**Preprints of the 18th IFAC World Congress**
**Milano (Italy) August 28 - September 2, 2011**

# A Local Ordered Upwind Method for Hamilton-Jacobi and Isaacs Equations ⋆

**S. Cacace** * **E. Cristiani** ** **M. Falcone** ***

* *Dipartimento di Matematica, SAPIENZA - Università di Roma, Rome, Italy (e-mail: cacace@mat.uniroma1.it).*
** *Dipartimento di Matematica, SAPIENZA - Università di Roma, Rome, Italy (e-mail: emiliano.cristiani@gmail.com).*
*** *Dipartimento di Matematica, SAPIENZA - Università di Roma, Rome, Italy (e-mail: falcone@mat.uniroma1.it).*

**Abstract:** We present a generalization of the Fast Marching (FM) method for the numerical solution of a class of Hamilton-Jacobi equations, including Hamilton-Jacobi-Bellman and Hamilton-Jacobi-Isaacs equations. The method is able to compute an approximation of the viscosity solution concentrating the computations only in a small evolving *trial region*, as the original FM method. The main novelty is that the size of the trial region does not depend on the dynamics. We compare the new method with the standard iterative algorithm and the FM method, in terms of accuracy and order of computations on the grid nodes.

*Keywords:* dynamic programming, optimal control, differential games, fast marching methods, numerical methods

## 1. INTRODUCTION

The solution of optimal control problems and differential games via dynamic programming requires first the computation of the value function (see e.g. Bardi and Capuzzo Dolcetta (1997)) which is characterized in terms of a first order nonlinear partial differential equation of the form

$$H(x, u(x), \nabla u(x)) = 0\,, \quad x \in D\,, \tag{1}$$

where $H : D \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}$ is called the *Hamiltonian* and $D$ is an open subset of $\mathbb{R}^d$. Typically, Dirichlet boundary conditions complement the equation. In particular, for games, the Hamiltonian is nonconvex with a minmax structure (for details see Bardi and Capuzzo Dolcetta (1997) and Section 2 below). Despite the fact that the theoretical framework and the approximation theory for the classical iterative algorithms (e.g. finite differences, semi-Lagrangian) are still valid in any dimension, the computational problem which has to be solved is so huge that no one has been able to compute a solution for a dimension greater than 6. This is a real bottleneck for the applications of this technique to real-life problems because those problems are often high dimensional. For Pursuit-Evasion games this difficulty becomes very serious even for rather low dimensions since, in this case, the dimension of the state variable appearing in the Isaacs equation doubles the number of variables used for each player (we pass from $d$ to $2d$ variables in the continuous model). Some techniques have been proposed to overcome this difficulty. One can accelerate convergence via specially adapted methods that exploit monotonicity as in Falcone (1997) or can introduce efficient methods to implement the algorithm in high-dimension as in Carlini et al. (2004), Bokanowski et al. (2010). Moreover, one can adopt a domain decomposition technique and split the problem into sub-problems with a small number of nodes so that every sub-problem can fit the memory allocation requirements as in Camilli et al. (1994).

Another interesting idea is to localize the computation to avoid useless computations. A class of Fast Marching (FM) methods has been proposed for the eikonal equation, starting from Tsitsiklis (1995) and Sethian (1996). The main feature of these methods is that they are *single-pass* and that their computational cost is of order $O(N \log N)$, where $N$ is the number of grid nodes.

More recently, the methods for the eikonal equation have been improved in order to speed-up the computation as in the Group Marching method (see Kim (2001)) and to obtain more accurate results (see Cristiani and Falcone (2007)). Moreover, FM-like methods have been proposed for more general Hamilton-Jacobi equations and various local solvers (see Carlini et al. (2006), Sethian and Vladimirsky (2003), Prados and Soatto (2005), Cristiani (2009)). Other efficient methods have been introduced, like the Fast Sweeping methods, see e.g. Tsai et al. (2003), Zhao (2005).

We present here a new method which looks as an improvement of the Buffered Fast Marching (BFM) method proposed in Cristiani (2009). Both BFM and the new method solve the same class of equations, but the new method does not need to introduce the *buffer* zone as the BFM does, thus keeping the computation strictly *local*. In this paper we are interested in the solution of problems with convex Hamiltonians where the standard FM method fails, as e.g. the anisotropic eikonal equation (see Sethian and Vladimirsky (2003)) and with minmax nonconvex Hamiltonians which appear in the analysis of differential

⋆ This work has been partially supported by the AFOSR Grant FA9550-10-1-0029.

games. We will discuss the main ideas which are behind this new algorithm also showing some numerical results on classical problems.

