SAPIENZA - UNIVERSITÀ DI ROMA

DOTTORATO DI RICERCA IN "METODI E MODELLI MATEMATICI PER LA TECNOLOGIA E LA SOCIETÀ"

XIX CICLO

Emiliano Cristiani

Fast Marching and Semi-Lagrangian Methods for Hamilton-Jacobi Equations with Applications

TESI DI DOTTORATO - Ph.D. THESIS

A.A. 2005-2006

Everything should be made as simple

as possible, but not simpler.

Albert Einstein

Contents

Acknowledgments

A special thank to my supervisor Maurizio Falcone.

I would like to thank Alessandra Seghini, Jean-Denis Durou and Frédéric Courteille for the interesting discussions on the Shape-from-Shading problem.
I would like also thank Manuela Sagona, Elisabetta Carlini, Regis Monneau and Nicolas Forcadel for the helpful discussions on Fast Marching methods and Marco Rorro and Piero Lanucara for supporting me in parallel computing. I wish to thank Martino Bardi for the useful suggestions about Fast Marching methods for differential games and Simone Cacace and Daniele Graziani for the passionate discussions on the Tag-Chase game. I thank Gabriella Bretti for some hints about LaTeX.
I thank Michel Delfour for his kind hospitality at Université de Montréal during my three-months stage and CASPUR (Consorzio interuniversitario per le Applicazioni di Supercalcolo per Università e Ricerca) where I ran the parallel algorithms developed in this thesis.
Finally I thank Erika Cotognini who is at my side throughout the journey of life.

Introduction

The numerical approximation of nonlinear partial differential equations is a challenging problem with a great impact on applications in several fields. This thesis deals with the development and the analysis of numerical methods for the resolution of first order nonlinear differential equations arising in the study of the propagation of fronts via the level set method, the Perspective Shape-from-Shading problem and the Pursuit-Evasion games with state constraints. In all these cases our goal is to compute an approximation of the weak solution in the viscosity sense [34, 15, 7].
In order to set this work into perspective let us mention some previous works on these topics that have been the basis for our research. It is well known that the level set method, introduced by Osher and Sethian in [72] for the study of a front evolution, produces a first order evolutive equation in the case of a normal velocity c which only depends on space and time,

$$u_t(x,t) + c(x,t)|\nabla u(x,t)| = 0 \tag{1}$$

whereas it yields a second order equation when the velocity also depends on the geometric properties of the front, typically on its curvature (see the monographs [84], [71]). The front at time t can be recovered as the 0-level set of the function $u(\cdot,t)$. The techniques used to approximate these problems are based on finite difference schemes, semi-Lagrangian schemes and, more recently, finite element schemes. It should also be mentioned that classical approximation methods require the computation of an approximate solution on every node of the grid at every iteration and are generally considered rather expensive. Starting with [85, 97] new methods have been proposed in order to reduce the computational effort and obtain the solution in a finite number of steps. These new methods save a lot of floating point operations concentrating the computation around the interface in a small subset of nodes which is called *narrow band.* Following this idea various algorithms have been proposed, generally these methods are called Fast Marching (FM) methods [83].
The contribution of this thesis to the development of Fast Marching methods is twofold. On one hand we propose [38] a new FM method for eikonal type equations based on the semi-Lagrangian approximation scheme presented in [48]. That FM scheme requires a larger *narrow band* with respect to the classical FM method based on finite difference therefore a new proof of convergence is needed. A detailed analysis has been carried out to show that the approximation scheme converges to the viscosity solution and that it is more accurate with respect to the classical FM method as presented in [85]. Moreover,

we have done a lot of work comparing the new semi-Lagrangian FM method with other FM methods in the literature on a wide range of problems. The second contribution in this area is the extension of the FM method to non-convex *minmax* Hamiltonians which arise in the study of differential games and to front propagation problems where the scalar velocity $c(x,t)$ is time-dependent and can change its sign in space and/or in time. This means that the front can increase or decrease and it can pass several times on the same point. As far as we know the papers [39, 25] are the first dealing with these problems.

The thesis deals also with an application to *image processing* which results in a rather complicated first order equation. The Shape-from-Shading problem is a classical inverse problem. It consists in reconstructing the *shape* of a three-dimensional object from the brightness variation (*shading*) in a greylevel photograph of that object [57]. Despite the huge amount of articles that deal with it, few real applications have been developed because the usual assumptions considered in the theory are too restrictive. Only recently two new PDE models have been proposed in order to include in the model the perspective deformation of the image, this allows to drop the unrealistic assumption requiring that the point of view is very far from the object (see Prados and Faugeras [77], Tankus, Sochen and Yeshurun [88], Courteille, Crouzil, Durou and Gurdjos [30]). Since those papers, the Shape-from-Shading was finally applied to some real problems like the reconstruction of faces [76, 78], the reconstruction of human organs [90] and the digitization of documents without scanners [31, 32, 29]. The first model, proposed in [30], takes into account the perspective deformation due to the finite distance between the camera and the scene. In this model the distance of the light source is infinite so that all the rays are parallel (in the sequel this model will be denoted by PSFS_∞). In the second model, proposed by Prados and Faugeras [77], the light source is placed at the optical center as in [70] so that this model is more realistic under flash lighting conditions (in the sequel this model will be denoted by PSFS_r).
The contribution of this thesis on the Perspective Shape-from-Shading problem consists in the proposition of two semi-Lagrangian schemes for the PSFS_r and PSFS_∞ models, an analysis of their convergence and a comparison of the results on some test problems based on synthetic and real images. Moreover, we address the problem of determining the uniqueness of weak solutions and show with two counterexamples that the PSFS_∞ and PSFS_r problems are not well-posed (see [40, 36]).

The thesis also deals with the numerical approximation of the Isaacs equation and Pursuit-Evasion games. It is well known (see for example [7]) that via the Dynamic Programming method one can characterize the value function v of a differential game as the unique viscosity solution of the Isaacs equation

$$v(x) + \min_{b\in B}\max_{a\in A}\{-f(x,a,b)\cdot\nabla v(x)\} - 1 = 0. \tag{2}$$

As far as the approximation of (2) is concerned we should mention that several convergence results as well as *a-priori* error estimates are now available, see for example [8, 2, 3, 14] (see also the survey papers [49] and [9, 47] for a comprehensive presentation of this theory respectively for control problems and games). We will present in the sequel the

main results of this approach which is based on a discretization in time of the original control/game problem followed by a discretization in space which result in a fixed point problem. This approach is natural for control problems since at every discretization we keep the meaning of the approximate solutions in terms of the control problem and we have *a-priori* error estimates which just depend on the data of the problem and on the discretization steps. Moreover, by the approximate value function one can easily compute approximate feedback controls and optimal trajectories. For the synthesis of feedback controls we have some error estimates in the case of control problems [50] but the problem is still open for games. Before starting our presentation let us quote other numerical approaches related to the approximation of games. The theory of *minmax* solutions has also a numerical counterpart developed by the Russian school (see [91, 73]) which is based on the construction of generalized gradients adapted to finite difference operators which approximate the value function. Another approximation for the value and for the optimal policies of dynamic zero-sum stochastic games has been proposed in [92, 94] and it is based on the approximation of the game by a finite state approximation (see also [93] for a numerical solution of zero-sum differential games with stopping time). The theory of viability [4] is based on set-valued analysis and gives a different characterization of the value function of control/game problems: the value function is the boundary of the viability kernel. The numerical counterpart of this approach is based on the approximation of the viability kernel and can be found in [21, 23, 24].
Finally, let us mention that other numerical methods based on the approximation of open-loop control have been proposed. The advantage is of course to replace the Dynamic Programming equation (which can be difficult to solve for high-dimensional problems) by a large system of ordinary differential equations. The interested reader can find in Pesch [74] a general presentation.
The contribution of this thesis on differential games consists in the proof of convergence of the fully-discrete semi-Lagrangian scheme for Pursuit-Evasion games with state constraints. It is obtained by coupling the convergence of the fully-discrete value function to the time-discrete value function (see [37]) with the convergence of the time-discrete value function already developed in [11]. We also state some results on the classical Tag-Chase game and its implementation, complementing this work with a number on numerical tests on parallel architectures.

Plan of the thesis

The thesis is organized as follows.
Chapter 1 is devoted to the theoretical background necessary to deal with Hamilton-Jacobi equations and viscosity solutions. Particular attention is given to the eikonal equation since it appears many times in a number of different contexts. In Section 1.2 we introduce the Hamilton-Jacobi-Bellman (HJB) equations related to optimal control problems and the minimum time problem. Section 1.3 shows how a front propagation problem can be seen as a minimum time problem and vice versa. This background result is crucial in the rest of the thesis. Then we deal with non-convex Hamiltonians which arise in the framework of 2-player zero-sum differential games. In Section 1.4 we present the Hamilton-Jacobi-Isaacs

(HJI) equation associated to the natural generalization of the minimum time problem and state a number of recent results about Pursuit-Evasion games with state constraints, *i.e.* when the two players have to keep the system in a given bounded domain. The final part of the chapter is dedicated to the semi-Lagrangian approximation of optimal control problems and differential games without state constraints. We introduce the time-discrete scheme via a Discrete Dynamic Programming Principle as well the fully-discrete scheme and analyze their main properties.

Chapter 2 is a small survey on fast and efficient numerical methods for the eikonal equation (and some other more general equations) which have been developed in the last ten years. We present the Fast Marching (FM) method (Sethian, 1996 [85]) and its further generalizations as Group Marching method (Kim, 2001 [63]), FM method for anisotropic front propagation (Sethian and Vladimirski, 2003 [86]) and FM method for HJB equations related to the Shape-from-Shading problem (Prados, 2006 [79]). Finally, we present the main features of the Fast Sweeping method (Zhao, 2005 [100]).

Chapter 3 is devoted to the original results achieved with regard to FM methods (see [38, 39, 25]). First of all, we focus our attention on the original FM method showing by an explicit example that, under a particular choice of the velocity of the front and a particular choice of the finite difference discretization, some imaginary solutions appear. To overcome this problem, we introduce a new CFL-like condition which guarantees real solutions in all the domain of computation. After that, a new semi-Lagrangian version of FM method is given. Since the definition of *narrow band* slightly changes, a new proof of convergence is needed. In Section 3.2.3 a number of numerical tests are performed in order to compare the classical FM method with its new semi-Lagrangian version. In Section 3.3 we extend the FM semi-Lagrangian method to non-convex Hamiltonians, in particular to *minmax* Hamiltonians which appear in the analysis of differential games. Some numerical tests show the potential of this new method. Finally, we present a second extension of the FM technique to non-monotone evolution of fronts, *i.e.* when the speed of propagation is time-dependent and it can change sign in space and/or in time. The new scheme can be successfully applied to the study of dislocation dynamics.

Chapter 4 deals with the numerical approximation and well/ill-posedness of the Perspective Shape-from-Shading problem. We introduce a semi-Lagrangian scheme for the two equations related to the existing models for Perspective Shape-from-Shading, emphasizing if and how the classical convex/concave ambiguity changes according to the model's modifications (see [40, 36]).

Chapter 5 is devoted to the semi-Lagrangian approximation of Pursuit-Evasion games with state constraints. We prove that the solution of the fully-discrete scheme converges to the solution of the time-discrete scheme as the mesh size goes to 0. This result can be coupled with a recent result of Bardi et al. [11] to obtain (under suitable assumptions) the convergence of the fully-discrete scheme to the continuous problem as the time and space steps go to 0. Then we present some minor results with regard to the Tag-Chase game and its implementation. We prove that if a) the game is played in a bounded convex domain, b) the game ends when the distance between the Pursuer and Evader is smaller than a given positive tolerance and c) the two players run with the same velocity, the time of capture is finite (note that in a unbounded domain this is not true). Then we recall a fast method to make interpolations in high dimensional spaces studied in [26] giving a precise error estimate. Finally, we show how it is possible to take into account the symmetries of the problem in order to reduce the high computational cost whenever the game's field is a square. In Section 5.4 we present a number of numerical experiments for constrained Tag-Chase game. In addition to classical tests already studied by Alziary de Roquefort [3] we perform some test in which the velocity of the Evader is equal or even greater than the velocity of the Pursuer. To our knowledge this is the first time such an experiments are performed and the results are quite interesting.

Chapter 1

Background results on Hamilton-Jacobi equations

In this chapter we present all the definitions and the basic theoretical results we will refer to in the following. First of all we introduce the notion of *viscosity solution* of the Hamilton-Jacobi equation

$$H(x, u(x), \nabla u(x)) = 0\,, \qquad x \in \Omega \tag{HJ}$$

where Ω is an open domain of $\mathbb{R}^n$ and the Hamiltonian $H = H(x, r, p)$ is a continuous real valued function on $\Omega \times \mathbb{R} \times \mathbb{R}^n$. This notion allows us to obtain important existence and uniqueness results for some equations of the form (HJ). Then, we will focus our attention on optimal control problems and Hamilton-Jacobi-Bellman equation showing the strict relationship with front propagation problems in the framework of level set methods. In Section 1.4 we present the most important theoretical results for the Hamilton-Jacobi-Isaacs equation which arises in the study of 2-player zero-sum differential games. We focus on Pursuit-Evasion games and Pursuit-Evasion games with state constraints, *i.e.* in the case the two players have to keep the system in a given bounded domain. In the last section we introduce the semi-Lagrangian approximation of optimal control problems and differential games without state constraints. We derive the time-discrete scheme via a Discrete Dynamic Programming Principle as well the fully-discrete scheme and we analyze their main properties.
If no specific indication is given, the reader can find the proofs of all theorems in [7] and [47].

1.1 Viscosity solutions

It is well known that equation (HJ) is in general not well-posed. It is possible to show several examples in which any classical (that is of class C^1) solution exists or infinite weak (that is *a.e.* differentiable) solutions exist. Even for the very simple unidimensional eikonal equation complemented with homogeneous Dirichlet boundary conditions

$$\begin{cases} |\nabla u(x)| = 1\,, & x \in (-1, 1) \\ u(x) = 0\,, & x = \pm 1 \end{cases} \tag{1.1}$$

multiple weak solutions are found (see Fig. 1.1). The theory of *viscosity solution*s was

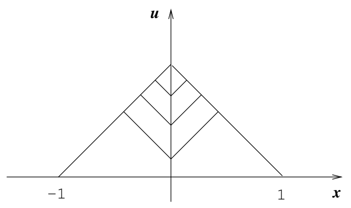


Figure 1.1: multiple *a.e.* differentiable solutions of the eikonal equation (1.1)

developed in order to overcome these problems. It gives a tool to get uniqueness of the solution and in some cases also to select the correct physical solution among all solutions of the equation. We give here two equivalent definitions of *viscosity solution.*

Definition 1.1 (I version) *A continuous function u is a* viscosity solution *of the equation (HJ) if the following conditions are satisfied:*

(i) $H(x,u(x),p) \le 0$ *for all* $x \in \mathbb{R}^n$, *for all* $p \in D^+u(x)$ (viscosity subsolution)

(ii) $H(x,u(x),q) \ge 0$ *for all* $x \in \mathbb{R}^n$, *for all* $q \in D^-u(x)$ (viscosity supersolution)

where

$$D^+u(x) = \left\{ p \in \mathbb{R}^n : \limsup_{y\to x} \frac{u(y)-u(x)-p\cdot(y-x)}{|y-x|} \le 0 \right\}$$

$$D^-u(x) = \left\{ q \in \mathbb{R}^n : \liminf_{y\to x} \frac{u(y)-u(x)-q\cdot(y-x)}{|y-x|} \ge 0 \right\}.$$

Definition 1.2 *(*II version) *A continuous function u is a* viscosity solution *of the equation (HJ) if the following conditions are satisfied:*

(i) for any test function $\phi \in C^1(\Omega)$, *if* $x_0 \in \Omega$ *is a local maximum point for* $u-\phi$, *then*

$$H(x_0,u(x_0),\nabla\phi(x_0)) \le 0 \qquad \text{(viscosity subsolution)}$$

(ii) for any test function $\phi \in C^1(\Omega)$, *if* $x_1 \in \Omega$ *is a local minimum point for* $u-\phi$, *then*

$$H(x_1,u(x_1),\nabla\phi(x_1)) \ge 0 \qquad \text{(viscosity supersolution).}$$

The motivation for the terminology ”viscosity solutions” is that this solution can be recovered as the limit function $u = \lim_{e\to 0^+} u^\varepsilon$ where $u^\varepsilon \in C^2(\Omega)$ is the classical solution of the perturbed problem

$$-\varepsilon\triangle u^\varepsilon + H(x,u^\varepsilon,\nabla u^\varepsilon) = 0\,, \qquad x \in \Omega$$

in the case u^ε exists and converges locally uniformly to some continuous function u. This method is named *vanishing viscosity.*

Before recalling the existence and uniqueness results for equation (HJ), we want to show the useful parallelism between this equation and its evolutive version

$$u_t(y,t) + \widehat{H}(y,t,u(y,t),\nabla_y u(y,t)) = 0\,, \qquad (y,t) \in (0,T_f) \times D \tag{1.2}$$

where D is an open subset of $\mathbb{R}^{n-1}$, $T_f > 0$ is the final time and ∇_y is the gradient with respect to y. Indeed, by the positions

$$x = (y,t)\,, \quad \Omega = (0,T_f) \times D \subseteq \mathbb{R}^n\,, \quad H(x,r,p) = p_n + \widehat{H}(x,r,p_1,\ldots,p_{n-1})\,,$$

equation (1.2) is reduced to the form (HJ). As a consequence, all definitions and results for one equation are easily transferred to the other.

In the following we present some comparison results between viscosity sub- and supersolutions. As simple corollary, each comparison result produces a uniqueness theorem for the associated Dirichlet problem.

Theorem 1.3 *Let Ω be a bounded open subset of $\mathbb{R}^n$. Assume that $u_1, u_2 \in C(\overline{\Omega})$ are, respectively, viscosity sub- and supersolution of*

$$u(x) + H(x,\nabla u(x)) = 0\,, \qquad x \in \Omega \tag{1.3}$$

and

$$u_1 \le u_2 \qquad \text{on } \partial\Omega.$$

Assume also that H satisfies

$$|H(x,p) - H(y,p)| \le \omega_1(|x-y|(1+|p|)), \tag{1.4}$$

for $x, y \in \Omega$, $p \in \mathbb{R}^n$, where ω_1 is a modulus*, that is $\omega_1 : [0,+\infty) \to [0,+\infty)$ is continuous nondecreasing with $\omega_1(0) = 0$. Then $u_1 \le u_2$ in $\overline{\Omega}$.*

Theorem 1.4 *Assume that $u_1, u_2 \in C(\mathbb{R}^n) \cap L^\infty(\mathbb{R}^n)$ are, respectively, viscosity sub- and supersolution of*

$$u(x) + H(x,\nabla u(x)) = 0\,, \qquad x \in \mathbb{R}^n \tag{1.5}$$

Assume also that H satisfies (1.4) *and*

$$|H(x,p) - H(x,q)| \le \omega_2(|p-q|)\,, \qquad \text{for all } x,p,q \in \mathbb{R}^n. \tag{1.6}$$

where ω_2 is a modulus. *Then $u_1 \le u_2$ in $\mathbb{R}^n$.*

Remark 1.5 *Theorem 1.4 can be generalized to cover the case of a general unbounded open set $\Omega \subset \mathbb{R}^n$. Moreover, the assumptions $u_1, u_2 \in C(\mathbb{R}^n) \cap L^\infty(\mathbb{R}^n)$ can be replaced by $u_1, u_2 \in UC(\mathbb{R}^n)$.*

A comparison result can be formulated for the more general case

$$H(x, \nabla u(x)) = 0\,, \qquad x \in \Omega \tag{1.7}$$

only if we assume the convexity of H with respect to the p variable. This assumption plays a key role in many theoretical results.

Theorem 1.6 *Let Ω be a bounded open subset of $\mathbb{R}^n$. Assume that $u_1, u_2 \in C(\overline{\Omega})$ are, respectively, viscosity sub- and supersolution of* (1.7) *with $u_1 \leq u_2$ on $\partial\Omega$. Assume also that H satisfies* (1.4) *and the two following conditions*

$$p \mapsto H(x,p) \text{ is convex on } \mathbb{R}^n \text{ for each } x \in \Omega; \tag{1.8}$$

$$\begin{cases} \text{there exists } \phi \in C(\overline{\Omega}) \cap C^1(\Omega) \text{ such that } \phi \leq u_1 \text{ in } \overline{\Omega} \\ \text{and } \sup\limits_{x \in \Omega'} H(x, \nabla\phi(x)) < 0\,, \text{ for all } \Omega' \subset\subset \Omega. \end{cases} \tag{1.9}$$

Then, $u_1 \leq u_2$ in Ω.

1.1.1 The eikonal equation

The classical model problem for (1.7) is the eikonal equation of geometric optics

$$c(x)|\nabla T(x)| = 1\,, \qquad x \in \Omega. \tag{1.10}$$

Theorem 1.6 applies to the eikonal equation (1.10) whenever $c(x) \in Lip(\Omega)$ and it is strictly positive. In fact (1.9) is satisfied by taking $\phi(x) \equiv \min\limits_{\overline{\Omega}} u_1$.
It easy to prove that the distance function from an arbitrary set $S \subseteq \mathbb{R}^n$, $S \neq \emptyset$ defined by

$$d_S(x) = d(x, S) := \inf_{z \in S} |x - z| = \min_{z \in \overline{S}} |x - z|$$

is continuous in $\mathbb{R}^n$. Moreover, for smooth ∂S it is smooth near ∂S and satisfies in the classical sense the equation (1.10) in $\mathbb{R}^n \backslash \overline{S}$ for $c(x) \equiv 1$.
For a general set S, it can be shown that the function d_S is the unique *viscosity solution* of $|\nabla T(x)| = 1$ in $\mathbb{R}^n \backslash \overline{S}$.

Remark 1.7 *If we consider the eikonal equation in the form $|\nabla T(x)| = f(x)$ where f is a function vanishing at least in a single point in Ω, then the uniqueness result does not hold. This situation is referred to as* degenerate eikonal equation. *It can be proved that in this case many viscosity or even classical solutions may appear. Consider for example the equation $|u'| = 2|x|$ for $x \in (-1, 1)$ complemented by Dirichlet boundary condition $u = 0$ at $x = \pm 1$. It is easy to see that $u_1(x) = x^2 - 1$ and $u_2(x) = 1 - x^2$ are both classical solutions.*

1.2 Optimal control problems

We introduce here the basic notations and theory for optimal control controls, focusing in particular on minimum time problems and the related Hamilton-Jacobi-Bellman equation. Let us consider the controlled nonlinear dynamical system

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

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 there exists a constant $L > 0$ such that

$$|f(y_1, a) - f(y_2, a)| \leq L|y_1 - y_2| \qquad \text{for all } y_1, y_2 \in \mathbb{R}^n, \ a \in A. \tag{1.11}$$

By Caratheodory's theorem the choice of measurable controls guarantees that for any given $a(\cdot) \in \mathcal{A}$, there is a unique trajectory solution of (DS) which will be denoted by $y_x(t; a(\cdot))$.
The final goal is to find an optimal control $a^*(t)$ such that the corresponding trajectory $y_x(t; a^*(\cdot))$ is the "most convenient" one with respect to some given criterion between all possible trajectories starting from x.

1.2.1 The infinite horizon problem

In the infinite horizon problem the *cost functional* J associated to every trajectory which has to be minimized is

$$J(x, a(\cdot)) = \int_0^\infty l(y_x(t; a(\cdot)), a(t)) e^{-\lambda t} dt$$

where the constant interest rate λ is strictly positive and the running cost $l(x, a) : \mathbb{R}^n \times A \to \mathbb{R}$ is continuous in both variables, bounded and there exists $C > 0$ such that

$$|l(x, a) - l(y, a)| \leq C|x - y| \qquad \text{for all } x, y \in \mathbb{R}^n, \ a \in A.$$

We are looking for the *value function* $v(x)$ defined as

$$v(x) := \inf_{a(\cdot) \in \mathcal{A}} J(x, a(\cdot))$$

and possibly for the optimal control

$$a^*(\cdot) = \arg \min_{a(\cdot) \in \mathcal{A}} J(x, a(\cdot)).$$

Proposition 1.8 *Under the assumption on f and l introduced above, the value function v is bounded and Lipschitz continuous.*

Proposition 1.9 ***(Dynamic Programming Principle).*** *For all $x \in \mathbb{R}^n$ and $t > 0$ the value function satisfies*

$$v(x) = \inf_{a(\cdot)\in\mathcal{A}} \left\{ \int_0^t l(y_x(s; a(\cdot)), a(s))e^{-\lambda s} ds + v(y_x(t; a(\cdot)))e^{-\lambda t} \right\}. \tag{1.12}$$

Quoting Bellman [16], equation (1.12) means that "An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" where in our setting the "decisions" are the choices of the control.

Proposition 1.10 *The value function v is a viscosity solution of*

$$\lambda v + \sup_{a\in A} \{-f(x, a) \cdot \nabla v - l(x, a)\} = 0\,, \qquad x \in \mathbb{R}^n. \tag{1.13}$$

The equation (1.13) is called the Hamilton-Jacobi-Bellman equation for the infinite horizon problem.

1.2.2 The finite horizon problem

In the finite horizon problem the *cost functional* J associated to every trajectory which has to be minimized is

$$J(x, t, a(\cdot)) = \int_0^t l(y_x(s; a(\cdot)), a(s))e^{-\lambda s} ds + g(y_x(t; a(\cdot)))e^{-\lambda t}$$

where $\lambda \geq 0$, $l(x, a)$ satisfies the same hypothesis as in the infinite horizon case above and the terminal cost $g : \mathbb{R}^n \to \mathbb{R}$ is bounded and uniformly continuous. We are looking for the *value function* $v(x)$ defined as

$$v(x, t) := \inf_{a(\cdot)\in\mathcal{A}} J(x, t, a(\cdot)).$$

Proposition 1.11 *Under the assumption on f, l and g introduced above, the value function v is bounded and continuous in $\mathbb{R}^n \times [0, T]$ for all $T > 0$.*

Proposition 1.12 ***(Dynamic Programming Principle).*** *For all $x \in \mathbb{R}^n$ and $0 < \tau \leq t$ the value function satisfies*

$$v(x, t) = \inf_{a(\cdot)\in\mathcal{A}} \left\{ \int_0^\tau l(y_x(s; a(\cdot)), a(s))e^{-\lambda s} ds + v(y_x(\tau; a(\cdot)))e^{-\lambda\tau}, t - \tau \right\}. \tag{1.14}$$

Proposition 1.13 *The value function v is the unique viscosity solution of*

$$\begin{cases} v_t + \lambda v + \sup_{a\in A} \{-f(x, a) \cdot \nabla_x v - l(x, a)\} = 0 & (x, t) \in \mathbb{R}^n \times (0, +\infty) \\ v(x, 0) = g(x) & x \in \mathbb{R}^n. \end{cases} \tag{1.15}$$

The equation (1.15) is called the Hamilton-Jacobi-Bellman equation for the finite horizon problem.

1.2.3 The minimum time problem

In the minimum time problem the cost associated to every trajectory which has to be minimized is the time needed by the system to reach a given closed *target* $\mathcal{T} \subset \mathbb{R}^n$, that is

$$J(x, a(\cdot)) = t_x(a(\cdot))$$

where

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

The *value function* is

$$T(x) := \inf_{a(\cdot)\in\mathcal{A}} t_x(a(\cdot)). \tag{1.16}$$

We will refer to T also as the minimum time function and we set $T = 0$ on $\mathcal{T}$.

Definition 1.14 *The* reachable set *is* $\mathcal{R} := \{x \in \mathbb{R}^n : T(x) < +\infty\}$, i.e. *it is the set of starting points from which it is possible to reach the target.*

Note that the reachable set depends on the target, the dynamics and on the set of admissible controls and it is not a datum in our problem.

Proposition 1.15 ***(Dynamic Programming Principle)**. For all $x \in \mathcal{R}$, $0 \leq t < T(x)$ (so that $x \notin \mathcal{T}$) the value function satisfies*

$$T(x) = \inf_{a(\cdot)\in\mathcal{A}} \{t + T(y_x(t; a(\cdot)))\}. \tag{DPP}$$

Let us derive formally the Hamilton-Jacobi-Bellman equation associated to the minimum time problem from the Dynamic Programming Principle. Rewrite (DPP) as

$$T(x) - \inf_{a(\cdot)\in\mathcal{A}} T(y_x(t; a(\cdot))) = t$$

and divide by $t > 0$

$$\sup_{a(\cdot)\in\mathcal{A}} \left\{ \frac{T(x) - T(y_x(t; a(\cdot)))}{t} \right\} = 1 \qquad \text{for all } \ t < T(x).$$

We want to pass to the limit as $t \to 0^+$.
Assume that T is differentiable at x and $\lim_{t\to 0^+}$ commutes with $\sup_{a(\cdot)}$. Then, if $\dot{y}_x(0; a(\cdot))$ exists,

$$\sup_{a(\cdot)\in\mathcal{A}} \{-\nabla T(x) \cdot \dot{y}_x(0; a(\cdot))\} = 1\,,$$

so that, if $a(0) = a_0$, we get

$$\max_{a_0\in A} \{-\nabla T(x) \cdot f(x, a_0)\} = 1. \tag{1.17}$$

Note that in the final equation (1.17) the maximum is taken over A and not over the set of measurable controls $\mathcal{A}$.

Proposition 1.16 *If $\mathcal{R}\backslash\mathcal{T}$ is open and $T \in C(\mathcal{R}\backslash\mathcal{T})$, then T is a viscosity solution of*

$$\max_{a\in A}\{-f(x,a)\cdot\nabla T(x)\} - 1 = 0\,, \qquad x \in \mathcal{R}\backslash\mathcal{T}. \tag{1.18}$$

Natural boundary conditions for (1.18) are

$$\begin{cases} T(x) = 0 & x \in \partial\mathcal{T} \\ \lim\limits_{x\to\partial\mathcal{R}} T(x) = +\infty. \end{cases}$$

In order to achieve uniqueness of the viscosity solution of equation (1.18) is useful an exponential transformation named *Kružkov transform*

$$v(x) := \begin{cases} 1 - e^{-T(x)} & \text{if } T(x) < +\infty \quad (x \in \mathcal{R}) \\ 1 & \text{if } T(x) = +\infty \quad (x \notin \mathcal{R}) \end{cases} \tag{1.19}$$

It easy to check (at least formally) that if T is a solution of (1.18) than v is a solution of

$$v(x) + \max_{a\in A}\{-f(x,a)\cdot\nabla v(x)\} - 1 = 0\,, \qquad x \in \mathbb{R}^n\backslash\mathcal{T}. \tag{1.20}$$

This transformation has many advantages.

1. The equation for v has the form (1.3) so that we can apply the uniqueness result already developed in this chapter.
2. v takes values in $[0,1]$ whereas T is generally unbounded (for example if f vanishes in some points) and this helps in the numerical approximation.
3. The domain in which the equation has to be solved is no more unknown.
4. One can always reconstruct T and $\mathcal{R}$ from v by the relations

$$T(x) = -\ln(1 - v(x))\,, \qquad \mathcal{R} = \{x : v(x) < 1\}.$$

Proposition 1.17 *v is the unique viscosity solution of*

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

Computation of optimal feedback and trajectories

As mentioned above, the final goal of the minimum time problem (and of every control problem) is to find the optimal control

$$a^*(\cdot) = \arg\min_{a(\cdot)\in\mathcal{A}} t_x(a(\cdot))$$

and the associated optimal trajectory, *i.e.* the solution $y^*(t)$ of

$$\begin{cases} \dot{y}(t) = f(y(t), a^*(t))\,, & t > 0 \\ y(0) = x \end{cases} \tag{1.21}$$

The next theorem shows how to compute a^* in feedback form, *i.e.* as a function of the state $y(t)$. This form is obviously better then open-loop optimal control where a^* depends only on time t. In fact, the feedback control leads the state to the target even in presence of perturbations or noise.

Theorem 1.18 *Let $T \in C^1(\mathcal{R}\backslash\mathcal{T})$ be the unique solution of* (1.18) *and define $a_*(x)$ as*

$$a_*(x) := \arg\max_{a\in A}\{-f(x,a)\cdot\nabla T(x)\}\,, \qquad x \in \mathcal{R}\backslash\mathcal{T}. \tag{1.22}$$

Let $y^*(t)$ be the solution of

$$\begin{cases} \dot{y}^*(t) = f(y^*(t), a_*(y^*(t)))\,, & t > 0 \\ y^*(0) = x \end{cases}. \tag{1.23}$$

Then, $a^*(t) = a_*(y^*(t))$ is the optimal control.

Proof. First we note that

$$\frac{d}{ds}T(y^*(s)) = \dot{y}^*(s)\cdot\nabla T(y^*(s)) = f(y^*(s), a_*(y^*(s)))\cdot\nabla T(y^*(s))$$

and then, by choosing $x = y^*(s)$ in (1.18),

$$\frac{d}{ds}T(y^*(s)) = -1. \tag{1.24}$$

By (1.24) we have

$$T(y^*(t_x(a_*(y^*(t))))) - T(x) = \int_0^{t_x(a_*(y^*(t)))}\frac{d}{ds}T(y^*(s))ds = -t_x(a_*(y^*(t))).$$

Since $T(y^*(t_x(a_*(y^*(t))))) = 0$, we obtain

$$T(x) = t_x(a_*(y^*(t)))$$

and thus the conclusion by the definition of $T(x)$. ■

Note that the differentiability assumption on T in Theorem 1.18 can be relaxed. We do not give details on this since we will deal only with approximated optimal trajectories computed by the discrete solution of (1.18).

1.3 Front propagation problems

Given a closed $(n-1)$-dimensional hypersurface $\Gamma_{t=0}$, we want to produce an *Eulerian* formulation for the motion of the hypersurface Γ_t propagating along its exterior normal direction with speed $c(x)$. The main idea of the level set methodology (see [84]) is to embed this propagating interface as the zero-level set of a higher dimensional function ϕ defined as

$$\phi(x,t) := \pm d(x,\Gamma_t)\,, \qquad x \in \mathbb{R}^n,\ t \geq 0$$

where $d(x,\Gamma_t)$ is the distance between a point $x \in \mathbb{R}^n$ and the interface Γ_t with the "+" sign if x is outside the interface and the "-" sign if it is inside. The datum of the problem is the initial condition $\phi(x,0)$ that is the signed distance function to the initial hypersurface. Once we defined ϕ we can recover a partial differential equations which models the front propagation. Let $x(t)$ be the trajectory that a generic point of the hypersurface traces in the space. We want that

$$\phi(x(t),t) = 0 \qquad \forall t > 0$$

that is the point $x(t)$ is a point of the interface at time t for every $t > 0$. Differentiating with respect to t we obtain

$$\phi_t + \nabla\phi(x(t),t) \cdot x_t(t) = 0\,, \qquad t > 0$$

where ∇ is the gradient with respect to x. We know that the interface is propagating along its exterior normal direction, *i.e.* $x_t(t) = c(x)\eta(x)$ for all $x \in \Gamma_t$, where $\eta(x)$ is the exterior normal unit vector to the interface. Since

$$\eta(x(t)) = \frac{\nabla\phi(x(t),t)}{|\nabla\phi(x(t),t)|},$$

we formally obtain that the equation modeling the evolution of the interface is the following evolutionary Hamilton-Jacobi equation

$$\begin{cases} u_t(x,t) + c(x)|\nabla u(x,t)| = 0 & x \in \mathbb{R}^n,\ t > 0 \\ u(x,0) = \pm d(x,\Gamma_0) & x \in \mathbb{R}^n. \end{cases} \tag{1.25}$$

and the front at any time is given by

$$\Gamma_t = \{x \in \mathbb{R}^n : u(x,t) = 0\}.$$

Remark 1.19 *As remarked in [42] it would be tempting to say that for each t the function ϕ is the solution of the equation (1.25) not only on Γ_t but in all $\mathbb{R}^n$. This is not true because the equivalence $x_t(t) = c(x)\eta(x)$ holds only on Γ_t. Therefore, the zero-level of the solution $u(x,t)$ of (1.25) is actually the interface at time t but u is* not *the distance function to Γ_t for any $t > 0$.*
In order to get an equation which describes the evolution of the distance-to-Γ_t function ϕ we have to replace the equation (1.25) with the following equation (see also [55])

$$\begin{cases} \phi_t(x,t) + c(x - \phi(x,t)\nabla\phi(x,t)) = 0 & x \in \mathbb{R}^n,\ t > 0 \\ \phi(x,0) = \pm d(x,\Gamma_0) & x \in \mathbb{R}^n. \end{cases} \tag{1.26}$$

If the evolution of the front is strictly monotone, *i.e.* $c(x) > 0$ for all $x \in \mathbb{R}^n$, then equation (1.25) can be written in a stationary form introducing the function

$$T(x) := u(x,t) + t \tag{1.27}$$

and then we can recover the interface by the knowledge of T at any time using

$$\Gamma_t = \{x \in \mathbb{R}^n : T(x) = t\}.$$

Substituting (1.27) in the first equation of (1.25) we obtain

$$c(x)|\nabla T(x)| = 1$$

and then we can reformulate the boundary value problem as

$$\begin{cases} c(x)|\nabla T(x)| = 1 & x \in \mathbb{R}^n \backslash \Omega_0 \\ T(x) = 0 & x \in \partial\Omega_0 \end{cases} \tag{1.28}$$

where Ω_0 is a subset of $\mathbb{R}^n$ such that $\partial\Omega_0 = \Gamma_0$. Note that the Dirichlet boundary condition $T(x) = 0$ on $\partial\Omega_0$ is quite natural considering that $T(x)$ represents the time needed by the interface to reach the point x.

Let us recover equation (1.28) in the setting of optimal control problems. Let us consider again the problem (DS) and choose

$$f(y, a(t)) = -c(y)a(t)\,, \quad c : \mathbb{R}^n \to \mathbb{R}\,, \quad c > 0$$

and $\mathbb{R}^n \supset A = B(0,1)$ where B(0,1) is the unit ball centered in the origin. This choice means that the controller can move the state of the system with speed $c > 0$ (which depends on the position of the system itself) in every direction he wants. Obviously he can reach the target from every initial position x and then $\mathcal{R} = \mathbb{R}^n$. Equation (1.18) becomes

$$c(x) \max_{a \in B(0,1)} \{-a \cdot \nabla T(x)\} = 1\,, \qquad x \in \mathbb{R}^n \backslash \mathcal{T} \tag{1.29}$$

and it easy to check that the unit vector a which realizes the max in the above equation is

$$a^*(x) := \arg \max_{a \in B(0,1)} \{-a \cdot \nabla T(x)\} = -\frac{\nabla T(x)}{|\nabla T(x)|} \tag{1.30}$$

so that equation (1.29) can be written as

$$c(x)|\nabla T(x)| = 1\,, \qquad x \in \mathbb{R}^n \backslash \mathcal{T}.$$

If $\Omega_0 = \mathcal{T}$, we just proved that the front at time t is the set of the points which can be driven to the target in time t. Moreover, the trajectory $x(t)$ of a point of the front (*i.e.* a characteristic curve of equation (1.28)) coincides with the optimal trajectory starting at $x(t)$ and reaching the target in time t. Lastly, by (1.30) we obtain that characteristic and gradient directions coincide. This is not true for a general control/front propagation problem as we will see in Section 2.2.

1.4 Differential games

In this section we introduce basic notations and theory for 2-player zero-sum deterministic differential games and the related Hamilton-Jacobi-Isaacs equation. Our setting will be very similar to that for minimum time problem, the main difference here is that two players instead of one can control the dynamics of the system. The two players are opponents since the first player wants to minimize the cost associate to the solution of the system whereas the second player wants to maximize it. The first player is called P, standing for *Pursuer*, the second is called E, standing for *Evader*. The game is said "zero-sum" because any gain of one player is a loss of the same size for the other player.
The modified version of the dynamical system (DS) is

$$\begin{cases} \dot{y}(t) = f(y(t), a(t), b(t))\,, & t > 0 \\ y(0) = x \end{cases} \tag{1.31}$$

where
$y(t)$ is the state of the system,
$a(\cdot) \in \mathcal{A}$ and $b(\cdot) \in \mathcal{B}$ are respectively the controls of the first and the second player, $\mathcal{A}$ and $\mathcal{B}$ being the sets of admissible controls defined as

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

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

and A and B are given compact sets of $\mathbb{R}^m$. Assume again $f : \mathbb{R}^n \times A \times B \to \mathbb{R}^n$ is continuous in all three variables and there exists $L > 0$ such that

$$|f(y_1, a, b) - f(y_2, a, b)| \le L|y_1 - y_2| \qquad \text{for all } y_1, y_2 \in \mathbb{R}^n,\ a \in A,\ b \in B. \tag{1.32}$$

We will denote the solution of (1.31) by $y_x(t; a(\cdot), b(\cdot))$. We will only deal with the natural extension of the minimum time problem presented in section 1.2.3 so we define the *payoff* of the game as

$$t_x(a(\cdot), b(\cdot)) := \begin{cases} \min\{t : y_x(t; a(\cdot), b(\cdot)) \in \mathcal{T}\} & y_x(t; a(\cdot), b(\cdot)) \in \mathcal{T} \text{ for some } t \ge 0 \\ +\infty & y_x(t; a(\cdot), b(\cdot)) \notin \mathcal{T} \text{ for all } t \ge 0 \end{cases}$$

where $\mathcal{T} \subseteq \mathbb{R}^n$ is a given closed target.

