^{(s)}$ is correct (induction hypothesis) while any boundary condition on $\Gamma_{NB}^{(s)}$ between 0 and 1 has not influence. As a consequence, the new accepted nodes are correct. ■

3.5 Some considerations on the computational cost

We always denote by N the total number of nodes in the grid. We assume for simplicity that we are working on a square grid in dimension 2 so each dimension has $\sqrt{N}$ nodes.

Classical iterative method. The iterative (fixed point) method consists in computing the solution of the equation node by node on all over the grid until convergence is reached. Since it needs $O(\sqrt{N})$ iterations to converge, its complexity is of order $O(N\sqrt{N})$ and we expect that doubling the nodes in each dimension leads to multiply by 8 the CPU time. Conversely, we expect that for an $O(N)$ algorithm the CPU time is multiplied only by 4.

FM method. The FM method has a complexity of order $O(N \ln N_{NB})$ where

N_{NB} is the number of nodes in the narrow band (it varies at each step). N_{NB} is bounded by N but in general it is expected to be of order $\sqrt{N}$ because the front has dimension 1, so its computational cost drops to $O(N \ln \sqrt{N})$. The term $O(\ln \sqrt{N})$ comes from the need of keeping an order in the list containing the nodes of the narrow band (f.e. by an heap-tree structure), so that it is fast to pick the node with the minimal value at each step. To this end it is important to note that the sequence of the minimal values of the narrow band is in many cases very close to be increasing, this means that a simple insertion sort is not so costly as in the randomly-ordered case. *For this reason we did not implement an heap structure to store the nodes but a simple dynamic linked list.* By experiments it seems that the factor $O(\ln \sqrt{N})$ is in fact $O(1)$, at least for low-dimensional problems.
BFM method. As we already remarked, in the case of isotropic front propagation problems the BFM method changes back to the FM method. So we expect a computational cost of the same order in the best case. In the worse case the buffer becomes larger and larger and we need to solve an iterative problem on the buffer to accept just few nodes or even any node at all. This leads to a computational cost greater than that of the iterative algorithm. Experiments says (see next section) that BFM method behaves like FM method in most cases, although the constant in front of $N \ln \sqrt{N}$ is larger for the BFM method.

Remark **3.2** There exist a number of methods to solve the shortest path problem on graphs. The FM method is clearly inspired by the Dijsktra method. Another method named *threshold algorithm* [13] can resemble the BFM because of the presence of two different sets for the tentative values (the narrow band and the buffer in our notations). Nevertheless the way how the nodes enter and exit the sets is completely different.

4 Numerical tests

In this section we perform some tests on equations of the form (15) and (17). The aim is to compare numerical results and CPU times for the classical iterative method, the FM method and the BFM method. All the three methods are based on the semi-Lagrangian scheme (16) or (18). For the classical iterative method we use the Fast Sweeping (FS) technique to speed up the convergence. This technique consists in computing over the grid in four alternate directions (for example from North to South, from South to North, from East to West and from West to East) until convergence is reached (see [25, 20] and references therein for details). The algorithms are implemented in C++ on a PC with a Pentium IV processor and 256 MB RAM. In the following we will consider the solution of the iterative

method as the exact solution and we compute the error of the other two methods with respect to that solution. Although this is obviously not true (Semi-Lagrangian scheme can produce bad results in some cases due to the numerical diffusion) we can not expect neither FM nor BFM overcome the iterative method since the three methods are based on the same numerical scheme. We compare the methods on 51^2, 101^2 and 201^2 structured grids (the number of nodes is chosen to have a grid node corresponding to the point $(0,0)$).
Computation is done in a square domain Q on a structured grid. As stopping criterion for the iterative Fast Sweeping method we used

$$\|v^{(k+1)} - v^{(k)}\|_\infty < 10^{-16}.$$

The L^1 error is defined as

$$E_1 = \frac{1}{|Q|} \iint_Q |T - T^{FS}|$$

where T is the solution computed by FM or BFM and T^{FS} is the solution computed by FS. Note that the difference with respect to the relative error $\widetilde{E}_1 = \frac{1}{|Q|} \iint_Q |T - T^{FS}|/T^{FS}$ is not significant so it will be not reported.
For the BFM, the values of the parameters ε (see step M4) and p (see step M5) are reported in the tables test by test while the value of ε' (see step M6) is fixed to 10^{-6}.