## 2. HJB AND HJI EQUATIONS AND THEIR DISCRETIZATION

Let us consider the Hamilton-Jacobi-Isaacs (HJI) equation for the lower value of two-player zero-sum differential game related to a target problem where each player lives in $\mathbb{R}^d$, i.e.

$$\begin{cases} v(x) + \min\limits_{b\in B}\max\limits_{a\in A}\big\{ -f(x,a,b)\cdot\nabla v(x)\big\} = 1, & x\in\mathbb{R}^{2d}\backslash\overline{\Omega} \\ v(x) = 0\,, & x\in\partial\Omega \end{cases} \tag{2}$$

where $f : \mathbb{R}^{2d}\times A\times B\to\mathbb{R}^{2d}$ is the dynamics of the game, $A$ and $B$ are two compact sets in $\mathbb{R}^m$ representing the control set for the first player and for the second player respectively and $\Omega$ is an open set representing the target for the game (see Falcone (2006) for more details). Dropping the second player, we obtain the Hamilton-Jacobi-Bellman (HJB) equation associated to a minimum time problem in dimension $d$,

$$\begin{cases} v(x) + \max\limits_{a\in A}\big\{ -f(x,a)\cdot\nabla v(x)\big\} = 1, & x\in\mathbb{R}^{d}\backslash\overline{\Omega} \\ v(x) = 0\,, & x\in\partial\Omega. \end{cases} \tag{3}$$

As in Falcone (1997), Cristiani and Falcone (2009) we will focus our attention on the fully-discrete semi-Lagrangian (SL) scheme obtained via discrete dynamic programming, which for (2) reads

$$\begin{cases} w(x_i) = \max\limits_{b\in B}\min\limits_{a\in A}\big\{\beta w(z_i(a,b))\big\} + 1-\beta\,, x_i\in(Q\backslash\overline{\Omega})\cap G \\ w(x_i) = 0\,, & x_i\in\overline{\Omega}\cap G \end{cases} \tag{4}$$

where $h>0$, $\beta = e^{-h}$, $w$ represents the approximation of $v$, $z_i(a,b) := x_i + hf(x_i,a,b)$, $Q$ is (tipically) a polyedral set containing the target $\Omega$ and $G$ is the set of the grid nodes $x_i$, $i=1,\dots,N$. This leads to a numerical scheme which can be written as a fixed point iteration in abstract form

$$w_i^{n+1} = S_i[w^n], \quad i=1,\dots,N, \quad n\in\mathbb{N}\,, \tag{5}$$

where $w_i^n$ represents the numerical solution at iteration $n$ and at the node $x_i$. For an extensive presentation of semi-Lagrangian schemes for linear and Hamilton-Jacobi equations we refer the interested reader to the book Falcone and Ferretti (2011).

It is well known that at every node $x_i\in G$ we need to compute by interpolation the value $w(z_i(a,b))$ for all $a\in A$ and $b\in B$, using the values of the grid nodes. For a reconstruction based on linear interpolation in $\mathbb{R}^2$, we use three nodes close to the point $z_i$. A crucial point in this scheme is the choice of the discretization step $h$. An appropriate choice is essential to make the algorithm suitable for the Fast Marching technique. First, we choose a regular grid with uniform spatial space steps $\Delta x$. Then, for every node $x_i$ and every choice of $a,b$, we set $h = h_i(a,b) = \Delta x/|f(x_i,a,b)|$. In this way the point $z_i$ belongs to one of the first four cells surrounding the node $x_i$ and then the computation is kept "local". Moreover, we avoid to use the value $w(x_i)$ in the linear interpolation, as this allows for the convergence in a finite number of steps. Note that, as $h$ depends on $a$ and $b$, the term $\beta = e^{-h}$ should be now included in the minmax (or max) evaluation.

## 3. PREVIOUS FAST MARCHING METHODS

Here we briefly resume, for reader's convenience, the main features of some FM methods well known in literature. This will be useful later for a comparison with our method.