Nonanticipating strategies

Defining the value function T for this problem is not an easy task. A direct generalization of (1.16) leads to

$$\inf_{a(\cdot)\in\mathcal{A}} \sup_{b(\cdot)\in\mathcal{B}} t_x(a(\cdot), b(\cdot))$$

which is quite unrealistic because it means that player P can choose his control a using the information of the whole future response of player E to any control function a and this will give him a big advantage. Moreover this approach is not fit to be analyzed by the Dynamic Programming method.
A more unbiased information pattern can be modeled by means of the notion of *nonanticipating strategies* (see [44] and the references therein).

Definition 1.20 *A* strategy *for the first player is a map $\alpha : \mathcal{B} \to \mathcal{A}$; it is* nonanticipating *if $\alpha \in \Gamma$, where*

$$\Gamma = \{\alpha : \mathcal{B} \to \mathcal{A} : b(t) = \tilde{b}(t) \textit{ for all } t \le t' \textit{ implies } \alpha[b](t) = \alpha[\tilde{b}](t) \textit{ for all } t \le t'\}.$$

Similarly, a strategy *for the second player is a map $\beta : \mathcal{A} \to \mathcal{B}$; it is* nonanticipating *if $\beta \in \Delta$, where*

$$\Delta = \{\beta : \mathcal{A} \to \mathcal{B} : a(t) = \tilde{a}(t) \textit{ for all } t \le t' \textit{ implies } \beta[a](t) = \beta[\tilde{a}](t) \textit{ for all } t \le t'\}.$$

The above definition is fair with respect to the two players. In fact, if the player who controls $a(\cdot)$ (respectively $b(\cdot)$) chooses his control in Γ (respectively Δ) he will not be influenced by the future choices of the other player. As a consequence, all decisions are made by the knowledge of only the present and past history.
Now we can define the lower and the upper values of a game.

Definition 1.21 *The* lower value *of a game is*

$$T_{low}(x) = \inf_{\alpha\in\Gamma}\sup_{b\in\mathcal{B}} t_x(\alpha[b], b);$$

the upper value *is*

$$T_{upp}(x) = \sup_{\beta\in\Delta}\inf_{a\in\mathcal{A}} t_x(a, \beta[a]).$$

If $T_{low} \equiv T_{upp}$, we say that the game has a value and we set $T = T_{upp} = T_{low}$.

Hereafter we will consider only games which have a value, therefore we do not make any distinction between T_{low} and T_{upp} but note that the following results are valid in general only for T_{low}.
Recalling Definition 1.14, we can enunciate the following

Theorem 1.22 ***(Dynamic Programming Principle).*** *For all $0 \le t < T(x)$*

$$T(x) = \inf_{\alpha\in\Gamma}\sup_{b\in\mathcal{B}} \{t + T(y_x(t;\alpha[b], b))\}, \qquad x \in \mathcal{R}\backslash\mathcal{T} \tag{1.33}$$

and

$$v(x) = \inf_{\alpha\in\Gamma}\sup_{b\in\mathcal{B}} \left\{\int_0^t e^{-s}ds + e^{-t}v(y_x(t;\alpha[b], b))\right\}, \qquad x \in \mathbb{R}^n\backslash\mathcal{T} \tag{1.34}$$

where v is the Kružkov transform of T defined as in (1.19).

The proof of this Theorem can be obtained by standard arguments (see for example [46, 5, 7]). We write it here for reader's convenience.
Proof.
We give only the proof of (1.33) since the other is similar. Define

$$w(x) := \inf_{\alpha\in\Gamma}\sup_{b\in\mathcal{B}} \{t + T(y_x(t;\alpha[b], b))\}.$$

We fix $\varepsilon > 0$ and for any $z \in \mathbb{R}^n\backslash\mathcal{T}$ we pick $\alpha_z \in \Gamma$ such that

$$T(z) \ge \sup_{b\in\mathcal{B}} \{t_z(\alpha_z[b], b)\} - \varepsilon. \tag{1.35}$$

We first prove that $T(x) \le w(x)$. We choose $\overline{\alpha} \in \Gamma$ such that

$$w(x) \ge \sup_{b\in\mathcal{B}} \{t + T(y_x(t;\overline{\alpha}[b], b))\} - \varepsilon. \tag{1.36}$$

Now we define $\gamma \in \Gamma$ as follows

$$\gamma[b](s) := \begin{cases} \overline{\alpha}[b](s) & s \le t \\ \alpha_z[b(\cdot + t)](s - t) & s > t \end{cases}$$

with $z := y_x(t;\overline{\alpha}[b], b)$. Finally we have

$$\begin{aligned} w(x) \ge \sup_{b\in\mathcal{B}} \{t + T(y_x(t;\overline{\alpha}[b], b))\} - \varepsilon = \\ t + \sup_{b\in\mathcal{B}} \{T(z)\} - \varepsilon \ge \\ t + \sup_{b\in\mathcal{B}} \{t_z(\alpha_z[b], b) - \varepsilon\} - \varepsilon = \\ \sup_{b\in\mathcal{B}} \{t_x(\gamma[b], b)\} - 2\varepsilon \ge T(x) - 2\varepsilon \end{aligned}$$

which gives the desired inequality because ε is arbitrary.
To prove the inequality $T(x) \geq w(x)$ we pick $b_1 \in \mathcal{B}$ such that

$$w(x) \leq \inf_{\alpha \in \Gamma} \{t + T(y_x(t; \alpha[b_1], b_1))\} + \varepsilon \tag{1.37}$$

and for any $x \in \mathbb{R}^n \backslash \mathcal{T}$ we pick $\alpha_x \in \Gamma$ such that

$$T(x) \geq \sup_{b \in \mathcal{B}} \{t_x(\alpha_x[b], b)\} - \varepsilon. \tag{1.38}$$

Obviously we have

$$w(x) \leq t + T(y_x(t; \alpha_x[b_1], b_1)) + \varepsilon. \tag{1.39}$$

Now for each $b \in \mathcal{B}$ define $\widetilde{b} \in \mathcal{B}$ by

$$\widetilde{b}(s) := \begin{cases} b_1(s) & s \leq t \\ b(s-t) & s > t \end{cases} \tag{1.40}$$

and then we define

$$\widetilde{\alpha}[b](s) := \alpha_x[\widetilde{b}](s+t). \tag{1.41}$$

Next set

$$z := y_x(t; \alpha_x[b_1], b_1) \tag{1.42}$$

and choose $b_2 \in \mathcal{B}$ such that

$$T(z) \leq t_z(\widetilde{\alpha}[b_2], b_2) + \varepsilon. \tag{1.43}$$

By (1.38) we obtain that

$$T(x) \geq \sup_{b \in \mathcal{B}} \{t_x(\alpha_x[b], b)\} - \varepsilon \geq t_x(\alpha_x[\widetilde{b}_2], \widetilde{b}_2) - \varepsilon. \tag{1.44}$$

Now we claim that

$$w(x) \leq t_x(\alpha_x[\widetilde{b}_2], \widetilde{b}_2) + 2\varepsilon \tag{1.45}$$

which gives, together with (1.44),

$$w(x) \leq T(x) + 3\varepsilon$$

and thus the conclusion by the arbitrariness of $\varepsilon > 0$. To prove the claim observe that, by (1.40) and (1.41)

$$y_x(\tau; \alpha_x[\widetilde{b}_2], \widetilde{b}_2) = \begin{cases} y_x(\tau; \alpha_x[b_1], b_1) & \tau \leq t \\ y_z(\tau - t; \widetilde{\alpha}[b_2], b_2) & \tau > t \end{cases}$$

so we have

$$t_z(\widetilde{\alpha}[b_2], b_2) = t_x(\alpha_x[\widetilde{b}_2], \widetilde{b}_2) - t \tag{1.46}$$

and then by (1.39), (1.42), (1.43) and (1.46)

$$\begin{aligned} w(x) &\leq t + T(y_x(t; \alpha_x[b_1], b_1)) + \varepsilon = t + T(z) + \varepsilon \leq \\ &t + t_z(\widetilde{\alpha}[b_2], b_2) + 2\varepsilon \leq t_x(\alpha_x[\widetilde{b}_2], \widetilde{b}_2) + 2\varepsilon. \end{aligned}$$

■

Let us derive the Hamilton-Jacobi-Isaacs equation associated to the 2-player minimum time problem. We point out that this derivation is not easy as that for the 1-player minimum time problem presented in section 1.2.3 due to the use of the nonanticipating strategies. We have the following

Proposition 1.23 *1. If $\mathcal{R}\backslash\mathcal{T}$ is open and $T \in C(\mathcal{R}\backslash\mathcal{T})$, then T is a viscosity solution of*

$$\min_{b\in B}\max_{a\in A}\{-f(x,a,b)\cdot\nabla T\} - 1 = 0\,, \qquad x\in\mathcal{R}\backslash\mathcal{T}; \tag{1.47}$$

2. If v is continuous then it is a viscosity solution of

$$v + \min_{b\in B}\max_{a\in A}\{-f(x,a,b)\cdot\nabla v\} - 1 = 0\,, \qquad x\in\mathbb{R}^n\backslash\mathcal{T}. \tag{1.48}$$

Proposition 1.24 *$v(x)$ is the unique viscosity solution of*

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

Once we computed v, we can recover the optimal feedbacks as in the 1-player case (see section 1.2.3)

$$(a_*(x), b_*(x)) := \arg\min_{b\in B}\max_{a\in A}\{-f(x,a,b)\cdot\nabla v\} \qquad x\in\mathbb{R}^n\backslash\mathcal{T}$$

and the optimal trajectory as the solution of

$$\begin{cases} \dot{y}^*(t) = f(y^*(t), a_*(y^*(t)), b_*(y^*(t)))\,, & t>0 \\ y^*(0) = x \end{cases}$$

1.4.1 Pursuit-Evasion games

In the case of Pursuit-Evasion games the vector field f and the target $\mathcal{T}$ have a particular form so that the equation can be simplified. The problem is set in $\mathbb{R}^{2n}$ and we denote the coordinates of the space by $x = (x_P, x_E)$ where $x_P, x_E \in \mathbb{R}^n$. Each player can control only own dynamics, *i.e.* f has the form

$$f(x,a,b) = f(x_P, x_E, a, b) = \begin{pmatrix} f_P(x_P, a) \\ f_E(x_E, b)\end{pmatrix}, \qquad f_P, f_E \in \mathbb{R}^n$$

so that

$$\begin{aligned} &\min_{b\in B}\max_{a\in A}\{-\nabla v(x)\cdot f(x,a,b)\} = \\ &\qquad \min_{b\in B}\{\max_{a\in A}\{-\nabla_{x_P} v(x)\cdot f_P(x_P,a) - \nabla_{x_E} v(x)\cdot f_E(x_E,b)\}\} = \\ &\qquad\qquad \min_{b\in B}\{-\nabla_{x_E} v(x)\cdot f_E(x_E,b)\} + \max_{a\in A}\{-\nabla_{x_P} v(x)\cdot f_P(x_P,a)\}.\end{aligned}$$

Then, equation (HJI) can be written as

$$\begin{cases} v + \min\limits_{b\in B}\{-\nabla_{x_E} v\cdot f_E(x_E,b)\} + \max\limits_{a\in A}\{-\nabla_{x_P} v\cdot f_P(x_P,a)\} - 1 = 0 & x\in\mathbb{R}^n\backslash\mathcal{T} \\ v = 0 & x\in\partial\mathcal{T}\end{cases} \tag{1.49}$$

In this kind of games the target is

$$\mathcal{T} = \{(x_P, x_E)\in\mathbb{R}^{2n} : |x_P - x_E| \le \varepsilon\}\,, \qquad \varepsilon \ge 0. \tag{1.50}$$

This choice means that the game ends when P touches E or P enters in a ball of radius ε centered in E. Note that the target $\mathcal{T}$, as defined above, is *unbounded* and then it is unsuitable for a numerical solution of the problem.
Finally note that the behavior of the solution v can be very different if we choose in (1.50) $\varepsilon = 0$ or $\varepsilon \geq \varepsilon_0 > 0$. Some games which show this particular behavior can be found in the classical book of Isaacs [58]. See, for example, the wall pursuit game and the football players game.

Tag-Chase game

As a model problem for Pursuit-Evasion games, we present here the classical Tag-Chase game. In this game the two players can move in every direction of the space with an own constant speed. The vector field has the form

$$f = \begin{pmatrix} f_P(x_P, a) \\ f_E(x_E, b) \end{pmatrix} = \begin{pmatrix} V_P\, a \\ V_E\, b \end{pmatrix}, \qquad V_P, V_E > 0\,, \quad a, b \in B(0,1).$$

Equation (1.49) can be furthermore simplified as follows

$$\begin{aligned} 0 = v(x) + \min_{b\in B(0,1)} \{(-\nabla_{x_E} v(x) \cdot b) V_E\} + \max_{a \in B(0,1)} \{(-\nabla_{x_P} v(x) \cdot a) V_P\} - 1 = \\ v(x) - V_E \max_{b\in B(0,1)} \{(\nabla_{x_E} v(x) \cdot b)\} + V_P \max_{a\in B(0,1)} \{(-\nabla_{x_P} v(x) \cdot a)\} - 1 = \\ v(x) - V_E |\nabla_{x_E} v(x)| + V_P |\nabla_{x_P} v(x)| - 1 = 0. \quad (1.51) \end{aligned}$$

Reduced coordinates

The Tag-Chase game can be reformulated in a new coordinate system in order to reduce the dimension of the problem.
We start noting that both players choose their strategy only considering their mutual position and the problem is completely invariant with respect to any rigid motion of both players. These considerations suggest the following new coordinate system

$$\tilde{x} := x_P - x_E,$$

so that the dynamics

$$\begin{cases} \dot{x}_P = V_P\, a \\ \dot{x}_E = V_E\, b \end{cases}$$

becomes

$$\dot{\tilde{x}} = V_P\, a - V_E\, b.$$

Moreover, in this new coordinate system the target (1.50) becomes

$$\widetilde{\mathcal{T}} = \{\tilde{x} \in \mathbb{R}^n : |\tilde{x}| \leq \varepsilon\}\,, \qquad \varepsilon \geq 0.$$

This new system (named relative coordinate system or reduced coordinates system) greatly simplifies the numerical solution of the problem for three reasons:

1. The problem gets back to dimension n (instead of $2n$);

2. The target is bounded so that we can fix a domain Q containing $\widetilde{\mathcal{T}}$ and solving the Isaacs equation in it;

3. The Dirichlet boundary condition we will have to impose on ∂Q changes meaning. For example, fixing the value $v = 1$ ($T = \infty$) on ∂Q in the natural coordinate system means that the Evader wins if he reaches the boundary of the domain and this can change completely his strategy with respect to the exact strategy. In reduced coordinate system the same boundary condition means that the Evader wins whenever he is far from the Pursuer more then a given length. So he will try to stay as far as possible from the Pursuer as he does in the natural coordinate system.

1.4.2 Pursuit-Evasion games with state constraints

In real applications it is useful to consider differential games with state constraints, where the first player has to keep the system in a given set $\overline{\Omega}_1 \subset \mathbb{R}^n$ and the second player in $\overline{\Omega}_2 \subset \mathbb{R}^n$. The game is set in $\overline{\Omega} \subset \mathbb{R}^{2n}$, where $\Omega := \Omega_1 \times \Omega_2$. We denote by x_P and x_E respectively the coordinates of $\overline{\Omega}_1$ and $\overline{\Omega}_2$ so that $x = (x_P, x_E) \in \overline{\Omega}$. We assume for simplicity $\mathcal{T} \subset \overline{\Omega}$.
The dynamical system for the state $y = (y^P, y^E)$ is

$$\begin{cases} \dot{y}^P(t) = f_P(y^P(t), a(t)), & t > 0 \\ \dot{y}^E(t) = f_E(y^E(t), b(t)), & t > 0 \\ y^P(0) = x_P \\ y^E(0) = x_E \end{cases} \tag{1.52}$$

where

$$a(\cdot) \in \mathcal{A}_x := \left\{a \in \mathcal{A} : y_x^P(t; a(\cdot)) \in \overline{\Omega}_1 \text{ for all } t \geq 0\right\}, \qquad x \in \overline{\Omega}, \tag{1.53}$$

$$b(\cdot) \in \mathcal{B}_x := \left\{b \in \mathcal{B} : y_x^E(t; b(\cdot)) \in \overline{\Omega}_2 \text{ for all } t \geq 0\right\}, \qquad x \in \overline{\Omega}. \tag{1.54}$$

The modified sets of the admissible controls introduced in (1.53)-(1.54) allow to force the system to respect the state constraints. Let Γ_x denote the set of nonanticipating strategies for the first player, *i.e.* the functions $\alpha : \mathcal{B}_x \to \mathcal{A}_x$ satisfying

$$b(t) = \tilde{b}(t) \text{ for all } t \leq t' \text{ implies } \alpha[b](t) = \alpha[\tilde{b}](t) \text{ for all } t \leq t'.$$

As in definition 1.21 we define the (lower) value function of the game as

$$T(x) := \inf_{\alpha \in \Gamma_x} \sup_{b \in \mathcal{B}_x} t_x(\alpha[b], b)$$

and its Kružkov transform

$$v(x) = 1 - e^{-T(x)}.$$

Note that it is reasonable to drop the attribute "lower" since in [22] it was proved that, under suitable assumptions, Pursuit-Evasion games with state constraints have a value, *i.e.* the lower value function coincides with the upper value function.
We write $y_x^P(t; a)$ and $y_x^E(t; b)$ to indicate respectively the solution of (1.52) whenever $a(t) \equiv a$ and $b(t) \equiv b$ for some $a \in A$ and $b \in B$.
Following [11, 59, 66] we will select subsets of admissible controls denoted by $A(x)$ and $B(x)$, for $x \in \Omega \backslash \mathcal{T}$

$$A(x) = \left\{a \in A : \text{ there is } r > 0 \text{ such that } y_{x'}^P(t; a) \in \overline{\Omega}_1 \text{ for } t \in [0, r], x' \in B(x, r) \cap \overline{\Omega}_1\right\}$$

$$B(x) = \{b \in B: \text{ there is } r > 0 \text{ such that } y^E_{x'}(t;b) \in \overline{\Omega}_2 \text{ for } t \in [0,r], x' \in B(x,r) \cap \overline{\Omega}_2\}$$

Note that the set $A(x)$ depends only on x_P and $B(x)$ only on x_E. Moreover, we notice that $A(x) = A$ and $B(x) = B$ provided $x_P \in \Omega_1$ and $x_E \in \Omega_2$, respectively. We will always assume that $A(x) \times B(x) \neq \emptyset$.
In [66] it is shown that v satisfies the following boundary value problem

$$\begin{cases} v(x) + \inf\limits_{b \in B(x)} \{-\nabla_{x_E} v(x) \cdot f_E(x_E, b)\} + \sup\limits_{a \in A(x)} \{-\nabla_{x_P} v(x) \cdot f_P(x_P, a)\} - 1 = 0 & x \in \overline{\Omega} \backslash \mathcal{T} \\ v(x) = 0 & x \in \partial \mathcal{T} \end{cases} \tag{HJI-Ω}$$

Note that equation (HJI-Ω) must be interpreted in the viscosity sense due to the state constraints boundary condition on $\partial\Omega$ and the fact that v can be not differentiable. See [11] for an exact definition.
In order to provide sufficient conditions for the continuity of v, we need to introduce further assumptions. Whenever we say that $\omega : [0, +\infty) \to [0, +\infty)$ is a *modulus* we mean that ω is nondecreasing, it is continuous at zero and $\omega(0) = 0$. The first assumption is about the behavior of the value function v near the target $\mathcal{T}$.

$$\text{There is a modulus } \omega \text{ such that } v(x) \leq \omega(d(x, \mathcal{T})) \quad \text{for all } x \in \overline{\Omega}_1 \times \overline{\Omega}_2. \tag{C1}$$

where $d(x, \mathcal{T}) = \inf_{z \in \mathcal{T}} \{|x - z|\}$.
The second is a small time controllability assumption for the Pursuer.

$$\begin{cases} \text{There is } \omega_P(\cdot, R) \text{ modulus for all } R > 0 \text{ such that} \\ \text{for all } w_1, w_2 \in \overline{\Omega}_1 \text{ there are } a(\cdot) \in \mathcal{A}_{w_1} \text{ and } t(w_1, w_2) \geq 0 \\ \text{satisfying } y^P_{w_1}(t(w_1, w_2); a(\cdot)) = w_2 \text{ and } t(w_1, w_2) \leq \omega_P(|w_1 - w_2|, |w_2|). \end{cases} \tag{C2}$$

The third is a small time controllability assumption for the Evader.

$$\begin{cases} \text{There is } \omega_E(\cdot, R) \text{ modulus for all } R > 0 \text{ such that} \\ \text{for all } z_1, z_2 \in \overline{\Omega}_2 \text{ there are } b(\cdot) \in \mathcal{B}_{z_1} \text{ and } t(z_1, z_2) \geq 0 \\ \text{satisfying } y^E_{z_1}(t(z_1, z_2); b(\cdot)) = z_2 \text{ and } t(z_1, z_2) \leq \omega_E(|z_1 - z_2|, |z_2|). \end{cases} \tag{C3}$$

The proof of the next theorem can be found in [11].

Theorem 1.25 *Assume that* (C1), (C2) *and* (C3) *hold. Then, under our assumptions on f, the value function v is continuous in $\overline{\Omega}_1 \times \overline{\Omega}_2$.*

1.5 Semi-Lagrangian approximation for Hamilton-Jacobi equations

In this section we recall how to obtain a convergent numerical scheme for Hamilton-Jacobi equations. As a model we will consider the minimum time problem for one player and two players described in previous sections. In our approach the numerical approximation is based on a time-discretization of the original control problem via a discrete version of the Dynamic Programming Principle. Then, the functional equation for the time-discrete problem is "projected" on a grid to derive a finite dimensional fixed point problem. We also show how to obtain the same numerical scheme by a direct discretization of the directional derivatives in the continuous equation. Note that the scheme we study is different to that obtained by a Finite Difference approximation. In particular, our scheme has a built-in up-wind correction.

1.5.1 Semi-Lagrangian schemes for optimal control problems

The aim of this section is to build a numerical scheme for equation (HJB). In order to do this, we first make a discretization of the original control problem (DS) introducing a time step $h = \Delta t > 0$.
We obtain a discrete dynamical system associated to (DS) just using any one-step scheme for the Cauchy problem. A well known example is the explicit Euler scheme which corresponds to the following discrete dynamical system

$$\begin{cases} y_{n+1} = y_n + hf(y_n, a_n)\,, & n = 1, 2, \dots \\ y_0 = x \end{cases} \tag{DS$_h$}$$

where $y_n = y(t_n)$ and $t_n = nh$. We will denote by $y_x(n; \{a_n\})$ the state at time nh of the discrete time trajectory verifying (DS$_h$). Let us define the discrete analogue of the admissible controls

$$\mathcal{A}^h := \{\{a_n\}_{n\in\mathbb{N}} : a_n \in A \text{ for all } n\}$$

and that of the reachable set

$$\mathcal{R}^h := \Big\{x \in \mathbb{R}^n : \text{ there exists } \{a_n\} \in \mathcal{A}^h \text{ and } \bar{n} \in \mathbb{N} \text{ such that } y_x(\bar{n}; \{a_n\}) \in \mathcal{T}\Big\}.$$

Let us also define

$$n_h(x, \{a_n\}) := \begin{cases} \min\{n \in \mathbb{N} : y_x(n; \{a_n\}) \in \mathcal{T}\} & x \in \mathcal{R}^h \\ +\infty & x \notin \mathcal{R}^h \end{cases}$$

and

$$N_h(x) := \inf_{\{a_n\}\in\mathcal{A}^h} n_h(x, \{a_n\}).$$

The discrete analogue of the minimum time function $T(x)$ is $T_h(x) := hN_h(x)$

Proposition 1.26 ***(Discrete Dynamic Programming Principle)*** *Let $h > 0$ fixed. For all $x \in \mathcal{R}^h$, $0 \le n < N_h(x)$ (so that $x \notin \mathcal{T}$)*

$$N_h(x) = \inf_{\{a_n\}\in\mathcal{A}^h} \{n + N_h(y_x(n; \{a_n\}))\}. \tag{1.55}$$

The proof of the Proposition 1.26 can be found in [8]. Choosing $n = 1$ in (1.55) and multiplying by h we obtain the time-discrete Hamilton-Jacobi-Bellman equation

$$T_h(x) = \min_{a\in A}\{T_h(x + hf(x, a))\} + h. \tag{1.56}$$

Note that we can obtain the equation (1.56) also by a direct discretization of equation (1.18)

$$0 = \max_{a\in A}\{-f(x, a) \cdot \nabla T(x))\} - 1 \approx \max_{a\in A}\left\{-\frac{T_h(x + hf(x, a)) - T_h(x)}{h}\right\} - 1$$

and, multiplying by h,

$$-\min_{a\in A}\{T_h(x + hf(x, a)) - T_h(x)\} - h = -\min_{a\in A}\{T_h(x + hf(x, a))\} + T_h(x) - h = 0.$$

As in the continuous problem, we apply the Kružkov change of variable

$$v_h(x) = 1 - e^{-T_h(x)}.$$

Note that, by definition, $0 \le v_h \le 1$ and v_h has constant values on the set of initial points x which can be driven to $\mathcal{T}$ by the discrete dynamical system in the same number of steps (of constant width h). This shows that v_h is a piecewise constant function. By (1.56) we easily obtain that v_h satisfies

$$v_h(x) = \min_{a\in A}\{\beta v_h(x + hf(x,a))\} + 1 - \beta$$

where $\beta = e^{-h}$ and we have the following

Proposition 1.27 *v_h is the unique bounded solution of*

$$\begin{cases} v_h(x) = \min_{a\in A}\{\beta v_h(x + hf(x,a))\} + 1 - \beta & x \in \mathbb{R}^n\backslash\mathcal{T} \\ v_h(x) = 0 & x \in \partial\mathcal{T} \end{cases} \tag{HJB$_h$}$$

Note that the time step h we introduced for the discretization of the dynamical system is still present in the time-*in*dependent equation (HJB$_h$) and then it will be interpreted as a *fictitious* time step.

Definition 1.28 *Assume $\partial\mathcal{T}$ smooth. We say that the Small Time Local Controllability (STLC) assumption is verified if*

$$\text{for any } x \in \partial\mathcal{T}, \text{ there exists } \hat{a} \in A \text{ such that } f(x,\hat{a})\cdot\eta(x) < 0 \tag{1.57}$$

where $\eta(x)$ is the exterior normal to $\mathcal{T}$ at x.

Theorem 1.29 *Let $\mathcal{T}$ be compact with nonempty interior. Then, under our assumptions on f and STLC, v_h converges to v locally uniformly in $\mathbb{R}^n$ for $h \to 0^+$.*

Fully discrete scheme

In order to solve the problem numerically we need to project equation (HJB$_h$) on a finite grid. First of all, we restrict our problem to a compact subdomain Q containing $\mathcal{T}$ and we build a regular triangulation of Q denoting by X the set of its nodes x_i, $i \in I := \{1,\dots,N\}$ and by S the set of simplices S_j, $j \in J := \{1,\dots,L\}$. Let us denote by k the size of the mesh *i.e.* $k = \Delta x := \max_j\{\text{diam}(S_j)\}$. Note that one can always decide to build a structured grid for Q as it is usual for Finite Difference scheme, although for dynamic programming/semi-Lagrangian scheme this is not an obligation. Main advantage of using structured grid is that one can easily find the simplex containing the point $x_i + hf(x_i,a)$ for every node x_i and every control $a \in A$ and make interpolations.
We will divide the nodes into three subsets, the algorithm will perform different operations in the three subsets.

$$\begin{aligned} I_{\mathcal{T}} &= \{i \in I : x_i \in \mathcal{T}\} \\ I_{in} &= \{i \in I\backslash I_{\mathcal{T}} : \text{ there exists } a \in A \text{ such that } x_i + hf(x_i,a) \in Q\} \\ I_{out} &= \{i \in I\backslash I_{\mathcal{T}} : x_i + hf(x_i,a) \notin Q \text{ for all } a \in A\} \end{aligned}$$

Now we can define the fully discrete scheme simply writing (HJB$_h$) at every node of the grid adding the boundary condition on ∂Q

$$\begin{cases} v_h^k(x_i) = \min\limits_{a\in A}\{\beta v_h^k(x_i + hf(x_i,a))\} + 1 - \beta & i \in I_{in} \\ v_h^k(x_i) = 0 & i \in I_{\mathcal{T}} \\ v_h^k(x_i) = 1 & i \in I_{out} \\ \\ v_h^k(x) = \sum_j \lambda_j(x) v_h^k(x_j)\,, \quad 0 \le \lambda_j(x) \le 1\,, \quad \sum_j \lambda_j(x) = 1 & x \in Q \end{cases} \qquad (\mathrm{HJB}_h^k)$$

Let us make a number of remarks on the above scheme:

1. The function v_h^k is extended on the whole space Q in a unique way by linear interpolation, *i.e.* as a convex combination of the values of $v_h^k(x_i)$, $i \in I$. It should be noted that one can choose any other interpolation operator.

2. The condition on I_{out} assigns to those nodes a value greater than the maximum value inside Q. It is like saying that once the trajectory leaves Q it will never come back to $\mathcal{T}$ (which is obviously false). Nonetheless the condition is reasonable since we will never get the information that the real trajectory (living in the whole space) can get back to the target unless we compute the solution in a larger domain containing Q. In general, the solution will be correct only in a subdomain of Q and it is greater than the real solution everywhere in Q. This means also that the solution we get strictly depends on Q (see also [79] on this point).

3. By construction, v_h^k belongs to the set

$$W^k := \{w : Q \to [0,1] \text{ such that } w \in C(Q), \nabla w = \text{constant in } S_j\,, j \in J\} \qquad (1.58)$$

of the piecewise linear functions.

We map all the values at the nodes onto a N-dimensional vector $V = (V_1, \ldots, V_N)$ so that we can rewrite (HJB$_h^k$) in a fixed point form

$$V = F(V) \qquad (1.59)$$

where F is defined componentwise as follows

$$[F(V)]_i := \begin{cases} \min\limits_{a\in A}\{\beta \sum_j \lambda_j(x_i + hf(x_i,a))V_j\} + 1 - \beta & i \in I_{in} \\ 0 & i \in I_{\mathcal{T}} \\ 1 & i \in I_{out} \end{cases} \qquad (1.60)$$

Theorem 1.30 *The operator F defined in* (1.60) *has the following properties:*

(i) F is monotone, i.e. *$U \le V$ implies $F(U) \le F(V)$;*

(ii) $F : [0,1]^N \to [0,1]^N$;

(iii) F is a contraction mapping in the max *norm $\|W\|_\infty = \max\limits_{i\in I} |W_i|$*

$$\|F(U) - F(V)\|_\infty \le \beta \|U - V\|_\infty.$$

Corollary 1.31 (HJB$_h^k$) *has a unique solution in W^k. Moreover, the solution can be approximated by the fixed point sequence*

$$V^{(n+1)} = F(V^{(n)}) \tag{1.61}$$

starting from any initial guess $V^{(0)} \in \mathbb{R}^N$.

A typical choice for $V^{(0)}$ is

$$V_i^{(0)} = \begin{cases} 0 & i \in I_{\mathcal{T}} \\ 1 & \text{elsewhere} \end{cases}$$

which guarantees a monotone decreasing convergence to the fixed point V^*.

For a convergence result regarding the fully-discrete value function v_h^k we refer to the following section where we will consider the discretization of differential games. In fact, any optimal control problem can be seen as a particular 2-player differential game in which the set of admissible controls for one player is just a singleton (in other words, one player has a fixed strategy).

It is also interesting to remark that in the algorithm the information flows from the target to the other nodes of the grid. In fact, on the nodes in $Q \backslash \mathcal{T}$ we start from $V_i^{(0)} = 1$ but these values immediately decrease in a neighborhood of $\mathcal{T}$ since the Euler scheme drives them to the target in just one step. At the next iteration other values, in a larger neighborhood of $\mathcal{T}$, will decrease due to the same mechanism and so on. This fact obviously relies on the hyperbolic nature of the equation and it is the basic idea the Fast Marching technique stems from. Moreover, the numerical evidence shows that the fixed point iterations converge in a finite number of steps, *i.e.* when the information has propagated from the target to the whole space Q.

1.5.2 Semi-Lagrangian schemes for differential games

The aim of this section is to build a numerical scheme for equation (HJI). In order to do this, we proceed as in the previous section considering the following discrete dynamical system

$$\begin{cases} y_{n+1} = y_n + hf(y_n, a_n, b_n)\,, & n = 1, 2, \dots \\ y_0 = x \end{cases} \tag{1.62}$$

We will denote by $y_x(n; \{a_n\}, \{b_n\})$ the state at time nh of the discrete time trajectory verifying (1.62). Let us define the discrete analogue of the admissible controls

$$\mathcal{A}^h := \{\{a_n\}_{n\in\mathbb{N}} : a_n \in A \text{ for all } n\}\ , \qquad \mathcal{B}^h := \{\{b_n\}_{n\in\mathbb{N}} : b_n \in B \text{ for all } n\}\,.$$

In order to define the value of the game we need the notion of discrete nonanticipating strategy.

Definition 1.32 *A* strategy *for the first player is a map $\alpha : \mathcal{B}^h \to \mathcal{A}^h$; it is* nonanticipating *if $\alpha \in \Gamma^h$, where*

$$\Gamma^h = \{\alpha : \mathcal{B}^h \to \mathcal{A}^h : b_n = \tilde{b}_n \textit{ for all } n \le n' \textit{ implies } \alpha[\{b_k\}]_n = \alpha[\{\tilde{b}_k\}]_n \textit{ for all } n \le n'\}.$$

Let us define the discrete analogue of the reachable set

$$\mathcal{R}^h := \Big\{ x \in \mathbb{R}^n : \text{ for all } \{b_n\} \in \mathcal{B}^h \text{ there exists } \alpha \in \Gamma^h \text{ and } \bar{n} \in \mathbb{N} \text{ such that } y_x(\bar{n}; \alpha[\{b_n\}], \{b_n\}) \in \mathcal{T} \Big\}. \tag{1.63}$$

Then we define

$$n_h(x, \{a_n\}, \{b_n\}) := \begin{cases} \min\{n \in \mathbb{N} : y_x(n; \{a_n\}, \{b_n\}) \in \mathcal{T}\} & x \in \mathcal{R}^h \\ +\infty & x \notin \mathcal{R}^h \end{cases}$$

and

$$N_h(x) := \inf_{\alpha \in \Gamma^h} \sup_{\{b_n\} \in \mathcal{B}^h} n_h(x, \alpha[\{b_n\}], \{b_n\}).$$

The discrete analogue of the (lower) value of the game $T(x)$ is $T_h(x) := hN_h(x)$.

Theorem 1.33 ***(Discrete Dynamic Programming Principle)*** *Let $h > 0$ fixed. For all $x \in \mathcal{R}^h$, $0 \le n < N_h(x)$ (so that $x \notin \mathcal{T}$)*

$$N_h(x) = \inf_{\alpha \in \Gamma^h} \sup_{\{b_n\} \in \mathcal{B}^h} \{n + N_h(y_x(n; \alpha[\{b_n\}], \{b_n\}))\}. \tag{1.64}$$

The proof of Theorem 1.33 traces the proof of Theorem 1.22 (*Dynamic Programming Principle*) once we define the sets of piecewise constant controls

$$a(t) := a_n \qquad t \in [nh, (n+1)h]\,, \quad n \in \mathbb{N}$$

and

$$b(t) := b_n \qquad t \in [nh, (n+1)h]\,, \quad n \in \mathbb{N}.$$

Similarly to the 1-player case, but with more technicalities due to the use of nonanticipating strategies, we can prove that T_h solves the following time-discrete Hamilton-Jacobi-Isaacs equation (see [7, 13])

$$T_h(x) = \max_{b \in B} \min_{a \in A} \{T_h(x + hf(x, a, b))\} + h \qquad x \in \mathcal{R}^h \backslash \mathcal{T} \tag{1.65}$$

and

$$v_h(x) = \max_{b \in B} \min_{a \in A} \{\beta v_h(x + hf(x, a, b))\} + 1 - \beta \qquad x \in \mathbb{R}^n \backslash \mathcal{T}. \tag{1.66}$$

where $v_h(x) = 1 - e^{-T_h(x)}$. Note that we can obtain the equation (1.66) also by a direct discretization of equation (HJI)

$$\begin{aligned} 0 = v(x) + \min_{b \in B} \max_{a \in A} \{-\nabla v(x) \cdot f(x, a, b)\} - 1 \approx \\ v_h(x) + \min_{b \in B} \max_{a \in A} \left\{ -\frac{v_h(x + hf(x, a, b)) - v_h(x)}{h} \right\} - 1 = \\ v_h(x) + \min_{b \in B} \max_{a \in A} \left\{ -\frac{1}{h} v_h(x + hf(x, a, b)) \right\} + \frac{1}{h} v_h(x) - 1 \end{aligned}$$

so we get

$$v_h(x) = \left(\frac{1}{1+h} \right) \max_{b \in B} \min_{a \in A} \{v_h(x + hf(x, a, b))\} + \frac{h}{1+h}. \tag{1.67}$$

Since it is a first order scheme, we can substitute the constant $\frac{1}{1+h}$ by $\beta = e^{-h}$ having

$$e^{-h} = \frac{1}{1+h} + O(h^2).$$

Let us define the space W as

$$W := \{w : \mathbb{R}^n \to [0,1]\}$$

and the operator F as

$$F[w](x) := \max_{b\in B} \min_{a\in A} \{\beta w(x + hf(x,a,b))\} + 1 - \beta. \tag{1.68}$$

Theorem 1.34 *The operator F defined in* (1.68) *has the following properties:*

(i) $F : W \to W$*;*

(ii) F is a contraction mapping in the L^∞ norm, i.e.

$$\|F[v] - F[w]\|_\infty \le \beta \|v - w\|_\infty.$$

Proof. (i) Since $w(z) \in [0,1]$ for every $z \in \mathbb{R}^n$, we have

$$0 < 1 - \beta \le \beta w(x + hf(x,a,b)) + 1 - \beta \le 1 \quad \text{for every } x,\ a,\ b$$

and then $F[w](x) \in [0,1]$ for every $x \in \mathbb{R}^n$.
(ii) We have

$$F[v](x) - F[w](x) = \beta \max_{b\in B} \min_{a\in A} \{v(x + hf(x,a,b))\} - \beta \max_{b\in B} \min_{a\in A} \{w(x + hf(x,a,b))\}.$$

Let b^* be the control such that

$$b^* = \arg\max_{b\in B} \{\min_{a\in A} \{v(x + hf(x,a,b))\}\}$$

and let a^* be the control such that

$$a^* = \arg\min_{a\in A} \{w(x + hf(x,a,b^*))\}\}.$$

Then,

$$\begin{aligned} F[v](x) - F[w](x) &\le \beta \min_{a\in A} \{v(x + hf(x,a,b^*))\} - \beta \min_{a\in A} \{w(x + hf(x,a,b^*))\} \\ &\le \beta v(x + hf(x,a^*,b^*)) - \beta w(x + hf(x,a^*,b^*)) \le \beta \|v - w\|_\infty. \end{aligned}$$

Swapping the role of v and w we easily conclude. ■

Proposition 1.35 *v_h is the unique bounded solution of*

$$\begin{cases} v_h(x) = \max_{b\in B} \min_{a\in A} \{\beta v_h(x + hf(x,a,b))\} + 1 - \beta & x \in \mathbb{R}^n \backslash \mathcal{T} \\ v_h(x) = 0 & x \in \partial \mathcal{T} \end{cases} \tag{HJI$_h$}$$

Fully discrete scheme

As in the case of minimum time problem for one player, we need to project equation (HJI$_h$) on a finite grid. First of all, we restrict our problem to a compact subdomain Q containing $\mathcal{T}$ and we build a regular triangulation of Q denoting by X the set of its nodes x_i, $i \in I := \{1, \dots, N\}$ and by S the set of simplices S_j, $j \in J := \{1, \dots, L\}$. We denote by k the size of the mesh *i.e.* $k = \Delta x := \max_j\{\mathrm{diam}(S_j)\}$. As in the case of a single player we need to impose boundary condition on ∂Q. However, the situation for games is much more complicated. In fact, setting the value of the solution outside Q equal to 1 (as in the 1-player case) will imply that the Pursuer looses every time the Evader drives the dynamics outside Q. On the contrary, setting the value to 0 outside Q will give a great advantage to the Pursuer. We will not take the matter further because in the following we will consider either state constraint boundary condition (so the trajectory can not leave Q) or we will get rid of the problem by means of the reduced coordinates (see Section 1.4.1).
The discretization in time and in space leads to a fully-discrete scheme

$$\begin{cases} v_h^k(x_i) = \max\limits_{b\in B}\min\limits_{a\in A}\{\beta v_h^k(x_i + hf(x_i,a,b))\} + 1 - \beta & x_i \in I_{in} \\ v_h^k(x_i) = 0 & x_i \in I_{\mathcal{T}} \\ v_h^k(x) = \sum_j \lambda_j(x) v_h^k(x_j)\,, \quad 0 \le \lambda_j(x) \le 1\,, \quad \sum_j \lambda_j(x) = 1 & x \in Q \end{cases} \tag{HJI$_h^k$}$$

where

$$\begin{aligned} I_{\mathcal{T}} &= \{i \in I : x_i \in \mathcal{T}\} \\ I_{in} &= \{i \in I \backslash I_{\mathcal{T}} : x_i + hf(x_i,a,b) \in Q \text{ for any } a \in A, b \in B\}. \end{aligned}$$

As mentioned above, we leave off all issues about those nodes which are in Q but not in $I_{\mathcal{T}} \cup I_{in}$.
Note that, as in the 1-player case, v_h^k belongs to the set W^k defined in (1.58).
We map all the values at the nodes onto a N-dimensional vector $V = (V_1, \dots, V_N)$ so that we can write (HJI$_h^k$) in a fixed point form

$$V = F(V) \tag{1.69}$$

where F is defined componentwise as follows

$$[F(V)]_i := \begin{cases} \max\limits_{b\in B}\min\limits_{a\in A}\{\beta \sum_j \lambda_j(x_i + hf(x_i,a,b))V_j\} + 1 - \beta & i \in I_{in} \\ 0 & i \in I_{\mathcal{T}} \end{cases} \tag{1.70}$$

Theorem 1.36 *The operator F defined in* (1.70) *has the following properties:*

(i) F is monotone, i.e. *$U \le V$ implies $F(U) \le F(V)$;*

(ii) $F : [0,1]^N \to [0,1]^N$;

(iii) F is a contraction mapping in the max *norm $\|W\|_\infty = \max\limits_{i\in I} |W_i|$*

$$\|F(U) - F(V)\|_\infty \le \beta \|U - V\|_\infty.$$

The proof can be found in [10].