Test 1: Eikonal equation

In this test we solve the eikonal equation $|\nabla T(x,y)| = 1$ in $[-2,2]^2$ coupled with a Dirichlet boundary condition $T(0,0) = 0$. This equation can be written in the form (15) choosing $f(x,a) = a$ and $A = B(0,1) \subset \mathbb{R}^2$. We discretized the unit ball with 16 points equally spaced on the boundary. The level sets of the solution $T(x,y) = x^2 + y^2$ correspond to an isotropic front propagation so the FM method can be used and it is the best choice. By Table 1 we can see that the three methods compute the same solution. Here FS method needs just four iterations to reach convergence. Nevertheless, FM is the fastest method. BFM is slower than FM due to the time spent to menage the buffer (even if here the buffer contains only one node at each step). The last column reports the ratio between the CPU time for a $4N$ grid and a N grid. The value 5.0 for the FM with $N = 101^2$ is not completely correct because the FM with $N = 51^2$ is too fast to be precisely measured.

Table 1: Errors and CPU times for Test 1

method	Δx	N	p	ε	E_1	CPU time (sec)	CPU $N \to 4N$
FS	0.08	51^2	–	–	–	0.04	–
BFM	0.08	51^2	25	10^{-3}	0	0.07	–
FM	0.08	51^2	–	–	0	0.02	–
FS	0.04	101^2	–	–	–	0.17	4.2
BFM	0.04	101^2	50	10^{-3}	0	0.27	3.8
FM	0.04	101^2	–	–	0	0.1	5.0
FS	0.02	201^2	–	–	–	0.7	4.1
BFM	0.02	201^2	100	10^{-3}	0	1.12	4.1
FM	0.02	201^2	–	–	0	0.38	3.8

Table 2: Errors and CPU times for Test 2

method	Δx	N	p	ε	E_1	CPU time (sec)	CPU $N \to 4N$
FS	0.08	51^2	–	–	–	0.37	–
BFM	0.08	51^2	25	10^{-3}	0.02	0.09	–
FM	0.08	51^2	–	–	0.87	0.02	–
FS	0.04	101^2	–	–	–	2.49	6.7
BFM	0.04	101^2	50	10^{-3}	0.01	0.45	5.0
FM	0.04	101^2	–	–	1.02	0.09	4.5
FS	0.02	201^2	–	–	–	13.55	5.4
BFM	0.02	201^2	70	10^{-3}	0.02	1.67	3.7
FM	0.02	201^2	–	–	1.01	0.4	4.4

Test 2: Eikonal equation on a manifold

In this test we solve the equation related to anisotropic front propagation described in section 2.2 with $\alpha = \beta = 5$ and the unit ball is discretized by means of 16 points. Q is again $[-2, 2]^2$. As shown in [23], the FM method is not able to compute the right solution even if we increase the number of nodes. So in this test (and in the following ones) we show the CPU time for FM just for reference as the ideal time but it is not a real alternative to FS and BFM. On the other hand, by Table 2 and Fig. 3–4 we see that the behavior of BFM is quite good since it preserves the order of the scheme with a significantly difference in CPU time.
Note that the error of BFM does not converge to zero as the grid is refined because we used a fixed ε in all cases.

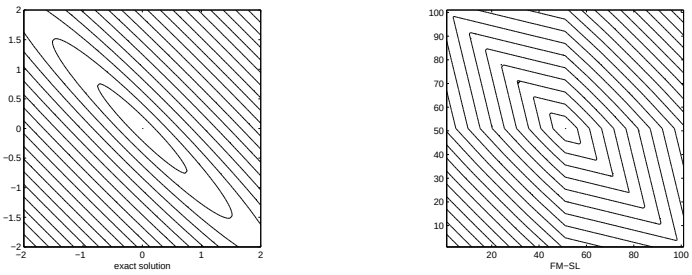

Figure 3: Test 2: exact solution (left) and solution computed by FM method (right), 101×101 grid.