### *3.1 The original Fast Marching method*

The FM method was originally developed for the eikonal equation

$$\begin{cases} c(x)|\nabla u(x)| = 1\,, & x\in\mathbb{R}^d\setminus\overline{\Omega} \\ u(x) = 0\,, & x\in\partial\Omega. \end{cases} \tag{6}$$

Note that this is the stationary version of the equation describing, via the level set method, the propagation of a front with speed $c$ in the normal direction (see Falcone (1994) for a detailed presentation of the relation between the minimum time and the front propagation problem). By the (monotone) change of variable $v(x) = 1-\exp(-u(x))$, the particular choice of the dynamics $f(x,a) = c(x)a$ and of the control set $A = B(0,1)$, equation (6) can be written in the form (3). Note that $0\leqslant v<1$ because the minimum time $u$ is always positive by physics. We discretize the eikonal equation by means of the SL scheme described before.

The idea of the FM method is to concentrate the computational effort in a small subset of the grid at each time. In more detail, at a generic step $n$ of the algorithm the grid is divided in three sets, $A_n$ (*accepted* nodes), $NB_n$ (*narrow band* nodes) and $F_n$ (*far* nodes). The nodes in $A_n$ are already computed and their value is considered as final, while the nodes in $F_n$ are not yet computed. The computation only takes place in $NB_n$. At each step only one node in $NB_n$ is moved to $A_n$ and $NB_n$ is updated. The FM method is able to compute the solution following the level sets of the solution itself. Moreover, by accepting one node at a time, the algorithm orders the grid nodes, and this order turns to be the one that respects the causality principle (the value of each node only depends on the values of the previous nodes).
Let us describe the algorithm.

*FM Algorithm*

1. Locate the nodes belonging to the target $\overline{\Omega}$ and label them as $A_0$, setting their values to $w=0$. Label all the neighbors of nodes in $A_0$ as $NB_0$ and compute their values solving the discrete equation. Set to $w=1$ the value of all other nodes and label them as $F_0$.
2. Move the node $X_{min} := \arg\min_{X\in NB_n}\{w(X)\}$ into the *accepted* region, i.e. $A_{n+1} = A_n\cup\{X_{min}\}$.
3. Remove $X_{min}$ from the *narrow band* and include not-accepted neighbors of $X_{min}$ in the *narrow band*, i.e. $NB_{n+1}$=$(NB_n\backslash\{X_{min}\})\cup\{$not-accepted neighbors of $X_{min}\}$. Solve the equation in $NB_{n+1}\backslash NB_n$.
4. If $NB_{n+1}$ is not empty go to Step 2 with $n\leftarrow n+1$, else stop.

Note that the SL scheme is compatible with the FM technique, as proved in Cristiani and Falcone (2007). It is also important to note that this method converges in a finite number of iterations. It has been proved that for

classical local solvers like finite difference or SL schemes the FM method has a complexity of order $O(N \log N)$ operations, where $N$ is the total number of nodes.

### *3.2 Limitations of the FM method*

The classical FM method accepts at each iteration the node with the minimal value among all the nodes in *NB* and this yields to compute the solution following its gradient lines and not the characteristics. This choice is correct in the case of the eikonal equation (6), since gradient lines and characteristics coincide. That geometric property does not hold any more for general Hamilton-Jacobi equations (1), as in the anisotropic eikonal equation extensively studied in Sethian and Vladimirsky (2003). The FM method could fail in the sense that it could accept a node in *NB* which has not reached convergence yet. Then we face the new problem of finding a rule to determine which node (if any) in *NB* should be accepted.

### *3.3 The Buffered Fast Marching method*

Before introducing the new method, it is worth to recall the main ideas of the Buffered FM (BFM) method, which is a generalization of the FM method proposed in Cristiani (2009). The BFM method divides the domain in four sets instead of three. The additional set, called *buffer*, is between the *accepted* region and the *narrow band*. It collects all the nodes which exit the *narrow band* (with the same accept-the-minimum rule of FM method). When the *buffer* is large enough, a new acceptance condition is used to move nodes to the *accepted* zone. To check this condition it is required to compute the solution in the *buffer* iteratively until convergence, substituting a test value for the values of the *narrow band* (which act as a boundary for the *buffer* region), see Cristiani (2009) for details. Choosing in an appropriate way the test value, we can find in the *buffer* those nodes that can not depend on the outcome of future computations. Then they necessarily depend on the *accepted* nodes.
It is important to note that the minimal size of the *buffer* which allows to accept at least one node depends on the dynamics $f$ and it can be very large. In the worst case, the BFM method becomes equivalent to the classical iterative method which computes all the grid nodes at the same time. This is the main drawback which we try to overcome with the new method.