Corollary 1.37 (HJI$_h^k$) *has a unique solution in W^k. Moreover, the solution can be approximated by the fixed point sequence*

$$V^{(n+1)} = F(V^{(n)}) \tag{1.71}$$

starting from any initial guess $V^{(0)} \in \mathbb{R}^N$.

Proposition 1.38 *Let $\mathcal{T}$ be the closure of an open set with Lipschitz boundary. Let w_m be the solution of HJI$_h^k$ with $Q = Q_m$, $h = h_m$, $k = k_m$ such that*

$$h_m \to 0\,, \quad \frac{k_m}{h_m} \to 0 \quad \textit{as } m \to \infty$$

and

$$Q_{m+1} \supseteq Q_m \textit{ for all } m \in \mathbb{N}\,, \qquad \bigcup_m Q_m = \mathbb{R}^n.$$

Let v be the bounded continuous viscosity solution of (HJI).
Then, w_m converges to v as $m \to \infty$ uniformly on any compact set of $\mathbb{R}^n$.

The proof of Proposition 1.38 can be found in [10]. The interested reader can find an error estimate for the quantity $\|v_h^k - v\|_\infty$ in terms of h and k in [87].

Remark 1.39 *Once we constructed the fully-discrete scheme, we need an extra work in order to make the problem completely finite-dimensional. In fact, if the sets A and B are constituted by an infinite number of points, we have either to choice a finite selection of them or choice a numerical method to evaluate the* maxmin *(or the* min *in the 1-player case). In both cases we have to add a further numerical error to those already considered in the discretization. In section 3.2.1 we will see how we can avoid this error at least in the 1-player minimum time problem.*

SL scheme for Pursuit-Evasion games

In section 1.4.1 we presented the Pursuit-Evasion games as a particular case of differential game. The time-discrete semi-Lagrangian scheme for the (simplified) equation (1.49) can be obtained by a direct discretization of the directional derivatives as before.

$$v_h(x)+\min_{b\in B}\left\{-\frac{v_h(x_P,x_E+hf_E)-v_h(x_P,x_E)}{h}\right\}+\max_{a\in A}\left\{-\frac{v_h(x_P+hf_P,x_E)-v_h(x_P,x_E)}{h}\right\}-1 =$$

$$v_h(x)+\min_{b\in B}\left\{-\frac{1}{h}v_h(x_P,x_E+hf_E)\right\}+\frac{v_h(x)}{h}+\max_{a\in A}\left\{-\frac{1}{h}v_h(x_P+hf_P,x_E)\right\}+\frac{v_h(x)}{h}-1$$

then the final equation is

$$v_h(x_P,x_E) = G[v_h](x_P,x_E) \tag{1.72}$$

where

$$G[v](x_P,x_E) := \frac{1}{2+h}\max_{b\in B}\{v(x_P,x_E+hf_E)\} + \frac{1}{2+h}\min_{a\in A}\{v(x_P+hf_P,x_E)\}+\frac{h}{2+h}. \tag{1.73}$$

As in games with a general dynamics (see Theorem 1.34), we have the following result

Theorem 1.40 *The operator G defined in* (1.73) *has the following properties:*

(i) $G : W \to W$*;*

(ii) G *is a contraction mapping in the* L^∞ *norm,* i.e.

$$|G[v](x) - G[w](x)| \leq \lambda ||v - w||_\infty, \quad \lambda < 1.$$

Proof. We prove only (ii).

$$\begin{aligned} &G[v](x) - G[w](x) = \\ &\quad \frac{1}{2+h} \max_{b \in B}\{v(x_P, x_E + hf_E)\} + \frac{1}{2+h} \min_{a \in A}\{v(x_P + hf_P, x_E)\} \\ &\quad - \frac{1}{2+h} \max_{b \in B}\{w(x_P, x_E + hf_E)\} - \frac{1}{2+h} \min_{a \in A}\{w(x_P + hf_P, x_E)\} = \\ &\quad \frac{1}{2+h} \max_{b \in B}\{v(x_P, x_E + hf_E)\} + \frac{1}{2+h} \max_{a \in A}\{-w(x_P + hf_P, x_E)\} \\ &\quad - \frac{1}{2+h} \max_{a \in A}\{-v(x_P + hf_P, x_E)\} - \frac{1}{2+h} \max_{b \in B}\{w(x_P, x_E + hf_E)\}. \end{aligned}$$

Let

$$a^* := \arg\max_{a \in A}\{-w(x_P + hf_P, x_E)\} \quad \text{and} \quad b^* := \arg\max_{b \in B}\{v(x_P, x_E + hf_E)\},$$

then

$$\begin{aligned} &G[v](x) - G[w](x) \leq \\ &\quad \leq \frac{1}{2+h} v(x_P, x_E + hf_E(x_E, b^*)) + \frac{1}{2+h}\Big(-w(x_P + hf_P(x_P, a^*), x_E)\Big) \\ &\quad - \frac{1}{2+h}\Big(-v(x_P + hf_P(x_P, a^*), x_E)\Big) - \frac{1}{2+h} w(x_P, x_E + hf_E(x_E, b^*)) \leq \\ &\quad \leq \frac{1}{2+h}||v - w||_\infty + \frac{1}{2+h}||v - w||_\infty = \frac{2}{2+h}||v - w||_\infty. \end{aligned}$$

Changing the role of v and w we obtain

$$G[v](x) - G[w](x) \leq \frac{2}{2+h}||v - w||_\infty$$

and this concludes the proof. ■

Equation (1.72) must be carefully compared with equation (1.67). In particular, it is important to select the most convenient equation for numerical purposes.

Remark 1.41 *Since*

$$\frac{1}{1+h} < \frac{2}{2+h} \quad \text{for all } h > 0$$

the convergence of the simplified scheme for Pursuit-Evasion games (1.72) *is expected to be slower than that of the general scheme* (1.67)*. This is confirmed by the numerical tests we did. On the other hand, every iteration of* (1.72) *is faster than that of* (1.67) *since* *the evaluation of the* max *and the* min *is faster than that of the* maxmin. *In fact, if we denote by* n_c *the number of points which constitute the discrete version of the sets* A *and* B *(see Remark 1.39), the algorithm associated to the equation* (1.72) *needs* $2n_c$ *evaluations of* f *whereas the algorithm associated to the equation* (1.67) *needs* n_c^2 *evaluations of* f. *In conclusion, the CPU times* are comparable.

Finally, note that once we have abandoned equation (HJI) for the time-discrete equation (HJI$_h$), we can no longer take into account the particular form of the dynamics for Pursuit-Evasion games in order to simplify the equation. This is probably the justification of the fact that the particular case (1.72) works worse than the general case (1.67).

Chapter 2

Fast Marching methods

In this chapter we give an overview of fast and efficient numerical methods for Hamilton-Jacobi equations developed in the last ten years. All the literature on this subject mainly stems from a paper by Tsitsiklis [97] in which the author introduced a Dijkstra-like algorithm to solve efficiently the discretized Hamilton-Jacobi equation associated to a particular minimum time problem. One year later, Sethian [85] refined Tsitsiklis' arguments introducing the so-called Fast Marching (FM) method for the eikonal equation modelling the monotone front propagation. This method will be extensively studied in the first section and in Chapter 3.
Later on, some authors improved the FM method in order to make it more efficient. For example, Kim [63] introduced the Group Marching (GM) method and Chopp [28] extended the FM method to high order approximation. Sethian himself proposed a wide class of applications for FM method (see, among others, [64, 65]).
More recently, some authors tried to modify the FM method in order to deal with more general equations. Sethian and Vladimirsky [86] introduced an extension which covers a wider class of equations including the monotone anisotropic front propagation and Vladimirsky [98] introduced an extension which covers the front evolution with time-depending velocity. Prados [79] suggested a modification which allows to deal with quite general Hamilton-Jacobi equations provided a subsolution of the equation is given. In parallel, other authors competed with FM method introducing the so-called Fast Sweeping (FS) method which provides a very efficient technique in order to speed up the classical iterative algorithm (see for example [100, 96, 80, 61, 62]).

2.1 Fast Marching method

The classical Fast Marching method based on finite differences (FM-FD in the sequel) has been proposed by Sethian in 1996 [85] as an acceleration method for the monotone first order iterative finite difference scheme for the following eikonal equation modelling a front which evolves monotonically along its normal direction

$$c(x)|\nabla T(x)| = 1\,, \qquad x \in \mathbb{R}^n \backslash \Omega_0\,, \qquad c > 0 \tag{2.1}$$

complemented with the Dirichlet boundary condition

$$T(x) = 0\,, \qquad x \in \partial\Omega_0. \tag{2.2}$$

The basic finite difference algorithm is based on an iterative procedure $T^{(n+1)} = F(T^{(n)})$ which computes the approximate solution everywhere in $\mathbb{R}^n \setminus \Omega_0$ at every iteration. The FM-FD method instead follows the front concentrating the computational effort where is needed, *i.e.* in a small neighborhood of the front, and it updates that neighborhood at every iteration to avoid useless computations. This is done dividing the grid nodes into three subsets: *far* nodes, *accepted* nodes and *narrow band* nodes. The *narrow band* nodes are the nodes where the computation actually takes place and their value can still change at the following iterations. The *accepted* nodes are those where the solution has been already computed and where the value can not change in the following iterations. Finally, the *far* nodes are the remaining nodes where an approximate solution has never been computed. In physical terms, the *far* nodes are those in the space region which has never been touched by the front, 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 (see Fig. 2.1).

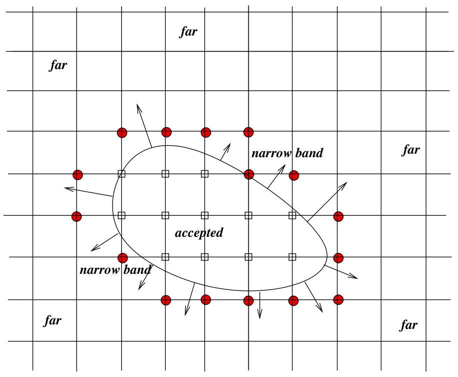


Figure 2.1: *far*, *narrow band* and *accepted* nodes

The algorithm starts labeling as *accepted* only the nodes belonging to the initial front, *i.e.* belonging to $\Gamma_0 = \partial\Omega_0$, and ends only when all the nodes have been *accepted.* In this section, we will briefly sketch the FM-FD scheme for (2.1)-(2.2). In order to avoid cumbersome notations we will restrict the presentation to the case $n = 2$. In the sequel, we will always consider the case of a positive velocity, *i.e.* we assume $c(x) > 0$ to guarantee a monotone (increasing) evolution of the front. The results in this section can be easily generalized to the n-dimensional case and to the case $c(x) < 0$.
We will take a square Q large enough to contain Ω_0, this is the domain where we want to compute T. Boundary conditions will be given on ∂Q and Γ_0 but, as a first step, we will consider the algorithm *without* boundary conditions on ∂Q. The implementation of boundary conditions in the scheme will be discussed later on at the end of section 3.2.2.
We will assume to work on a structured grid of $M \times N$ nodes (x_i, y_j), $i = 1, \ldots, N$ and $j = 1, \ldots M$. Δx and Δy will denote the (uniform) discretization steps respectively on the x and y axis. We will denote by $T_{i,j}$ and $c_{i,j}$ respectively the values of T and c at (x_i, y_j).
Let us write (2.1) as

$$T_x^2 + T_y^2 = \frac{1}{c^2(x,y)}. \tag{2.3}$$

We replace the partial derivatives T_x and T_y by first order finite differences and we choose for simplicity $M = N$ and $\Delta x = \Delta y$.

$$T_x(x_i, y_j) \approx \frac{T_{i,j} - T_{i-1,j}}{\Delta x} \qquad T_y(x_i, y_j) \approx \frac{T_{i,j} - T_{i,j-1}}{\Delta x}$$

or

$$T_x(x_i, y_j) \approx \frac{T_{i+1,j} - T_{i,j}}{\Delta x} \qquad T_y(x_i, y_j) \approx \frac{T_{i,j+1} - T_{i,j}}{\Delta x}.$$

It is well known that, in order to obtain an approximation of the viscosity solution, an *up-wind* correction must be introduced. This leads to the following equation

$$\begin{aligned} &\left(\max\left\{ \max\left\{ \frac{T_{i,j} - T_{i-1,j}}{\Delta x}, 0 \right\}, -\min\left\{ \frac{T_{i+1,j} - T_{i,j}}{\Delta x}, 0 \right\} \right\} \right)^2 + \\ &+ \left(\max\left\{ \max\left\{ \frac{T_{i,j} - T_{i,j-1}}{\Delta x}, 0 \right\}, -\min\left\{ \frac{T_{i,j+1} - T_{i,j}}{\Delta x}, 0 \right\} \right\} \right)^2 = \frac{1}{c_{i,j}^2}. \end{aligned} \tag{2.4}$$

Let us note that the above discretization differs from another popular *up-wind* correction which is the so-called *minmod* scheme

$$\begin{aligned} &\frac{1}{\Delta x^2} \operatorname{minmod}^2(T_{i+1,j} - T_{i,j},\ T_{i,j} - T_{i-1,j}) + \\ &+ \frac{1}{\Delta x^2} \operatorname{minmod}^2(T_{i,j+1} - T_{i,j},\ T_{i,j} - T_{i,j-1}) = \frac{1}{c_{i,j}^2} \end{aligned} \tag{2.5}$$

where

$$\operatorname{minmod}(a, b) = \begin{cases} a & \text{if } |a| < |b| \text{ and } ab > 0 \\ b & \text{if } |b| < |a| \text{ and } ab > 0 \\ 0 & \text{if } ab \le 0 \end{cases}.$$

The main difference is that (2.5) sets to 0 the value of the derivative whenever the right and the left derivative have different signs whereas (2.4) always chooses the derivative with maximum absolute value. It should be noted that the two schemes do not coincide also when the right and left derivatives have the same sign.
Let us briefly recall the main definitions and steps of the FM-FD method.

Definition 2.1 (Neighboring nodes for the FD scheme) *Let $X = (x_i, y_j)$ be a node. We define the set of neighboring nodes to X as*

$$N_{FD}(X) = \Big\{ (x_{i+1}, y_j), (x_{i-1}, y_j), (x_i, y_{j+1}), (x_i, y_{j-1}) \Big\}.$$

These are the nodes appearing in the stencil of the first order finite difference discretization. The definition can be easily extended to the n-dimensional case.

Sketch of the FM-FD algorithm

Initialization (see Figure 2.2)

1. The nodes belonging to the initial front Γ_0 are located and labeled as *accepted*. Their value is set to $T = 0$ (they form the set $\widetilde{\Gamma}_0$).

2. The initial *narrow band* is defined taking the nodes belonging to $N_{FD}(\widetilde{\Gamma}_0)$, external to Γ_0. These nodes are labeled as *narrow band*, setting the value to $T = \frac{\Delta x}{c}$.
3. The remaining nodes are labeled as *far*, their value is set to $T = +\infty$ (in practice, the maximum floating point number).

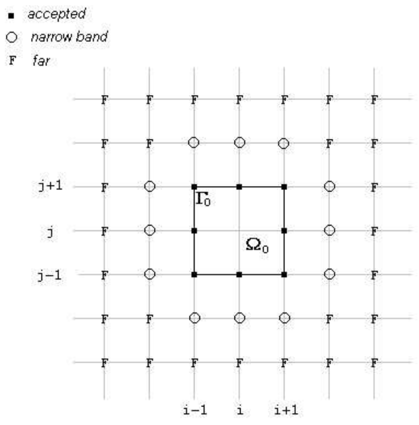


Figure 2.2: Initialization for FM-FD method, case $c > 0$

Main Cycle

1. Among all the nodes in the *narrow band* we search for the minimum value of T. Let us denote this node by A.
2. A is labeled as *accepted* and it is removed from the *narrow band.*
3. The nodes in $N_{FD}(A)$ which are not *accepted* are labeled as *active.* If among these nodes there are nodes labeled as *far*, they are transferred to the *narrow band.*
4. The value of T in the nodes *active* is computed (or recomputed) solving the second order equation (2.4) and taking the largest root.
5. If the *narrow band* is not empty go back to 1.

Note that the *narrow band* is a reasonable approximation of the level set of $T(x, y)$.
The main interest in the FM-FD method is that its computational cost is bounded. In fact, every node can not be accepted more than one time and every node has just four neighbors, so the bound on the maximum number of times a single node can be recomputed is four. This corresponds to a computational cost of $O(N)$ where N is the total number of nodes. We should add to that cost the search for the minimum value of T among the nodes in the *narrow band* which costs $O(\ln(N_{nb}))$ where N_{nb} is the number of nodes in the *narrow band.* In conclusion, the algorithm has a global cost of $O(N \ln(N_{nb}))$ operations (see [97], [84], [85] for further details on the computational cost). This is not the case for the usual iterative/fixed point algorithm approximating the solution of a general optimal control problem since in that case the approximate solution is obtained in the limit and, in practice, no one knows when the stopping criterion will apply, *i.e.* the maximum number of iterations is virtually unbounded.

2.2 Fast Marching methods for more general equations

Since its first presentation, many people tried to adapt the FM method to more general equations due to its great efficiency. Unfortunately this seems to be a difficult task since 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 characteristic 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 T in the ascending order and then to maintain the right upwinding *only* in the case the optimal control a^* satisfies

$$a^*(x) := \arg\max_{a\in B(0,1)}\{-f(x,a)\cdot\nabla T(x)\} = -\frac{\nabla T(x)}{|\nabla T(x)|} \tag{2.6}$$

(see (1.30) and all Section 1.3). In [86] it is extensively studied an example in which this problem arises. Consider the eikonal equation $|\nabla d| = 1$ complemented by Dirichlet boundary condition $d(0,0) = 0$ in a plane $z = \lambda x + \mu y$ for some vector (λ, μ). 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

$$f(x,y,a) = \hat{c}(x,y,a)a\,, \qquad a \in B(0,1)$$

where

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

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

$$T(x,y) = \sqrt{(1+\lambda^2)x^2+(1+\mu^2)y^2+2\lambda\mu xy} \tag{2.7}$$

and it is a viscosity solution of

$$\max_{a\in A}\{-\hat{c}(x,y,a)a\cdot\nabla T(x,y)\} - 1 = 0 \tag{2.8}$$

or, in a equivalent way, of

$$\sqrt{\frac{(1+\mu^2)T_x^2+(1+\lambda^2)T_y^2-2\lambda\mu T_xT_y}{1+\lambda^2+\mu^2}} = 1.$$

If we now consider the particular case $\lambda = 1$, $\mu = 0$, $(x,y) = (1,1)$ we obtain, by (2.7) and (2.8)

$$a^*(1,1) = \frac{(-1,-1)}{\sqrt{2}} \qquad \text{and} \qquad -\nabla T(1,1) = \frac{(-2,-1)}{\sqrt{3}}$$

(see Fig. 2.3). This example shows that in an optimal control problem with general n-dimensional dynamics $f(x,a)$ we expect that the characteristic and the gradient directions are different and that a simplex $S(x)$ may contain the characteristic for the node x even

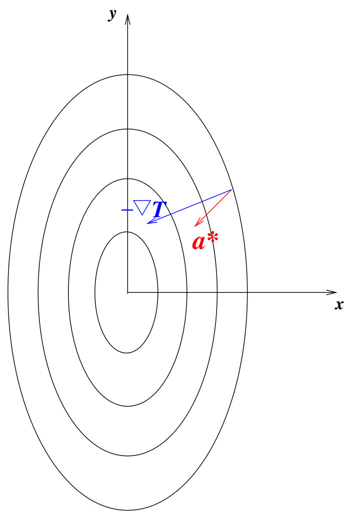


Figure 2.3: gradients vs characteristic directions

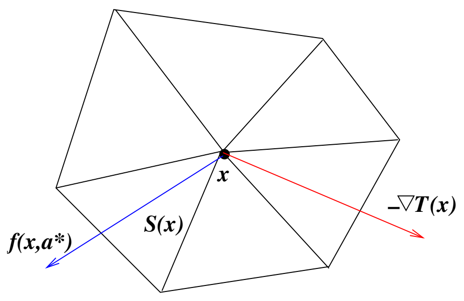


Figure 2.4: gradients vs characteristic directions at discrete level

if the direction $-\nabla T(x)$ is not contained into that simplex (see Fig. 2.4). This is an intrinsic problem with Dijkstra-like methods: to produce the numerical solution these methods attempt to compute T in the ascending order ($i.e.$ from the simplex containing $-\nabla T$), whereas, in order to maintain the upwinding, $T(x)$ has to be computed from the simplex containing the characteristic.
Taking into account these considerations, Sethian and Vladimirky [86] extended the FM method to the following equation modelling the anisotropic front propagation

$$c\left(x, \frac{\nabla T}{|\nabla T|}\right)|\nabla T| = 1\,, \qquad x \in \mathbb{R}^n \backslash \Omega_0\,, \qquad c > 0 \tag{2.9}$$

which can be seen as an equation of the form (1.20) in the special case $f(x,a) = c(x,a)a$ and $A = B(0,1)$. The computational complexity of the proposed algorithm increases with respect the standard FM method and becomes of $O(\Upsilon^{n-1} N \ln(N))$ where N is the total number of nodes and Υ is an *anisotropy coefficient* defined by

$$\Upsilon := \frac{\max_{x,a} c(x,a)}{\min_{x,a} c(x,a)}.$$

This is due to the fact that the scheme makes use of more simplices than those in the neighborhood of the considered point.

Finally, let us mention that Prados [79] proposed a modified FM method which can deal with a generic Hamilton-Jacobi-Bellman equation of the form

$$\lambda F(u(x)) + \sup_{a\in A}\{-f(x,a)\cdot\nabla u(x) - l(x,a)\} = 0\,, \qquad x\in\Omega \tag{2.10}$$

where Ω is an open set of $\mathbb{R}^n$, $\lambda \geq 0$, and $F:\mathbb{R}\to\mathbb{R}$ is a strictly increasing function. Note that in this case the cost function l is *not* assumed to be positive so that the solution of the equation does not necessarily increase along the characteristic curves and, of course, the characteristic curves does not coincide in general with the gradient lines. The proposed algorithm is identical to the standard FM method in all the steps but when the node to be *accepted* is chosen. Here we have to choose the node with the minimal value of $u-\psi$ (instead of u), ψ being any subsolution of the considered equation, *i.e.*

$$\lambda F(\psi(x)) + \sup_{a\in A}\{-f(x,a)\cdot\nabla \psi(x) - l(x,a)\} \leq 0\,, \qquad x\in\Omega.$$

Prados successfully applied this scheme to Hamilton-Jacobi equations related to Shape-from-Shading problems since for those equations it is possible to recover a subsolution (see [76]).
Obviously all difficulties mentioned above related to the research of the characteristic directions are shifted onto the research of a subsolution of the equation.

2.3 Group Marching method

The Group Marching (GM) method has been introduced by Kim [63] to solve the eikonal equation (2.1)-(2.2) on a structured grid by a discretization as that in FM-FD.
Let us denote by Γ the set of nodes belonging to the *narrow band* and let us choose $\Delta x = \Delta y$.
Define

$$T_{\Gamma,min} = \min\{T_{i,j} \mid (x_i,y_j)\in\Gamma\}$$
$$c_{\Gamma,\max} = \max\{c_{i,j} \mid (x_i,y_j)\in\Gamma\}.$$

The GM method labels as *accepted*, *at the same time*, all the nodes belonging to the set G defined by

$$G := \left\{(x_i,y_j)\in\Gamma \;:\; T_{i,j} \leq T_{\Gamma,min} + \frac{\Delta x}{\sqrt{2}}\,\frac{1}{c_{\Gamma,\max}}\right\}. \tag{2.11}$$

The procedure described above is justified by the fact that all the values in G can not be in principle affected by the future updates of the other nodes and then it is safe to *accepted* them all at a time. In fact, a value of a node can be considered "not modifiable" if it is lower then the minimal value in the *narrow band* plus the time needed to cover the minimal distance with the maximal velocity. Considering that the position of the front must be recovered by interpolation of the values on the nodes, the minimal distance on a structured grid between a node and the front is $\frac{\sqrt{2}}{2}\Delta x$, which corresponds to the half-diagonal of a square cell with side Δx (see Fig. 2.5).

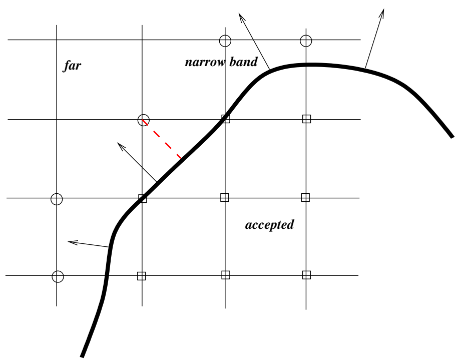


Figure 2.5: minimal distance between the front and a node considering that the position of the front must be recovered by interpolation of the values on the nodes

At every iteration the up-date of the *narrow band* is obtained as in FM-FD method, including the four neighbors of every node that has been labeled as *accepted*. It is clear that if the set G is large the GM method can be much faster than FM-FD method because more than one node at a time is *accepted*. On the other hand, it is rather difficult to give an estimate of the acceleration parameter since the cardinality of G depends on the values $\{T_{i,j} \ : \ (x_i, y_j) \in \Gamma\}$ and on the velocity of propagation. It could be that

$$G = \{T_{\Gamma,min}\}$$

and this would imply a computational cost $O(N \ln(N_{nb}))$ instead of getting $O(N)$ as one would expect by some tests in [63]. Moreover, it should be noted that, in order to avoid possible instabilities, it is necessary to update all the neighboring points of the group G twice, increasing the computational cost.

2.4 Fast Sweeping method

The Fast Sweeping (FS) method is based on an idea first introduced in [41] and it was extensively analyzed in [100] and [96]. The crucial idea is that the algorithm sweeps the whole (two-dimensional) domain with four alternating orderings repeatedly

$$(1)\ i = 1, \ldots, N\,,\ j = 1, \ldots, M \qquad (2)\ i = N, \ldots, 1\,,\ j = 1, \ldots, M \tag{2.12}$$

$$(3)\ i = N, \ldots, 1\,,\ j = M, \ldots, 1 \qquad (4)\ i = 1, \ldots, N\,,\ j = M, \ldots, 1 \tag{2.13}$$

(where N and M are the number of nodes in each dimension) and it updates the value at a grid point only if the new value is smaller than the current one. This idea can be easily extended to the n-dimensional case.

Computing the values in this special ordering the algorithm is able to follow simultaneously a family of characteristic in a certain direction. As proved in [100] the Fast Sweeping method converges in 2^n iterations where n is the dimension of the problem if the initial front Ω_0 is just a point on the grid and the function c is constant. If those assumptions do not hold, the FS has been shown to be of complexity $O(N)$ and to converge in a finite number of iterations although the number of iterations is not explicitly written out. In

[80] an extension on triangular meshes is given and it is proved that the upper bound to the number of iterations needed by FS to reach convergence depends only on the quantity

$$\frac{DK\lambda_M}{\lambda_m}$$

where D is the diagonal of the domain Q and

$$f = 1/c\,, \qquad K = \sup_{x\in Q}\left|\frac{\nabla f(x)}{f(x)}\right|\,, \qquad \lambda_M = \sup_{x\in Q} f(x)\,, \qquad \lambda_m = \inf_{x\in Q} f(x).$$

Let us note that the discretization used in [100] is the same used in the FM-FD method described in section 2.1 and that in any case the numerical evidence shows that the convergence is more rapid with respect to the classical iterative method.

Chapter 3

New Fast Marching methods

In this chapter we present a number of new results on Fast Marching (FM) methods introduced in the previous chapter. In section 3.1 we give an exhaustive analysis of the FM-FD method. An example which proves that the FM-FD scheme can produce imaginary solutions is exhibited. Then, we give a complete proof of its convergence under a new CFL-like condition which guarantees that the scheme is always meaningful and there are not imaginary solutions. To our knowledge this condition appears for the first time in the literature.
Section 3.2 is devoted to the presentation of a new Fast Marching semi-Lagrangian method (FM-SL in the sequel) for the eikonal equation first appeared in [38]. Since semi-Lagrangian schemes (SL in the sequel) introduced in Section 1.5.1 have shown to be more accurate than the Finite Difference schemes corresponding to the same order, it is natural to extend the ideas behind the FM-FD method to this class of schemes. Note that in the framework of semi-Lagrangian schemes several convergence results and *a-priori* error estimates have been obtained via control arguments (see Section 1.5). Moreover, these schemes will not require an explicit and restrictive CFL condition for stability (see [52]).
In Section 3.2.2 convergence of FM-SL is proved. In the same section we analyze the computational complexity showing that the FM-SL has a complexity of order $O(N \ln(N_{nb}))$ where N is the total number of nodes and N_{nb} is the number of nodes in the *narrow band* (bounded by N). We also give the sweeping version of our method. Section 3.2.3 is devoted to numerical tests and to the comparison between several schemes on a number of benchmarks.
In Section 3.3 we extend the FM-SL method to non-convex Hamiltonians, in particular to *minmax* Hamiltonians which appear in the analysis of differential games. Some numerical tests show the potential of this new method presented in [39].
Finally, in Section 3.4 we present a second extension of the FM technique to non-monotone evolution of fronts appeared for the first time in [25]. The new scheme covers the case when the speed of propagation is time-dependent and it can change sign in space and/or in time and can be successfully applied to the study of dislocation dynamics.

3.1 A new CFL-like condition for FM method

Let us observe that it is necessary to introduce some conditions or to modify the scheme in order to avoid inconsistencies due to the appearance of imaginary solutions. In fact, let us consider the discretization (2.4) and suppose that

$$T_{i,j} < T_{i+1,j} \qquad T_{i,j} < T_{i,j-1} \qquad T_{i,j} > T_{i-1,j} \qquad T_{i,j} > T_{i,j+1}.$$

It is easy to check that (2.4) corresponds to

$$\left(\frac{T_{i,j} - T_{i-1,j}}{\Delta x}\right)^2 + \left(\frac{T_{i,j+1} - T_{i,j}}{\Delta x}\right)^2 = \frac{1}{c_{i,j}^2},$$

which gives

$$T_{i,j} = \frac{T_{i-1,j} + T_{i,j+1} \pm \sqrt{2\left(\frac{\Delta x}{c_{i,j}}\right)^2 - (T_{i-1,j} - T_{i,j+1})^2}}{2}. \tag{3.1}$$

We already noted that the term under the square root can be negative. Obviously this must avoided since complex roots have no physical meaning. A situation where this occurs is the following example.
Consider the case where the initial front is the union of two points, *i.e.* $\Gamma_0 = P \cup Q$, $Q = (\Delta x, \Delta y)$ and $P = (2\Delta x, 4\Delta y)$ (see Figure 3.1). Let us consider the following velocity

$$c(x,y) = \begin{cases} \varepsilon & y \le \Delta y \\ \varepsilon + \frac{1-\varepsilon}{\Delta y}(y - \Delta y) & \Delta y \le y \le 2\Delta y \\ 1 - \frac{1-\varepsilon}{\Delta y}(y - 2\Delta y) & 2\Delta y \le y \le 3\Delta y \\ \varepsilon & y \ge 3\Delta y \end{cases} \tag{3.2}$$

In this case the algorithm initialize the *narrow band* computing a large value for B when

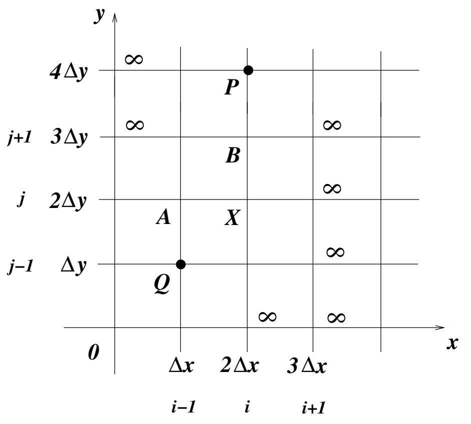


Figure 3.1: A configuration with complex roots

ε is small and a small value for the node A which will be the first node *accepted* (after Γ_0). When the node X have to be computed, its value depends on $T(A)$ and $T(B)$. Since $c(X) = 1$ and $T(A) - T(B)$ is large (for ε small) the radicand in (3.1) will be negative (as numerical tests confirm).
This difficulty can be solved either choosing the positive part of the radicand (as suggested in [63]) or changing discretization as in [100]. However, both choices lead to a modification

of the scheme which can be difficult to handle when looking for theoretical results. We prefer to avoid to change the scheme and we prove that under the CFL-like condition

$$\Delta x \leq (\sqrt{2}-1)\frac{c_{min}}{L_c} \tag{3.3}$$

the algorithm computes always *real* solutions at every node (here c_{min} is the minimum value of c and L_c is its Lipschitz constant). Condition (3.3) has a clear meaning and allows to give a proof of convergence to the viscosity solution. At our knowledge this is the first time this condition appears in the literature.

Let us denote by A the node in the *narrow band* where the minimum value of T is attained.
The algorithm labels A as *accepted* and starts to compute the neighboring nodes which are not *accepted*.

Proposition 3.1 *Let $X = (x_i, y_j) \in N_{FD}(A)$ be the node where the FM-FD method computes a solution. Let us assume that*

$$c_{min} = \min_{Q\setminus\Omega_0} c(x) > 0 \tag{3.4}$$

and that the following CFL-like condition holds true

$$\Delta x \leq (\sqrt{2}-1)\frac{c_{min}}{L_c} \tag{3.5}$$

where L_c denotes the Lipschitz constant of c. Then, we have

$$T(A) \leq T(X) \leq T(A) + f_X \tag{3.6}$$

where $f_X := \Delta x/c(X)$.

The above result is crucial to have convergence in a finite number of steps. In fact, it shows that the minimum value of the nodes in the *narrow band* (which is actually the only value *accepted* at every iteration) is exact within the consistency error of the scheme. An approximate value is considered to be exact if the algorithm can not replace it with a strictly lower value at any of the following iterations.

Proof of Proposition 3.1. We will assume that B, C and D are the neighbors of X which can have a label *accepted*, *narrow band* or *far* (see Fig. 3.2). We will prove the result by

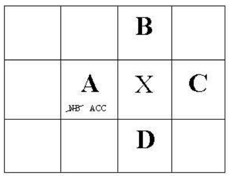


Figure 3.2: The neighbouring nodes to X

induction on the number of iterations of the algorithm. We will always assume

$$T(B) \leq T(D) \tag{3.7}$$

which is not restrictive since we can always switch the B and D.
In the first iteration we simply have

$$T(X) = 0 + f_X$$

and (3.6) is satisfied. Let us consider the n-th step of the algorithm. The induction hypothesis implies that at each iteration the values of nodes in the *narrow band* are greater than values of nodes labeled as *accepted* at the same iteration. Therefore, by construction we have that taken two nodes Y and Z

if Y has become *accepted* before Z then $T(Y) \leq T(Z)$.

The proof will be divided into four cases:

CASE 1
B is *far.*
C and D are *narrow band* o *far.*
By assumption $T(B) = +\infty$ and since $T(B) \leq T(D)$ this imply that D must be *far.* Moreover, we have

$$T(C) \geq T(A)$$

since A has been chosen among all the nodes of the *narrow band* to become *accepted.* Also X must be *far*, since it has never been computed. Then by (2.4) we get

$$T(X) = T(A) + f_X \tag{3.8}$$

Since $f_X > 0$, (3.8) implies

$$T(A) \leq T(X) \leq T(A) + f_X.$$

CASE 2
B is *narrow band.*
C and D are *narrow band* or *far.*
Also in this case X is *far.* We have

$$T(A) \leq T(B)\,, \qquad T(A) \leq T(C)$$

since A is the minimal node in *narrow band* . Moreover, the assumption (3.7) implies that $T(X)$ will be computed by the values at A and B. From (2.4) we get

$$T(X) = \frac{T(A) + T(B) + \sqrt{2f_X^2 - (T(A) - T(B))^2}}{2} \tag{3.9}$$

and then

$$T(X) \geq \frac{T(A) + T(B)}{2} \geq \frac{T(A) + T(A)}{2} = T(A). \tag{3.10}$$

Since $T(X)$ solves

$$(T(X) - T(A))^2 + (T(X) - T(B))^2 = f_X^2$$

we have

$$(T(X) - T(A))^2 \leq f_X^2.$$

Since all the terms in the above equation are positive we conclude that

$$T(X) - T(A) \leq f_X. \tag{3.11}$$

CASE 3
B is *accepted.*
C and D are *narrow band* or *far.*
This situation occurs when X has been already computed once (when B has been labeled as *accepted*). Let us denote its value by $T_{old}(X)$. The node X is then in the *narrow band* and has to be recomputed because A has just been labeled as *accepted.* Let us note that in the previous computation $T_{old}(X)$ has been computed according to the rules examined in CASE 1 or CASE 2. Then we have

$$T(B) \leq T_{old}(X) \leq T(B) + f_X.$$

Moreover $T(A) \leq T_{old}(X)$ because A just became *accepted* and $T(B) \leq T(A)$ since B became *accepted* before A (induction).
These inequalities imply

$$T(B) \leq T(A) \leq T_{old}(X) \leq T(B) + f_X \tag{3.12}$$

and

$$0 \leq T(A) - T(B) \leq f_X. \tag{3.13}$$

The value at X, which will be denoted by $T_{new}(X)$, will depend on $T(A)$ and $T(B)$.
By (3.13) and (3.12) we derive

$$\begin{aligned} T_{new}(X) &= \frac{T(A) + T(B) + \sqrt{2f_X^2 - (T(A) - T(B))^2}}{2} \geq \\ &\geq \frac{T(A) + [T(B) + f_X]}{2} \geq \frac{T(A) + T(A)}{2} = T(A) \end{aligned} \tag{3.14}$$

and

$$T_{new}(X) \leq \frac{T(A) + T(B) + \sqrt{2}f_X}{2} \leq T(A) + \frac{\sqrt{2}}{2}f_X \leq T(A) + f_X.$$

CASE 4
B is *narrow band* or *far.*
C *accepted.*
D is *narrow band* or *far.*
In this case X has already been computed because is a neighbor of C. It belongs to the *narrow band* and has a value $T_{old}(X)$. Besides

$$T(A) \leq T_{old}(X) \tag{3.15}$$

since on the contrary X would have been chosen instead of A as the node to be *accepted* and

$$T(A) \leq T(B) \tag{3.16}$$

for the same reason. Moreover we have $T(C) \leq T(A)$ by induction and $T(B) \leq T(D)$ by assumption. In conclusion, the nodes contributing to the computation of $T(X)$ are C and B or only C. The fact that A has been labeled as *accepted* has no effect on the computation so we are again in *CASE* 1 o 2. This implies,

$$T(C) \leq T_{new}(X) \leq T(C) + f_X \leq T(A) + f_X.$$

Now we prove that $T_{new} \geq T(A)$. When $T_{old}(X)$ was computed the algorithm was in the *CASE* 1, 2 so

$$T(C) \leq T_{old}(X) \leq T(C) + f_X \tag{3.17}$$

Moreover we have

$$T(C) \leq T(B) \tag{3.18}$$

by induction.
If $T(B) > T_{old}(X)$ than the node contributing to the computation of $T(X)$ is only C so we have

$$T_{new}(X) = T(C) + f_X \geq T_{old}(X) \geq T(A).$$

Otherwise, if $T(B) \leq T_{old}(X)$ the nodes contributing to the computation of $T(X)$ are C and B.
Using this last assumption, (3.17) and (3.18) we have

$$T(C) \leq T(B) \leq T_{old}(X) \leq T(C)+f_X \Rightarrow 0 \leq T(B)-T(C) \leq f_X \Rightarrow (T(B)-T(C))^2 \leq f_X^2 \tag{3.19}$$

Moreover, by (3.15) and (3.17) we have

$$T(C) + f_X \geq T(A). \tag{3.20}$$

Computation of X leads to

$$\begin{aligned} T_{new}(X) &= \frac{T(C) + T(B) + \sqrt{2f_X^2 - (T(C) - T(B))^2}}{2} = \\ &= \frac{(T(C) + f_X) - f_X + T(B) + \sqrt{2f_X^2 - (T(C) - T(B))^2}}{2}. \end{aligned} \tag{3.21}$$

Using (3.20), (3.19) and (3.16) we obtain

$$\begin{aligned} T_{new}(X) &\geq \frac{T(A) - f_X + T(B) + \sqrt{2f_X^2 - (T(C) - T(B))^2}}{2} \geq \\ &\geq \frac{T(A) - f_X + T(B) + \sqrt{f_X^2}}{2} \geq \frac{T(A) + T(A)}{2} = T(A). \end{aligned} \tag{3.22}$$

Finally, let us remark that the cases when two or more nodes among B, C and D are *accepted*, can be treated as the previous cases. Note that if D is *accepted*, then B must also be *accepted* since $T(B) \leq T(D)$.
To complete the proof, it is necessary to show that the expression appearing under the square root in the computation of $T(X)$ expressed as a function of its two neighbors is nonnegative.