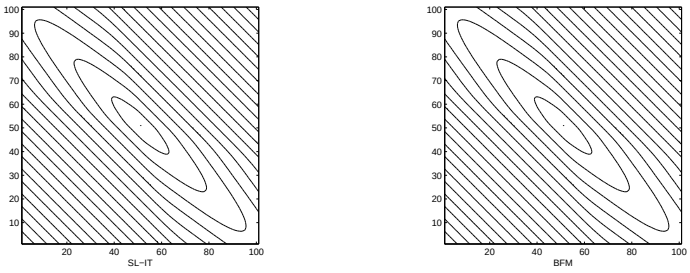

Figure 4: Test 2: solution computed by FS method (left) and BFM method (right).

Test 3: Zermelo navigation problem

In this test we solve equation (15) with

$$\begin{cases} f_1(x,y,a) = (2+2.1)\cos(a) \\ f_2(x,y,a) = 2.1\sin(a) \end{cases}$$

and Dirichlet boundary condition $v(1,0) = 0$ (corresponding to $T(1,0) = 0$). We chose $A = [0, 2\pi) \subset \mathbb{R}$ and we discretize this interval in 32 points. Q is $[-2,2]^2$. It corresponds to the classical Zermelo navigation problem when the speed of the current is 2 and the boat can move in any direction with speed 2.1. Accordingly to that dynamics it is possible to reach the target from every point of the space but the solution has strong variations and the anisotropy is strong too.
The SL scheme is not very efficient in this case due to the linear interpolation which results in some numerical diffusion. Anyway, by Table 3 and

Table 3: Errors and CPU times for Test 3

method	Δx	N	p	ε	E_1	CPU time (sec)	CPU $N \to 4N$
FS	0.08	51^2	–	–	–	0.53	–
BFM	0.08	51^2	25	10^{-3}	0.002	0.34	–
FM	0.08	51^2	–	–	0.17	0.04	–
FS	0.04	101^2	–	–	–	2.46	4.6
BFM	0.04	101^2	50	10^{-3}	0.003	1.02	3.0
FM	0.04	101^2	–	–	0.24	0.18	4.5
FS	0.02	201^2	–	–	–	11.35	4.6
BFM	0.02	201^2	100	10^{-3}	0.005	3.58	3.5
FM	0.02	201^2	–	–	0.26	0.76	4.2

Fig. 5 we see that the BFM is again very close to FS while FM is not. In the last column, the ratio of the CPU time for the BFM method is better than the optimal one (which is 4) because ε is fixed in the three cases.

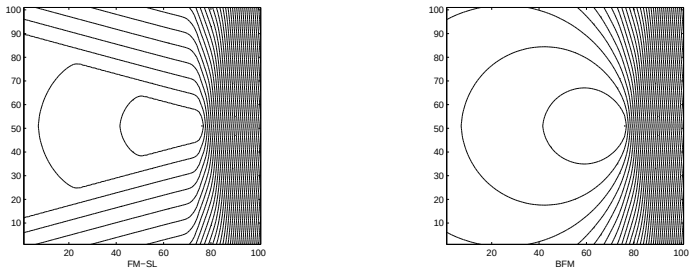

Figure 5: Test 3: solution computed by FM method (left) and BFM method (right).

Test 4: Lunar landing

In this test we solve equation (15) with

$$\begin{cases} f_1(x,y,a) = y \\ f_2(x,y,a) = a \end{cases}$$

and Dirichlet boundary condition $v(0,0) = 0$ (corresponding to $T(0,0) = 0$). We chose $A = \{-1, 1\}$. This test correspond to the classical one-dimensional

Table 4: Errors and CPU times for Test 4

method	Δx	N	p	ε	E_1	CPU time (sec)
FS	0.2	51^2	–	–	–	0.13
BFM	0.2	51^2	12	10^{-3}	0.07	0.01
FM	0.2	51^2	–	–	1.54	0
FS	0.1	101^2	–	–	–	0.67
BFM	0.1	101^2	25	10^{-4}	0.07	0.15
FM	0.1	101^2	–	–	3.21	0.02
FS	0.05	201^2	–	–	–	3.91
BFM	0.05	201^2	50	10^{-5}	0.05	2.05
FM	0.05	201^2	–	–	6.11	0.11

minimum time problem in which the dynamics is $\ddot{x} = u$ and u can be chosen in $\{-1, 1\}$. This is a difficult test because of the strong mutual dependency of nodes. Moreover, the effect of boundary condition is very strong so we decided to perform computation on the domain $[-5, 5]^2$ and to analyze the results on the subdomain $[-2, 2]^2$.
As the grid size increased we needed to decrease the constant ε in order to maintain the same order in the error. Not surprisingly, the FM computes a vary bad solution (see Fig. 6).