## 4. THE PROGRESSIVE FAST MARCHING METHOD

Let us assume for a while that we have at our disposal the solution of the equation computed by the classical iterative scheme (5) which computes on the full grid. We will refer to this solution as the "exact" solution. Then, we could in principle use this solution to select the node in the *narrow band* to be accepted, i.e. we simply select the node which has the exact value. Running this dumb algorithm we have checked that the *narrow band* does not always contain an exact value, meaning that it is not possible to build a truly *single-pass* scheme for general equations. It is then necessary to solve iteratively in the *narrow band* the numerical scheme in order to stabilize the solution and get at least one acceptable node.

We are now ready to describe the Progressive FM (PFM) method.

*PFM Algorithm*

1. Locate the nodes belonging to the target $\overline{\Omega}$ and label them as $A_0$, setting their values to $w = 0$. Label all the neighbors of the nodes in $A_0$ as $NB_0$ and compute their values. Set the values of all other nodes to $w = 1$ and label them as $F_0$.
2. Solve the numerical scheme iteratively in $NB_n$ until all values are stabilized.
3. Find $w_{min} = \min_{X \in NB_n}\{w(X)\}$ and set $w_{out} = w_{min}$.
4. For each node $X \in NB_n$, replace the value of all its not-accepted neighbors with $w_{out}$, and re-solve in $X$ computing $w_{new}(X)$. Compare $w(X)$ with $w_{new}(X)$. If the node $X$ has not changed its value, name it $X_{acc}$, set $A_{n+1} = A_n \cup \{X_{acc}\}$ and go to Step 5. If, after cycled in $NB_n$, *all* the nodes in $NB_n$ changed their value, increase slightly $w_{out}$ and go to Step 4.
5. Update the *narrow band*, $NB_{n+1}$=$(NB_n \setminus \{X_{acc}\}) \cup$ {not-accepted neighbors of $X_{acc}$}.
6. If $NB_{n+1}$ is not empty go to Step 2 with $n \leftarrow n + 1$, else stop.

Let us clarify the main idea behind the algorithm. In order to find the node to be accepted in the *narrow band* we should know in advance the solution in the *far* zone, since this would allow to find the node which does not depend on the outcome of future computations. Since we do not have this information, in order to safely make a choice of acceptance we are forced to consider the two extreme possibilities, namely $w = w_{min}$ and $w = 1$: the first represents the minimal value the solution can attain in the current *narrow band* and *far* zone (because the solution is increasing along characteristics), the second represents an upper bound for the solution. The maximal case is somehow included in the choice of the initial guess in the *far* zone and then it must not be further considered. Let us come to the minimal case.
After Step 2 of the PFM we can assume that at least one node in the *narrow band* has the "exact" value, so that at least one node in the *narrow band* is not affected by future computations. In Step 4, let us assume that all the nodes in the *narrow band* changed their value. This means that now all these nodes depend on the test value $w_{out}$, but this contradict the result of Step 2. Then, we are allowed to increase a little bit $w_{out}$ and repeat the computation. When the threshold value $\overline{w}_{out}$ is found, we have found the actual lowest value which can come out from future computations and then we can accept the first node whose value is not affected by it. In this way we introduce a completely new rule of acceptance for the nodes in the *narrow band*, which turns out to be the correct one. Moreover, in the case where $\overline{w}_{out}$ coincides with $w_{min}$, we recover the classical FM method. Then, we can also interpret the gap $\overline{w}_{out} - w_{\min}$ as an index of anisotropy of the problem.
In order to speed up the algorithm, we can compute and keep in memory the threshold $\overline{w}_{out}(X)$ for all $X \in NB$, and then accept the node $\overline{X} = \arg\min_{X \in NB} \overline{w}_{out}(X)$. This allows to re-compute at each iteration only few thresholds, since $\overline{w}_{out}(X)$ will change only if $X$ is a neighbor of $\overline{X}$.