Let us start proving that the hypothesis (3.5) guarantees that

$$\frac{c(Z)}{c(Z')} \leq \sqrt{2} \tag{3.23}$$

for any couple of nodes Z and Z' such that

$$|Z - Z'| = \Delta x.$$

In fact, by assumption we have

$$|c(Z) - c(Z')| \leq L_c |Z - Z'|.$$

If $|Z - Z'| = \Delta x$, we have that

$$|c(Z) - c(Z')| \leq L_c \Delta x \leq (\sqrt{2} - 1)c_{min} \leq (\sqrt{2} - 1)c(Z')$$

which implies

$$c(Z) - c(Z') \leq (\sqrt{2} - 1)c(Z')$$

and then

$$c(Z) \leq \sqrt{2}\, c(Z').$$

Let us examine the three cases where we need to show that the radicand is nonnegative.

CASE 2
Since B is in the *narrow band*, there must be at least one neighbour belonging to *accepted*. Let E be this node (see Figure 3.3 and 3.1). Moreover, $T(A) \leq T(B)$ since A has been

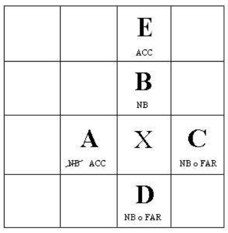


Figure 3.3: proof that radicand is positive under CFL condition (3.5)

chosen to be labeled as *accepted* and $T(E) \leq T(A)$ because E became *accepted* before A. By the previous results, we get

$$T(E) \leq T(B) \leq T(E) + f_B$$

which implies

$$T(A) \leq T(B) \leq T(E) + f_B \leq T(A) + f_B$$

and

$$0 \leq T(B) - T(A) \leq f_B. \tag{3.24}$$

Choosing $Z = X$ and $Z' = B$ in (3.23), we get

$$\frac{c(X)}{c(B)} \leq \sqrt{2}$$

and then

$$\sqrt{2} f_X \geq f_B. \tag{3.25}$$

Finally (3.24) and (3.25) imply

$$\sqrt{2} f_X \geq T(B) - T(A) \geq 0$$

so we can conclude that

$$2 f_X^2 - (T(B) - T(A))^2 \geq 0.$$

CASE 3 and *4*
In these cases, (3.13) and (3.19) guarantee respectively that the expression appearing under the radicand is always positive. ■

Let us show now that the value at the node which is labeled as *accepted* at every iteration is exact. Let us denote this value T_{min}. Since all the nodes in the *narrow band* have values greater than T_{min}, the previous proposition implies that using those nodes we cannot assign to a node a value lower than T_{min}. In conclusion (see [85]), the *up-winding* is respected and the value T_{min} can be considered as exact since it cannot be improved on the same grid (of course it can be improved if we reduce the discretization steps).

3.2 A semi-Lagrangian FM method

In this section we present our new Fast Marching semi-Lagrangian (FM-SL) scheme for the eikonal equation appeared for the first time in [38].
The idea which is behind the FM-SL method is rather simple: we follow the initialization and all the steps of the classical FM-FD method but the step where the value at the node x_i is actually computed. That step would require to iterate until convergence the scheme

$$v_h^k(x_i) = \min_{a \in B(0,1)} \{\beta v_h^k(x_i - hc(x_i)a)\} + 1 - \beta, \tag{3.26}$$

where $\beta = e^{-h}$ so that the typical fixed point iteration is applied "locally" at every single node following the order indicated by the FM-FD method. Note that we can replace a with $-a$ in (3.26) since we have $a \in A = B(0,1)$. We will prove that for a SL scheme based on a piecewise linear space reconstruction just a single iteration is needed to compute the exact (within the accuracy of the scheme) value at every node so that the computational effort is very limited and of the same order of the FM-FD method.
At the end of the section we will also give a sweeping version of our algorithm.

To avoid cumbersome notations we will denote the fully-discrete value function v_h^k by w.

3.2.1 The FM-SL scheme

We will always approximate the function $v = 1 - e^{-T}$ instead of T and use the fact that Kružkov transform is monotone. In fact, since $T_1 > T_2$ if and only if $v_1 > v_2$ we can work on the v variable without changing the rules for the up-date of the *narrow band* because the crucial point is to label as *accepted* the node in the *narrow band* where T (or v) attains its minimum. The above rule guarantees that we will process the nodes in an ordering which corresponds to increasing values of v.
Let us note that by means of the Kružkov transform one can also deal with the case when $c = 0$ in some regions (obstacles) since in that case the minimum time function to the target will have infinite value at some points whereas v will always stay bounded by 1.

Fast minimum search in $B(0, 1)$

We will start improving the minimum search which is typical of the SL-schemes. The search for a minimum in the unit ball $B(0, 1)$ will be solved algebraically for a linear interpolation which allows to compute the values $w(x_i - hc(x_i)a)$ using the known values at the nodes. Clearly, a new algebraic solution must be obtained (if possible) for other high order interpolations. Let us just recall that for the standard SL scheme the search for the minimum is usually restricted to a discretization of the unit ball $B(0, 1)$ which takes into account r points (or *controls* in the minimum time terminology) $a_1, a_2, \dots, a_k, \dots, a_r \in B(0, 1)$.
For example, one can construct a uniform grid on $\partial B(0, 1)$ with step $\Delta\theta = 2\pi/r$. To find the minimum, for every a_k the value $w(x_i - hc(x_i)a_k)$ is actually computed by interpolation. Although the choice of the type and order of the interpolation is completely free the most popular choices are *linear*, using the three values at the nodes which are closer to $x_i - hc(x_i)a_k$ and *bilinear*, using the four values of w at the vertices of the cell containing $x_i - hc(x_i)a_k$.
Once all the values for a_k, $k = 1, \dots r$ are computed the minimum is obtained by comparison. It is worth to note that this algorithm is quite slow and requires an high computational cost, however it can be applied to every high order interpolation. Moreover, it should be noted that this minimization problem is quite difficult since we expect to have non differentiable or even discontinuous solutions (if state constraints/obstacles are present in the domain) and that the comparison algorithm is very simple to implement and reasonably fast in low dimension especially when the search for the minimum can be restricted to the boundary of $B(0, 1)$ (as it will be the case in many examples). However, other algorithms for the minimization of non smooth functions can be applied and the interested reader can find in [26] and [50] recent improvements on the solution of this problem. These algorithms converge to the minimum in the limit so that they cannot be applied here since we want to have an exact evaluation of the computational cost.
It is important to note that the time step h in (3.26) can vary at every node. We will denote by $h_i = h(x_i)$ the time step corresponding to the node x_i, by $c_i = c(x_i)$ the velocity at x_i and by $\beta_i = e^{-h_i}$. When $c_i > 0$ it is always possible to choose

$$h_i = \frac{\Delta x}{c_i}. \tag{3.27}$$

In this way (3.26) can be written as

$$w(x_i) = \min_{a \in B(0,1)} \{\beta_i w(x_i - \Delta x\; a)\} + 1 - \beta_i. \tag{3.28}$$

In this situation, the nodes where $c_i = 0$ are actually treated apart from the other nodes: we just assign them the value $w = 1$ (which corresponds to $T = +\infty$) without any additional computation.

The method we propose here for the minimization problem has a low dimensional cost since for linear interpolation the search is restricted to the boundary of the unit ball. This is not a real restriction since, for our applications, the minimum in the unit ball is attained at the boundary, see (1.30). Later in this section we will show how this algorithm can be applied as a building block of our FM-SL scheme.
For simplicity, let us examine the situation in $\mathbb{R}^2$ considering a set of four cells each of side length Δx centered at the origin (see Figure 3.4). We want to compute the minimum

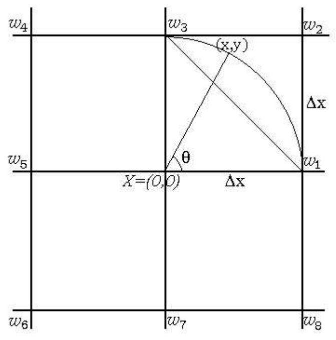


Figure 3.4: search for optimal control

of the function $w((0,0) - \Delta x\; a)$ for $a = (\cos\theta, \sin\theta)$ and $\theta \in [0, 2\pi)$. Let us introduce a vector $\mathbf{m} = (m_1, m_2, \ldots, m_8)$, the values of its components will be defined below. The minimum value we search will be given by $p = \min\{m_1, m_2, \ldots, m_8\}$.
Let us define the first four components of $\mathbf{m}$

$$m_1 = w(\Delta x, 0)\,, \quad m_2 = w(0, \Delta x)\,, \quad m_3 = w(-\Delta x, 0)\,, \quad m_4 = w(0, -\Delta x)$$

and let us search for the minimum in every orthant.

I orthant
Let w_1, w_2 and w_3 be the values of w corresponding respectively to the nodes $(\Delta x, 0)$, $(\Delta x, \Delta x)$ e $(0, \Delta x)$. The unique linear function $f(x, y)$ satisfying the conditions

$$f(\Delta x, 0) = w_1\,, \quad f(\Delta x, \Delta x) = w_2\,, \quad f(0, \Delta x) = w_3$$

is

$$f(x, y) = ax + by + c \tag{3.29}$$

where

$$a = \left(\frac{w_2 - w_3}{\Delta x}\right), \qquad b = \left(\frac{w_2 - w_1}{\Delta x}\right), \qquad c = w_1 - w_2 + w_3.$$

Let us define the real function

$$F(\theta) = f(\Delta x \cos\theta, \Delta x \sin\theta) = a\Delta x \cos\theta + b\Delta x \sin\theta + c\,, \quad \theta \in [0, 2\pi) \tag{3.30}$$

and look for the minimum of $F(\theta)$ in the interval $(0, \pi/2)$. Note that the extreme values $\theta = 0$ and $\theta = \pi/2$ are not included since the values at the extrema of that interval have been already included in $\mathbf{m}$ (they are m_1 and m_2). By differentiating with respect to θ we obtain

$$F'(\theta) = 0 \Leftrightarrow \theta = \arctan(b/a).$$

The interesting case is when $w_2 < w_1$ and $w_2 < w_3$, otherwise the minimum is w_1 or w_3. In this case, we get

$$a \neq 0\,, \quad b \neq 0\,, \quad b/a > 0\,, \quad \arctan(b/a) \in (0, \pi/2)$$

which means that the relative minimum is at $\theta_1^* = \arctan(b/a)$ and we set $m_5 = F(\theta_1^*)$. If $w_2 \geq w_1$ or $w_2 \geq w_3$ we set $m_5 = +\infty$ (or the highest machine number).

II orthant

Let w_3, w_4 and w_5 be the values of w respectively at the nodes $(0, \Delta x)$, $(-\Delta x, \Delta x)$ e $(-\Delta x, 0)$. The unique linear function $f(x, y)$ such that

$$f(0, \Delta x) = w_3\,, \quad f(-\Delta x, \Delta x) = w_4\,, \quad f(-\Delta x, 0) = w_5$$

is

$$f(x, y) = ax + by + c$$

where

$$a = \left(\frac{w_3 - w_4}{\Delta x}\right), \qquad b = \left(\frac{w_4 - w_5}{\Delta x}\right), \qquad c = w_3 - w_4 + w_5.$$

Again we will consider the composite function $F(\theta)$ defined in (3.30), and we observe that it has a relative minimum in $(\pi/2, \pi)$ if and only if $w_4 < w_3$ and $w_4 < w_5$. In this case we have

$$a \neq 0\,, \quad b \neq 0\,, \quad b/a < 0\,, \quad \arctan(b/a) \in (-\pi/2, 0). \tag{3.31}$$

Since we are in the second orthant the value of θ where the minimum for F is attained is $\theta_2^* = \arctan(b/a) + \pi$. Proceeding as in the first orthant we set $m_6 = F(\theta_2^*)$.
If $w_4 \geq w_3$ or $w_4 \geq w_5$ we set $m_6 = +\infty$.

The analysis of the third and fourth orthant follows in the same way and it will be skipped. Once all the components of $\mathbf{m}$ have been set, we just compute $p = \min\{m_1, m_2, \ldots, m_8\}$ and substitute it in the expression

$$w(0, 0) = \beta p + 1 - \beta. \tag{3.32}$$

This is done at every fixed point iteration till convergence. It is important to note that the above linear interpolation has a great advantage: the computation of the correct value of $w(0, 0)$ does not require more than one iteration given the values at the neighbouring nodes (along the axis directions and the diagonals) since $F(\theta)$ will not depend on $w(0, 0)$. This property will *not* hold for other high-order interpolations, *e.g.* quadratic interpolation. Another advantage of linear interpolation with respect to the comparison of the values in a discrete unit ball is that it gives the exact value of the optimal direction at the cost corresponding to a discretization of $B(0, 1)$ by just 8 directions.

The FM-SL algorithm

This section is devoted to the presentation of the fast marching version of the SL-algorithm, for simplicity the presentation is given in $\mathbb{R}^2$, the algorithm can be easily extended to $\mathbb{R}^n$. Let us start introducing the following definitions.

Definition 3.2 (Neighboring nodes for the SL scheme) *Let $X = (x_i, y_j)$ be a node of the grid. We define*

$$N_{FD}(X) = \Big\{(x_i, y_{j+1}), (x_i, y_{j-1}), (x_{i-1}, y_j), (x_{i+1}, y_j)\Big\},$$

$$D(X) = \Big\{(x_{i+1}, y_{j+1}), (x_{i+1}, y_{j-1}), (x_{i-1}, y_{j+1}), (x_{i-1}, y_{j-1})\Big\},$$

and

$$N_{SL}(X) = N_{FD}(X) \cup D(X).$$

The above definition is the natural extension of the Definition 2.1 for the semi-Lagrangian scheme. According to the new definition, the nodes in the *narrow band* will include also the diagonal directions and not only the four directions N, S, E, W as in the FM-FD method of section 2.

Sketch of the FM-SL algorithm

Initialization (see Figure 3.5)

1. The nodes belonging to the initial front Γ_0 are located and labeled as *accepted.* Their value is set to $w = 0$. We will denote by $\widetilde{\Gamma}_0$ this set of nodes.
2. The initial *narrow band* is defined, according to the Definition 3.2, taking the nodes belonging to $N_{SL}(\widetilde{\Gamma}_0)$ external to Γ_0. These nodes are labeled as *narrow band.* Their value is set to $w = 1 - e^{-\frac{\Delta x}{c}}$ (which corresponds to $T = \Delta x/c$) if they belong to $N_{FD}(\widetilde{\Gamma}_0)$ or to $w = 1 - e^{-\frac{\sqrt{2}\Delta x}{c}}$ (which corresponds to $T = \sqrt{2}\Delta x/c$) if they belong to $D(\widetilde{\Gamma}_0)$.
3. We label as *far* all the remaining nodes of the grid, their value is set to $w = 1$ (which corresponds to the value $T = +\infty$).

Main Cycle

1. Among all the nodes in the *narrow band* we search for the minimum value of w. Let us denote this node by A.
2. The node A is labeled as *accepted* and it is removed from the *narrow band.*
3. We label as *active* the nodes in $N_{SL}(A)$ which are not *accepted.* If there are *far* nodes, they are moved into the *narrow band.*
4. We compute (or recompute) the value w at the nodes belonging to $N_{FD}(A)$ which are *active*, iterating the fixed point operator

$$w(x_i) = \min_{a \in B(0,1)} \{\beta_i w(x_i - h_i c_i a)\} + 1 - \beta_i\,, \tag{3.33}$$

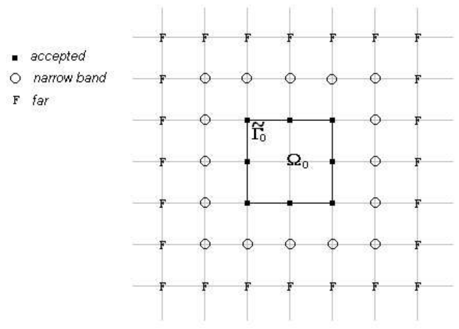


Figure 3.5: Initialization for FM-SL method, case $c > 0$

where $h_i c_i = \Delta x$. Note that just one iteration is needed as we will see in the following sections. Then, we compute by the same formula the value at the remaining *active* nodes in $N_{SL}(A) \setminus N_{FD}(A)$.

5. If the *narrow band* is empty the algorithm stops, else it goes back to Step 1.

Although the algorithm advances the *narrow band* also in the diagonal directions, according to the new definition, it computes first the values at the neighbouring nodes in the directions N, S, E, W (*i.e.* the FD directions) and then passes to the diagonal directions.

Some extensions: obstacles, infinite velocity.
We have seen that one can use our algorithm to deal with a front propagation with obstacles, *i.e.* regions where c vanishes. In [95] the problem has been analyzed and several tests have been presented for a SL-method based on the linear interpolation which treats the obstacle in a very simple way. The algorithm just assigns to the nodes belonging to an obstacle the value $w = 1$ in order to impose (indirectly and easily) a state constraints boundary conditions. In order to use the Fast Marching technique we just have to be careful and distinguish between nodes initialized to the value $w = 1$ because they are *far* and the ones to which was assigned the value $w = 1$ because they belong to an obstacle. In section 3.2.3 (Test 5) we will show a front propagating in presence of obstacles.
Another interesting extension for applications to image processing is when the domain of computation contains points with infinite velocity. This is the case, *e.g.* for the Shape-from-Shading problem when we have point of maximal light intensity in the image (see *e.g.* [81], [65] and Chapter 4). Let us illustrate the idea which is behind our solution. Let x_{i_0} be a node such that

$$\lim_{x \to x_{i_0}} c(x) = +\infty.$$

Our equation $c(x)|\nabla T(x)| = 1$ can be written as

$$|\nabla T(x)| = g(x) \tag{3.34}$$

where $g(x) = 1/c(x)$. Clearly, (3.34) is a degenerate eikonal equation since g vanishes at x_{i_0} (see Remark 1.7).

In order to compute $w(x_{i_0})$, we can set, according to (3.27), $h_{i_0} = 0$ and $\beta_{i_0} = 1$ and proceed as before setting in (3.33)

$$h_{i_0} c_{i_0} = \Delta x. \tag{3.35}$$

Let us extend the function $h(x)$ outside the nodes in the domain $Q \backslash \Omega_0$. Our choice (3.35) can be justified by the fact that we would expect in our algorithm

$$\lim_{x \to x_{i_0}} c(x) = +\infty\,, \quad \lim_{x \to x_{i_0}} h(x) = 0 \quad \text{and} \quad \lim_{x \to x_{i_0}} c(x)h(x) = \Delta x.$$

Note that, even if this argument is heuristic, it assigns to the node x_{i_0} the exact value for w. In fact, by (3.33), we get

$$w(x_{i_0}) = \min_{a \in B(0,1)} \{1 \cdot w(x_{i_0} - \Delta x\ a)\} + 1 - 1 = w(x_{i_0} - \Delta x a^*)$$

where a^* is the *optimal control.* Since the front has an infinite velocity at x_{i_0} the minimum time of arrival on it coincides with the minimum time of arrival on the circle of radius Δx centered at x_{i_0}. In section 3.2.3 (Test 6 and 7) we will show an application to a front propagation problem and to the Shape-from-Shading problem.

Properties of the FM-SL scheme

We start with the following easy result on the semi-Lagrangian discretization.

Proposition 3.3 *Let X be a node and assume that $w(X)$, defined by (3.33), is computed by interpolation using the three values $w^{(1)}$, $w^{(2)}$, $w^{(3)}$. Then,*

$$w(X) \geq \min \left\{ w^{(1)}, w^{(2)}, w^{(3)} \right\}. \tag{3.36}$$

Proof. Let $\beta = e^{-h}$, $h > 0$ and a^* be the optimal direction/control at X. The inequality

$$\beta w(X - h_i c_i a^*) + 1 - \beta \geq w(X - h_i c_i a^*)$$

is satisfied if and only if $w(X - h_i c_i a^*) \leq 1$. Since w is always less or equal to 1 (due to the Kružkov transform) we proved that

$$w(X) \geq w(X - h_i c_i a^*). \tag{3.37}$$

Since a simple property of linear interpolation guarantees that

$$\max \left\{ w^{(1)}, w^{(2)}, w^{(3)} \right\} \geq w(X - h_i c_i a^*) \geq \min \left\{ w^{(1)}, w^{(2)}, w^{(3)} \right\} \tag{3.38}$$

by (3.37) and (3.38) we end the proof. ∎

In order to prove that the Fast Marching version of our SL-scheme converges to the viscosity solution in a finite number of steps we have to prove first that the fast method for the minimum analyzed in section 3.2.1 matches with the Fast Marching technique. This is necessary since the *narrow band* of FM-SL method is larger than the *narrow band* of FM-FD method as a consequence of the new definition of neighboring nodes. In particular

we will show that the algorithm automatically reject *far* nodes from the computation as in the standard *up-wind* finite difference discretization.
Let X be the node where we want to compute $w(X)$. Without loss of generality, we will assume that the optimal value is attained at a direction $\theta^* \in [0, \pi/2]$, *i.e.*

$$a^* = (\cos\theta, \ \sin\theta), \quad \theta \in [0, \pi/2]. \tag{3.39}$$

We will examine in detail all the possible configurations for this situation which will be referred to in the sequel as the "minimum in I orthant" case (see Figure 3.6). For simplicity, let us assume $c > 0$ so that a node is labeled as *far* if and only if its value is $w = 1$.

Proposition 3.4 *Let X be a node and let $w(X)$ be defined by (3.33). The value $w(X)$ will* not *be computed by interpolation using nodes labeled as* far*.*

Proof. Let us give the proof for minimum in the I orthant. The analysis for the other orthants is similar and can be easily obtained by symmetry arguments.

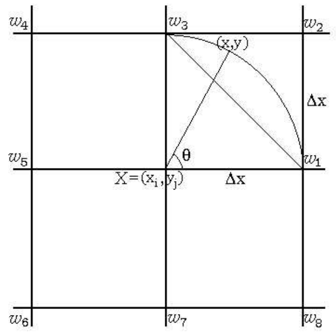


Figure 3.6: Analysis of the minimum in the I orthant

1. $w_1 = w_2 = w_3 = 1$.
   This configuration can not occur. In fact, since the minimum is attained in the I orthant we should have

   $$w_4 = w_5 = w_6 = w_7 = w_8 = 1.$$

   But this is not possible since we compute at X only when at least one of the nodes belonging to $N_{SL}(X)$ has been labeled to *accepted* in one of the previous iterations and an *accepted* node must have a value lower than 1.

2. Among w_1, w_2 and w_3 there are two values equal to 1.

   (a) $w_1 = w_3 = 1$: this case can not occur. In fact, since the minimum is attained in the I orthant we must have $w_2 \leq w_1, w_3, w_4, \ldots, w_8$. The node that must be labeled as *accepted* is the one corresponding to the value w_2. This implies that the values w_1 and w_3 must be computed before X (see the sketch of the algorithm).

(b) $w_1 = w_2 = 1$: the minimum value is w_3. A new iteration to compute $w(X)$ would not give a lower value, so the optimal value is obtained in just one iteration.

(c) $w_2 = w_3 = 1$: the minimum value is w_1. Again, we will not get a lower value iterating and the optimal value is obtained in just one iteration.

3. Among w_1, w_2 and w_3 only one value is equal to 1

   (a) $w_2 = 1$: since f is linear the minimum will be attained by w_1 or w_3. The optimal value is obtained in just one iteration.

   (b) $w_1 = 1, \quad w_3 \leq w_2$: the minimum is w_3.

   (c) $w_1 = 1, \quad w_3 > w_2$: this is the most delicate case since $w_2 < w_1,\ w_3$. The minimum for F will be attained at some $\theta^* \in (0, \pi/2)$. The value $w(X)$, obtained by linear interpolation will not be correct since it depends on $w_1 = 1$, which is a conventional value. Moreover, note that a new iteration of the fixed point map at X will not make $w(X)$ decrease since w_1 is frozen and so does $w(X)$. If this case could occur we will not get convergence to the correct value even in the limit on the number of iterations. Note that this difficulty can not occur neither for the global SL scheme where *all* the nodes are computed at the same iteration nor for the FM-FD method where the values corresponding to *far* nodes are not used in the stencil. The following argument shows that this case can not occur also for the FM-SL scheme. Since $w_1 = 1$, the corresponding node is labeled as *far* at the current iteration. This implies the nodes labeled as *accepted* at the previous iteration do not belong to $N_{SL}(w_1)$. As a consequence, w_2 belongs to the *narrow band*. By Proposition 3.3 we have $w(X) > w_2$. This implies that X can not be labeled as *accepted* before the nodes corresponding to w_2. Once w_2 becomes *accepted* the algorithm computes w_1 and w_3 before computing $w(X)$ so that the values at nodes labeled as *far* will not contribute.

   (d) $w_3 = 1, \quad w_1 \leq w_2$: the minimum is w_1. The optimal value is obtained in just one iteration.

   (e) $w_3 = 1, \quad w_1 > w_2$: analogous to case (3c).

■

3.2.2 Convergence of FM-SL

As for the FM-FD method we have to prove that the minimal value of the nodes of the *narrow band* can not decrease if we iterate the fixed point operator, *i.e.* it coincides with the value obtained by the discrete operator working on all the nodes. As we have seen, the values at the nodes belonging to the *narrow band* are not accepted all together. Only the minimal value is accepted at every iteration (this is a very pessimistic choice which simplifies the theoretical result). The following proposition shows the bounds on the number of times that one node can be recomputed and it is a building block for the convergence of the scheme.

Proposition 3.5 *Let X be a node in the* narrow band *such that $w(X) = w_{old}(X)$. Let us assume that at the current iteration the algorithm needs to compute a new value $w_{new}(X)$ for X. Moreover, let us assume that at the current iteration the following property holds true:*

$$\textit{If } A \textit{ belongs to the } \text{narrow band} \textit{ and } B \textit{ is labeled as } \text{accepted}\ , \ \textit{then } w(A) \geq w(B) \tag{3.40}$$

The following properties hold:

1. *If the value $w_{old}(X)$ was computed at an iteration in which a grid point $A_1 \in N_{FD}(X)$ was labeled as* accepted *then it is impossible that $w_{new}(X) < w_{old}(X)$.*

2. *If the value $w_{old}(X)$ was computed at an iteration in which a grid point $A_2 \in D(X)$ was labeled as* accepted *then to the node X it can be assigned a new value $w_{new}(X) < w_{old}(X)$ but it will always satisfy the inequality $w_{new}(X) \geq w(A_2)$.*

Proof. Let us start from the first statement.

1. Let us assume that when the value w_{old} was assigned to X the node A_1 has been the (unique) node belonging to $N_{FD}(X)$ which has been labeled as *accepted.* When the algorithm computed $w(X) = w_{old}(X)$ we certainly had

$$\min_{a \in \partial B(0,1)} w(X - \Delta x\ a) = w^* \leq w(A_1)$$

   since there is a direction/control $\bar{a} \in \Big\{(1,0),(0,1),(-1,0),(0,-1)\Big\}$ such that $w(X - \Delta x\ \bar{a}) = w(A_1)$. The only possibility to have at X a value lower than $w_{old}(X)$ in the following iterations of the algorithm is that a value assigned to a node belonging to $N_{SL}(X)$ be lower than w^*. However, by Proposition 3.3 we know that this value can not be computed using in the stencil the values at the nodes of the actual *narrow band* because they are all greater than $w(A_1) \geq w^*$ which has been *accepted* (as (3.40) assures). A lower value could be computed only using a stencil which contains nodes already *accepted* in one of the previous iterations since they all have values lower than $w(A_1)$. This is not possible since all the nodes which are neighbors of those *accepted* nodes have been already computed and they have a value greater or equal to $w(A_1)$ since they have not been labeled as *accepted.*

2. Let us assume, for simplicity, that the node A_2 is the unique node belonging to $D(X)$ which has been labeled as *accepted* and let $w_{old}(X)$ be the value assigned at X at the same iteration. When a node $A_1 \in N_{FD}(X)$ has been labeled as *accepted* before A_2, the result holds true by the arguments of the above Case 1.
   Let us assume that A_2 be the unique neighbor of X which has been labeled as *accepted.* Then we have

$$\min_{a \in \partial B(0,1)} w(X - \Delta x\ a) = w^* \geq w(A_2)$$

   It is always possible that using $w(A_2)$ one can obtain a new value $w_{new}(X)$ lower than $w_{old}(X)$. However, by (3.40) and Proposition 3.3 all the new values will be greater or equal to $w(A_2)$ therefore $w_{new}(X) \geq w(A_2)$. ∎

Remark 3.6 *Note that the previous proposition allows to accelerate the algorithm. In fact, one can save CPU time avoiding to recompute the values at the nodes corresponding to Case 1. However, they can not be labeled as* accepted *before their value is the minimum in the* narrow band. *An important consequence of Proposition 3.5 and the above observation is that every node can be computed at most* 5 *times, this is one of the reasons why the CPU time for FM-SL is slightly larger than that for the FM-FD method where a node can be computed at most 4 times. We will see in the last section that the FM-SL method produces a more accurate approximation of the viscosity solution which justifies a small increment in the CPU time.*

The following result is an analogue of Proposition 3.1 and it is crucial to prove convergence in a finite number of steps.

Proposition 3.7 *Let w be defined in (3.33) and let $w(X)$ be the value assigned at X at the same iteration when a node $Z \in N_{SL}(X)$ is labeled as* accepted. *Assume that*

$$c(x) \geq 0\,, \quad \textit{for any} \;\; x \in Q \backslash \Omega_0.$$

Then, we have

$$w(X) \geq w(Z). \tag{3.41}$$

Proof. We examine all the cases corresponding to a minimum in the I orthant (see Figure 3.6). The proof will be obtained by induction on the number of iterations of the algorithm. At the first step the result holds true by our initialization.
Let us consider the n-th step of the algorithm. The induction hypothesis implies that at the current iteration the values of nodes in the *narrow band* are greater than values of nodes labeled as *accepted*. Therefore (3.40) holds true so we can apply Proposition 3.5. Our proof will be divided into three parts.

CASE 1 : $w_1, \dots, w_8$ are *narrow band* or *far* (before Z is labeled as *accepted*).
If Z belongs to the I orthant we have seen by Proposition 3.3 that

$$w(X) \geq \min \Big\{ w_1, w_2, w_3 \Big\} = w(Z).$$

If Z does not belong to the I orthant we have

$$w(X) \geq \min \Big\{ w_1, w_2, w_3 \Big\} \geq w(Z)$$

since Z as been labeled as *accepted*.

CASE 2: one node $w_1, \dots, w_8$ is *accepted* (before Z is labeled as *accepted*).
Let us denote by P this node. When P was accepted the value at X was $w_{old}(X)$. Now the value at X has to be recomputed. We can only have one of the following situations:

1. P belongs to the I orthant.
    (a) Z belongs to the I orthant
        i. See Figure 3.7-(a). By Proposition 3.5 Z and B cannot be assigned to a lower value after P became *accepted*, so $w_{new}(X) = w_{old}(X)$ and $w_{old}(X) \geq w(Z)$ since Z is the node chosen to be labeled as *accepted*.

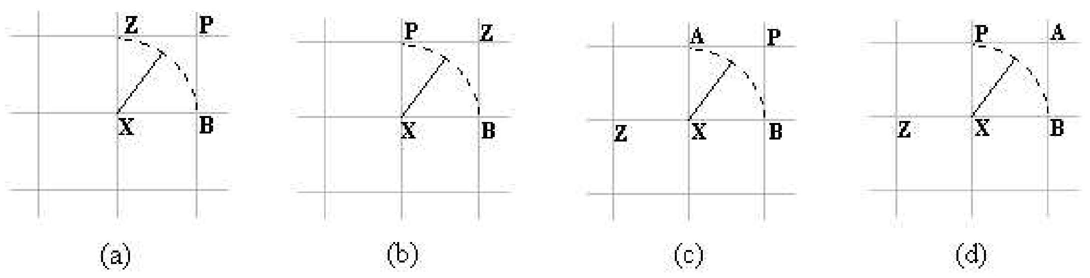


Figure 3.7: Four different configurations for CASE 2.

ii. See Figure 3.7-(b). When Z is *accepted* the minimum is attained at P and this implies again $w_{new}(X) = w_{old}(X)$.

(b) Z does not belong to the I orthant

i. See Figure 3.7-(c). In the iterations between the acceptation of P and that of Z the values $w(A)$ and $w(B)$ can not be changed. Moreover, the minimum is attained in the I orthant so we have $w_{new}(X) = w_{old}(X)$.

ii. See Figure 3.7-(d). We know that the value $w(A)$ has not been replaced, $w(B)$ can not be lower than $w(P)$ and that the minimum is attained in the I orthant. Then, the minimum is attained at P and $w_{new}(X) = w_{old}(X)$.

2. P does not belong to the I orthant.
Since the minimum is attained in the I orthant this means that P does not have effect on the computation at X and we are back to CASE 1.

CASE 3: more than one value among $w_1, \dots, w_8$ has been labeled as *accepted* (before Z is labeled as *accepted*).
This case can be solved by the arguments in *CASE 2*. ■

As for the FM-FD method (see [85]) we can now conclude that the value of the node which is labeled as *accepted* at every iteration can not be decreased if we iterate the fixed point operator. In fact, let us denote this value w_{min}. Since all the nodes in the *narrow band* have values greater than w_{min}, the previous result implies that using those nodes we cannot assign to a node a value lower than w_{min}. In conclusion, the *up-winding* is respected and the value w_{min} can be considered exact since it cannot be improved on the same grid (of course it can be improved if we reduce the discretization steps).
It is interesting to note that the FM-SL scheme does not require a stability CFL type condition as required by the FM-FD scheme.

The SL scheme is consistent as it has been proved, *e.g.* in [52]. Moreover, choosing $\Delta x = \Delta y$ we get that local truncation error is $O(\Delta x)$.
We will prove that the solution computed by the FM-SL method is identical to the solution computed by the standard semi-Lagrangian scheme where the computation is repeated on every node of the grid until convergence. Naturally, if the two schemes compute the same values, convergence of the FM-SL method to the viscosity solution is just a consequence of that of the standard SL scheme.

Theorem 3.8 *Let $(V_i)_{i=1,\dots,M}$ be the matrix containing the final values on the n-dimensional grid and let*

$$V_i = F(V_{i-k}, \dots, V_{i+l}) \tag{3.42}$$

be the iteration corresponding to the numerical scheme. Let $\widehat{V}$ be the matrix of the approximate solution corresponding to the fixed point iteration (3.42) *and let $\overline{V}$ the matrix containing the final values of the approximate solution corresponding to the FM technique applied to the* same *scheme (*i.e. *the result obtained when the* narrow band *is empty). Then, $\overline{V} = \widehat{V}$.*

Proof. The two matrices coincide if and only if

$$\overline{V}_i = F(\overline{V}_{i-k}, \dots, \overline{V}_{i+l}), \qquad \text{for any } \; i = 1, \dots, M. \tag{3.43}$$

Assume the *narrow band* is empty and take $\overline{V}$ as initial guess for the fixed point technique, this will not change the solution since the value is computed by the same scheme. When all the nodes are *accepted* the equality (3.43) must hold for every i. In fact, if the equality is not true at one node than its value can still be improved and this implies that the list of *narrow band* or *far* nodes is not empty, which gives the contradiction. ■

The above results allow us to draw some conclusions about the order of complexity of the FM-SL scheme. The values $w(X)$ computed by (3.33) is an approximation of $v(X)$ which has been computed in at most 5 times for every nodes. This means that the computational cost can be estimated as in FM-FD scheme. One component is given by the cost of the heap-sort method to select the minimum value in the *narrow band* , the other component is given by the computational cost at every node. This globally gives a cost $O(N \log(N_{nb}))$, where N is the total number of nodes and N_{nb} the number of nodes in the narrow band.
Since the values which have been labeled as *accepted* at every iteration can not be improved by the global fixed point iteration, *i.e.* they coincide with the same values obtained by the global fixed point operator, the *a-priori* error estimates are still valid for the solution obtained by the FM-SL method. In the last section we will present several tests which confirm these theoretical results.

Boundary conditions on ∂Q

We define outside Q a strip of ghost nodes where we set $w = 1$. If they enter in the *narrow band*, at the end of the iteration their value is set back to $w = 1$ to avoid their contribution to the computation of other internal nodes. When the minimal value on the nodes of the *narrow band* is 1, the ghost cells will be the only non *accepted* nodes and we can stop the computation. In general, any constant larger than the maximum of the solution in Q can be used to assign the value at the ghost nodes (a typical choice is to set the solution to $+\infty$ if there is no *a-priori* estimate on the solution).

A semi-Lagrangian Fast Sweeping methods

The Fast Sweeping method introduced in Section 2.4 has an easy extension to the semi-Lagrangian case. In fact, we can easily substitute the FD discretization by the SL discretization maintaining the ordering in which nodes are visited. Obviously, we expect that at least in the case $c(x) \equiv \textit{constant}$ the Fast Sweeping semi-Lagrangian (FS-SL in the sequel) can compute in four iterations exactly the same solution of FM- SL. In the next section we run this algorithm in the case $c(x) \equiv 1$ with two different initial fronts and we will see that this intuition is actually true.

3.2.3 Numerical experiments

In this section we present some numerical experiments performed with Matlab 7 on a Personal Computer equipped with Pentium IV 2.80 GHz processor, 512 MB RAM.
The main goal is to compare the FM-FD method and the FM-SL method. We also compare these methods with the iterative semi-Lagrangian method and Fast Sweeping method based on a semi-Lagrangian discretization. First two tests are devoted to approximate the solution of model problems where we know the exact solution, so we can compute the L^∞ error and L^1 error. Other tests are devoted to solve more complicated problems and applications in which the velocity function $c(x)$ does not satisfy standard assumptions such as Lipschitz continuity and boundedness.
If not specified otherwise, we choose $Q = [-2, 2]^2$ as our computational domain.

Tests on model problems

In the following tests we compare the exact solution T with the solution $\widehat{T}$ computed by FM-FD method and FM-SL method described above. Note that in the implementation of the FM-SL algorithm we have used the observation in Remark 3.6 to speed up the computation.
We compute

$$E_{\infty,\Delta x} = \max_{i,j} |T_{i,j} - \widehat{T}_{i,j}|\,, \qquad E_{1,\Delta x} = (\Delta x)^2 \sum_{i,j} |T_{i,j} - \widehat{T}_{i,j}| \tag{3.44}$$

and the rate of convergence r in some model problems in $\mathbb{R}^2$. We consider 51×51, 101×101 and 201×201 grids[1] corresponding respectively to $\Delta x = 0.08$, $\Delta x = 0.04$ and $\Delta x = 0.02$. Since we know that there is a constant C such that

$$E_{p,\Delta x} \le C\Delta x^r \quad \text{and} \quad E_{p,\Delta x/2} \le C\left(\frac{\Delta x}{2}\right)^r, \qquad p = 1, \infty$$

we obtain that the numerical rate of convergence is

$$r = \log_2\left(\frac{E_{p,\Delta x}}{E_{p,\Delta x/2}}\right), \qquad p = 1, \infty.$$

Moreover, we compare these algorithms with the classical iterative semi-Lagrangian method (SL) in which we chose $\max_{i,j} |w^{(k)}_{i,j} - w^{(k-1)}_{i,j}| < \varepsilon$, $\varepsilon = 10^{-7}$ as stopping criterion and with the Fast Sweeping method based on a semi-Lagrangian discretization (FS-SL) performing just four iterations in different order.
Let us finally remark that in all cases the condition (3.5) holds.