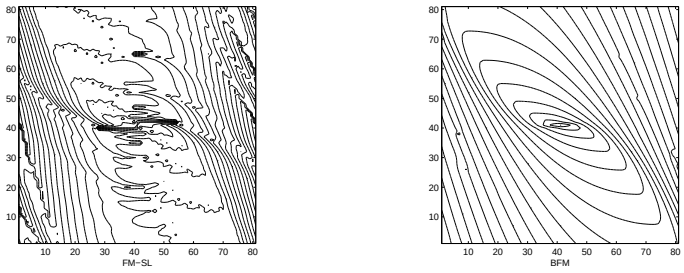

Figure 6: Test 4: solution computed by FM method (left) and BFM method (right).

Test 5: Tag-Chase game in reduced coordinates

In this test we solve equation (17) with

$$f(x, y, a, b) = V_A a - V_B b$$

where

$$a \in \{(\cos\theta_a, \sin\theta_a), \theta_a \in [0, 2\pi) \backslash [-\pi/4, \pi/4]\},$$
$$b \in \{(\cos\theta_b, \sin\theta_b), \theta_b \in [0, 2\pi)\}.$$

The target is the point $(0, 0)$. This example models a Tag-Chase game where a boy A running with speed V_A in an unbounded domain wants to catch another boy B running with speed $V_B < V_A$. We choose $V_A = 2$ and $V_B = 1$. While B can run in every direction, A has a constraints moving in the right direction. Denoting respectively by (x_A, y_A) and (x_B, y_B) the coordinates of the two players on the plane, the problem is solved in the reduced coordinates $(x, y) = (x_A - x_B, y_A - y_B)$. The target corresponds to the capture (see [11] for more details on the model). In Fig. 7 we show the level sets of the solution and an optimal trajectory in both reduced and real coordinates. Although this problem can be clearly seen as an anisotropic front propagation problem, the characteristics and gradient lines lie on the same simplex so the FM is able to compute the right solution (as stated in Criterion 5.1 of [23]). So this test is just to show that the FM technique can be extended to *minmax* operator (although some modifications are needed to avoid nodes with fictitious value $v = 1$ ($T = +\infty$) are used in the interpolation).

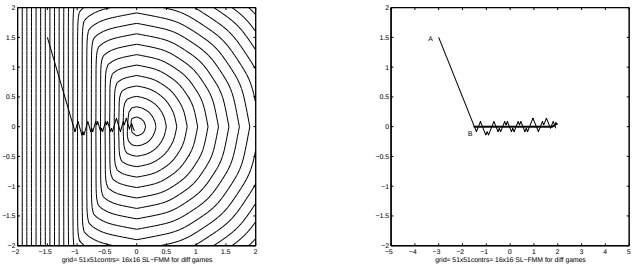

Figure 7: Test 5: solution computed by FM method with on optimal trajectory starting from the point $A = (-3, 1.5)$, $B = (-1.5, 0)$. Reduced coordinate (left) and real plane (right).

Test 6: Tag-Chase game with state constraints

In this test we solve equation (17) with

$$\begin{cases} f_1(x, y, a) = V_A a \\ f_2(x, y, b) = V_B b \end{cases}$$

where $a, b \in \{-1, 0, 1\}$. This test models the one-dimensional Tag-Chase game where the two players A and B are constrained to run in the segment $[-2, 2]$. Due to the state constraints, it is not possible to use reduced coordinates so that the game is set in $Q = [-2, 2]^2 \subset \mathbb{R}^2$. The velocities V_A for the pursuer and V_B for the evader are constant. We choose $V_A = 2$ and $V_B = 1$. The axis of abscissas represents the coordinate x_A of the Pursuer and the axis of ordinate represents the coordinate x_B of the Evader. The target is $\mathcal{T} = \{(x_A, x_B) : x_A = x_B\}$ that is the set of point where the capture occurs (see [11, 9, 6] for more details on the model and recent results on differential games with state constraints).
In Fig. 8 we show the exact solution (left) with one optimal trajectory starting from the point $(-1.5, 0)$ and the solution computed by FM (right). We show the result only in half domain due to the symmetry of the solution. Clearly in this case characteristics and gradient lines does not lie on the same simplex so the FM fails. In Fig. 9 we show the solution computed by FS (left) and by BFM (right). We can see very well the effect of numerical diffusion due to the scheme but again the two solutions are very similar.