PFM method shares with BFM method some important features. The most important one is the use of a test value ($w_{out}$ in PFM method) to be assigned outside the region in which the next accepted node must be found. It is interesting to note that the enlargement of the *buffer* zone in the BFM method is now replaced by the increments of $w_{out}$ (Step 4 of PFM).
Concerning memory usage, apart from the storage of a full matrix for the solution, both methods need extra data structures to work, but PFM method has some advantages. Indeed, BFM allocates two lists, one containing the nodes belonging to the *narrow band* and one containing the nodes belonging to the *buffer*. Since the *narrow band* is an object of co-dimension 1, the size of the first list, e.g. in dimension 2, is about $\sqrt{N}$, where $N$ is the total number of grid nodes. On the other hand, the size of the *buffer* list is *a-priori* unknown, since it strongly depends on the dynamics driving the system and in the worst case it may contain the nodes of the whole grid. On the contrary, we recall that PFM method is designed to be a strictly local algorithm and then it only requires a *narrow band* list to perform all the computations. This is an important improvement in view of the application of the method to high dimensional problems.

## 5. NUMERICAL EXPERIMENTS

All numerical experiments were performed with a Matlab (version 7) implementation on a HP Compaq 8510w with an Intel Core 2 Duo T7500 2.20 GHz processor and 2 GB RAM. The numerical domain is $Q = [-2, 2]^2$, the grid $G$ has $51 \times 51$ nodes (corresponding to spatial discretization steps $\Delta x = \Delta y = 0.08$) and the unit ball $B(0, 1)$, representing the control set, is discretized by means of 16 directions uniformly distributed on the boundary. If no otherwise specified, we choose $\Omega = B(0, \varepsilon)$ with $\varepsilon = 0.001$ as target set. Moreover, in all the figures below, we will plot the physical minimum time function $u = -\log(1 - v)$ instead of its Kružkov transform $v$.

**Test 1** (Anisotropic eikonal equation)

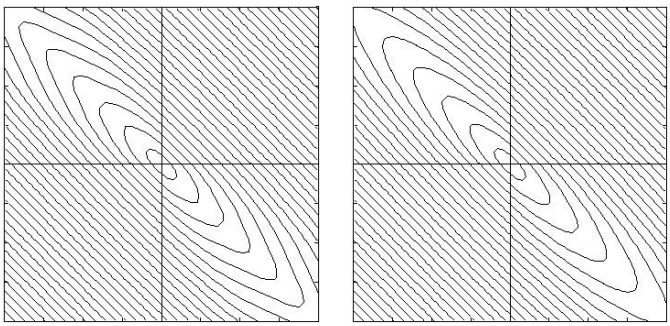
Let us consider equation (3) for dimension $d = 2$ and set

$$f(x, y, a) = c(x, y, a)\, a, \qquad a = (a_1, a_2) \in B(0, 1)$$

with

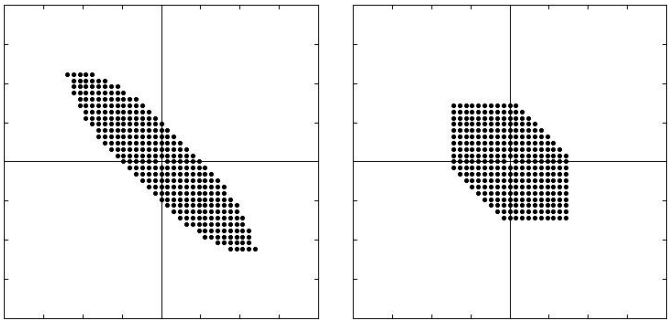
$$c(x, y, a) = \frac{1}{\sqrt{1 + (\lambda a_1 + \mu a_2)^2}} \qquad \lambda = \mu = 5.$$

This is a well known case where the classical FM method fails (see Sethian and Vladimirsky (2003)), so it is a good benchmark for our PFM algorithm.
In Fig.1 we compare the level sets of the solutions obtained by the FM method and by the PFM method respectively. Focusing on the II and IV quadrant, we can see that the ellipses computed by the FM method are quite a bit distorted with respect to those computed by PFM method. As discussed in Section 3.2, this depends on the fact that in these regions gradient lines and characteristics fall in different cells of the grid, so that FM method propagates a wrong information producing an error as large as we move away from the origin.
In Fig.2 we show how the grid nodes are accepted by the two methods at some intermediate stage. It is quite evident that FM method computes the solution following its level sets (i.e. the gradient lines), while PFM method (see again II and IV quadrant in Fig.2(b), focusing on the corner points of the hexagon) employs information coming from the $x = 0$ and $y = 0$ axes, which belong to the right numerical domain of dependence, thus computing the correct solution. Indeed, the error in the $L^\infty$ norm with respect to the solution computed by the standard iterative scheme (5) is about 0.7 for the FM solution whereas the error is of order $10^{-15}$ for the PFM solution.