[1] In these grids there is a node corresponding to the point $(0, 0)$

Test 1

$$\Gamma_0 = (0, 0)\,, \quad c(x, y) \equiv 1$$

Exact solution:

$$T(x, y) = \sqrt{(x^2 + y^2)}$$

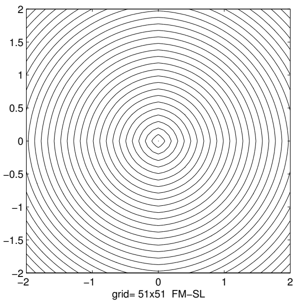


Figure 3.8: level sets of $T(x)$ computed by FM-SL method, 51×51 grid

method	Δx	L^∞ error	L^1 error	CPU time (sec)
FM-FD	0.08	0.0875	0.7807	0.5
FM-SL	0.08	0.0329	0.3757	0.7
SL (46 it)	0.08	0.0329	0.3757	8.4
FS-SL	0.08	0.0329	0.3757	0.8
FM-FD	0.04	0.0526	0.4762	2.1
FM-SL	0.04	0.0204	0.2340	3.1
SL (86 it)	0.04	0.0204	0.2340	60
FS-SL	0.04	0.0204	0.2340	3.2
FM-FD	0.02	0.0309	0.2834	9.4
FM-SL	0.02	0.0122	0.1406	14
SL (162 it)	0.02	0.0122	0.1406	443.7
FS-SL	0.02	0.0122	0.1406	12.5

Table 3.1: errors for Test 1

method	L^∞ $(0.08 \to 0.04)$	L^∞ $(0.04 \to 0.02)$	L^1 $(0.08 \to 0.04)$	L^1 $(0.04 \to 0.02)$
FM-FD	0.7342	0.7675	0.7132	0.7487
FM-SL	0.6895	0.7417	0.6831	0.7349

Table 3.2: rate of convergence in L^∞ and L^1 computed by errors in Table 3.1

Results are summarized in Table 3.1. As expected, in all cases errors reduce as Δx decreases. The numerical rate of convergence (Table 3.2) is in the interval $[0.5, 1]$ for both

methods.
FM-SL method and SL give exactly the same errors in accordance with Theorem 3.8 and they are also equal to the errors of FS-SL as expected since FS-SL converges in four iterations in the case c is constant. These errors are about the half of the FM-FD method errors although both are first order methods. This is due to the fact that semi-Lagrangian discretization is able to follow every direction of the characteristic flow.
Both methods based on Fast Marching technique are dramatically faster then iterative method SL. Nevertheless we want to note that only one iteration of the iterative scheme is less expensive with respect to the single iteration needed by Fast Marching based algorithms. This is due to the fact that the *narrow band* technique requires 1) to compute a minimum over nodes in the *narrow band* and 2) to access the data in an almost random manner rather than a systematic way along the loop indices (see [63]). Finally we note that CPU time needed by FM-SL method is slightly larger than CPU time needed by FM-FD method. This due to the fact that 1) the *narrow band* is bigger in the first method therefore the search for the minimum in the *narrow band* is more expensive and 2) in FM-SL method we need to compute the minimum over the unit ball $B(0, 1)$.

Test 2

$$\begin{gathered}\Gamma_0 = \text{unit square centered in } (-1,1) \text{ and rotated by } 11.25^\circ \cup \\ \text{circle with radius } R = 0.5 \text{ centered in } (0,-1) \cup \\ \text{square with side 0.4 centered in } (1.4,1.4)\end{gathered}$$

$$c(x, y) \equiv 1$$

Exact solution: $T(x, y)$ = minimum between the distance function of the square rotated, the circle and the square.

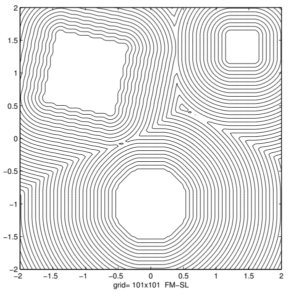


Figure 3.9: level sets of $T(x)$ computed by FM-SL method, 101×101 grid

Results are summarized in Table 3.3. In this test the shape of the initial front is much more complicate but errors have the same behavior as in the previous simple Test 1, although the difference between errors is smaller.
The FS-SL seems to be the best method. It has the smallest error and the CPU time is slightly larger than that of FM-FD. This is probably due to the fact that the structure of the *narrow band* is very complicate and it is very large in terms of nodes.
Also in this case the rate of convergence (Table 3.4) is grater than 0.5.

method	Δx	L^∞ error	L^1 error	CPU time (sec)
FM-FD	0.08	0.0625	0.2154	0.5
FM-SL	0.08	0.0440	0.1849	0.7
SL (30 it)	0.08	0.0440	0.1849	4.9
FS-SL	0.08	0.0440	0.1849	0.7
FM-FD	0.04	0.0393	0.1120	2.2
FM-SL	0.04	0.0215	0.1044	3.1
SL (55 it)	0.04	0.0215	0.1044	34.1
FS-SL	0.04	0.0215	0.1044	2.9
FM-FD	0.02	0.0248	0.0669	10.2
FM-SL	0.02	0.0135	0.0633	14.5
SL (102 it)	0.02	0.0135	0.0633	246.6
FS-SL	0.02	0.0135	0.0633	11.4

Table 3.3: errors for Test 2

method	L^∞ $(0.08 \to 0.04)$	L^∞ $(0.04 \to 0.02)$	L^1 $(0.08 \to 0.04)$	L^1 $(0.04 \to 0.02)$
FM-FD	0.6693	0.6642	0.9435	0.7434
FM-SL	1.0332	0.6714	0.8246	0.7218

Table 3.4: rate of convergence in L^∞ and L^1 computed by errors in Table 3.3

Applications

In the following we try to use FM-SL method in some classical applications of the eikonal equation like the minimum time problem and Shape-from-Shading. We consider some cases not covered by the theory in which $c(x,y)$ is discontinuous, $c(x,y)$ vanishes in some regions (state constraints) and $c(x,y)$ has infinite values. We also consider the *anisotropic* case in which the velocity field c depends on (x,y) and on the control a. The results we obtained are very satisfactory even in these cases.

Test 3: non-constant velocity.

$$\Gamma_0 = \partial B(0,\varepsilon)\,, \quad \varepsilon = \Delta x/2$$

$$c(x,y) = |x+y|$$

In this case the velocity field is non-constant. Figure 3.10 shows the value function $T(x,y)$ and level sets of T. On the line $x=-y$ the solution T is not defined since its correct value is $T=+\infty$.
The FS-SL method needs 12 iterations to reach convergence and it is more than three times slower than FM-SL method.

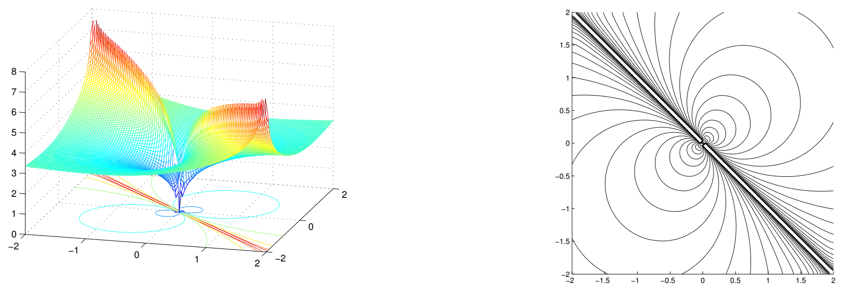


Figure 3.10: value function T (left) and level sets of T (right)

Test 4: discontinuous vector field.

$$\Gamma_0 = (-1, 0).$$

$$c(x,y) = \begin{cases} 0.4 & (x,y) \in [0.5,1] \times [0,0.5] \\ 1 & \text{elsewhere} \end{cases}$$

In this case the velocity field is discontinuous. Figure 3.11 shows the value function

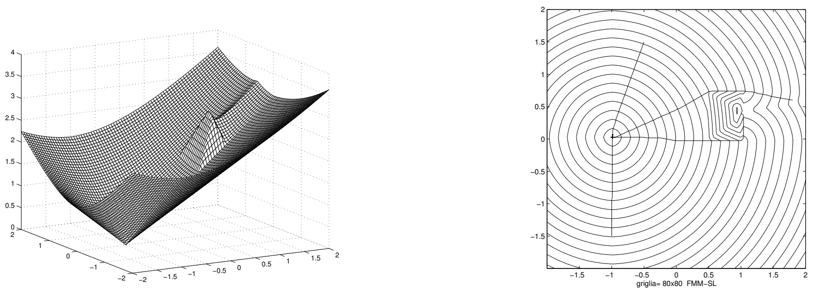


Figure 3.11: value function T (left) and level sets of T with some optimal trajectories (right)

$T(x,y)$ and level sets of T. Figure 3.11-right also shows some optimal trajectories which start from four different points and reach the target Γ_0 in the minimum time with speed $c(x,y)$. The FS-SL method converges in 8 iterations.

Test 5: state constraint problem.

$$\Gamma_0 = (-1, -1).$$

$$c(x,y) = \begin{cases} 0 & (x,y) \in ([0,0.5] \times [-2,1.5]) \cup ([1,1.5] \times [-1.5,2]) \\ 1 & \text{elsewhere} \end{cases}$$

In this test the velocity field vanishes in two different regions (the obstacles). Figure 3.12 shows the computational domain, the value function $T(x,y)$ and level sets of T. Figure 3.12-right also shows one optimal trajectory which starts from the point $(1.8, 1.5)$

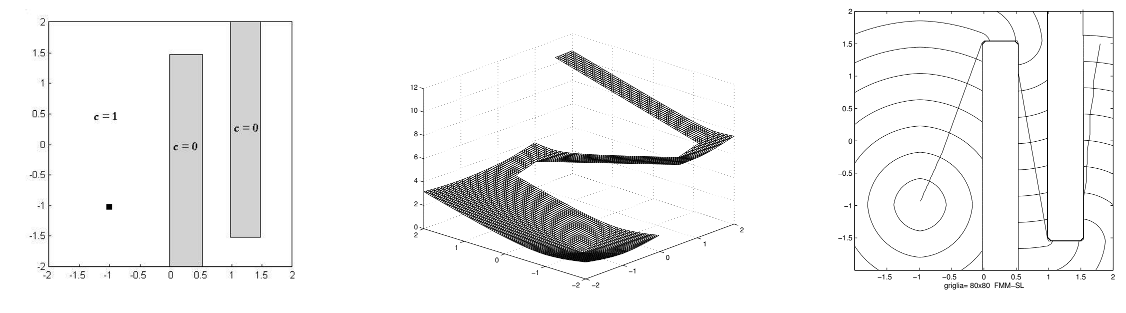


Figure 3.12: domain of the equation (left), value function T (center) and level sets of T with one optimal trajectory (right)

and reaches Γ_0 in the minimum time avoiding obstacles. We remark that since we use Kružkov transform and compute v, we do not need to modify the numerical scheme to deal with state constraints. Also in this case the FS-SL method converges in 8 iterations.

Test 6: infinite velocity.

$$\Gamma_0 = (-1, 0).$$

$$c(x, y) = \begin{cases} +\infty & x \geq 1 \\ 1 & \text{elsewhere} \end{cases}$$

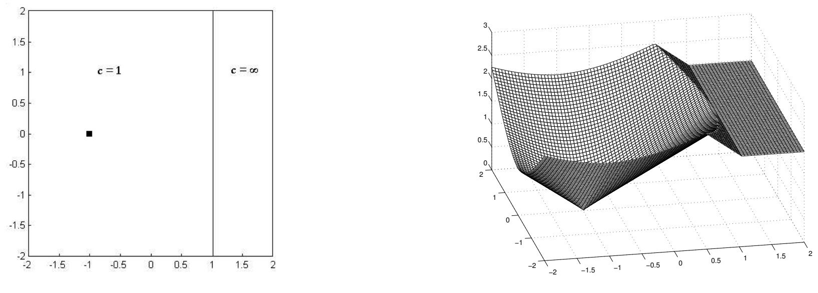


Figure 3.13: domain of the equation (left) and value function T (right)

In this case the front can propagate instantaneously in the region $R = \{x \geq 1\}$. It corresponds to the case of the following degenerate eikonal equation (see Remark 1.7)

$$|\nabla T| = f(x, y) \quad \text{with } f = 0 \text{ in } R.$$

Figure 3.13 shows the computational domain and value function $T(x, y)$. In this test we used the technique described in section 3.2.1 in order to deal with this kind of vector field. This technique allows to reconstruct a perfect *flat* surface on R as the theory and the physical sense require. This technique can be very useful in Shape-from-Shading problems.

Test 7: Shape-from-Shading.

$$Q = [-1,1]^2\,, \qquad \Gamma_0 = \text{silhouette of a vase}$$
$$c(x,y) = \left(\sqrt{\frac{1}{I^2}-1}\right)^{-1}\,, \quad I(x,y) = \text{intensity light function}$$

In this test we solve the Shape-from-Shading problem in the simple case of a vase. Figure

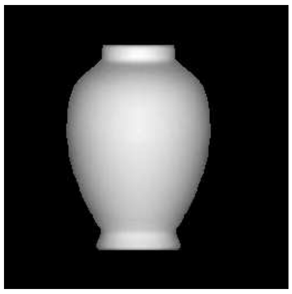
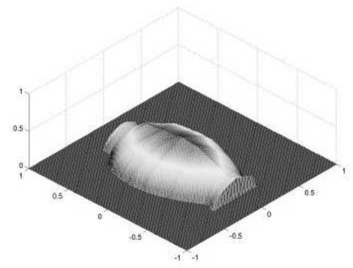



Figure 3.14: initial image (left) and reconstructed surface (right)

3.14-left shows the initial image and Figure 3.14-right shows the reconstructed surface. By the symmetry of the problem we guess that all characteristic curves start from the right and left side of the image, so we can impose Dirichlet boundary condition just on the right and left side of the domain and state constraints elsewhere (see Chapter 4 for some considerations about boundary conditions in Shape-from-Shading problem.

Test 8: Poincaré model.

$$Q = [-1,1]^2\,, \qquad \Gamma_0 = (-0.65,-0.65).$$
$$c(x,y) = \begin{cases} 1-(x^2+y^2)\,, & x^2+y^2<1 \\ 0 & \text{elsewhere} \end{cases}$$

This example is an interesting application of the eikonal equation to the Poincaré model of the hyperbolic geometry. Figure 3.15 shows the computational domain, the value function $T(x,y)$ and level sets of T. The FS-SL method converges in 8 iterations.
As result of the particular choice of the velocity field (see [69]), the optimal trajectories of the associated minimum time problem correspond to the hyperbolic straight lines. Moreover, the level sets of T are hyperbolic circles with center Γ_0 (*i.e.* the sets of points which have the same hyperbolic distance from Γ_0).

Test 9: geodesics on a nonsmooth surface.

$$Q = [-1.5,1.5]^2\,, \qquad \Gamma_0 = (0,-0.6).$$
$$\text{Surface}: z(x,y) = \begin{cases} 1-(|x|+|y|)\,, & |x|+|y|<1 \\ 0 & \text{elsewhere} \end{cases}$$

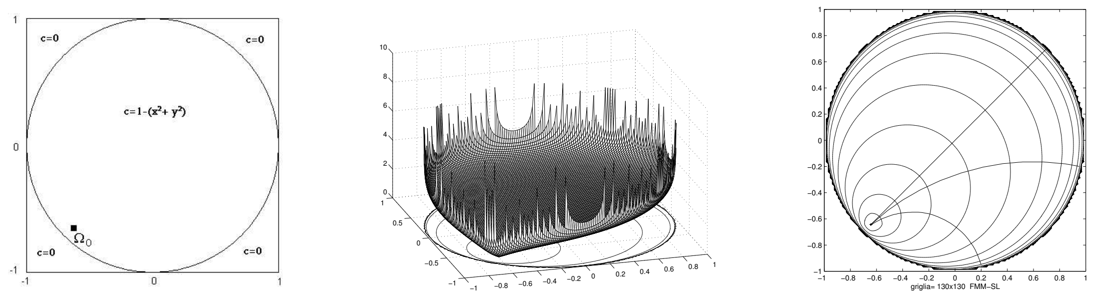


Figure 3.15: domain of the equation (left), value function T (center) and level sets of T with some optimal trajectories (right)

$$\tilde{c}(x, y) \equiv 1 \quad \text{(on the surface)}$$

In this case we want to solve a minimum time problem on a surface $z = z(x, y)$. The 3D

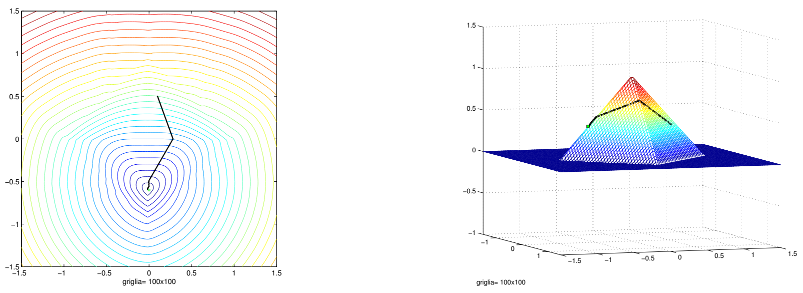


Figure 3.16: level sets of T (left) and an optimal trajectory on the surface z (right)

problem can be easily reduced to a 2D problem modifying the velocity field in according to the function z. In fact, if the intrinsic velocity on the surface in equal to 1, it can be shown (see [84, 64]) that the velocity of the corresponding 2D problem becomes

$$c(x, y, a) = \frac{1}{\sqrt{1 + (\nabla z \cdot a)^2}}.$$

Figure 3.16 shows the level sets of T and the surface with an optimal trajectory on it. The starting point is $(0, 0.5)$.

We remark that the dependence of c on a changes the properties of the solution of the equation. In fact the equation for anisotropic front propagation is

$$\begin{cases} v(x) + \max\limits_{a \in B(0,1)} \{c(x, a) a \cdot \nabla v(x)\} = 1 & x \in \mathbb{R}^n \backslash \Omega_0 \\ v(x) = 0 & x \in \partial\Omega_0 \end{cases} \tag{3.45}$$

In this case Fast Marching technique is no more directly applicable (there is no guarantee that convergence is reached in just one iteration, see Section 2.2). This is true for FM-SL method too, but we stress out that scheme (3.33) requires tiny modifications to deal with this kind of velocity field. Moreover, if we use the function w computed by FM-SL method as starting point of the iterative scheme SL, we can reach convergence in very few iterations.

3.3 A FM method for Pursuit-Evasion games without state constraints

In this section we present a generalization of the semi-Lagrangian Fast Marching method to Pursuit-Evasion games (see Section 1.4.1 for the definition of the problem). This work is appeared in [39]. We point out that this is the first time the FM technique is extended to *non-convex* Hamiltonians, in particular to minmax Hamiltonians which appear in the analysis of differential games. We will discuss the main ideas which are behind this new algorithm also showing some numerical results on classical problems.

3.3.1 The FM-SL scheme for Pursuit-Evasion games

Let us consider the Isaacs equation for the lower value of a differential game

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

where f is the dynamics for the game, A and B are two compact sets in $\mathbb{R}^m$ representing respectively the control set for the first player (Pursuer) and the second player (Evader) and $\mathcal{T}$ is a closed set with non empty interior representing the target for the game. The fully-discrete scheme based on the Discrete Dynamic Programming Principle is (see 1.5.2)

$$\begin{cases} w(x_i) = \max_{b\in B}\min_{a\in A}\big\{e^{-h}w(x_i+hf(x_i,a,b))\big\} + 1 - e^{-h} & x_i\in I_{in} \\ w(x_i) = 0 & x_i\in I_{\mathcal{T}} \end{cases} \tag{3.46}$$

where we denoted by w the fully-discrete value function v_h^k. It is well known that at every node x_i we need to compute the value $w(x_i+hf(x_i,a,b))$ for all $a\in A$ and $b\in B$ by interpolation using the values of the neighboring nodes. For a reconstruction based on linear interpolation in $\mathbb{R}^2$, we will write

$$w(x_i+hf(x_i,a,b)) = I(w(x_{i,1}), w(x_{i,2}), w(x_{i,3}))$$

where I is the interpolation operator and the values $x_{i,k}$, $k=1,2,3$ correspond to the vertexes of the triangle containing $x_i+hf(x_i,a,b)$. Obviously the choice of the nodes $x_{i,1}, x_{i,2}, x_{i,3}$ to be used in the interpolation depends on a and b via the dynamics f. We will denote by $x^*_{i,1}, x^*_{i,2}, x^*_{i,3}$ the triple corresponding to the optimal controls a^* and b^* (see Fig. 3.17). Now we can introduce the following important

Definition 3.9 *We define the* reachable set *at iteration n as the set*

$$\mathcal{R}^n = \{x_i\in G : x^*_{i,1}, x^*_{i,2}, x^*_{i,3} \textit{ are } \text{narrow band} \textit{ or } \text{accepted} \textit{ nodes}\}.$$

The above definition means that if the state of the system is in $\mathcal{R}^n$ then player P(ursuer) can drive the dynamics in the computed zone (that is he wins) whatever player E(vader) does. This allows, in some sense, to get rid of the second player E so that the problem is reduced to a 1-player game. Now let us give a brief sketch of the algorithm (see the FM-SL algorithm for details).

Algorithm

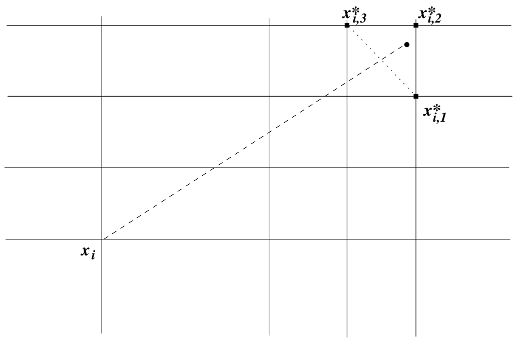


Figure 3.17: the triple $x^*_{i,1}, x^*_{i,2}, x^*_{i,3}$

1. The nodes belonging to the target $\mathcal{T}$ are located and labeled as *accepted* setting their values to $w = 0$. All other nodes are set to $w = 1$ and labeled as *far.*
2. The initial *narrow band* is defined as all the neighbors of the *accepted* nodes. Their value is "valid" only if they are in the *reachable set.*
3. The node in the *narrow band* with the minimal "valid" value is labeled as *accepted* and it is removed from the *narrow band.*
4. Neighbors not *accepted* of the last *accepted* node are computed and inserted in the *narrow band.* Their value is "valid" only if they are in the *reachable set.*
5. If the *narrow band* is not empty go to Step 3, else stop.

It should be noted that the algorithm can stop even if some nodes are not yet *accepted.*

Theorem 3.10 *Let $f(x,a,b)$ be a dynamics for a game and assume that*

$$\hat{f}(x,a) := f(x, a(b^*), b^*)$$

is a dynamics for which the 1-player FM-SL converges. Then, the FM-SL algorithm for differential games described above computes an approximate solution of (HJI).

Proof.
Let us consider the last *accepted* value w_{min} at the n-th iteration. We have to prove that this value is optimal for both players P and E. If the value w_{min} was assigned to the node x_i, it means that there exists at least a trajectory which drives the dynamics from x_i to the target in time $\widehat{T} = -\ln(1 - w_{min})$, that is P can win in a time $T \leq \widehat{T}$ even if E plays optimally. Therefore any value $w > w_{min}$ can not be optimal for P. On the other hand, it is impossible to get a value w lower than w_{min} at the same node x_i. This result follows by the proof of the FM-SL method. ■

Although it is hard to check if the assumptions of the Theorem 3.10 hold true, it makes a "bridge" between 1-player and 2-player games. Once we can count on a generalized Fast Marching method which can deal with general Hamiltonians, Theorem 3.10 gives a criterion to its applicability to the solution of differential games.

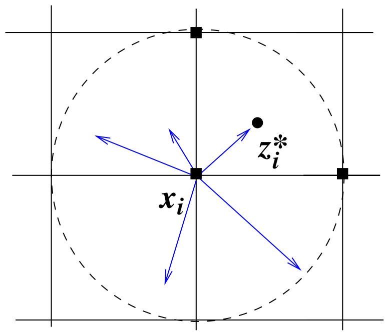
In the following we will only deal with unconstrained Tag-Chase game in reduced coordinates introduced in section 1.4.1. We assume that the two players P and E run in a infinite plane. The discrete problem will be set in a domain $Q \subset \mathbb{R}^2$ discretized by a uniform structured grid and we denote the space step by Δx.

Choice of discretization step h

The choice of discretization step h is fundamental to achieve the convergence of the numerical scheme. Unfortunately, we noted that the 2-player case is more complicated with respect to the 1-player case. Let us consider the dynamics $f(x,a) = c(x)a$ for the 1-player case. If $A = \partial B(0,1)$ we have $|f(x_i,a)| = c(x_i)$ for every a, *i.e.* $|f|$ is constant with respect to the choice of control. On the contrary, if we consider the dynamics for reduced Tag-Chase game $f(x,a,b) = V_P a - V_E b$ and we choose $A = B = \partial B(0,1)$ then $|f|$ is no longer constant with respect to the choice of controls a and b. This fact can be a difficulty whenever we want that the two following statements hold true at the same time:

1. the triple $x^*_{i,1}, x^*_{i,2}, x^*_{i,3}$ is in a neighborhood of the considered node (since the *narrow band* must be "narrow")
2. computation of $w(x_i)$ does *not* make use of the node x_i (in order to achieve convergence in a finite number of steps).

This is our strategy. First, we compute the optimal controls a^* and b^* choosing a time



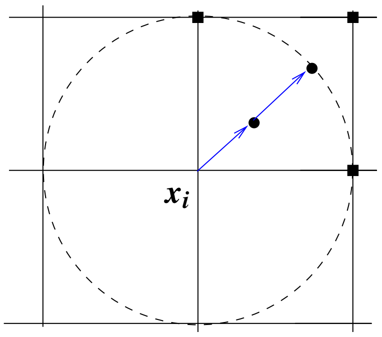


Figure 3.18: Choice of discretization step h

step h_i such that the point $z_i := x_i + h_i f(x_i,a,b)$ is in the four neighboring cells of the point x_i. More precisely, we choose $h_i = \Delta x / \max_{a,b} |f(x_i,a,b)|$. At this level it is possible that the point $z_i^* := x_i + h_i f(x_i,a^*,b^*)$ is very close to x_i so that the node x_i is in the triple $(x^*_{i,1}, x^*_{i,2}, x^*_{i,3})$ and this must be avoided (see Fig. 3.18-left).
Once (a^*,b^*) is computed, we define $h_i^* = \Delta x / |f(x_i,a^*,b^*)|$ and then we compute $w(x_i + h_i^* f(x_i,a^*,b^*))$ by interpolation (see Fig. 3.18-right). This choice allows to avoid to use the value at x_i in the linear interpolation and this is crucial to establish convergence in a finite number of steps as in the FM-SL.

method	Δx	L^∞ error	L^1 error	CPU time (sec)
FM-SL for games	0.16	0.0433	0.5416	51
SL iterative	0.16	0.0449	0.5407	276
FM-SL for games	0.08	0.0257	0.2918	180
SL iterative	0.08	0.0286	0.2927	1845

Table 3.5: error table for Test 1

3.3.2 Numerical experiments

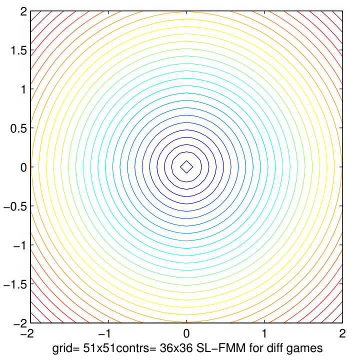
All numerical experiments were performed on a PC with a Pentium IV 3.06 GHz processor and 512 MB RAM. The CPU times refer to a Matlab (version 7) implementation.

Test 1: Tag-Chase game

$$f(x,y,a,b) = V_P a - V_E b\,, \qquad A = B = B(0,1)$$

$$Q = [-2,2]^2\,, \qquad \mathcal{T} = B(0,0.1)$$

The grid has 51×51 nodes (corresponding to $\Delta x = \Delta y = 0.08$) and the unit ball $B(0,1)$ is discretized in 36 controls (all placed at the boundary) for both players P and E. We have chosen $V_P = 2$, $V_E = 1$ in order to guarantee the capture of E.
It is easy to show that the optimal controls are $a^* = \frac{(-x,-y)}{|(-x,-y)|}$ and $b^* = -a^*$.
The exact solution is $T(x,y) = -\ln(1-v) = \sqrt{x^2+y^2} - 0.1$ (*i.e.* the distance from the target).
In Table 3.5 we compare our algorithm and the classical iterative SL scheme computing the L^∞ and L^1 error and CPU time. In Fig. 3.19 we show the level sets of the value



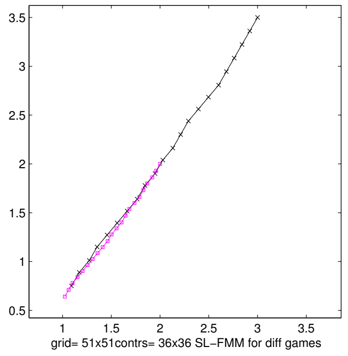


Figure 3.19: Test 1. Level sets of $T = -\ln(1-w)$ (left) and an optimal trajectory (right).

function T computed by our algorithm and an optimal trajectory in the real plane when P starts from the point $(3,3.5)$ and E starts from the point $(2,2)$.

Test 2: Tag-Chase game with constraints on the directions
This game has the same dynamics of the previous one. The only difference is that now

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 chose a grid of 51×51 nodes, (corresponding to $\Delta x = \Delta y = 0.08$) and 16 controls for both players P and E. The CPU time of our algorithm was 36 seconds and that of the classical iterative SL scheme was 662 seconds. In Fig. 3.20 we show the value function $T = -\ln(1-v)$ computed by our algorithm and its level sets. We can see some oscillations in the value function. This is due to the fact that in this case the propagation of the front is anisotropic and the FM method (as in the original version proposed by Sethian in [85]) can not deal with this kind of velocity field. In other words, assumptions of Theorem 3.10 does not hold true.

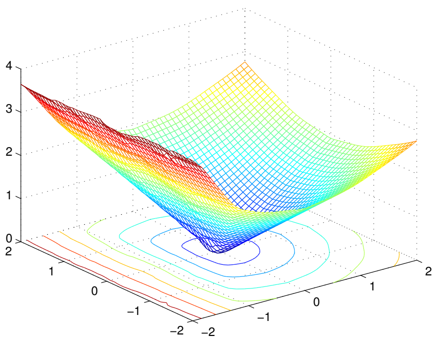
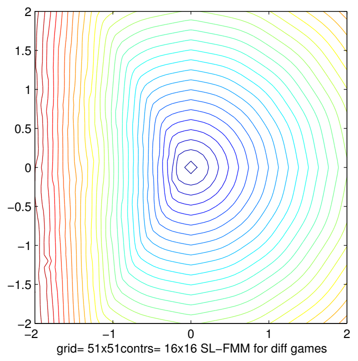



Figure 3.20: Test 2. Value function $T = -\ln(1-v)$ (left) and its level sets (right).

Fig. 3.21 shows the level sets of the solution computed by the classical iterative SL scheme. Any oscillation appears in this case.
Finally, Fig. 3.22 shows the optimal trajectories in the real plane when P starts from the point $(-3.5, 0)$ and E starts from the point $(-2, 1.5)$.

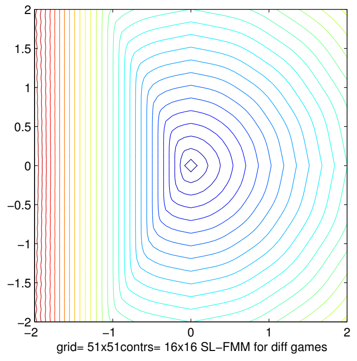


Figure 3.21: Test 2. Iterative SL scheme. Level sets of the value function.

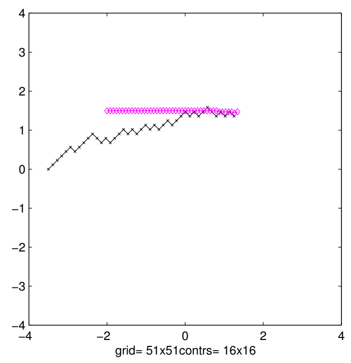


Figure 3.22: Test 2. Optimal trajectories.

3.4 A non-monotone Fast Marching method

In this section we propose a new Fast Marching method based on a finite difference discretization for the following time-dependent eikonal equation

$$\begin{cases} u_t(x,y,t) = c(x,y,t)|\nabla u(x,y,t)| & \mathbb{R}^2 \times (0,T) \\ u(x,y,0) = u^0(x,y) & \mathbb{R}^2. \end{cases} \tag{3.47}$$

The above equation describes the propagation of a front $\Gamma_0 = \partial\Omega_0$ moving along its normal direction with speed $c(x,y,t)$ (see section 1.3).
The new algorithm was presented for the first time in [25]. It is an extension of the FM method since it can deal with a time-dependent velocity *without any restrictions on its sign.*
As we know by previous chapter, the FM method is based on the following equation

$$c(x,y)|\nabla T(x,y)| = 1 \tag{3.48}$$

which is the stationary version of equation (3.47) whenever $c = c(x,y) > 0$. In FM method the computation of the solution proceeds in an increasing order accepting at each iteration the smallest value of the nodes in the current *narrow band*. The minimal value of the *narrow band* can be considered *exact* (within the discretization error) in the sense that it can not be improved in the following iterations. This fact allows us to deal easily with a *time-dependent* speed function using the current minimal value of the *narrow band* as time t and then to evaluate the speed function $c(x,y,t)$ during the computation. Using this basic idea, Vladimirsky [98] extended the FM method to a *signed* explicit time-depending function $c = c(x,y,t)$ and proved that in this case the evolution of front can be recovered as the level set of the time-independent function $T(x,y)$ which is the unique viscosity solution of the equation

$$c(x,y,T(x,y))|\nabla T(x,y)| = 1. \tag{3.49}$$

In order to treat the *non-monotone* case in which speed is allowed to have different signs in different regions and/or to change sign in time, we introduce some important modifications to the classical scheme.

1) We perform a slight modification of the function c. If there are two or more regions with different sign for c at the same time, we force the speed to be exactly zero on the boundaries of these regions so that the evolution of the front in each region can be considered completely separate. We will refer to the modified function as *numerical speed* and it will be denoted with $\hat{c}$.
2) Our new *narrow band* is the set of nodes which are going to be reached by the front *and* the nodes just reached by the front. This allows to deal with changes of sign of the velocity in time.

3.4.1 The FM scheme for unsigned velocity

In this section we give details for our FM algorithm for unsigned velocity. We describe the evolution of the front using an auxiliary function:

$$\theta(x,y,t) = \begin{cases} -1 & \text{if } u(x,y,t) \geq 0 \\ +1 & \text{otherwise.} \end{cases}$$

Notations and preliminary definitions

We consider a subset Q of $\mathbb{R}^2$ where we want to compute the approximate solution and we define a structured grid $Q_\Delta = \{(i,j) \in \mathbb{Z}^2 : (x_i, y_j) = (i\Delta, j\Delta) \in Q\}$ with space step Δ. Moreover, we denote by $0 < t_1 < ... < t_n < ... < t_N \leq T$ a non uniform grid on $[0,T]$, where t_n is the physical evolution time computed in each iteration of the FM method. We note that the partition of the time interval is not known *a priori*.
We introduce some definitions which will be useful in the sequel.

Definition 3.11 *We define* **neighborhood of the node** (i,j) *the set*

$$N_{FD}(i,j) = \{(l,m) \in Q_\Delta \text{ such that } |(l,m)-(i,j)| = 1\}.$$

Definition 3.12 *Given the speed* $c^n_{i,j} := c(x_i, y_j, t_n)$ *for all* i, j, n, *we define the* **numerical speed** *as*

$$\hat{c}^n_{i,j} = \begin{cases} 0 & \text{if there exists } (l,m) \in N_{FD}(i,j) \text{ such that} \\ & c^n_{i,j} c^n_{l,m} < 0 \text{ and } |c^n_{i,j}| \leq |c^n_{l,m}|, \\ c^n_{i,j} & \text{otherwise.} \end{cases}$$

Definition 3.13 *Given* $\theta^n_{ij} = \theta(x_i, y_j, t_n)$ *for all* i, j, n, *we define the* **fronts** F^n_+ *and* F^n_- *by*

$$F^n_\pm = N_{FD}(E) \backslash E, \text{ where } E = \{(i,j) \in Q_\Delta : \theta^n_{i,j} = \mp 1\} \quad \text{and} \quad F^n := F^n_+ \cup F^n_-.$$

Remark 3.14 . *We should point out that the main difference with respect to the classical FM algorithm is the presence of two* fronts*:* F_+ *and* F_- *(see Fig. 3.23).*
If the speed is positive (negative) the front propagates using only the information coming from F_+ *(*F_-*).*

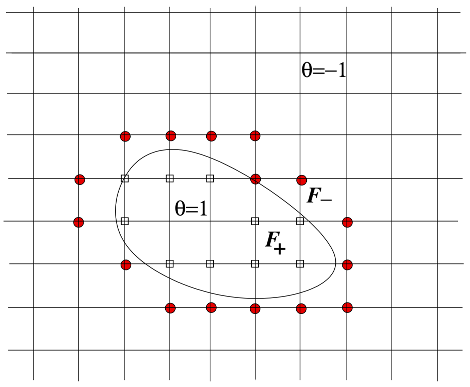


Figure 3.23: the fronts F_+ and F_-

We give here a brief description of the algorithm. We note that it is more complicated with respect the classical FM method, this is mainly due to the fact that it was developed in such a way it is possible to prove its convergence to the viscosity solution of equation (3.47) by a *new direct proof.* The interested reader is referred to [27] for the convergence of the scheme.

Description of the FM algorithm for unsigned velocity

We need a discrete function T_I to indicate the approximate physical time for the front propagation on the nodes $I = (i, j)$ of the fronts.

Initialization

1. $n = 1$
2. *Initialization of the matrix* θ^0
   $\theta^0_I = \begin{cases} 1 & \text{if } (x_i, y_j) \in \Omega_0 \\ -1 & \text{if } (x_i, y_j) \in Q \setminus \Omega_0 \end{cases}$
3. *Initialization of the time on the fronts*
   $T^0_I = 0$ for all $I \in F^0$

Main cycle

4. *Computation of* $\tilde{T}^{n-1}_I$.
   We define $\hat{T}^{n-1}_{\pm,J} = \begin{cases} T^{n-1}_J & \text{if } J \in F^{n-1}_\pm \\ +\infty & \text{elsewhere.} \end{cases}$
   Let $I \in F^{n-1}_\mp$, then
   (a) if $\pm\hat{c}^{n-1}_I \le 0$, $\tilde{T}^{n-1}_I = +\infty$,
   (b) if $\pm\hat{c}^{n-1}_I > 0$, then we compute $\tilde{T}^{n-1}_I$ as the greater solution of the following second order equation:

$$\sum_{k=1}^{2} \left(\max_{\pm} \left(0, \tilde{T}^{n-1}_I - \hat{T}^{n-1}_{+,I^{k,\pm}}\right)\right)^2 = \frac{(\Delta x)^2}{|\hat{c}^{n-1}_I|^2} \text{ if } I \in F^{n-1}_-$$

$$\sum_{k=1}^{2}\left(\max_{\pm}\left(0,\tilde{T}_I^{n-1}-\hat{T}_{-,I^{k,\pm}}^{n-1}\right)\right)^2=\frac{(\Delta x)^2}{|\hat{c}_I^{n-1}|^2} \text{ if } I\in F_+^{n-1}$$

where

$$I^{k,\pm}=\begin{cases}(i\pm 1,j) & \text{if} \quad k=1\\ (i,j\pm 1) & \text{if} \quad k=2\end{cases}$$

5. $\hat{t}_n=\min\left\{\tilde{T}_I^{n-1},\ I\in F^{n-1}\right\}$

6. $\tilde{t}_n=\begin{cases}\hat{t}_n & \text{if } \hat{t}_n<\infty\\ t_{n-1}+\delta & \text{if } \hat{t}_n=\infty\end{cases}$
   where δ is a small constant, see following *Remark* 3.15

7. $t_n=\max(t_{n-1},\tilde{t}_n)$

8. if $t_n=t_{n-1}+\delta$ go to 4 with $n:=n+1$

9. *Initialization of new accepted points*
   $NA_\pm^n=\{I\in F_\pm^{n-1},\ \tilde{T}_I^{n-1}=\tilde{t}_n\}$, $NA^n=NA_+^n\cup NA_-^n$

10. *Reinitialization of* θ^n
    $\theta_I^n=\begin{cases}-1 & \text{if} & I\in NA_+^n\\ 1 & \text{if} & I\in NA_-^n\\ \theta_I^{n-1} & \text{elsewhere}\end{cases}$

11. *Reinitialization of* T^n
    (a) If $I\in F^n\backslash N_{FD}(NA^n)$ then $T_I^n=T_I^{n-1}$
    (b) If $I\in NA^n$ then $T_I^n=t_n$
    (c) If $I\in(F^{n-1}\cap N_{FD}(NA^n))\backslash(NA^n)$, then $T_I^n=T_I^{n-1}$
    (d) If $I\in N_{FD}(NA^n)\backslash F^{n-1}$ then $T_I^n=t_n$

12. Go to 4 with $n:=n+1$

Remark 3.15 *The time computed in step 5 is the physical time, instead $\tilde{t}$ in step 6 is an artificial time that allows to advance in time in any case. For example, if at the iteration n we have $\hat{c}_I^{n-1}=0$ for all $I\in F_\pm^{n-1}$, then there will not be new accepted point. Therefore the algorithm will be blocked. The term δ have to be small enough (like $\frac{\Delta}{|\hat{c}^{n-1}|}$).*
Possible large time step could be computed when the speed is close to zero so that $\tilde{t}_n$ could be very large. For this reason in step 8 we bound the size of the time step by δ.

Remark 3.16 *In step 11 we change T_I^n only if a point of the neighborhood of I has been accepted.*

Boundary conditions on ∂Q
The management of the boundary conditions is quite simple. As in the classical FM method we can assign to the nodes of the boundary a value, like $+\infty$, such that these nodes will not contribute at the computations. Then, at the end of the algorithm, these nodes will be cut off.

3.4.2 Numerical experiments

We present some simulations which show the good behavior of this new schemes.

Test 1
In the first test we choose a unit circle as initial front which evolves with speed

$$c(x,y,t) = \begin{cases} 2 & x \geq 0\,, \quad t \leq 0.3 \\ -1 & x < 0\,, \quad t \leq 0.3 \\ -2 & x \geq 0\,, \quad t > 0.3 \\ 1 & x < 0\,, \quad t > 0.3 \end{cases}$$

The approximated solution is shown in Fig. 3.24.