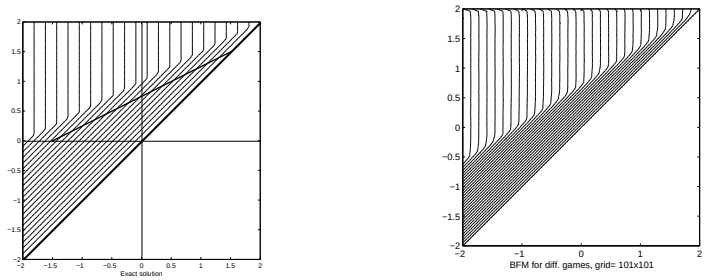

Figure 8: Test 6: exact solution with an optimal trajectory starting from $(-1.5, 0)$ (left) and solution computed by FM method (right).

FM and BFM methods vs. Fast Sweeping method

In this paper we used the Fast Sweeping technique to compute the solution of the classical iterative scheme (6) because it is in general fast and robust and it is proved to converge to the fixed point. The other advantage is that it is not restricted to isotropic front propagation problems like FM method. Unfortunately it is very difficult to estimate the number of sweepings needed to reach convergence in the case of a general velocity field but experiments say that FS method is much faster than the classical iterative method where the nodes are visited in only one fixed order.

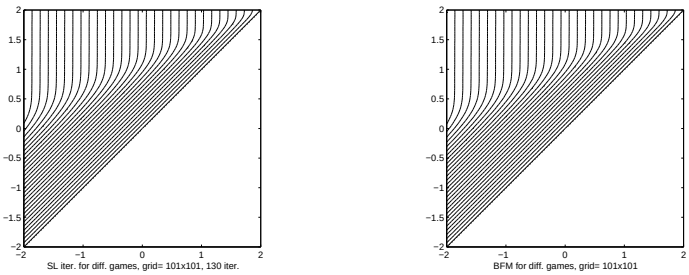

Figure 9: Test 6: solution computed by FS method (left) and BFM method (right).

Comparing FM/BFM and FS methods is not an easy task because their behavior is very case-dependent. For example, test 2 was chosen to be "difficult" for FM/BFM methods because of the strong anisotropy. On the other hand, the same test can be considered "easy" for FS method because the characteristics directions are straight lines to the origin and few sweepings are enough to compute a good approximation of the viscosity solution. In test 2 the FS method is faster than BFM method allowing the same L^1 distance from the exact solution.
The result is reversed in the test described in Fig. 10. The choice of the velocity field corresponds to an isotropic front propagation problem in presence of obstacles. The FS method is slower than BFM method which is slower than FM method.

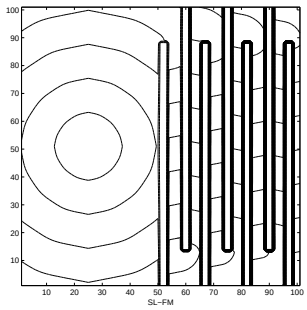

Figure 10: A difficult test for the FS method. The front moves in normal direction with speed 1. The rectangles represent obstacles.

Conclusions

In this paper we introduced a new fast method to solve Hamilton-Jacobi equations modeling a monotone front propagation problem, including Hamilton-Jacobi-Bellman and Hamilton-Jacobi-Isaacs equations related to optimal control problems and differential games. Although it does not compute exactly the same solution of the standard iterative (fixed point) method based on the same first order semi-Lagrangian scheme, the new method is able to compute a good approximation of the viscosity solution preserving the order of the scheme. By the experiments, it seems that the computational cost is near to $O(N)$ as for the FM method.

Acknowledgment

The author wishes to thank Maurizio Falcone, Frederic Bonnans, Hasnaa Zidani, Olivier Bokanowski, and Nicolas Forcadel for the useful discussions and suggestions.

References

[1] M. Bardi and I. Capuzzo Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*, Birkhäuser, Boston, 1997.

[2] M. Bardi and M. Falcone, *An approximation scheme for the minimum time function*, SIAM J. Control Optim., 28 (1990), pp. 950–965.