(a) (b)

Fig. 1. Level sets of the solution: (a) FM, (b) PFM



(a) (b)

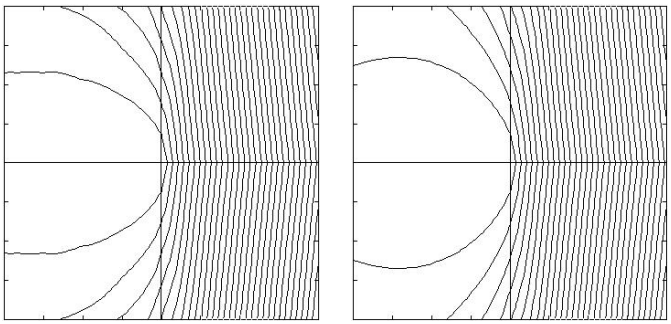
Fig. 2. Order of acceptance: (a) FM, (b) PFM

**Test 2** (Zermelo navigation problem)

By choosing in equation (3) the dimension $d = 2$ and

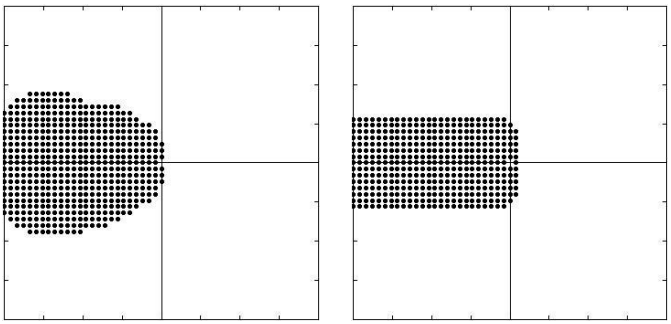
$$f(x, y, a) = 2.1a + (2, 0)\,, \qquad a \in B(0, 1)$$

we get 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. Again, the classical FM method fails due to the strong anisotropy of the problem and this gives us another interesting example to test our algorithm PFM.
In Fig.3 we compare the level sets of the solution obtained by the FM method and by the PFM method respectively. Here it is much more evident than in the previous test that there is a difference between the level sets in the half plane $\{x \leqslant 0\}$, which is a region with a strong anisotropy where gradient lines and characteristics do not fall in the same cell of the grid.
In Fig.4 we report the nodes accepted by the classical FM method and the PFM method at some intermediate stage. The error in the $L^\infty$ norm with respect to the solution computed by the standard iterative scheme (5) is about 0.4 for the FM solution and $10^{-15}$ for the PFM solution.



(a) (b)

Fig. 3. Level sets of the solution: (a) FM, (b) PFM



(a) (b)

Fig. 4. Order of acceptance: (a) FM, (b) PFM

**Test 3** (Tag-Chase game in $\mathbb{R}$ with state constraints)

Two boys $P$ and $E$ are running one after the other on the segment $[-2, 2]$. $P$ wants to catch $E$ in minimal time whereas $E$ wants to avoid the capture. Both of them are running with constant speed (respectively denoted by $v_P$ and $v_E$) and they can change their direction instantaneously. The game corresponds to equation (2) with the choices $d = 1$ and

$$f(x, y, a, b) = (v_P a, v_E b)\,, \quad a, b \in A = B = \{-1, 0, 1\}\,,$$
$$\Omega = \{(x, y) : x = y\}\,.$$

In the computations we have chosen $v_P = 2$, $v_E = 1$. Since the solution is symmetric with respect to the diagonal $\{x = y\}$ it suffices to compute the solution only in the upper triangle. We refer to Cristiani and Falcone (2009) for details on the implementation of state constraints.
In Fig.5 we show the value function computed by PFM and its level sets.
In Fig.6 we report the nodes accepted by the classical FM method and the PFM method at some intermediate stage.