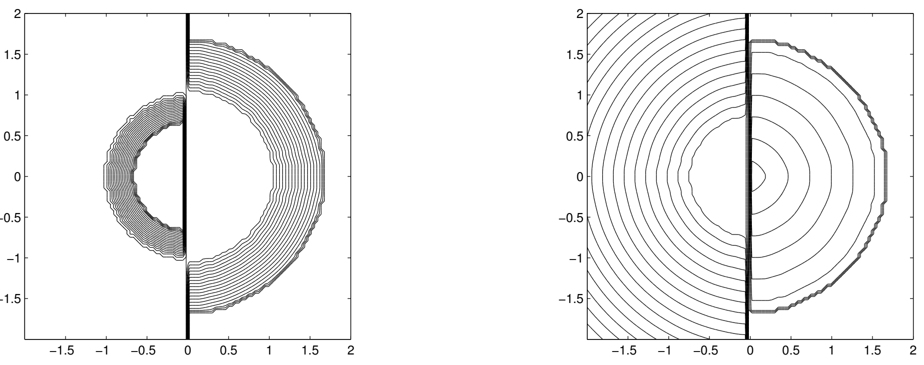


Figure 3.24: non monotone evolution of a circle, $t \leq 0.3$ (left) and $t > 0.3$ (right)

Test 2
The second test regards the rotation of a line. We consider the square $[-1,1]^2$ and we approximate the evolution of a line crossing the $\{x = 0\}$ axis with the velocity $c(x,y,t) = -x$. We set $\theta^0 = -1$ above the line and $\theta^0 = 1$ below. By easy computations,

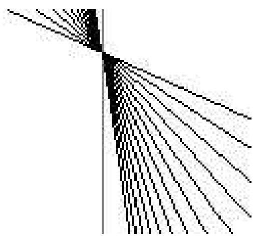


Figure 3.25: Rotation of a line

one expects that a straight line remains a straight line for all $t > 0$ and that it rotates around the axis $\{x = 0\}$ (where the velocity is zero). Indeed, let us consider a generic straight line $r = \{(x,y) : y = ax + b\}$ and a point $(x_0, y_0) \in r$. We denote by (x_1, y_1) the image of (x_0, y_0) after the time Δt. A first order expansion gives

$$(x_1, y_1) = (x_0, y_0) - \frac{\Delta t x_0}{\sqrt{1+a^2}}(-a, 1)$$

and then

$$y_1 = \left(\frac{a - \Delta t}{1 + a\Delta t}\right) x_1 + b.$$

Since (x_1, y_1) satisfies the equation of a line we deduce that a straight line always remains a straight line. Fig. 3.25 shows that our algorithm computes what one expects.
Moreover, one can observe that the velocity of rotation of the line decreases when it approaches the axis $\{x = 0\}$. This is due to the fact that the velocity decreases near this axis.

Test 3
We propose a third test regarding the evolution of a circle centered in the origin, with a speed $c(x, y, t) = 0.1t - x$. As shown in Fig. 3.26, the circle translates on the left and propagates in a self similar way. This test is run with $\Delta x = 2\pi/300$. The front is plotted every 0.5 physical time iterations with final time $T = 5$ and the solution is compared with that approximated by the classical finite difference scheme (FD) for equation (3.47).

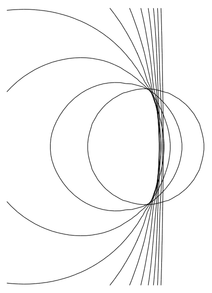


Figure 3.26: Evolution of a circle with FD

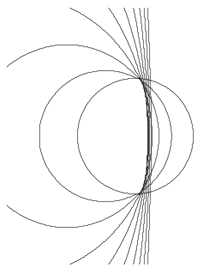


Figure 3.27: Evolution of a circle with the FM method

Test 4
Finally we propose a fourth test regarding the evolution of two circles. We set $\theta^0 = 1$ inside the circles and $\theta^0 = -1$ outside. We choose a velocity which changes sign in time, $c(x, y, t) = 1 - t$. This test is run with $\Delta x = 2\pi/300$. The front is plotted every 0.2 physical time with final time $T = 2.6$.
In Fig. 3.28 and 3.29 we show the result and we compare it with the approximation computed by the classical finite difference scheme for (3.47).

3.4.3 Application to dislocation dynamics

Our method can be extended to dislocations dynamic problems where the velocity of the front depends on the position of the front itself so that the velocity is time-dependent and it can also change sign. The Fast marching approach is motivated by the fact that the traditional iterative algorithm does not work very well since the representative function u

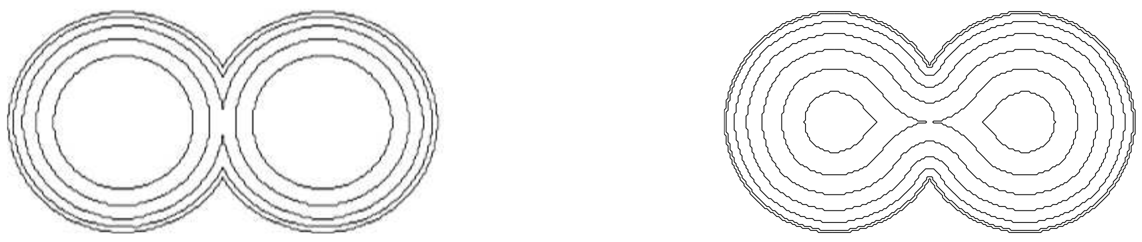


Figure 3.28: Increasing (left) and decreasing (right) evolution of two circles with FM

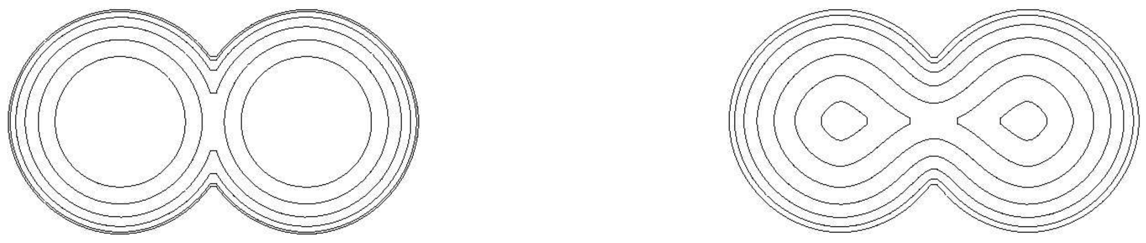


Figure 3.29: Increasing (left) and decreasing (right) evolution of two circles by classical FD scheme

becomes more and more flat around its 0-level set.
The problem consists in solving the following non-local Hamilton-Jacobi equation

$$\begin{cases} u_t(x,y,t) = c^0(x,y) \star [u](x,y,t)|\nabla u(x,y,t)| & \mathbb{R}^2 \times (0,T) \\ u(x,y,0) = u^0(x,y) & \mathbb{R}^2 \end{cases} \tag{3.50}$$

where the kernel c^0 is a given function and depends only on the space, $\star$ denotes the convolution in space and $[u]$ is defined by

$$[u] := \begin{cases} 1 & u \geq 0 \\ 0 & u < 0 \end{cases}$$

The 0-level set of the solution of (3.50) represents a dislocation line in a 2D plane. We refer to [1] for a physical presentation of the model for dislocation dynamics. In [25] there is a numerical test regarding the relaxation of a sinusoidal dislocation line.

Chapter 4

Numerical solution of the Perspective Shape-from-Shading problem

The Shape-from-Shading problem consists in reconstructing the three-dimensional *shape* of a scene from the brightness variation (*shading*) in a greylevel photograph of that scene (see Fig. 4.1).

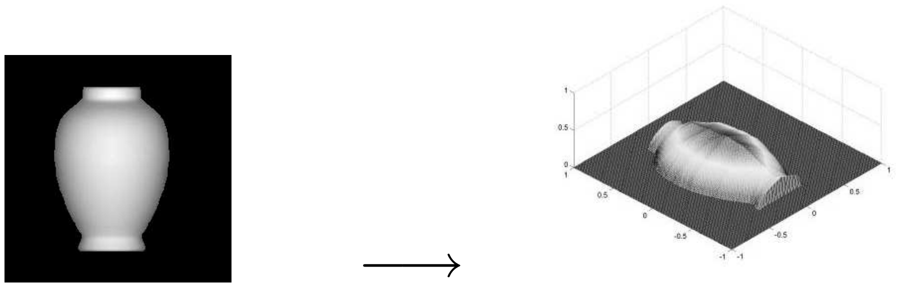


Figure 4.1: initial image (left) and reconstructed surface (right)

The study of the Shape-from-Shading problem started in the 70s (see [56, 57] and references therein) and since then a huge number of papers have appeared on this subject. More recently, the mathematical community was interested in Shape-from-Shading since its formulation is based on a first order partial differential equation of Hamilton-Jacobi type. Unfortunately, the numerous assumptions usually introduced in order to make the problem manageable highly reduce the relevance of the models.
The most common assumptions are (see [29]):
H_1 - The image reflects the light uniformly and then the albedo (ratio between energy reflected and energy captured) is constant.
H_2 - The material is Lambertian, *i.e.* the intensity of the reflected light is proportional to the scalar product between the direction of the light and the normal to the surface.
H_3 - The light source is unique and the rays of light which lighten the scene are parallel.
H_4 - Multiple reflections are negligible.

H_5 - The aberrations of the objective are negligible.
H_6 - The distance between the scene and the objective is much larger than that between the objective and the CCD sensor.
H_7 - The perspective deformations are negligible.
H_8 - The scene is completely visible by the camera, *i.e.* there are not hidden regions.

Some comments:
Assumptions H_1 and H_2 are often false for a common material.
Assumption H_3 means that we can describe the light direction by a unique and constant vector. Note that this is true only if the light source is very far from the scene (for example, if the scene is illuminated by the sun). Naturally, this assumption does not hold in case of flash illumination.
Assumption H_7 means that the camera is very far from the scene and it is obviously false in most cases.

This chapter is organized as follows. In Section 4.1 we recall the classical model for Shape-from-Shading problem (SFS in the sequel) which is derived under assumptions H_1-H_8. It is based on a very simple equation of eikonal type we already studied in previous chapters. This model is well known in the literature so that we do not make any study on it.

In Section 4.2 we get rid of assumption H_7 deriving a new Hamilton-Jacobi equation. The new model (PSFS_∞ in the sequel) takes into account the perspective deformation due to the finite distance between the camera and the scene. As we will see, such a deformation can be very relevant and the new model greatly overcomes SFS in some reconstructions. We perform a numerical approximation of the equation related to the new problem via a semi-Lagrangian discretization. In particular we focus our attention on the effect of boundary condition. We already presented these results in [40].

In Section 4.3 we abandon both H_3 and H_7 deriving a third Hamilton-Jacobi equation. The new model (PSFS_r in the sequel) takes into account the perspective deformation of the photograph like the previous model *and* the closeness of the light source so it can deal with photographs taken by means of a flash. Also for this model we perform a semi-Lagrangian discretization.
Note that the last two models were proposed very recently (after 2001) by several teams. See Prados and Faugeras [77] (see also Okatani and Deguchi [70]), Tankus, Sochen and Yeshurun [88], Courteille, Crouzil, Durou and Gurdjos [30]. Since those papers, the Shape-from-Shading was finally applied to some real problems like reconstruction of faces ([76, 78]), reconstruction of human organs ([90]) and the digitization of ancient books without scanners ([31, 32, 29]).

In Section 4.4 we will focus our attention on the concave/convex ambiguity in the PSFS_∞ and PSFS_r models. We will show some numerical examples and we will prove that even in these new models an ambiguity exists. This work was already presented in [36].
Finally, let us mention that an effort has been made to solve the (perspective) Shape-from-Shading problem by Fast Marching method. See for example [65, 99, 89, 79].

Let us briefly derive the model for Shape-from-Shading under general assumptions.
Let Ω be a bounded set of $\mathbb{R}^2$ and let $u(x, y) : \Omega \to \mathbb{R}$ be a surface which represents the three-dimensional image/surface we want to reconstruct. The partial differential

equation related to the Shape-from-Shading model can be derived by the "image irradiance equation"

$$R(\hat{n}(x,y)) = I(x,y) \tag{IE}$$

where I is the brightness function measured at all points (x, y) in the image, R is the reflectance function giving the value of the light reflection on the surface as a function of its orientation (*i.e.* of its normal) and $\hat{n}(x)$ is the unit normal to the surface at point $(x, y, u(x, y))$. If the surface is smooth we have

$$\hat{n}(x,y) = \frac{(-u_x(x,y), -u_y(x,y), 1)}{\sqrt{1+|\nabla u(x,y)|^2}}. \tag{4.1}$$

The brightness function I is the datum in the model since it is measured on each pixel of the image in terms of a gray level, for example from 0=black to 255=white or, after a rescaling, from 0 to 1. To construct a continuous model we will assume hereafter that I takes real values in the interval $[0, 1]$.
Clearly, equation (IE) can be written in different ways depending on which assumptions H_1-H_8 hold true.
It is important to note that, whatever the final equation is, in order to compute a solution we will have to impose some boundary conditions on $\partial\Omega$ and/or inside Ω. A natural choice is to consider Dirichlet type boundary conditions in order to take into account at least two different possibilities. The first corresponds to the assumption that the surface is standing on a flat background, *i.e.* we set

$$u(x,y) = 0 \qquad (x,y) \in \partial\Omega.$$

The second possibility occurs when the height of the surface on the boundary is known

$$u(x,y) = g(x,y) \qquad (x,y) \in \partial\Omega.$$

The above boundary conditions are widely used in the literature although they are often unrealistic since they assume a previous knowledge of the surface.
We will come back later on this problem.

4.1 The Shape-from-Shading problem

Under assumptions H_1-H_8, we have

$$R(\hat{n}(x,y)) = \omega \cdot \hat{n}(x,y)$$

where $\omega \in \mathbb{R}^3$ is a unit vector which indicates the direction of the light source. Then, equation (IE) can be written, using (4.1)

$$I(x)\sqrt{1+|\nabla u(x,y)|^2} + (\omega_1,\omega_2)\cdot\nabla u(x,y) - \omega_3 = 0\,, \quad (x,y)\in\Omega \tag{4.2}$$

which is a first order non-linear partial differential equation of Hamilton-Jacobi type.
If the light source is vertical, *i.e.* $\omega = (0, 0, 1)$, then equation (4.2) simplifies to the eikonal equation

$$\left(\sqrt{\frac{1}{I(x,y)^2}-1}\right)^{-1} |\nabla u| = 1\,, \quad (x,y)\in\Omega. \tag{4.3}$$

Points (x, y) where I is maximal (*i.e.* equal to 1) correspond to the particular situation when ω and $\hat{n}$ point in the same direction. These points are usually called "singular points" and, if they exist, equation 4.3 is said to be *degenerate* (see Remark 1.7). The notion of singular points is strictly related to that of concave/convex ambiguity which we briefly recall here.

4.1.1 Concave/convex ambiguity

The SFS problem is one of the most famous examples of ill-posed problem.

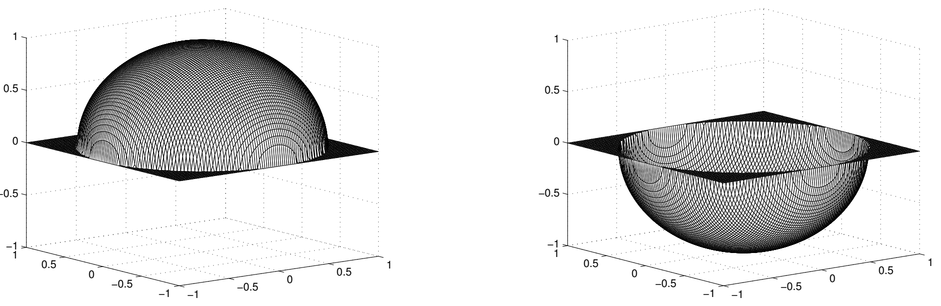


Figure 4.2: two different surfaces corresponding to the same brightness function I

Consider for example the two surfaces $z = +\sqrt{1 - x^2 - y^2}$ and $z = -\sqrt{1 - x^2 - y^2}$ (see Fig. 4.2). It is easy to see that they have the same brightness function I and verify the same boundary condition so that they are virtually indistinguishable by the model. As a consequence, even if we compute a viscosity solution of the equation, it is possible that the solution we obtained is different from the surface we expect. Note that this is an intrinsic problem and it can not be completely solved without a modification of the model.
In order to overcome this difficulty, the problem is usually solved by adding some informations such as the height at the singular points (see [68]). More recently, an attempt as been made to eliminate the need for *a priori* additional information by means of the characterization of the *maximal* solution (see [60, 19]). A result by Ishii and Ramaswamy [60] guarantees that if I is continuous and the number of singular points is finite, then a unique maximal solution exists. Following this approach, some algorithms to approximate the unique maximal solutions were proposed (see for example [82, 43] and references therein).

4.2 The PSFS$_\infty$ problem

In this section we get rid of assumption H$_7$ so we will take into account the perspective deformation due to the fact that the camera is close to the scene.

4.2.1 The model

Let us define the model adopting the same notations used in [31]. The point (X_0, Y_0) is the principal point of the image, d and d' are respectively the distance of the objective from the perspective plane (the CCD sensor) and the distance of the objective from the (flat) background, l and $l' = \frac{d'}{d} l$ are respectively the length of a generic segment in the perspective plane and the length of the real segment corresponding to it (see Figure 4.3 and [31] for more details). The representation of the surface in terms of the (X, Y) coordinates

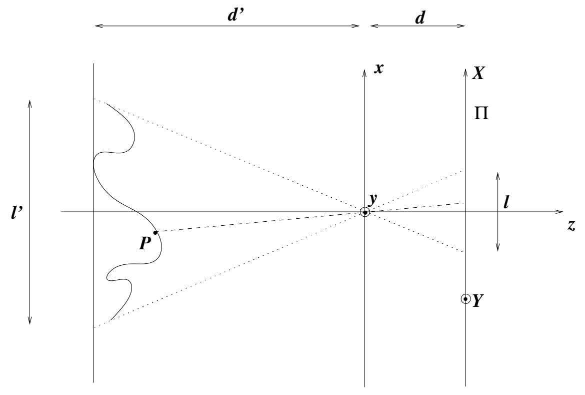


Figure 4.3: the PSFS$_\infty$ model

of the points in the perspective plane Π is given by three parametric equations

$$x = r(X, Y), \ y = s(X, Y), \ z = t(X, Y) \tag{4.4}$$

where (see [31])

$$\begin{cases} r(X, Y) = \frac{X - X_0}{d} t(X, Y) \\ s(X, Y) = \frac{Y - Y_0}{d} t(X, Y) \end{cases} . \tag{4.5}$$

Then the problem amounts to compute the third component t. This is the most difficult task since t is the solution of the following eikonal type equation

$$\left(\frac{d}{\bar{t}(X, Y)} \right)^2 |\nabla t(X, Y)|^2 = \frac{I_{max}^2}{I'(X, Y)^2} - 1 \quad \text{in } \Omega \tag{4.6}$$

where Ω is the internal region bounded by the silhouette of the object ($\partial\Omega$ will denote its boundary) which is embedded in a rectangular domain Q,

$$\bar{t}(X, Y) = t(X, Y) + (X - X_0, Y - Y_0) \cdot \nabla t(X, Y), \tag{4.7}$$

$$I'(X, Y) = \frac{I(X, Y)}{cos^4 \alpha(X, Y)}, \tag{4.8}$$

$$\cos^4(\alpha(X, Y)) = \frac{d^4}{((X - X_0)^2 + (Y - Y_0)^2 + d^2)^2}, \tag{4.9}$$

and I_{max} is a constant depending on parameters of the problem. The set $Q \setminus \Omega$ is the background.
Defining

$$f(X, Y) := \frac{1}{d^2} \left(\frac{I_{max}^2}{I'(X, Y)^2} - 1 \right) \tag{4.10}$$

we can write (4.6) as

$$|\nabla t(X,Y)| = \sqrt{f(X,Y)}\ \left|\bar{t}(X,Y)\right|. \tag{4.11}$$

We want to write (4.11) in a fixed point form and construct an approximation scheme for this equation. To this end it is important to note that $\bar{t}$ has a sign. In fact, the exterior normal to the original surface in the point P is given by

$$\hat{n}(P) = N(P)/|N(P)| \tag{4.12}$$

where

$$N(P) = (d\ t_X(X,Y), d\ t_Y(X,Y), -\bar{t}(X,Y)) \tag{4.13}$$

and since $-\bar{t}$ must be positive (according to the orientation of the z axis in Figure 4.3, $\bar{t}$ must be negative. This implies that (4.11) is in fact

$$|\nabla t(X,Y)| + \sqrt{f(X,Y)}(t(X,Y) + (X - X_0, Y - Y_0)\cdot \nabla t(X,Y)) = 0 \tag{4.14}$$

which can be written in short as

$$H((X,Y), t, \nabla t) = 0\,, \qquad \text{in } \Omega \tag{4.15}$$

where the Hamiltonian H represents the left-hand side of (4.14).
Finally, let us consider equation (4.14) complemented with the Dirichlet boundary condition

$$t = g(X,Y) \quad \text{ on } \partial\Omega, \quad \text{ where } -d' \le g \le 0. \tag{4.16}$$

4.2.2 Numerical approximation

The usual semi-Lagrangian scheme for (4.14)-(4.16) is

$$t(X,Y) = F[t](X,Y) \qquad \text{in } \Omega \tag{4.17}$$

where

$$F[t](X,Y) := \frac{1}{1+h} \inf_{a\in B(0,1)} \{t\,(b_h(X,Y,a))\} \qquad \text{in } \Omega\,, \tag{4.18}$$

$$b_h(X,Y,a) = (X,Y) + h\left(\frac{-a}{\sqrt{f}} - (X,Y)\right) \qquad (X,Y)\in\Omega,\ \ a\in B(0,1) \tag{4.19}$$

and $B(0,1)$ is the unit ball in $\mathbb{R}^2$.
Let us examine the properties of the F operator in order to guarantee convergence for the fixed point iteration. First, let us introduce the following space:

$$W = \{w : \Omega \to \mathbb{R},\ \text{such that } w|_{\partial\Omega} = g\} \tag{4.20}$$

Note that W is a space of functions satisfying the Dirichlet boundary condition $w = g$ on $\partial\Omega$.

Lemma 4.1 *Under the above assumptions, the following properties hold true:*
a) F is a contraction mapping in $L^\infty(\Omega)$;
b) F is monotone, i.e. $s \le t$ implies $F[s] \le F[t]$;
c) Let $V = \{w \in W : -d' \le w(X,Y) \le 0\}$, then $F : V \to V$;

Proof.
a) Let us take two functions, t and s. For every $\xi = (X, Y)$, we have

$$F[t](\xi) - F[s](\xi) \leq$$
$$\frac{1}{1+h}\left[\inf_{a\in B(0,1)}\left\{t\left(\xi + h\left(\frac{-a}{\sqrt{f}} - \xi\right)\right)\right\} - \inf_{a\in B(0,1)}\left\{s\left(\xi + h\left(\frac{-a}{\sqrt{f}} - \xi\right)\right)\right\}\right] \leq$$
$$\frac{1}{1+h}\left[t\left(\xi + h\left(\frac{-a^*}{\sqrt{f}} - \xi\right)\right) - s\left(\xi + h\left(\frac{-a^*}{\sqrt{f}} - \xi\right)\right)\right] \leq$$
$$\frac{1}{1+h}\|t - s\|_\infty$$

where a^* is the direction where the infimum for s is attained. Replacing the role of t and s one obtains the reverse inequality and proves

$$\|F[t] - F[s]\|_\infty \leq \frac{1}{1+h}\|t - s\|_\infty. \tag{4.21}$$

b) The monotonicity with respect to t is a direct consequence of the definition of F since the coefficient in front of the *inf* is strictly positive.

c) It is a direct consequence of *b)* and of

$$F[0] = 0, \qquad F[-d'] = \frac{-d'}{1+h} > -d'. \tag{4.22}$$

■

Now let us examine the algorithm. Lemma 4.1 assures that, starting from any initial guess t^0 which satisfies the boundary conditions, the fixed point iteration

$$t^{(n+1)} = F[t^{(n)}] \tag{4.23}$$

converges to the unique solution t^* (fixed point).
We note that a direct consequence of the above Lemma is that one can obtain a monotone increasing convergence just starting from any function below the final solution, *e.g.* choosing $t^{(0)} \equiv -d'$ in the internal nodes and imposing the Dirichlet boundary condition $t^{(0)} = g(X, Y)$ on $\partial\Omega$. Moreover, the property b) guarantees that $\bar{t}^{(n)}$ (defined in (4.7)) is negative for all $(X, Y) \in \Omega$ *at every iteration*, so the equation associated to the problem is always (4.14).

4.2.3 Creation of a virtual perspective image and computation

In this section we will describe how we construct a virtual image to test the PSFS$_\infty$ algorithm described in the previous section. We developed this procedure to produce some synthetic images that will be used as benchmarks for our algorithm.
The starting point is the choice of a surface $u = u(x, y)$. Note that all figures refer to the example of a tent $(u(x, y) = 1 - |x|, \ \ (x, y) \in [-1, 1]^2)$ but the procedure is valid for every

(graph) surface.

Pre-processing
Given $u = u(x, y)$, we compute (analitically or numerically) the unit normal vector $\hat{n}(P)$ at every point and we compute the light function

$$I(x, y) = \omega \cdot \hat{n}(P)$$

where ω is the direction of the source light. In our tests we fixed $\omega = (0, 0, 1)$ so we have

$$I(x, y) = \frac{1}{\sqrt{1 + u_x^2 + u_y^2}} .$$

We consider a rectangular $n \times n$ grid $G' := \{(x_{i'}, y_{j'})\}$ where $i' = 1, \ldots, n$ and $j' = 1, \ldots, n$

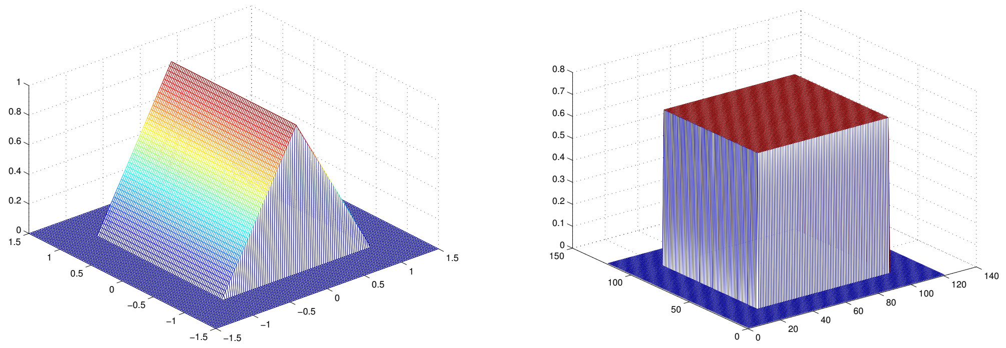


Figure 4.4: Initial surface $u(x, y)$ (left) and computed light function $I(i', j')$ (right)

on which we compute two matrices $I(i', j')$ and $u(i', j')$, $\quad i', j' = 1, \ldots, n$ (see figure 4.4).

Let us "take a photograph" of the surface. Every discrete coordinates (i', j') correspond to a point $(x, y, u(x, y))$ belonging to the surface and every point $(x, y, u(x, y))$ is associated to a pair (X, Y) on the perspective plane by the transformation

$$\begin{cases} \frac{X - X_0}{d} = \frac{x}{u} \\ \\ \frac{Y - Y_0}{d} = \frac{y}{u} \end{cases} \tag{4.24}$$

Varying i', j' in $\{1, \ldots, n\}$, we obtain the set

$$\Big\{(X_{i'j'}, Y_{i'j'})\Big\}_{i', j' = 1, \ldots, n} \tag{4.25}$$

which is the (discrete) domain of the perpective image (see Figure 4.5).

Remark 4.2 *The transformation (4.24) is not injective. In particular, a point (X, Y) associated to a point $(x, y, u(x, y))$ belonging to the background can coincide with a point $(\tilde{X}, \tilde{Y})$ associated to a point $(\tilde{x}, \tilde{y}, \tilde{u}(\tilde{x}, \tilde{y}))$ belonging to the surface. This the case of the tent as shown in Figure 4.6. This is due to the fact that distance camera-object is finite which implies the existence of some areas in full shade (see Figure 4.7).*

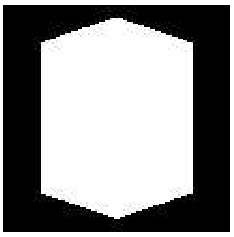


Figure 4.5: In white: domain of the perpective image on XY plane. In black: background

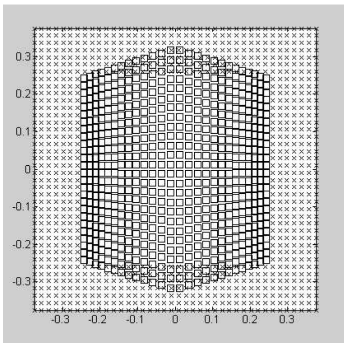


Figure 4.6: Discrete domain of the perspective image. CROSS: point coming from the background, SQUARE: point coming from the object (there are two overlapping regions)

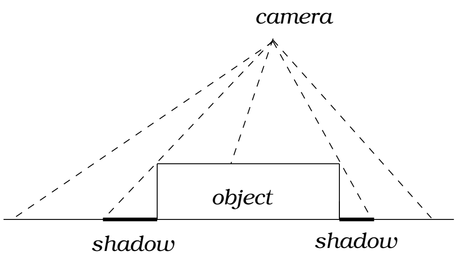


Figure 4.7: areas in full shade

We want to stress that, as a result of transformation (4.24), the set (4.25) loses the initial ordering in the sense that two adjacent points (X_1, Y_1), (X_2, Y_2) in the perspective XY-plane are not necessarily coming from two adjacent points $(x_1, y_1, u(x_1, y_1))$, $(x_2, y_2, u(x_2, y_2))$. Then, the resulting ordering does not coincide with that of a structured grid.
In order to overcome this difficulty and to have an easy implementation of the algorithm, we discretized again the perspective image using a *new* rectangular $n \times n$ grid G indexed by $i = 1 \ldots, n$ and $j = 1 \ldots, n$. This provides an easy correspondence between (i', j') and (x, y) and between (i, j) and (X, Y).
Then, we find the relationship between (i, j) and (i', j'). Roughly speaking, we associate a pair (X, Y) to every node (i, j) and then we find the four closest points to (X, Y) in

the set (4.25). These four points are associated to four nodes in the grid G' on which the original light function I and the original surface u are defined. Given this correlation, we easily obtain the matrix representation $I(i,j)$ of the function $I(X,Y)$, that is the perspective image in the XY-plane simply applying an interpolation rule (typically linear interpolation). Moreover, we can compute, in the same way, the function $t(X,Y)$, that is the solution to our problem.
Note that in the case of the tent the computation of $I(X,Y)$ is trivial because $I(x,y)$ is constant. See Figure 4.8.

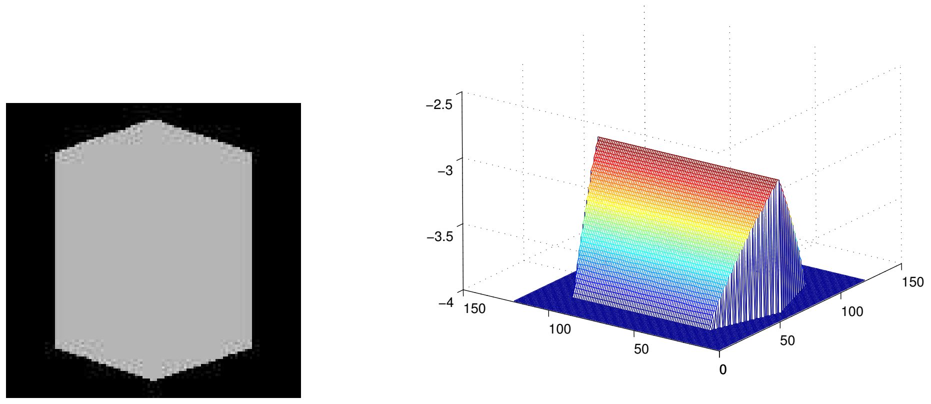


Figure 4.8: gray level of the photograph (left) and the solution $t(X,Y)$ of equation (right)

Boundary conditions and Computation
The assignment of boundary conditions is very easy and consists in computation of $t(X,Y)$ (as showed above) only in the required nodes. As mentioned before, we choose $t^{(0)} = -d'$ in internal nodes (that is inside Ω). Moreover, it is recommended initializing external nodes (that is outside Ω, where no computation is needed) with the value $t^{(0)} = 0$ which is the greatest value t can attain. This choice avoid the risk that the scheme uses some values coming from the background, because the evaluation of the infimum will automatically reject them.
Now we are ready to compute the solution using the scheme (4.17) and the fixed point technique.

Post-processing
Once we computed the approximate solution $t(X,Y)$, we can easily compute $r(X,Y)$ and $s(X,Y)$ as in (4.5) and then we can draw the surface

$$\{(r(X,Y), s(X,Y), t(X,Y)), \quad (X,Y) \in \Omega\}$$

given in parametric form to be compared with the exact solution in terms of u and I (see Figure 4.9).
Finally we note that, due to Remark 4.2, the computed surface with its background can have some "holes" in its domain in correspondence with areas in full shade. In other words, in the process we may lose some informations about the initial surface (see Figure 4.9 and 4.10). In Figure 4.10-right black areas correspond to points not visible by the objective.

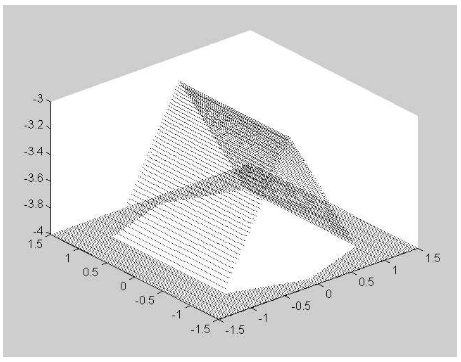


Figure 4.9: approximate solution (r, s, t) of the problem given in parametric form

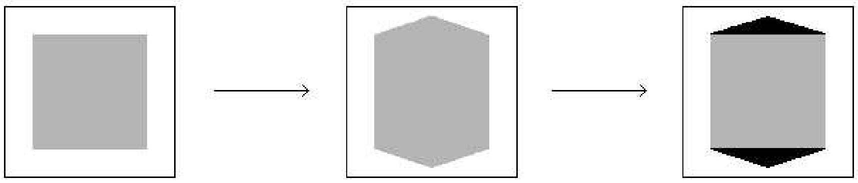


Figure 4.10: Initial domain of the object (left), perspective domain of the photograph (center) and reconstructed domain of the object (right)

4.2.4 Other boundary conditions and their effects

Beside the difficulty related to the concave/convex ambiguity which is behind the non uniqueness of viscosity solutions there is another difficulty which arises in the PDE approach. It is well known that in this approach one has to complement the equation with some boundary conditions to select a unique solution and to run the algorithm. This is a limitation with respect to minimization algorithms where such boundary conditions are not needed and the search for the solution is done via a gradient method or a line search algorithm. Naturally the solution computed by those algorithms will, in general, be different from the exact solution. However, in practical applications boundary conditions on the surface are seldom known, so it useful to analyse in more detail the effect of different types of boundary conditions on the solution in order to define a minimal set of conditions which will allow to compute the exact solution.

In this section, we will briefly analyse the effect of Dirichlet, Neumann and state constraints boundary conditions on subsets of the boundary. Let us note first that boundary conditions should be imposed in a weak sense. The typical condition which defines a viscosity subsolution u for (4.15) requires that for any test function $\varphi \in C^1(\overline{\Omega})$ and $x \in \partial\Omega$ local maximum point for $u - \varphi$

$$\min\{H(x, u(x), D\varphi(x)), B(x, u, D\varphi(x))\} \leq 0 \tag{4.26}$$

where the function B is the operator describing the boundary conditions, *f.e.* $B(x, u, Du) = u - g$ for the Dirichlet condition. Similarly, the boundary condition for supersolutions requires that for any test function $\varphi \in C^1(\overline{\Omega})$ and $x \in \partial\Omega$ local minimum point for $u - \varphi$

$$\max\{H(x, u(x), D\varphi(x)), B(x, u, D\varphi(x))\} \geq 0. \tag{4.27}$$

The effect of the Dirichlet condition is to impose a value on u according to the above conditions, in particular the value $u(x) = g(x)$ is set at every point where $H(x, u(x), D\varphi(x)) \geq 0$ (for subsolutions) and $H(x, u(x), D\varphi(x)) \leq 0$ (for supersolutions).

Neumann boundary conditions correspond to the operator $B(x, u, Du) = \partial u/\partial n(x) - m(x)$ where $n(\cdot)$ represents the outward normal to the domain Ω. A typical use of it is when we know (or we presume) that the level curves of the surface are orthogonal to the boundary $\partial\Omega$ or to a subset of it where we simply choose $m(x) = 0$.

The state constraints boundary condition is different from the above conditions since we do not impose neither a value for u nor a value for its normal derivative $\partial u/\partial n(x)$ (cfr. [18]). In this respect it has been interpreted as a "no boundary condition" choice although this interpretation is rather sloppy. In fact, a real function u bounded and uniformly continuous is said to be a *state constraints* viscosity solution if and only if it is a subsolution (in the viscosity sense) in Ω and a supersolution in $\overline{\Omega}$ (*i.e.* up to the boundary). It can be also stated as a Dirichlet boundary condition simply setting

$$g = C_g = constant \quad \text{provided} \quad C_g > \max_{x \in \Omega} u(x)$$

(note that in our problem, by Lemma 4.1, an easy choice satisfying the above condition is $C_g = 0$). By this choice (4.26) is trivially satisfied, whereas (4.27) requires (strictly)

$$H(x, u(x), D\varphi(x)) \geq 0. \tag{4.28}$$

In our algorithm, the fixed point operator F looks for a minimum on neighbouring points $b_h(X, Y, a)$ so we can obtain the same result if we define the solution outside Ω to be $t = C \geq 0$ so that searching for a minimum all directions $a \in B(0, 1)$ will be excluded. The effect of state constraints boundary condition is to look for the minimum inside Ω. It is interesting to note that if we adopt state constraints boundary conditions and we start from $t^{(0)} \equiv C = constant$, the scheme will produce the sequence $t^{(n)} \equiv (1 + h)^{-n}C$ which converges to 0 everywhere in Ω. Clearly, $t = 0$ is not a meaningful solution. However, if we fix the value at even a single point $x^* \in \Omega$ the solution will have a minimum at x^* and we will have $u(x) > u(x^*)$ for every $x \in \Omega$. So the effect of the state constraints boundary conditions is to produce solutions which increase when x gets close to the boundary $\partial\Omega$.

4.2.5 Numerical experiments

In this section we present some numerical experiments on synthetic images, on a real non perspective image with an artificial perspective deformation and finally on a real photograph with visible perspective deformation.

Synthetic images
For tests on synthetic images we have chosen in all cases the following parameters:

$$X_0 = 0, \quad Y_0 = 0, \quad d = 1, \quad d' = 4, \quad l = 0.75, \quad l' = 3.$$

The computational procedure follows the steps described in the previous sections. Both G and G' are 121×121 grids and the number of controls for the discretization of the unit

ball $B(0,1)$ is 16 (all placed on the boundary $\partial B(0,1)$). The iterative algorithm stops when $||t^{(n+1)} - t^{(n)}||_\infty \leq \varepsilon$.
In each subsection we present a) the original surface, b) its light function (in the xy-plane), c) the light function in the perspective XY-plane (that is the photograph) and d) the reconstructed surface. Finally, we compute the error estimate in L^∞-norm comparing the solution t (computed during the preprocessing step) with the approximate solution of the equation (4.14). Note that what we name "solution t" was actually computed by an interpolation so this is not a comparison with the real exact solution t. Nevertheless, this is a very reasonable way to calculate the accuracy of the algorithm because computation starts from the function $I(i,j)$ which was computed by the same interpolation too.

Remark: we choose a variable step discretization h in (4.18) depending on X, Y and a in such a way that

$$h(X,Y,a)\left(\frac{-a}{\sqrt{f}} - (X,Y)\right) = \Delta x \qquad \text{for all } X, Y, a$$

where Δx is the space step discretization. This trick reduces the number of iterations needed to reach convergence. Finally, we define $h_{min} := \min\limits_{X,Y,a} h(X,Y,a)$.