[3] E. Carlini, E. Cristiani and N. Forcadel, *A non-monotone Fast Marching scheme for a Hamilton-Jacobi equation modelling dislocation dynamics*, in A. Bermdez de Castro, D. Gmez, P. Quintela, P. Salgado (eds.), Numerical Mathematics and Advanced Applications, Proceedings of ENUMATH 2005 (Santiago de Compostela, Spain, July 2005), 723-731, Springer, 2006.

[4] E. Carlini, M. Falcone, N. Forcadel, and R. Monneau, *Convergence of a Generalized Fast Marching Method for an eikonal equation with a velocity changing sign*, submitted to SIAM J. Numer. Anal.

[5] E. Cristiani, *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.

[6] E. Cristiani and M. Falcone, *Numerical solution of the Isaacs equation for differential games with state constrants*, to appear in Proceedings of 17th IFAC World Congress (Seoul, Korea, July 6-11, 2008).

[7] E. Cristiani and M. Falcone, *Fast semi-Lagrangian schemes for the Eikonal equation and applications*, SIAM J. Numer. Anal., 45 (2007), 1979-2011.

[8] E. Cristiani and M. Falcone, *A characteristics driven Fast Marching method for the Eikonal equation*, submitted to Proceedings of ENUMATH 2007 (Graz, Austria, September 10-14, 2007).

[9] E. Cristiani and M. Falcone, *Fully-discrete schemes for the value function of Pursuit-Evasion games with state constraints*, to appear in Annals of International Society of Dynamic Games.

[10] M. Falcone, *The minimum time problem and its applications to front propagation*, in Motion by Mean Curvature and Related Topics, A. Visintin and G. Buttazzo, eds., de Gruyter, Berlin, 1994, pp. 70–88.

[11] M. Falcone, *Numerical methods for differential games based on partial differential equations*, International Game Theory Review, 8 (2006), pp. 231–272.

[12] M. Falcone, T. Giorgi, and P. Loreti, *Level sets of viscosity solutions: Some applications to fronts and rendez-vous problems*, SIAM J. Appl. Math., 54 (1994), pp. 1335–1354.

[13] F. Glover, R. Glover, D. Klingman, *The threshold shortest path algorithm*, Networks, 14 (1986), pp. 256-282.

[14] P. A. Gremaud and C. M. Kuster, *Computational study of fast methods for the eikonal equation*, SIAM J. Sci. Comput., 27 (2006), pp. 1803–1816.

[15] S.-R. Hysing and S. Turek, *The eikonal equation: numerical efficiency vs. algorithmic complexity on quadrilateral grids*, in Proceedings of ALGORITMY 2005, pp. 22–31.

[16] S. Kim, *An $\mathcal{O}(N)$ level set method for eikonal equations*, SIAM J. Sci. Comput., 22 (2001), pp. 2178–2193.

[17] R. Kimmel and J. A. Sethian, *Computing geodesic paths on manifold*, Proc. Natl. Acad. Sci. USA, 95 (1998), pp. 8431–8435.

[18] R. Kimmel and J. A. Sethian, *Optimal algorithm for shape from shading and path planning*, J. Math. Imaging Vision, 14 (2001), pp. 237–244.

[19] E. Prados and S. Soatto, *Fast marching method for generic shape from shading*, in Proceedings of VLSM 2005, pp. 320–331.

[20] J. Qian, Y.-T. Zhang, and H.-K. Zhao, *A fast sweeping method for static convex Hamilton-Jacobi equations*, J. Sci. Comput., 31 (2007), pp. 237–271.

[21] J. A. Sethian, *Level Set Methods and Fast Marching Methods. Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, Cambridge, UK, 1999.

[22] J. A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 1591–1595.

[23] 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.

[24] A. Spira and R. Kimmel, *An efficient solution to the eikonal equation on parametric manifolds*, Interfaces Free Bound., 6 (2004), pp. 315–327.

[25] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, and H.-K. Zhao, *Fast sweeping algorithms for a class of Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 41 (2003), pp. 673–694.

[26] J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control, 40 (1995), pp. 1528–1538.

[27] A. Vladimirsky, *Static PDEs for time-dependent control problems*, Interfaces and Free Boundaries, 8 (2006), pp. 281–300.