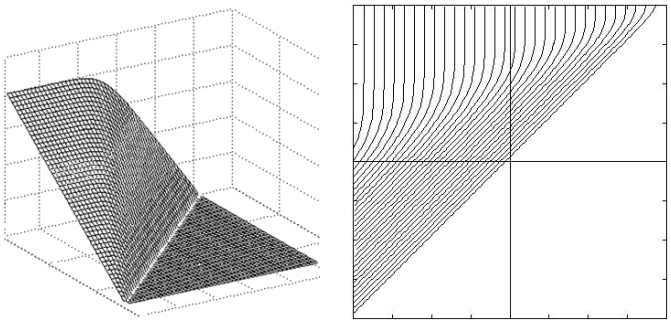
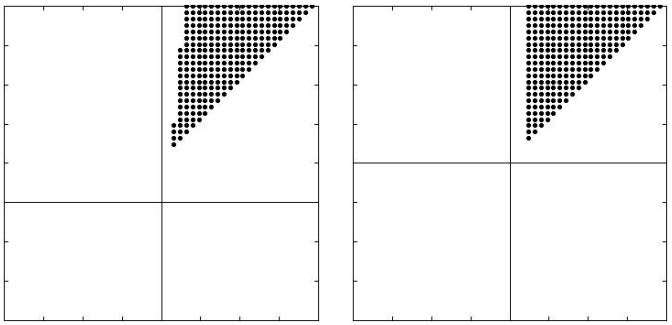


Fig. 5. The value function and its level sets



(a) (b)

Fig. 6. Order of acceptance: (a) FM, (b) PFM

The error in the $L^\infty$ norm with respect to the solution computed by the standard iterative scheme (5) is about $10^{-11}$ for the PFM solution. On the other hand, the classical FM method fails for this problem.

**Test 4** (Tag-Chase game in $\mathbb{R}^2$ with control constraints)

Here we extend the previous test to dimension 2 (without state constraints). The players $P$ and $E$ are now running in the plane $\mathbb{R}^2$. In reduced coordinates (see Falcone (2006) for details) the game corresponds to equation (2) with

$$f(x, y, a, b) = v_P a - v_E b\,, \quad A = B = B(0, 1)\,.$$

The pursuer $P$ has a constraint on his displacement directions. He can choose his control $a = (\cos\theta, \sin\theta)$ only for $\theta \in [\pi/4, 7\pi/4]$. We have chosen $v_P = 2$, $v_E = 1$ in order to guarantee the capture of $E$.
In Fig.7 we show the level sets of the value function computed by PFM and a couple of optimal trajectories in the real plane.
In Fig.8 we report the nodes accepted by the classical FM method and the PFM method at some intermediate stage.

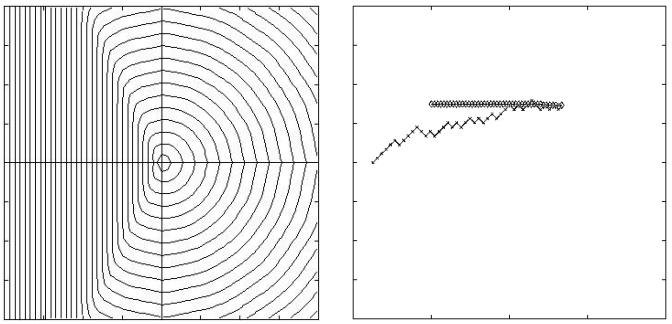
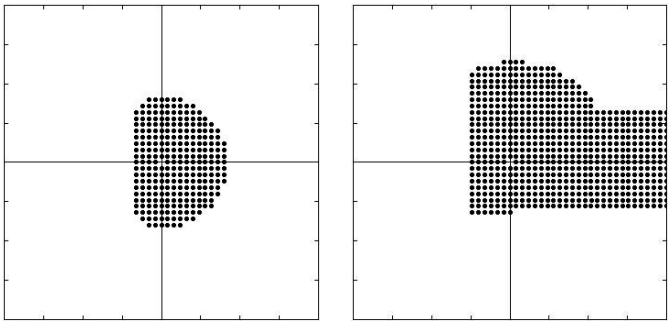


Fig. 7. Level sets and optimal trajectories



(a) (b)

Fig. 8. Order of acceptance: (a) FM, (b) PFM

It is interesting to note that while the Evader is running to the right on a straight line, the Pursuer follows a zig-zag path to intercept him. Since the Pursuer does not have the control direction corresponding to the interception line, he is forced to switch between two controls to approximate the optimal line of interception.
The error in the $L^\infty$ norm with respect to the solution computed by the standard iterative scheme (5) is about $10^{-10}$ for the PFM solution. Again, the classical FM method can not be applied to this problem.

## 6. CONCLUSIONS AND FUTURE WORK

We have proposed a new algorithm for the approximation of viscosity solutions of first order nonlinear partial differential equations related to control and games problems. In particular, as we have shown in our tests, the algorithm is able to compute the solution of a large variety of challenging nonlinear convex and nonconvex problems. Moreover, the method gives accurate results on equations which can not be solved by the original FM method.