Tent (I discontinuous)
This test is a slight modification of the tent used in Section 4.2.3. The main difference here is that I is discontinuous (whereas in the previous example I was constant). The solution t is non regular but the boundary conditions are very simple, 0 on every side of the square. One can see that the algorithm is accurate around the kinks and that the error in the max norm is about $10\Delta x$ (see Table 4.1).

$$u(x,y) = \begin{cases} 2(1-|y|) & x \in [-1,1], \quad y \in [-1, -\frac{1}{2}|x| - \frac{1}{2}] \\ 2(1-|y|) & x \in [-1,1], \quad y \in [\frac{1}{2}|x| + \frac{1}{2}, 1]) \\ 1-|x| & \text{otherwise} \end{cases}$$

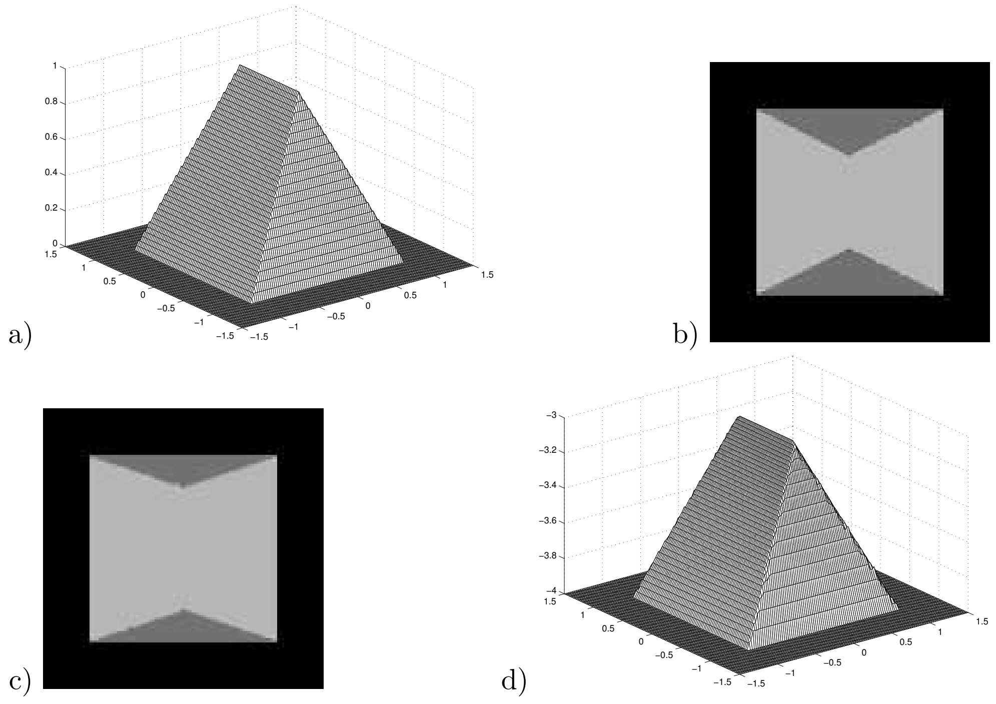


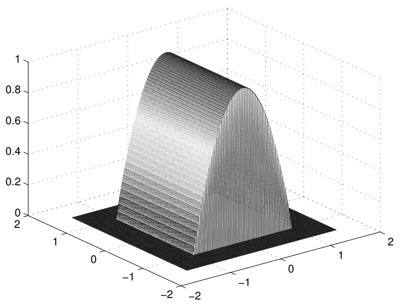
Figure 4.11: a) the original surface, b) its light function in the xy-plane, c) light function in the perspective XY-plane, d) the approximate surface

number of iterations	h_{min}	Δx	ε	L^∞ error
52	0.0046	0.00625	10^{-7}	0.064

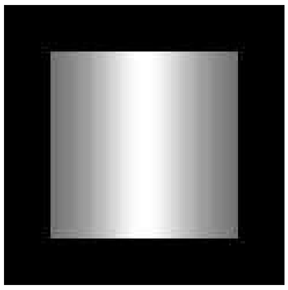
Table 4.1: L^∞ error

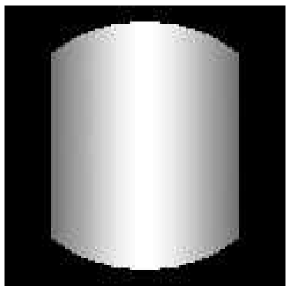
Very regular surface (I continuous)
This test has been created to check the accuracy on a regular surface which has a line of singular points (where $I(X,Y) = 1$). This line is truncated in the real computation and substituted by the value 0.9999. The boundary conditions are not homogeneous: they are 0 on the left- and right-hand sides of the square, and $b(x,y) = 1-x^2$ on the top and bottom sides of the square. The algorithm stops after 1613 iteration with a $h_{min} = 8.8 \cdot 10^{-5}$. It is interesting to note that also in this case the error is $10\Delta x$.

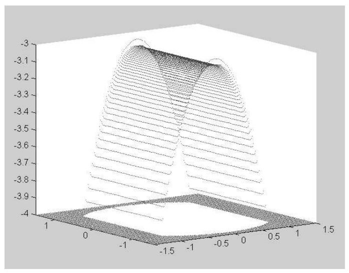
$$u(x,y) = 1 - x^2, \qquad (x,y) \in [-1,1]^2$$



a)


b)


c)



d)

Figure 4.12: a) the original surface, b) its light function in the xy-plane, c) light function in the perspective XY-plane, d) the approximate surface

number of iterations	h_{min}	Δx	ε	L^∞ error
1613	$8.8 \cdot 10^{-5}$	0.00625	10^{-7}	0.0418

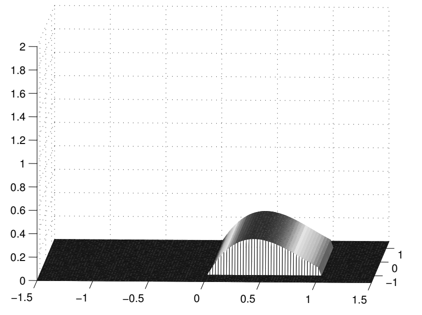
Table 4.2: L^∞ error

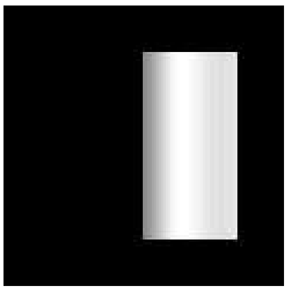
Synthetic book image (I continuous)
The last test is a tentative to reconstruct a synthetic surface as close as possible to the shape of a single page of a book. Again I and u are regular. The boundary conditions are not homogeneous: 0 on the left-hand side of the square, a positive constant on the right-hand side and a polynomial function $g(X, Y)$ on the top and bottom sides of the square. Note that $h_{min} = 1.428 \cdot 10^{-4}$ and that the approximate solution after 260 iterations has an error of the order $10\Delta x$.

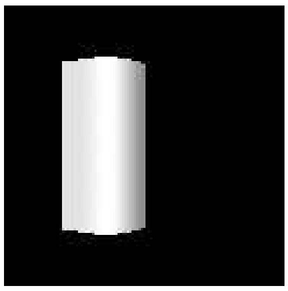
$$u(x, y) = \frac{1}{120}\Big(b\,(225|x|)^3 + c\,(225|x|)^2 + d\;225|x|\Big)$$

where

$$b = 1.09 \cdot 10^{-5}, \quad c = -6.21 \cdot 10^{-3}, \quad d = 0.883$$

a) 

b) 

c) 

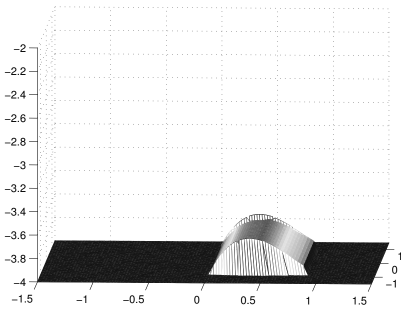
d) 

Figure 4.13: a) the original surface, b) its light function in the xy-plane, c) light function in the perspective XY-plane, d) the approximate surface

number of iterations	h_{min}	Δx	ε	L^∞ error
260	$1.428 \cdot 10^{-4}$	0.00625	10^{-7}	0.0536

Table 4.3: L^∞ error

Real image with synthetic perspective

In this test we used a *real* photograph of a vase with a negligible perspective deformation, so we modified it by an artificial perspective deformation as in the previous tests.
For this image the parameters values are $d = 1$, $l' = 2$, $d' = 4$, $l = \frac{d}{d'}l' = 0.5$, $\Delta x = 0.0042$ and $I_{max} = 1$. Figure 4.14 shows the photograph and the reconstructed surface computed

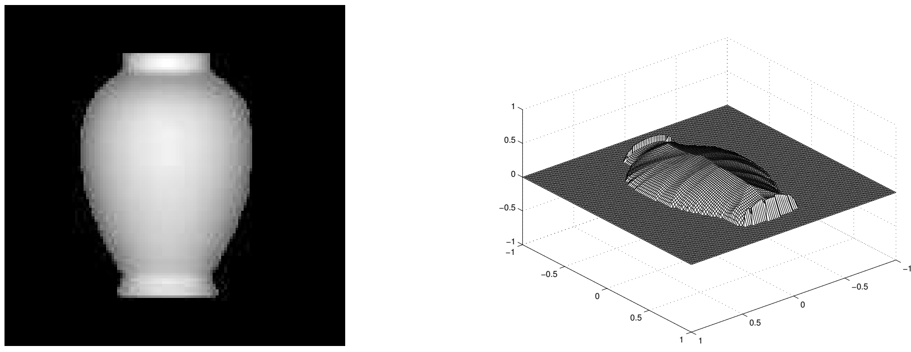


Figure 4.14: photograph, 121 x 121 pixels (left) and reconstructed surface with Dirichlet boundary condition (right)

using Dirichlet boundary conditions $u = g$, where g is the real height of the vase on $\partial\Omega$.
It is easy to verify that the numerical solution does not match the boundary conditions on the top and on the bottom of the vase (the error is high particularly on the top).
Numerical tests show that in this simple case the values of $t(X, Y)$ inside the domain depend only on its values on the left and right boundaries of the image. Therefore we can substitute Dirichlet boundary condition by state constraints on the top and bottom part of the boundary without any change in the solution. Figure 4.15-left shows the approximate solution in this case. As expected, the approximate solution is very good even if we do not impose Dirichlet boundary condition on the whole boundary. We want to emphasize that the knowledge of the exact solution t on $\partial\Omega$ can be considered in general a completely *non*-realistic assumption (because the height of the surface is exactly what we want to know) but in this simple case we are able to compute the exact solution also under realistic assumptions.
Finally, we computed the solution with Dirichlet boundary condition on the right and left side and Neumann boundary condition elsewhere. This is "realistic" boundary condition because we can assume that vase is flat on the top and on the bottom. Figure 4.15-right shows the result.
The average error (with respect to the exact solution) of the three tests is 0.043. In all cases, the iterative procedure converges in 65 iterations, with $\varepsilon = 10^{-7}$.

Real image

In this test we used a real photograph where the effect of perspective is visible.
The surface is a sheet of paper with the shape of a roof tile. For this image the parameter values are: $l = 6.91mm$, $d = 5.8mm$, $l' = 200mm$, $d' = \frac{l'}{l}d = 167.87mm$, $\Delta x = 0.05mm$.
We note that we performed the light correction (4.8) in the preprocessing step, so we can assume $I_{max} = 1$ during computation.

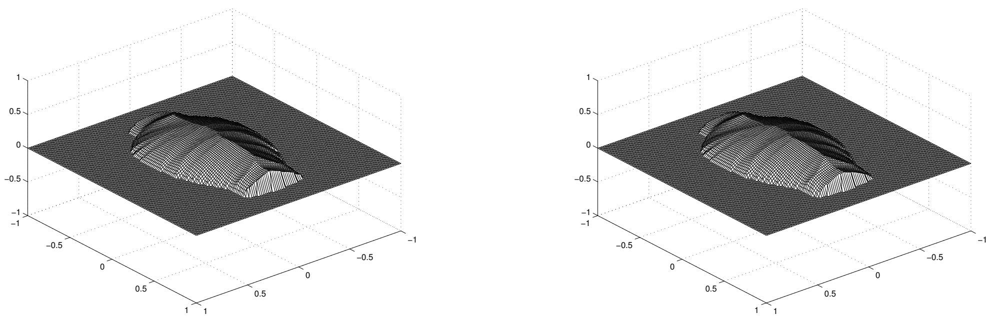


Figure 4.15: reconstructed surface with Dirichlet and state constraints boundary condition (left) and reconstructed surface with Dirichlet and Neumann boundary condition (right)

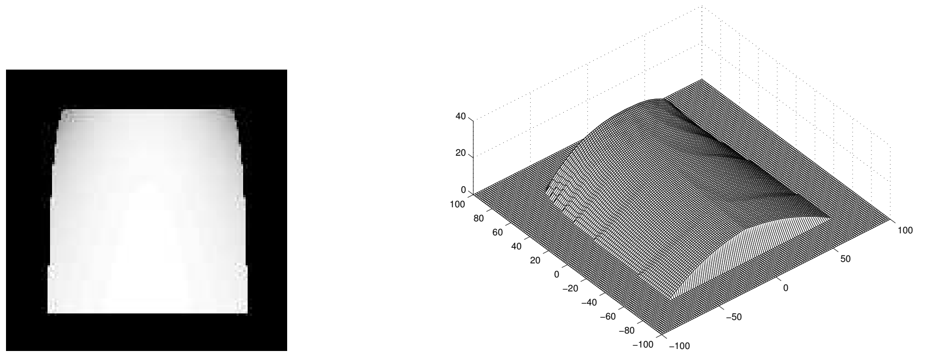


Figure 4.16: photographs, 128 x 128 pixels (left) and reconstructed surface with Dirichlet and state constraints boundary condition (right)

Figure 4.16 shows the photograph (128×128 pixels) and the surface reconstructed using Dirichlet boundary condition only on the left and right sides of the boundary and state constraints elsewhere (top and bottom sides). We can see that the solution is quite good considering the fact that light source (flash camera) is not far from the object and that direction of light source is not perfectly vertical as the mathematical model would have required.
We also tried to reconstruct the surface with two more practical boundary conditions. In the first case, we fixed a Dirichlet condition t^0 only on a vertical line in the center of the image (column 64) and then we turned over the computed surface with respect to the value t^0 (see Figure 4.17-left). Note that the solution is not very sensitive with respect to value t^0, so a rough knowledge of the behavior of the surface can be sufficient. We can see that the solution is quite good. We have a large maximum norm error on the boundary (17.7mm, 41% of the maximum height of the tile), but not inside. In fact, assuming that the reconstructed surface in Figure 4.16-right is the exact solution, the average error on all nodes for Figure 4.17-left is about 1.2mm.
In the second case (see Figure 4.17-right), we fixed a Dirichlet condition t^0 only on the point $(64, 64)$ (at the center of the image) and then we turned over the computed surface

as before. Note that in this case the solution has a shape very different from the expected solution since it has a global maximum at the central point $(64, 64)$.

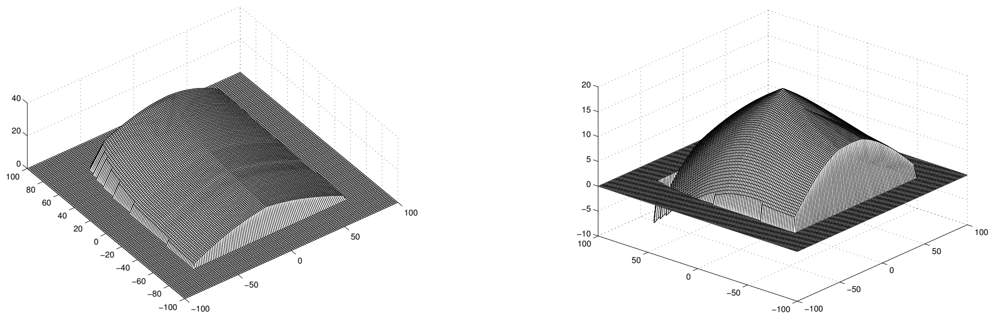


Figure 4.17: reconstructed surface with Dirichlet boundary condition on the center line (left) and reconstructed surface with Dirichlet boundary condition on one point (right, different scale)

In these three tests the iterative procedure converges respectively in 167, 185 and 190 iterations, with $\varepsilon = 10^{-6}$.

4.3 The PSFS$_r$ problem

In this section we get rid of assumptions H$_7$ and H$_3$ so we take into account the perspective deformation and the closeness of the light source which is now located at the optical center.

4.3.1 The model

Let us define the model adopting the same notations used in [78]. Let Ω be an open set of $\mathbb{R}^n$. Ω represents the image domain. We represent the scene by a surface $\mathcal{S}$ which can be explicitly parametrized by using the function $S = (S_1, S_2, S_3) : \Omega \to \mathbb{R}^3$, so that

$$\mathcal{S} = \{S(x, y) : (x, y) \in \overline{\Omega}\}.$$

We denote by O the optical center, by $f > 0$ the focal length and by M a generic point on the surface. There exists a function $u : \Omega \to \mathbb{R}$ such that (see Fig. 4.18)

$$M = M(x, y) = u(x, y)m' \tag{4.29}$$

where

$$m' = \frac{f}{\sqrt{x^2 + y^2 + f^2}}\, m \quad \text{and} \quad m = (x, y, -f). \tag{4.30}$$

We also denote by $r(x, y)$ the distance between the light source and the point $M(x, y)$ on the surface. We have

$$u(x, y) = \frac{r(x, y)}{f}.$$

By (4.29) and (4.30), we get

$$S(x, y) = u(x, y)m' = \frac{f u(x, y)}{\sqrt{x^2 + y^2 + f^2}}(x, y, -f), \quad (x, y) \in \Omega.$$

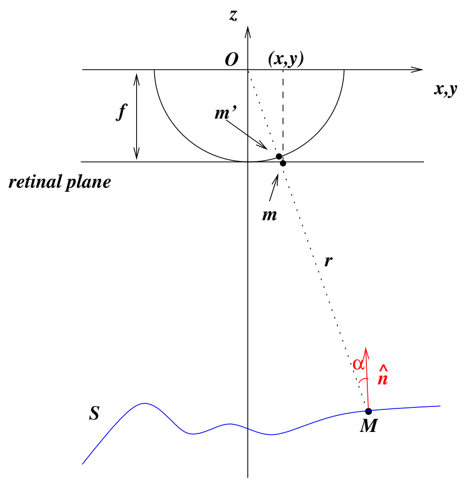


Figure 4.18: the PSFS$_r$ model

For such a surface $\mathcal{S}$, a normal vector $\hat{n}(x,y)$ at the point $M(x,y)$ is given by

$$\hat{n}(x,y) = \left(f\nabla u(x,y) - \frac{fxu(x,y)}{x^2+y^2+f^2},\ f\nabla u(x,y) - \frac{fyu(x,y)}{x^2+y^2+f^2},\ \nabla u(x,y)\cdot(x,y) + \frac{f^2u(x,y)}{x^2+y^2+f^2}\right). \tag{4.31}$$

For $M \in \mathcal{S}$, we denote by $\omega(M)$ the unit vector representing the light source direction at the point M. Since we assume that the light source is located at the optical center, we have

$$\omega(M(x,y)) = \frac{(-x,-y,f)}{\sqrt{x^2+y^2+f^2}}. \tag{4.32}$$

In this model we have

$$R(\hat{n}(x,y)) = \frac{\omega(M(x,y))\cdot\hat{n}(x,y)}{r^2}. \tag{4.33}$$

Remark 4.3 *The dependence of* ω *on* (x,y) *is due to the fact that the light source is close to the scene (indeed it coincides with the optical center). Moreover, the attenuation term* $1/r^2$ *is crucial and it is due to the fact that the power of the light source is finite and its effect decreases moving away from it. This is very realistic if, for example, the light source is a flash.*

Substituting (4.31) and (4.32) in (4.33), we can write the main equation (IE) as

$$\frac{I(x,y)f^2}{u(x,y)}\sqrt{\frac{f^2|\nabla u(x,y)|^2 + (\nabla u(x,y)\cdot(x,y))^2}{Q(x,y)^2} + u(x,y)^2} = u(x,y)^{-2}, \quad (x,y)\in\Omega \tag{4.34}$$

where

$$Q(x,y) = \sqrt{\frac{f^2}{x^2+y^2+f^2}}.$$

By assumption H$_8$, we know that the surface is completely visible, *i.e.* it is in front of the optical center. As a consequence, u is strictly positive. This allow us to simplify equation (4.34) by means of the transformation $v = \ln(u)$, obtaining

$$-e^{-2v(x,y)} + J(x,y)\sqrt{f^2|\nabla v(x,y)|^2 + (\nabla v(x,y)\cdot(x,y))^2 + Q(x,y)^2} = 0\,, \quad (x,y)\in\Omega \tag{4.35}$$

where

$$J(x,y) = \frac{I(x,y)f^2}{Q(x,y)}.$$

In [78] the authors state that equation (4.35) admits a "control formulation", *i.e.* it can be written as

$$-e^{-2v} + \sup_{a\in B(0,1)}\{-b(x,y,a)\cdot\nabla v - l(x,y,a)\} = 0\,, \quad (x,y)\in\Omega \tag{4.36}$$

where

$$l(x,y,a) = -I(x,y)f^2\sqrt{1-|a|^2}\,, \qquad b(x,y,a) = -J(x,y)\ {}^tR(x,y)\ D(x,y)\ R(x,y)\ a$$

$$D(x,y) = \begin{pmatrix} f & 0 \\ 0 & \sqrt{f^2+x^2+y^2}\end{pmatrix}, \qquad R(x,y) = \frac{1}{(x^2+y^2)^{1/2}}\begin{pmatrix} y & -x \\ x & y\end{pmatrix}$$

and ${}^tR(x,y)$ is the transpose of the matrix $R(x,y)$. Note that $l(x,y,a)=0$ on $\partial B(0,1)$, therefore in the numerical approximation we can not limit to consider the discretization of $\partial B(0,1)$ but we have to discretize the unit ball entirely.
The following result has been proved in [78].

Theorem 4.4 *Let Ω be bounded and smooth. If I is differentiable and if there exist $\delta > 0$ and M verifying $I \geq \delta$ and $|\nabla I| \leq M$, then equation* (4.35) *complemented with Dirichlet boundary condition $u = \phi$ on $\partial\Omega$ has a unique viscosity solution.*

This uniqueness result shows that, under certain assumptions, the PSFS$_r$ model is not ill-posed.
As we mentioned in section dedicated to the PSFS$_\infty$ model, the idea of state constraints provides a more convenient notion of boundary condition than Dirichlet's or Neumann's since it does not require any data. In [78] it is shown that a generic Hamiltonian $H(x,u,p)$ verifies state constraints as soon as

$$\nabla_p H(x,u,\nabla u(x))\cdot\eta(x) < 0 \qquad x\in\partial\Omega \tag{4.37}$$

where $\eta(x)$ is the unit inward normal vector to $\partial\Omega$ at the point x.
Consider for simplicity the unidimensional case $\Omega = [-x_0, x_0]$. It is easy to show that for our Hamiltonian $H_{PSFS_r}(x,v,v')$ defined as in the left-hand side of equation (4.35), the request (4.37) corresponds to $r'(-x_0) < 0$ and $r'(x_0) > 0$.

Theorem 4.5 *Under the assumptions of Theorem 4.4, equation* (4.35) *complemented with stat constraints boundary condition on $\partial\Omega$ has a unique viscosity solution.*

4.3.2 Numerical approximation

We present a semi-Lagrangian discretization for equation (4.36). The approximation scheme is much simpler than that presented in [78] and it has a built-in up-wind correction. By standard arguments, we get

$$-v_h(x,y)+\min_{a\in B(0,1)}\{v_h((x,y)+hb(x,y,a))+hl(x,y,a)\}+he^{-2v_h(x,y)}=0\,,\quad (x,y)\in\Omega \tag{4.38}$$

We want to solve equation (4.38) following the fixed point ideas of the previous chapters. Note that, unlike the standard control problems, once we compute the control a^* where the minimum is attained we need some extra work to compute $v_h(x,y)$. In fact, let us define

$$c := v_h((x,y)+hb(x,y,a^*))+hl(x,y,a^*)$$

and $t := v_h(x,y)$ for any (x,y) fixed. At every iteration we have to solve the equation

$$g(t) := -t+c+he^{-2t}=0.$$

We do that applying the Newton's method as in [78] (note that $g'(t)>0$ for all $t\in\mathbb{R}$). We start from any supersolution $v_h^{(0)}$ of (4.38) and we compute its solution, iterating the procedure until $\|v_n^{(n+1)}-v_h^{(n)}\|_\infty<\varepsilon$, where ε is a given tolerance. In [78] two supersolutions are suggested as initial guess $v^{(0)}$:

$$v^{(0)}=-\frac{1}{2}\ln(\delta f^2)\,,\quad \delta=\min I(x,y)\quad\text{and}\quad v^{(0)}=-\frac{1}{2}\ln(I(x,y)f^2).$$

We choose the time step $h=h(x,y)$ in such a way $|h(x,y)b(x,y,a^*)|\le\Delta x$ and we discretize the unit ball $B(0,1)$ in ($\#directions\times\#circles+1$) points (see Fig. 4.19).

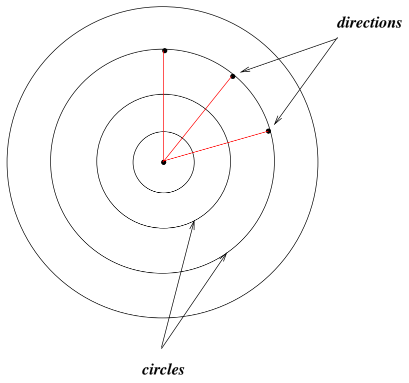


Figure 4.19: discretization of the unit ball $B(0,1)$

4.3.3 Numerical experiments

In this section we present two numerical experiments which show how promising and robust is the PSFS$_r$ model.

Test 1: flat surface

In this test we chose $\varepsilon = 10^{-4}$, $\Delta x = 0.03$, $f = 1$, 51×51 pixels and $8 \times 2 + 1$ controls. We impose state constraints on the boundary of the square. Convergence was attained after 66 iterations. The L^∞ error is 0.016. In Fig. 4.20 we show the initial photograph and the computed surface. In Fig. 4.21 we show the error function and the estimated optimal

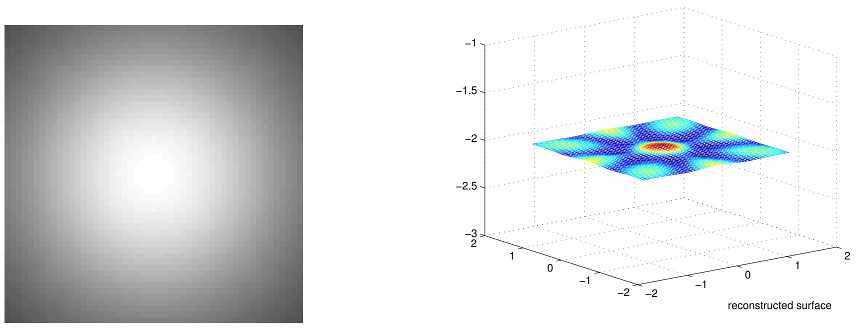


Figure 4.20: photograph (left) and reconstructed surface (right)

vector field. It is interesting to note that the optimal vector field points to the center of the square which is the first point to reach convergence in the fixed point procedure. In

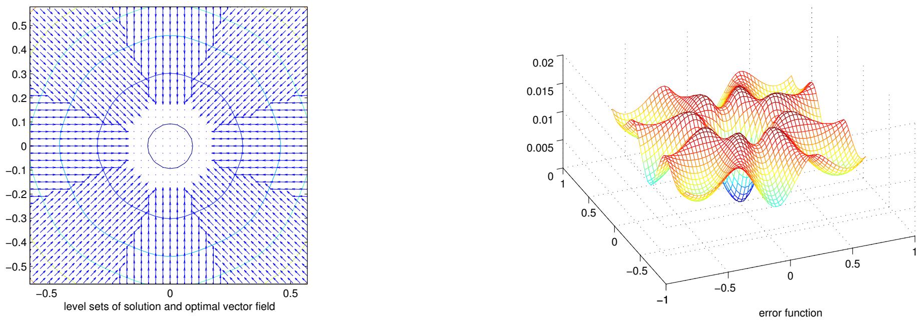


Figure 4.21: optimal vectorfield (left) and error function (right)

Table 4.4 and 4.5 we show the behavior of the L^∞ error with respect to the choice of

	$16 \times 2 + 1$	$32 \times 2 + 1$	$8 \times 8 + 1$
128×128	0.013	0.013	0.0081

Table 4.4: behavior of the L^∞ error with respect to the choice of controls

controls and of Δx. We can see that the error reduces particularly if $\#circles$ grows.

Test 2: pyramid upside down

In the second test we consider a pyramid upside down with the vertex standing on a flat

	$16 \times 4 + 1$
64×64	0.00479
128×128	0.00384
256×256	0.00344

Table 4.5: behavior of the L^∞ error with respect to the choice of Δx

background. We use a 128×128 pixels initial image and we chose $f = 1/4$. We impose Dirichlet boundary condition on the boundary of the pyramid (so any computation is done on the background). The initial image and the reconstructed surface are shown in Fig. 4.22. As in the previous case, we obtain good result. Consider that MATLAB connects all points of the surface despite there is a hole in the domain of the reconstructed surface as in Fig. 4.10.

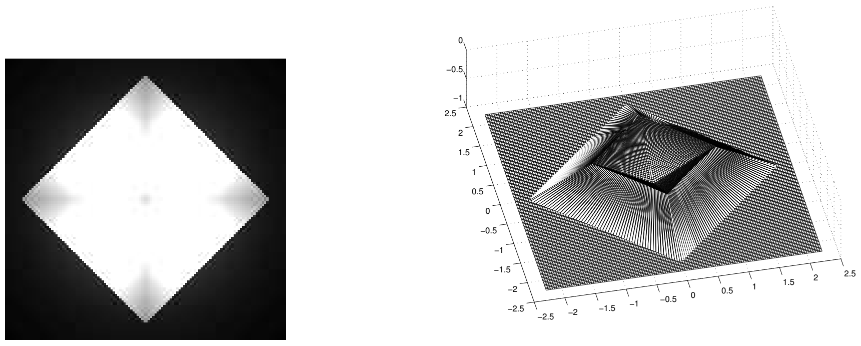


Figure 4.22: photograph (left) and reconstructed surface (right)

4.4 Shape from Shading: a well-posed problem?

In this section we investigate the well-posedness of both PSFS$_\infty$ and PSFS$_r$ models. The question is: does the concave/convex ambiguity still exist or it disappears? We will see by means of some numerical tests that in the PSFS$_\infty$ model the concave/convex ambiguity still exists, although it is slightly modified by the perspective deformation. With regard to PSFS$_r$ model, the matter is more delicate. Theorems 4.4 and 4.5 assure that the problem is well-posed under suitable assumption, but we will be able to prove the existence of two different surfaces S_1 and S_2 which are associated to the same brightness function I and the same boundary condition on $\partial\Omega$. The existence of such surfaces was first suggested by J.-D. Durou.
Note that the example we will give does not contradict Theorem 4.4 nor Theorem 4.5 since one of the two surfaces is not differentiable for all $x \in \Omega$ so that its brightness function I is not defined everywhere in Ω. On the other hand, we stress that the brightness function of the surface we construct is differentiable *a.e.* in Ω and it can be extended to a differentiable function in all Ω. Therefore, we believe that our example strongly limits the well-posedness of the problem declared in [78] and this is true in particular if we are

interested in applications to real problems.

We start showing two interesting numerical tests in which we compare the PSFS$_\infty$ and the PSFS$_r$ models. We consider a ridge tent upside down and its photograph (see Fig. 4.23). In Fig. 4.24 we show the result obtained by the PSFS$_\infty$ algorithm imposing Dirichlet

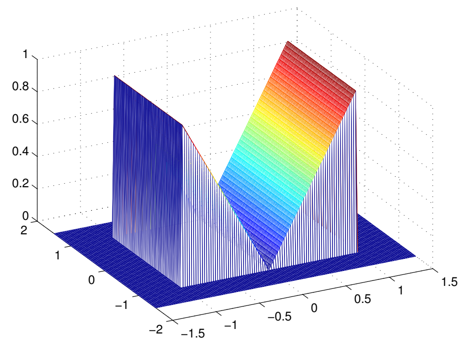
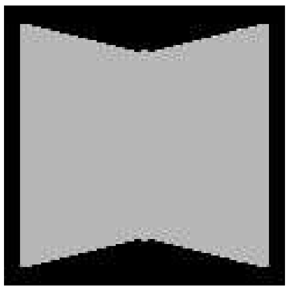



Figure 4.23: initial surface (left) and its photograph (right)

boundary condition on the silhouette of the tent. As we can see, the reconstruction fails

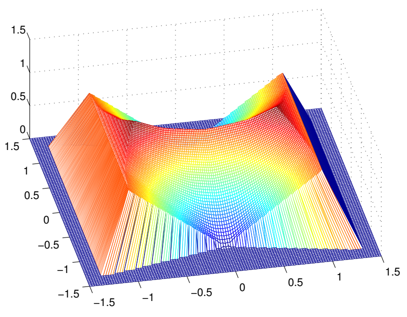
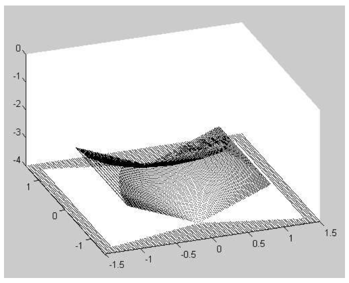



Figure 4.24: reconstructed surface with Dirichlet boundary condition. Interpolated (left) and not interpolated (right)

since the algorithm tries to compute the maximal solution instead of the correct solution. However, the shape of the domain (distorted in the photograph) is correctly straightened. Note that in Fig. 4.24-left MATLAB connects all points of the surface despite there is a hole in the domain of the reconstructed surface as in Fig. 4.10. In Fig. 4.24-right the same surface is plotted by a slighlty different point of view without interpolations.

In Fig. 4.25 we show the result obtained by the PSFS$_r$ algorithm imposing state constraints boundary condition on the boundary of the square (*i.e.* the background). In this case the reconstruction is definitively better than the previous one, considering that any boundary data was needed. On the other hand, we observe that the hole in the domain due to the regions in full shade is not reconstructed properly. In fact, the surfaces connecting the

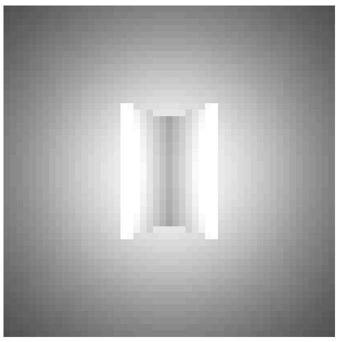
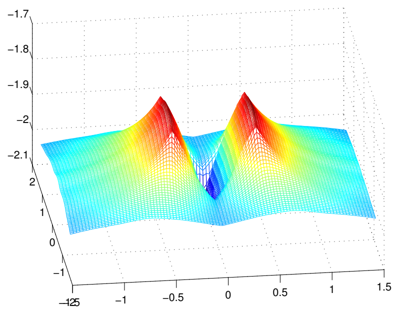



Figure 4.25: photograph (left) and reconstructed surface with state constraints boundary condition, 159 iter., 101 × 101 pixels, $f = 1$, $\Delta x = 0.015$, $16 \times 2 + 1$ controls (right)

tent and the background are really computed and they are not due to the MATLAB's interpolation.

We now come back to the existence of two different surfaces S_1 and S_2 which are associate to the same brightness function I in the PSFS$_r$ model. In order to do this, we need first to reformulate the problem in spherical coordinates (ρ, θ, ϕ) defined as usual (see Fig. 4.26), fixing the origin at the optical center. Given a brightness function $I(\theta, \phi)$ we are looking

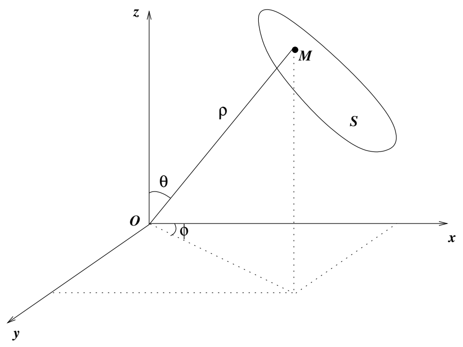


Figure 4.26: spherical coordinates

for a surface S in the form $\rho = \rho(\theta, \phi)$ such that

$$\frac{\omega(\theta,\phi) \cdot \hat{n}(\theta,\phi)}{\rho(\theta,\phi)^2} = I(\theta,\phi). \tag{4.39}$$

A generic point M on the surface S has coordinates

$$M = \begin{pmatrix} \rho \sin\theta \cos\phi \\ \rho \sin\theta \sin\phi \\ \rho \cos\theta \end{pmatrix}$$

with respect to the axis (x, y, z). We now introduce a new orthonormal system of coordinates $(u_\rho, u_\theta, u_\phi)$ defined by (see Fig. 4.27)

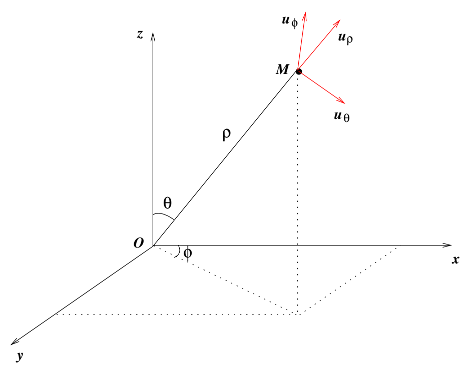


Figure 4.27: the new orthonormal system of coordinates $(u_\rho, u_\theta, u_\phi)$

$$u_\rho := \frac{M(\rho,\theta,\phi)}{\rho} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}, \qquad u_\theta := \frac{\partial_\theta u_\rho}{|\partial_\theta u_\rho|} = \begin{pmatrix} \cos\theta\cos\phi \\ \cos\theta\sin\phi \\ -\sin\theta \end{pmatrix}$$

and

$$u_\phi := \frac{\partial_\phi u_\rho}{|\partial_\phi u_\rho|} = \begin{pmatrix} -\sin\phi \\ \cos\phi \\ 0 \end{pmatrix}.$$

The new system $(u_\rho, u_\theta, u_\phi)$ is "mobile" and depends on the point $M \in S$. The coordinates of M in the new system are $(\rho, 0, 0)_{(u_\rho,u_\theta,u_\phi)}$.
The two vectors $\partial_\theta M$ and $\partial_\phi M$ are a basis in the plane orthogonal to the radial direction at the point $M = M(\rho(\theta,\phi),\theta,\phi)$.
Since $M = \rho u_\rho$ we have

$$\partial_\theta M = \rho_\theta u_\rho + \rho u_\theta \quad \text{and} \quad \partial_\phi M = \rho_\phi u_\rho + \rho u_\phi \sin\theta$$

and then

$$\partial_\theta M = (\rho_\theta, \rho, 0)_{(u_\rho,u_\theta,u_\phi)} \quad \text{and} \quad \partial_\phi M = (\rho_\phi, 0, \rho\sin\theta)_{(u_\rho,u_\theta,u_\phi)}.$$

Then, we can write the coordinates of the normal vector in the new system as

$$\hat{n}(\theta,\phi) = -\frac{\partial_\theta M \times \partial_\phi M}{|\partial_\theta M \times \partial_\phi M|} = \frac{(\rho^2\sin\theta, -\rho\rho_\theta\sin\theta, -\rho\rho_\phi)_{(u_\rho,u_\theta,u_\phi)}}{(\rho^4\sin^2\theta + \rho^2\rho_\theta^2\sin^2\theta + \rho^2\rho_\phi^2)^{1/2}}$$

so we have

$$\omega(\theta,\phi)\cdot\hat{n}(\theta,\phi) = u_\rho\cdot\hat{n} = (1,0,0)_{(u_\rho,u_\theta,u_\phi)}\cdot\hat{n}_{(u_\rho,u_\theta,u_\phi)} = \frac{\rho^2\sin\theta}{(\rho^4\sin^2\theta + \rho^2\rho_\theta^2\sin^2\theta + \rho^2\rho_\phi^2)^{1/2}}.$$

In conclusion, equation (4.39) can be written as

$$I(\theta,\phi) = \frac{\sin\theta}{(\rho^4\sin^2\theta + \rho^2\rho_\theta^2\sin^2\theta + \rho^2\rho_\phi^2)^{1/2}}$$

or, in a equivalent form,

$$\rho^2\left(\rho^2+\rho_\theta^2+\frac{\rho_\phi^2}{\sin^2\theta}\right)=\frac{1}{I^2}. \tag{4.40}$$

Equation (4.40) is the Image Irradiance Equation for the PSFS$_r$ problem in spherical coordinates.