To our knowledge, this is the first ordered upwind method for general Hamilton-Jacobi equations which is able to find the node to be accepted without the need of enlarging the *narrow band* as in Sethian and Vladimirsky (2003), or enlarging the region between the *narrow band* and the *accepted* zone as in Cristiani (2009), or using an *a priori* information as in Prados and Soatto (2005), where a subsolution of the equation is needed. Indeed, the correct value is found without computing other nodes than the *first* neighbors of the *accepted* zone.
The PFM method can be easily extended in any dimension and the "locality" of the method is also an advantage for memory usage, since only a single list containing *narrow band* nodes should be allocated to perform all the computations. In addition, more than one node can be accepted at a time.

These results motivate further investigations on the complexity of the method and its accuracy from a theoretical point of view. Several issues need to be improved in the implementation in order to reduce the amount of computations needed to get the solution. In our forthcoming papers we will proceed in these directions.

## ACKNOWLEDGEMENTS

We gratefully acknowledge Alexander Vladimirsky for some interesting discussions on the PFM method.

## REFERENCES

M. Bardi, I. Capuzzo Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Birkhäuser, 1997.

O. Bokanowski, E. Cristiani, H. Zidani. An efficient data structure and accurate scheme to solve front propagation problems. *J. Sci. Comput.*, 42:251–273, 2010.

F. Camilli, M. Falcone, P. Lanucara, A. Seghini. A domain decomposition method for Bellman equations. In D.E. Keyes, J. Xu, editors, *Domain Decomposition methods in Scientific and Engineering Computing*, Contemporary Mathematics 180, 477–483, AMS, 1994.

E. Carlini, E. Cristiani, N. Forcadel. A non-monotone Fast Marching scheme for a Hamilton-Jacobi equation modelling dislocation dynamics. In A. Bermudez de Castro, D. Gomez, P. Quintela, P. Salgado, editors, *Numerical Mathematics and Advanced Applications* (Proceedings of ENUMATH 2005, Santiago de Compostela, Spain, July 2005), 723–731, Springer, Berlin, 2006.

E. Carlini, M. Falcone, R. Ferretti. An efficient algorithm for Hamilton-Jacobi equations in high dimensions. *Computing and Visualization in Science*, 7:15–29, 2004.

E. Cristiani. A fast marching method for Hamilton-Jacobi equations modeling monotone front propagations. *J. Sci. Comput.*, 39:189–205, 2009.

E. Cristiani, M. Falcone. Fast semi-Lagrangian schemes for the Eikonal equation and applications. *SIAM J. Numer. Anal.*, 45:1979–2011, 2007.

E. Cristiani, M. Falcone. Fully-discrete schemes for the value function of Pursuit-Evasion games with state constraints, *Ann. Intl. Soc. Dyn. Games*, 10:177–206, 2009.

M. Falcone. The minimum time problem and its applications to front propagation. In A. Visintin, G. Buttazzo, editors, *Motion by mean curvature and related topics*, De Gruyter Verlag, Berlin, 1994.

M. Falcone. Numerical solution of dynamic programming equations. Appendix A in M. Bardi, I. Capuzzo Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Birkhäuser, 1997.

M. Falcone. Numerical methods for differential games based on partial differential equations. *International Game Theory Review*, 8:231–272, 2006.

M. Falcone, R. Ferretti. *Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations*. SIAM, to appear.

S. Kim. An $O(N)$ level set method for eikonal equations. *SIAM J. Sci. Comput.*, 22:2178–2193, 2001.

E. Prados, S. Soatto. Fast marching method for generic shape from shading. In N. Paragios, O. Faugeras, T. Chan, C. Schnoerr, editors, *Proceedings of the Third International Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, October 2005. LNCS 3752, 320–331. Springer, Berlin, 2005.

J.A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci. USA*, 93:1591–1595, 1996.

J.A. Sethian, A. Vladimirsky, Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms, *SIAM J. Numer. Anal.*, 41:325–363, 2003.

Y.R. Tsai, L.T. Cheng, S. Osher, H. Zhao. Fast sweeping algorithms for a class of Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 41:673–694, 2003.

J.N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. Autom. Control*, 40:1528–1538, 1995.

H. Zhao. A fast sweeping method for eikonal equations. *Math. Comp.*, 74:603–627, 2005.