We now come back to our purpose. We choose as first surface S_1 the surface $\rho(\theta,\phi)\equiv 1$, where $(\theta,\phi)\in[-\overline{\theta},\overline{\theta}]\times[-\overline{\phi},\overline{\phi}]$ for some $\overline{\theta},\overline{\phi}<\pi/2$ (see Fig. 4.28) which is associated

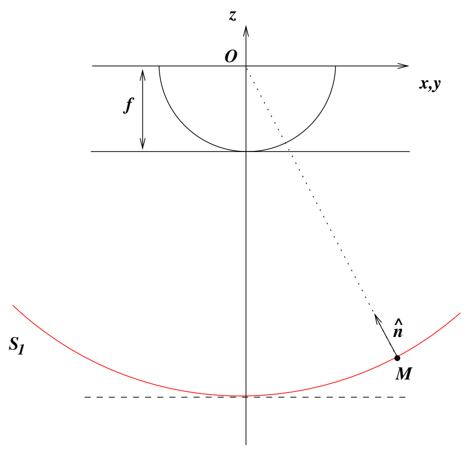


Figure 4.28: the surface S_1

to the brightness function $I_{S_1}(\theta,\phi)\equiv 1$. Then, we look for a second surface S_2 which is associated to the same brightness function as S_1 and it is of the form $\rho=\rho(\theta,\phi)=\rho(\theta)$. Equation (4.40) is simplified to the following ordinary differential equation

$$\rho^2(\rho^2+\rho_\theta^2)=1 \tag{4.41}$$

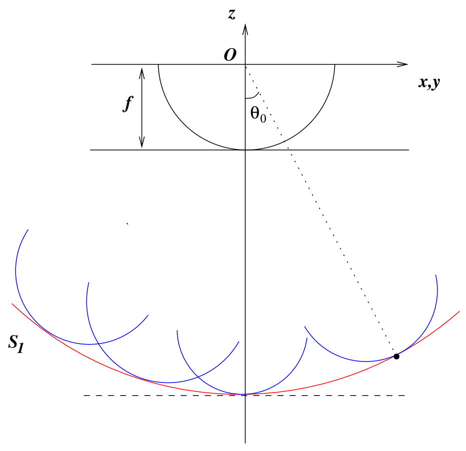
or, equivalently (if $\rho\leq 1$),

$$\rho_\theta=g(\rho)\,,\qquad \text{where } g(\rho):=-\frac{\sqrt{1-\rho^4}}{\rho} \tag{4.42}$$

It should be noted that g is not Lipschitz continuous if $\rho=1$. In fact, if we complement equation (4.42) whith the initial condition $\rho(\theta_0)=1$ for some θ_0 we obtain the family of solutions

$$\rho(\theta;\theta_0)=\sqrt{\cos(2(\theta-\theta_0))}\,,\quad \theta\in\left[-\frac{\pi}{4}+\theta_0,\frac{\pi}{4}+\theta_0\right]$$

which correspond to a family of surfaces $\{S_{\theta_0}\}$ associated to the same brightness function $I\equiv 1$ (see Fig. 4.29).
Note that, although we constructed a family of differentiable surfaces with the same brightness function, they are still distinguishable by the PSFS$_r$ model by means of the boundary condition (state constraints or Dirichlet). On the other hand, it is possible to construct a *a.e.* differentiable surface S_2 joining together two surfaces as in Fig. 4.30



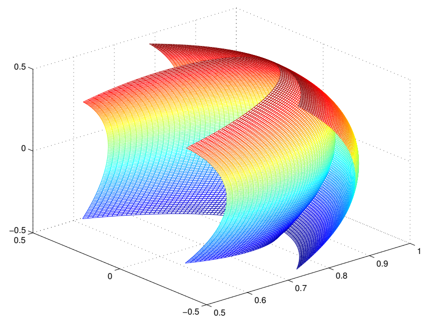
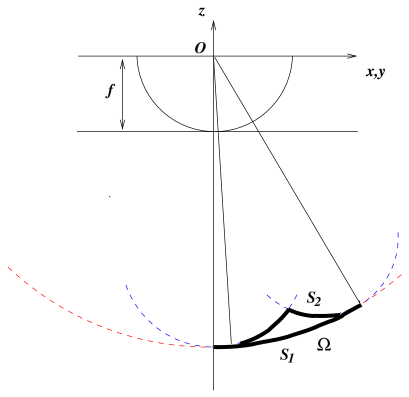


Figure 4.29: the surface S_1 ($\rho \equiv 1$) and some S_{θ_0}'s (left). The surfaces S_1 and $S_{\theta_0=0}$ (right, after a rotation)



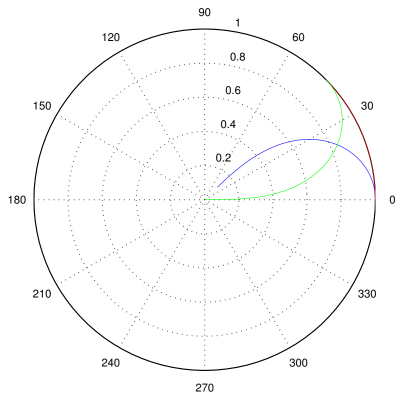


Figure 4.30: how to construct two surfaces with the same brightness function and same boundary condition

and such that S_1 and S_2 verify *the same Dirichlet condition at the boundary*. Moreover, although the brightness function of S_2 is defined only *a.e.*, it can be extended to the value 1 where it is not defined obtaining a differentiable brightness function in all Ω.

Finally, we tried to confirm our theoretical results by means of some numerical tests. First of all, we noted that if we solve the equation in a square, choosing $I \equiv 1$ and imposing suitable Dirichlet boundary condition we obtain the surface S_1, *i.e.* the hemisphere. In order to force the algorithm to compute the surface S_2 we need to fix the solution in some internal node. If we fix the height of the surface on the vertical line $x = 0$ the algorithm converges to the surface showed in Fig. 4.31-left. It is a credible approximation of the surface S_2. Fig. 4.31-right is obtained fixing the height of the surface in a single point (we chose the center point $(i, j) = (26, 26)$ on a 51×51 pixels grid). In this case the algorithm seems to join four functions in the set of the solutions of equation (4.40).

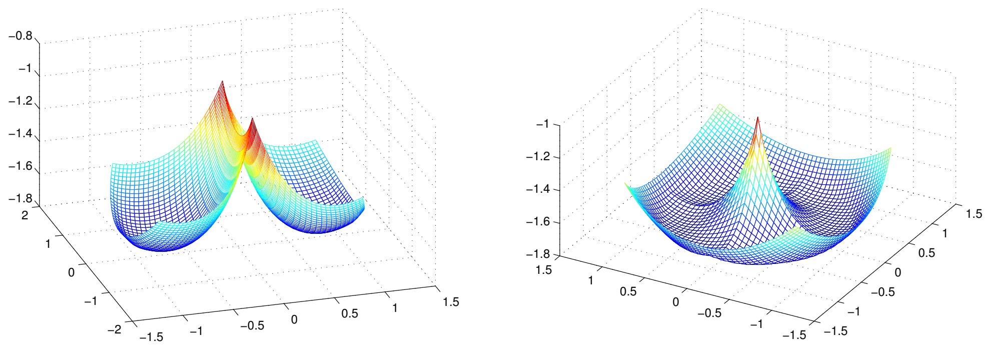


Figure 4.31: how to construct two surfaces with the same brightness function and same boundary condition

Chapter 5

Approximation of Pursuit-Evasion games with state constraints

In this chapter we study the fully-discrete semi-Lagrangian scheme for Pursuit-Evasion games with state constraints from the theoretical and numerical point of view.
In the first part we prove that the solution of the fully-discrete problem converges to the time-discrete value function as the mesh size goes to zero. Then, we couple this result with a recent result of Bardi et al. [11] to obtain, under suitable assumptions, the convergence to the solution v of the continuous problem when the time and space steps go to zero.

Afterwards, we analyze the constrained Tag-Chase game in which two boys run one after the other in a bounded convex domain with the same velocity. In this case the value function is discontinuous and most of the theoretical results we know for Tag-Chase game does not hold. We prove that the time of capture is finite if the capture occurs when the Pursuer enter in a small ball centered at the Evader's position.

In order to introduce the numerical tests, we give some hints for the implementation of the algorithm. First, we briefly recall a simple and useful technique to make interpolation in high dimensional spaces giving a precise error estimate. We will apply this technique in the last part of the chapter to the numerical solution of the Hamilton-Jacobi-Isaacs equation for games in $\mathbb{R}^4$.
Later on, we explain how it is possible to reduce the size of the problem taking advantage of symmetries in the case the Tag-Chase game is played in a square.

In the last section of the chapter we present a number of numerical tests for constrained Tag-Chase game. In some cases, we discover a bizarre behavior of the approximate optimal trajectories. Note that there are very few numerical results on constrained differential games. We cite the considerable and pioneering work of Alziary de Roquefort [3] where a large number of numerical tests are presented although they can not be considered a definitive study due to the very small power of the employed computer. In Bardi et al. [10] there are some interesting tests in $\Omega \subset \mathbb{R}^2$ with state constraints and discontinuous value function. In [6] the effect of the boundary conditions for the free problem in $\mathbb{R}^4$ is studied. Finally, we mention the paper of Pesch et al. [75] where the optimal trajectories are computed by means of neural networks (without solving Isaacs equation). Some other interesting results on optimal trajectories are in Breakwell [17], where no computers are used.

5.1 Convergence of the fully-discrete numerical scheme

In this section we present some results related to the approximation of Pursuit-Evasion games with state constraints. The main result is that the solution of the fully-discrete problem converges to the time-discrete value function as the mesh size goes to zero. The proof joins a number of techniques already presented in some papers. We mainly follow the ideas in Falcone [48] which presents a convergence result for the minimum time problem without state constraints. We generalize those ideas to games and we adapt all the notions involved to the state constrained case following [18].
Note that in [10] it is proved the convergence of the fully-discrete solution to the solution of the continuous problem in the free case, but this result can not be directly generalized to the constrained case. In [18] the convergence result is proved in the constrained case, but it strictly relies on the fact that the time-discrete value function is continuous so we can not apply the same ideas to Pursuit-Evasion games.
It should be noted that most of our results can be probably adapted to deal with generalized differential games. To do this, we can borrow some ideas introduced in [66] with regards to the definition of admissible controls which are not restricted to Pursuit-Evasion games.
Finally, we couple our result with a recent result of Bardi et al. [11] (see also [66, 59]) to obtain, under suitable assumptions, the convergence to the solution v of the continuous problem when the time and space steps go to zero.

We now formulate the time-discrete and fully-discrete version of Pursuit-Evasion games with state constraints. In order to do this we will adapt to the constrained case the work done in section 1.5.2 for the free case (see also section 1.4.2 for the formulation of the continuous problem with state constraints). For numerical purpose we suppose hereafter that $\mathcal{T} \subset \overline{\Omega}$.
We will consider for simplicity a discrete version of the dynamics based on the Euler scheme, namely

$$\begin{cases} y_{n+1} = y_n + hf(y_n, a_n, b_n) \\ y_0 = x \end{cases}$$

where $y_n = (y_n^P, y_n^E)$, $x = (x_P, x_E)$ and $f(y_n, a, b) = (f_P(y_n^P, a), f_E(y_n^E, b))$. The first player has to keep the system in $\overline{\Omega}_1 \subset \mathbb{R}^n$ and the second player in $\overline{\Omega}_2 \subset \mathbb{R}^n$. The game is set in $\overline{\Omega} \subset \mathbb{R}^{2n}$ where $\Omega := \Omega_1 \times \Omega_2$. Following the continuous case, let us define the discrete version of the admissible controls verifying the state constraints

$$\mathcal{A}_x^h := \left\{ \{a_n\}_{n\in\mathbb{N}} : a_n \in A \text{ and } y_n^P \in \overline{\Omega}_1\,, \text{ for all } n \right\}\,, \qquad x \in \overline{\Omega}$$

and

$$\mathcal{B}_x^h := \left\{ \{b_n\}_{n\in\mathbb{N}} : b_n \in B \text{ and } y_n^E \in \overline{\Omega}_2\,, \text{ for all } n \right\}\,, \qquad x \in \overline{\Omega}.$$

Then we define

$$A_h(x) := \left\{ a \in A : x_P + hf_P(x_P, a) \in \overline{\Omega}_1 \right\}\,, \qquad x \in \overline{\Omega}$$

and

$$B_h(x) := \left\{ b \in B : x_E + hf_E(x_E, b) \in \overline{\Omega}_2 \right\}\,, \qquad x \in \overline{\Omega}.$$

Clearly,

$$\{a_n\}_{n\in\mathbb{N}} \in \mathcal{A}^h_x \Leftrightarrow a_n \in A_h(y_n)\,, \qquad \text{for all } n \geq 1$$

$$\{b_n\}_{n\in\mathbb{N}} \in \mathcal{B}^h_x \Leftrightarrow b_n \in B_h(y_n)\,, \qquad \text{for all } n \geq 1.$$

Hereafter we will assume that

$$\text{There is } h_0 > 0 \text{ such that } A_h(x) \neq \emptyset \text{ and } B_h(x) \neq \emptyset \text{ for all } (h,x) \in (0,h_0] \times \overline{\Omega}. \tag{5.1}$$

Definition 5.1 *A* strategy *for the first player is a map $\alpha_x : \mathcal{B}^h_x \to \mathcal{A}^h_x$; it is* nonanticipating *if $\alpha_x \in \Gamma^h_x$, where*

$\Gamma^h_x := \{\alpha_x : \mathcal{B}^h_x \to \mathcal{A}^h_x : b_n = \tilde{b}_n$ *for all* $n \leq n'$ *implies* $\alpha_x[\{b_k\}]_n = \alpha_x[\{\tilde{b}_k\}]_n$ *for all* $n \leq n'\}$.

We define n_h as in Section 1.5.2 then we define the time-discrete value function T_h as

$$T_h(x) := \inf_{\alpha_x \in \Gamma^h_x} \sup_{\{b_n\}\in\mathcal{B}^h_x} h\, n_h(x, \alpha_x[\{b_n\}], \{b_n\})$$

and its Kružkov transform

$$v_h(x) := 1 - e^{-T_h(x)}\,, \quad x \in \overline{\Omega}. \tag{5.2}$$

The proof of the *Discrete Dynamic Programming Principle* in Section 1.5.2 can be easily adapted to the constrained case noting that

$$\{a'_m\}, \{a''_m\} \in \mathcal{A}^h_x \quad \Rightarrow \quad \{\tilde{a}_m\} := \{a'_0, a'_1, \ldots, a'_n, a''_0, a''_1, \ldots\} \in \mathcal{A}^h_x \qquad \text{for all } n \in \mathbb{N}$$

and similarly for $\mathcal{B}^h_x$.

Therefore, we can conclude that v_h is the unique bounded solution of

$$\begin{cases} v_h(x) = \max\limits_{b\in B_h(x)} \min\limits_{a\in A_h(x)} \{\beta v_h(x + hf(x,a,b))\} + 1 - \beta & x \in \overline{\Omega}\backslash\mathcal{T} \\ v_h(x) = 0 & x \in \mathcal{T} \end{cases} \tag{HJI$_h$–Ω}$$

where $\beta = e^{-h}$. In order to achieve the fully-discrete equation we build a regular triangulation of $\overline{\Omega}$ as in Sections 1.5.1-1.5.2 denoting by X the set of its nodes x_i, $i = 1, \ldots, N$ and by S the set of simplices S_j, $j = 1, \ldots, L$. $V(S)$ will denote the set of the vertices of a simplex S and the space discretization step will be denoted by k where $k := \max_j\{diam(S_j)\}$.

The fully-discrete approximation scheme is

$$\begin{cases} v^k_h(x_i) = \max\limits_{b\in B_h(x_i)} \min\limits_{a\in A_h(x_i)} \left\{\beta v^k_h(x_i + hf(x_i,a,b))\right\} + 1 - \beta & x_i \in (\overline{\Omega}\backslash\mathcal{T}) \cap X \\ v^k_h(x_i) = 0 & x_i \in \mathcal{T} \cap X \\ \\ v^k_h(x) = \sum_j \lambda_j(x) v^k_h(x_j)\,, \quad 0 \leq \lambda_j(x) \leq 1\,, \quad \sum_j \lambda_j(x) = 1 & x \in \overline{\Omega} \end{cases} \tag{HJI^k_h–Ω}$$

As in the unconstrained problem, the choice of linear interpolation is not an obligation and it was made for straightforwardness reasons. We denote by W^k the set

$$W^k := \left\{w \in C(\overline{\Omega}) : \nabla w(x) = \text{ constant for all } x \in S_j\,, j = 1, \ldots, L\right\}.$$

Theorem 5.2 *Equation* (HJI$_h^k$–Ω) *has a unique solution* $v_h^k \in W^k$ *such that* $v_h^k : \overline{\Omega} \to [0,1]$.

Proof. The proof follows by the same arguments already presented in Theorem 1.36 and its Corollary 1.37 by means of slightly modifications in order to take into account the state constraints. ■

Let us define

$$\mathcal{R}_0 := \mathcal{T}$$

and

$$\mathcal{R}_n := \left\{ x \in \overline{\Omega} \setminus \bigcup_{j=0}^{n-1} \mathcal{R}_j : \text{ for all } b_x \in B_h(x) \text{ there exists } \hat{a}_x(b_x) \in A_h(x) \text{ such that } x + hf(x, \hat{a}_x(b_x), b_x) \in \mathcal{R}_{n-1} \right\}, \quad n \geq 1. \tag{5.3}$$

See [48] for an analogous definition in the framework of the minimum time problem. Note that the minimum time problem can be seen as a particular Pursuit-Evasion game when the set B is a singleton, *i.e.* $B = \{b_0\}$.

Remark 5.3 *1. The shape of the sets* $\{\mathcal{R}_n\, ,\ n \in \mathbb{N}\}$, *strictly depends on* f *and* Ω*;*

2. $\mathcal{R}_n \cap \mathcal{R}_m = \emptyset$ *for all* $n \neq m$*;*

3. If $\mathcal{R}_p = \emptyset$ *for some* $p \in \mathbb{N}$, *then* $\mathcal{R}_q = \emptyset$ *for any* $q \geq p$*;*

4. The sets $\{\mathcal{R}_n\, ,\ n \in \mathbb{N}\}$ *are the level sets of* v_h *and* v_h *jumps on the boundary of each of them. Therefore,* v_h *is discontinuous.*

Hereafter we will always assume that

$$\overline{\Omega} = \bigcup_{j=0}^{\infty} \mathcal{R}_j. \tag{5.4}$$

Note that the assumption (5.4) can be seen as a sort of small time controllability assumption and that it is not really restrictive since if it exists a point $x \in \overline{\Omega} \setminus \bigcup_j \mathcal{R}_j$ this means that player P can not win the game from that point (he can not drive the system to the target) and then $v_h(x) = 1$.

We introduce two important assumptions on the triangulation. The first is the following CFL-like condition (see Fig. 5.1).

$$x \in S \cap \mathcal{R}_n \Rightarrow V(S) \subset \mathcal{R}_{n-1} \cup \mathcal{R}_n \cup \mathcal{R}_{n+1} \tag{5.5}$$

The second assumption is the "consistency" of the triangulation (see Fig. 5.1).

Definition 5.4 *We say that a triangulation is "consistent" if, for any* $n \in \mathbb{N}$ *and any* $x \in S \cap \mathcal{R}_n$ *it exists at least one vertex* $x_i \in V(S)$ *such that* $x_i \in \mathcal{R}_n$.

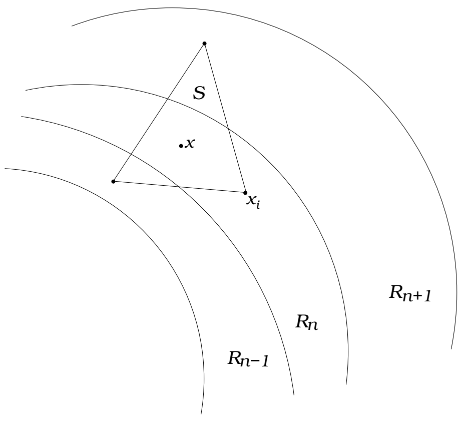


Figure 5.1: a simplex S crossing $\mathcal{R}_n$, $\mathcal{R}_{n-1}$ and $\mathcal{R}_{n+1}$

Let v_h and v_h^k denote respectively the solution of (HJI$_h$–Ω) and (HJI$_h^k$–Ω). We now state the main result of this chapter.

Theorem 5.5 *Let Ω an open bounded set. Let f be continuous and* (1.32) *holds. Moreover assume that* (5.4), (5.5) *hold true and let the triangulation be "consistent". Then, for $n \geq 1$*

a) $v_h(x) \leq v_h(y)$, *for any* $x \in \bigcup_{j=0}^{n} \mathcal{R}_j$, *for any* $y \in \overline{\Omega} \setminus \bigcup_{j=0}^{n} \mathcal{R}_j$;

b) $v_h(x) = 1 - e^{-nh}$, *for any* $x \in \mathcal{R}_n$;

c) $v_h^k(x) = 1 - e^{-nh} + O(k) \sum_{j=0}^{n} e^{-jh}$ *for any* $x \in \mathcal{R}_n$;

d) There exists a constant $C > 0$ *such that* $|v_h(x) - v_h^k(x)| \leq \frac{Ck}{1-e^{-h}}$, *for any* $x \in \mathcal{R}_n$.

Proof. a) By induction. For $n = 0$ the statement is true since

$$0 = v_h(x) \leq v_h(y) \quad \text{for all } x \in \mathcal{T}, \quad \text{for all } y \in \overline{\Omega} \setminus \mathcal{T}.$$

Let the statement be true up to $n-1$. Suppose by contradiction that

$$\text{there exists } x \in \bigcup_{j=0}^{n} \mathcal{R}_j \text{ and } y \in \overline{\Omega} \setminus \bigcup_{j=0}^{n} \mathcal{R}_j \text{ such that } v_h(y) < v_h(x).$$

Therefore there exists a (discrete) trajectory that starts from $\overline{\Omega} \setminus \bigcup_{j=0}^{n} \mathcal{R}_j$ and reaches the target in less then n time steps passing through $\mathcal{R}_n$. The contradiction follows by the definition of $\mathcal{R}_n$.
b) By the definition of $\mathcal{R}_n$, for any $x \in \mathcal{R}_n$ we can find $n+1$ points $x^{(q)}$, $q = 0, \dots, n$ such that $x^{(0)} = x$ and $x^{(q)} \in \mathcal{R}_{n-q}$. Introducing for simplicity the notations $a_q := a_{x^{(q)}}$ and $b_q := b_{x^{(q)}}$ we can write the sequence of the points $x^{(q)}$ more explicitly as

$$x^{(q+1)} = x^{(q)} + hf(x^{(q)}, \hat{a}_q(b_q^*), b_q^*)$$

where we use the * to indicate the optimal choice. As a consequence, the state of the system can reach the target in n steps and then $v_h(x) \le 1 - e^{-nh}$. Suppose by contradiction that $v_h(x) < 1 - e^{-nh}$. As in *b*), this means that the state has reached the target starting at x in less then n time steps but this is impossible since $x \in \mathcal{R}_n$.
c) By construction we have $v_h^k(x_i) = 0$ for all $x_i \in \mathcal{R}_0 \cap X$. We now consider a generic point $x \in \mathcal{R}_0$ and let S be the simplex containing x. Since the triangulation is "consistent", S must have at least a vertex $x_{i_0} \in \mathcal{R}_0$ and then $v_h^k(x) = O(k)$ for all $x \in \mathcal{R}_0$ since $v_h^k \in W^k$. This implies, for all $x_i \in \mathcal{R}_1 \cap X$,

$$v_h^k(x_i) = \beta v_h^k(x_i + hf(x_i, a^*, b^*)) + 1 - \beta = \beta O(k) + 1 - \beta$$

since $x_i + hf(x_i, a^*, b^*) \in \mathcal{R}_0$. We now consider a generic point $x \in \mathcal{R}_1$. By the same arguments there exists at least one vertex $x_{i_1} \in \mathcal{R}_1$ such that

$$v_h^k(x) = v_h^k(x_{i_1}) + O(k) = \beta O(k) + 1 - \beta + O(k) = 1 - \beta + (1+\beta)O(k).$$

For any $x_i \in \mathcal{R}_2 \cap X$

$$v_h^k(x_i) = \beta(1 - \beta + (1+\beta)O(k)) + 1 - \beta = 1 - \beta^2 + (\beta + \beta^2)O(k)$$

and, for any $x \in \mathcal{R}_2$ it exists $x_{i_2} \in \mathcal{R}_2$ such that

$$v_h^k(x) = v_h^k(x_{i_2}) + O(k) = 1 - \beta^2 + (1 + \beta + \beta^2)O(k).$$

Continuing by recursion we obtain, for any $x \in \mathcal{R}_n$

$$v_h^k(x) = 1 - \beta^n + O(k)\sum_{j=0}^{n} \beta^j.$$

d) By *b*) and *c*) it exists a constant C_1 (positive or negative) such that

$$|v_h(x) - v_h^k(x)| = |C_1| k \sum_{j=0}^{n} \beta^j \le \frac{|C_1|k}{1-\beta} = \frac{|C_1|k}{1-e^{-h}}.$$

■

Corollary 5.6 *Let Ω an open bounded set. Let f be continuous and* (1.32), (5.4) *hold. Moreover assume that*

$$\min_{x,a,b} |f(x,a,b| \ge f_0 > 0 \quad \textit{and} \quad 0 < k \le f_0 h. \tag{5.6}$$

Then, for $k \to 0^+$, v_h^k converges to v_h uniformly in $\overline{\Omega}$ for any $h > 0$ fixed.

Proof. First note that condition (5.6) is a sufficient condition for (5.5) and for the *consistency* of the triangulation. Therefore we can apply Theorem 5.5 and we easily conclude. ■

In order to obtain uniform convergence of v_h^k to the solution v of the continuous problem we couple our result with those in [11]. In Section 1.4.2 we proved that, under assumptions (C1), (C2) and (C3), the solution v of (HJI-Ω) is continuous in $\overline{\Omega}$. Let us introduce the following hypothesis on $\mathcal{T}$ (we always assume that $\mathcal{T} \subset \overline{\Omega}$).

$$\begin{cases} \text{For each } x \in \partial\mathcal{T} \text{ there are } r, \theta > 0 \text{ and } \Xi \in \mathbb{R}^{2n} \text{ such that} \\ \bigcup\limits_{0<t<r} B(x' + t\Xi, t\theta) \subset \Omega\backslash\mathcal{T} \text{ for } x' \in B(x,r) \cap \overline{\Omega\backslash\mathcal{T}}. \end{cases} \tag{5.7}$$

We have the following

Theorem 5.7 *Let Ω be an open bounded set. Let f be continuous and* (1.32)*,* (C1)*,* (C2)*,* (C3)*,* (5.1) *and* (5.7) *hold. Let v be a solution of* (HJI-Ω)*. Finally, assume that*

$$f_P(x_P, A(x_P)) \quad \textit{and} \quad f_E(x_E, B(x_E)) \quad \textit{are convex sets.} \tag{5.8}$$

Then, for $h \to 0^+$, v_h converges to v uniformly in $\overline{\Omega}$.

Proof. Hypothesis (5.8) guarantees that the value function v_h for the time-discrete problem defined in (5.2) coincides with that used in [11]. Moreover, assumptions of Theorem 1.25 are fulfilled so that $v \in C(\overline{\Omega})$. Then, the proof follows by Theorem 4.2 in [11]. ■

Now we can state the final result

Corollary 5.8 *Assume that there exists a constant C_2 such that $k \leq C_2 h^{1+\alpha}$, $\alpha > 0$. Then, under the assumptions of Corollary 5.6 and of Theorem 5.7, v_h^k converges to v uniformly in $\overline{\Omega}$ for $h \to 0^+$.*

Proof. Since $1 - e^{-h} \approx h$ for $h \to 0^+$ we have

$$|v_h^k(x) - v(x)| \leq |v_h^k(x) - v_h(x)| + |v_h(x) - v(x)| \leq C_3 h^\alpha + \|v_h(x) - v(x)\|_\infty$$

for some constant C_3. ■

5.2 Tag-Chase game with state constraints

In section 1.4.1 we introduced the Tag-Chase game as a particular case of Pursuit-Evasion games. We consider two boys P and E which run one after the other in the same 2-dimensional domain, so that the game is set in $\overline{\Omega} = \overline{\Omega}_1^2 \subset \mathbb{R}^4$ where Ω_1 is an open bounded set of $\mathbb{R}^2$. We denote by (x_P, x_E) the coordinates of $\overline{\Omega}$ where $x_P,\ x_E \in \overline{\Omega}_1$. P and E can run in every direction with velocity V_P and V_E respectively.
The case $V_P > V_E$ was completely studied in [2, 3]. The value function $T = -\ln(1-v)$, which represents the capture time, is continuous and bounded in all its domain of definition. Moreover the convergence result we obtained in section 5.1 applies to this case.
On the other hand, the most interesting case is certainly $V_P = V_E$, *i.e.* when any player has an advantage over the other. In this case it is easily seen that the value function T is discontinuous and then all theoretical results based on the continuity of the value function

does not hold.

In this section we will answer to the question: if $V_P = V_E$, the capture time is finite?
If the Tag-Chase game is played without constraints on the state and both players play optimally, it is immediately seen that the distance between P and E remains constant and then capture never happens (the optimal strategy for E is move as fast as he can along the line joining P and E in the opposite direction with respect to the position of P for ever). On the contrary, if the state is constrained in a bounded domain, such a restriction seems to play a key role against the Evader, as the following Proposition shows.

Proposition 5.9 *Let the target be*

$$\mathcal{T} = \{(x_P, x_E) \in \mathbb{R}^{2n} : d(x_P, x_E) \leq \varepsilon\}, \qquad \varepsilon \geq 0. \tag{5.9}$$

and Ω_1 an open bounded set. Then,

1. *If $V_P > V_E$ then the capture time $t_c = T(x_P, x_E) = -\ln(1 - v(x_P, x_E))$ is finite and bounded by*
$$t_c \leq \frac{|x_P - x_E|}{V_P - V_E}.$$
2. *If $V_P = V_E$, $\varepsilon \neq 0$ and Ω_1 is convex then the capture time t_c is finite.*

Proof.

1. This first part of the proof can be found in [2]. We fix a strategy for P and leave E free to decide his optimal strategy. First, P reaches the starting point of E covering the distance $|x_P - x_E|$ and then he follows the E's trace. The conclusion follows by elementary computations.

2. The basic idea of the proof is the same of the previous case but we have to change the strategy for the Pursuer in order to have a finite upper bound. P runs after E always along the line joining P and E (P can do it by the convexity of Ω_1) while E chooses his own optimal trajectory as before. We can characterize the strategy of E by a function $\theta(t) : [0, +\infty) \to [0, 2\pi)$ which represents at every time the angle between the velocity vector of E and the line joining P and E (see Fig. 5.2). Let us denote by $d_{PE}(t)$ the distance between P and E at time t. We claim that, for any fixed t,
$$\theta(t) \neq 0 \Rightarrow d'_{PE}(t) < 0 \tag{5.10}$$
where $' = \frac{d}{dt}$. Due to the state constraints, $\theta(t)$ can not be equal to 0 for a time interval longer then $diag(Q)$ and after that must be different from 0 for a finite time interval because E must change his trajectory. Therefore, if (5.10) holds then $d_{PE}(t) \to 0$ for $t \to \infty$ and then for any $\varepsilon > 0$ there exists a time $\bar{t}$ such that $d_{PE}(\bar{t}) \leq \varepsilon$ (the capture occurs).
In order to prove (5.10), let us define the two vectors $E(t)$ and $P(t)$ which are, respectively, the position of P and E at time t and the vector $r(t) := E(t) - P(t)$. Obviously we have $d_{PE}(t) = |r(t)|$. Without loss of generality, suppose that at time t, $P(t)$ is in the origin and $E(t)$ lies on the x-axis and that $V_P = V_E = 1$ (see Fig.

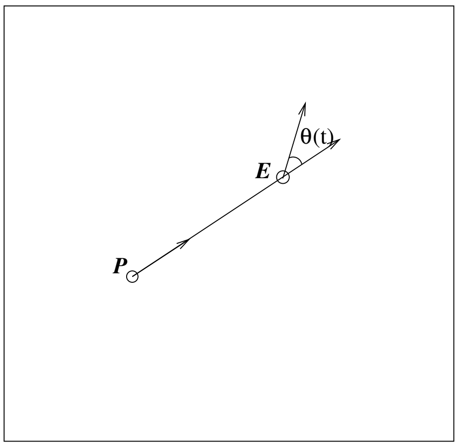


Figure 5.2: Trajectories of P and E in Proof of Proposition 5.9

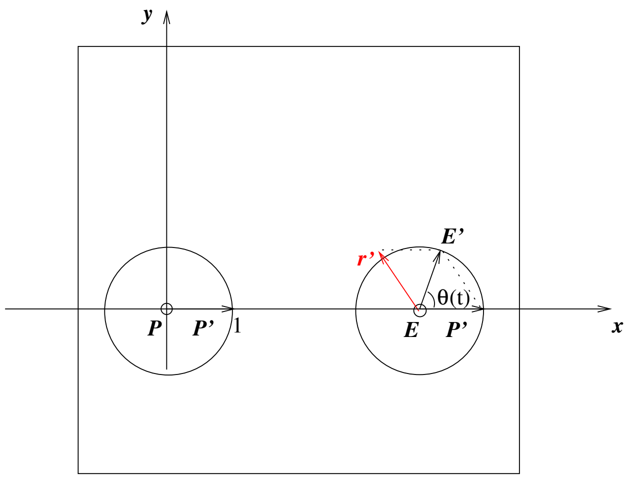


Figure 5.3: Vectors P, P', E, E' and r' in Proof of Proposition 5.9

5.3). Then

$$P'(t) = (1, 0) \quad \text{and} \quad \frac{r(t)}{|r(t)|} = (1, 0).$$

Moreover, by construction we have

$$E'(t) = (\cos\theta(t), \sin\theta(t)) \quad \text{and} \quad r'(t) = E'(t) - P'(t).$$

It follows that $r'(t) = (\cos\theta(t) - 1, \sin\theta(t))$ and

$$d'_{PE}(t) = \frac{r(t)}{|r(t)|} \cdot r'(t) = \cos\theta(t) - 1 \tag{5.11}$$

so that (5.10) holds. ■

5.3 Some hints for the algorithm

In this section we give some hints for an efficient implementation of the algorithm for the solution of differential games.

5.3.1 Interpolation in high dimension

In section 5.4 we will use the semi-Lagrangian scheme (HJI$_h^k$–Ω) to provide some numerical results for real problems. In every case we will consider 2-player differential games in which each player moves in a bounded domain $\overline{\Omega}_i \subset \mathbb{R}^2$, $i = 1, 2$, so that the game is set in $\overline{\Omega}$ where $\Omega = \Omega_1 \times \Omega_2 \subset \mathbb{R}^4$. Note that no reduction of the dimension of the problem seems to be possible due to the state constraints (see also section 5.3.2 below and [75, 3]).
It is important to note that the SL scheme requires that at every iteration and at every node the value $v_h^k(x_i + hf(x_i, a, b))$ is computed and this computation needs an interpolation of the values of v_h^k at the nodes. In order to overcome the difficulties coming from the high number of nodes we have to implement a fast method for making these interpolations. In [26] is extensively analyzed a fast and efficient interpolation method in high dimension suitable to our purposes. We recall briefly this method giving a precise error estimate.

Consider a point $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and the cell of the grid which contains it (see Fig. 5.4 for an example in 3D). Suppose that a function f is known in the 2^n vertexes of the cell and we want to compute the value $q(x)$ by a linear interpolation. The basic idea is to project the point x onto lower and lower dimensional subspaces until dimension 1 is reached. More precisely, choose a dimension (in Fig. 5.4 we chose x_1) and project the point x in that dimension on both sides of the cell finding the points P_1^1 and P_2^1. Then, choose a direction different from the first one (we chose x_2) and project the points P_1^1 and P_2^1 on the sides of the cell finding the points P_1^2, P_2^2, P_3^2 and P_4^2. Iterate the projection procedure

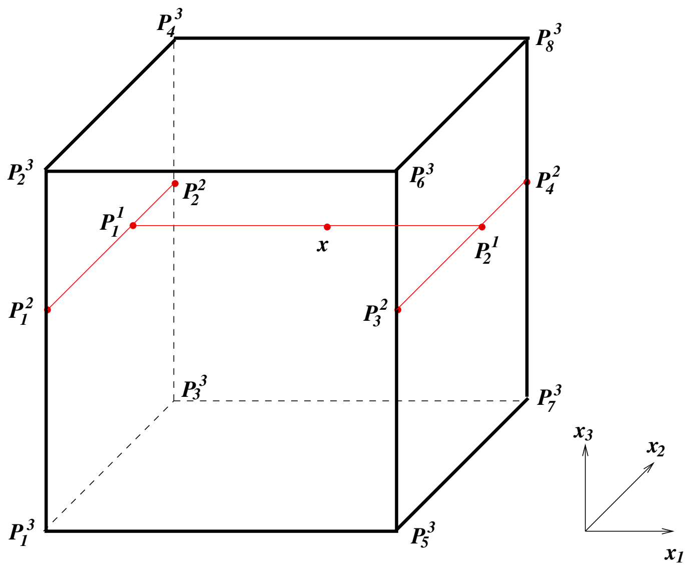


Figure 5.4: linear interpolation in 3D

$2^{n+1} - 2$ times in the same way until all vertexes of the cell are reached. At this stage the tree structure in Fig. 5.5 is computed from top to bottom. Now evaluate by *unidimensional* linear interpolations the values of q at the points $P_i^j\,, i = 1, \ldots, 2^n\,, \quad j = 1, \ldots, n$ in the reverse order with respect to that they was found (from bottom to top, see Fig. 5.6). This procedure leads to an approximate value of $f(x)$ obtained by $2^n - 1$ unidimensional linear interpolations.

$$\begin{tikzcd}
 & & & & & & x \arrow[dll] \arrow[drr] & & & & & & \\
 & & & & P_1^1 \arrow[dl] \arrow[dr] & & & & P_2^1 \arrow[dl] \arrow[dr] & & & & \\
 & & & P_1^2 & & P_2^2 & & P_3^2 & & P_4^2 & & & \\
 & & & \vdots & & \vdots & & \vdots & & \vdots & & & \\
 & P_1^{m-1} \arrow[dl] \arrow[dr] & \cdots & & & & & & & & \cdots & P_{2^{m-1}}^{m-1} \arrow[dl] \arrow[dr] & \\
P_1^m & & P_2^m & & & & & & & & P_{2^{m-1}}^m & & P_{2^m}^m
\end{tikzcd}$$

Figure 5.5: The tree structure for linear interpolation.

$$\begin{tikzcd}
q(P_1^m) \arrow[dr] & & q(P_2^m) \arrow[dl] & & & & & & & & q(P_{2^{m-1}}^m) \arrow[dr] & & q(P_{2^m}^m) \arrow[dl] \\
 & q(P_1^{m-1}) & \cdots & & & & & & & & \cdots & q(P_{2^{m-1}}^{m-1}) & \\
 & \vdots & & & & \vdots & & \vdots & & & & \vdots & \\
 & & & q(P_1^2) \arrow[dr] & & q(P_2^2) \arrow[dl] & & q(P_3^2) \arrow[dr] & & q(P_4^2) \arrow[dl] & & & \\
 & & & & q(P_1^1) \arrow[drr] & & & & q(P_2^1) \arrow[dll] & & & & \\
 & & & & & & q(x) & & & & & &
\end{tikzcd}$$

Figure 5.6: The tree for linear interpolation.

It is interesting to give a precise error estimates of this first order interpolation method.

Theorem 5.10 *Let $Q_n := [a_1, b_1] \times \ldots \times [a_n, b_n] \subset \mathbb{R}^n$ and $x = (x_1, \ldots, x_n)$.*
Assume $f \in C^2(Q_n; \mathbb{R})$ and let $q(x)$, $x \in Q_n$ be the approximate value of $f(x)$ obtained by the n-dimensional linear interpolation described above.

Then, the error $E(x) := f(x) - q(x)$ is bounded by

$$|E(x)| \leq \sum_{i=1}^{n} \frac{\Delta_i^2}{8} M_i \,, \qquad \textit{for all } x \in Q_n$$

where $M_i = \max\limits_{x \in Q_n} |\frac{\partial^2 f(x)}{\partial x_i^2}|$ and $\Delta_i = b_i - a_i$.

Proof. We proceed by induction.
If $n = 1$, by the basic theory of interpolation we know that for all $x \in [a_1, b_1]$ there exists $\xi_x \in [a_1, b_1]$ such that

$$|E(x)| = \left| f(x) - \left(\frac{b_1 - x}{b_1 - a_1} f(a_1) + \frac{x - a_1}{b_1 - a_1} f(b_1) \right) \right| = \frac{1}{2} |x - a_1| |x - b_1| |f''(\xi_x)|$$

and then

$$|E(x)| \leq \frac{(b_1 - a_1)^2}{8} M_1.$$

If $n = 2$ we have

$$|f(a_1, x_2) - q(a_1, x_2)| \leq \frac{(b_2 - a_2)^2}{8} M_2 \qquad \forall x_2 \in [a_2, b_2]$$

and

$$|f(b_1, x_2) - q(b_1, x_2)| \leq \frac{(b_2 - a_2)^2}{8} M_2 \qquad \forall x_2 \in [a_2, b_2].$$

Then

$$\begin{aligned}
|E| = |f(x_1, x_2) - q(x_1, x_2)| = \\
& \left| f(x_1, x_2) - \left(\frac{b_1 - x_1}{b_1 - a_1} q(a_1, x_2) + \frac{x_1 - a_1}{b_1 - a_1} q(b_1, x_2) \right) \right| = \\
& \left| f(x_1, x_2) - \left(\frac{b_1 - x_1}{b_1 - a_1} f(a_1, x_2) + \frac{x_1 - a_1}{b_1 - a_1} f(b_1, x_2) \right) + \right. \\
& \left(\frac{b_1 - x_1}{b_1 - a_1} f(a_1, x_2) + \frac{x_1 - a_1}{b_1 - a_1} f(b_1, x_2) \right) - \\
& \left. \left(\frac{b_1 - x_1}{b_1 - a_1} q(a_1, x_2) + \frac{x_1 - a_1}{b_1 - a_1} q(b_1, x_2) \right) \right| \leq \\
\frac{(b_1 - a_1)^2}{8} M_1 + \frac{b_1 - x_1}{b_1 - a_1} |f(a_1, x_2) - q(a_1, x_2)| + \frac{x_1 - a_1}{b_1 - a_1} |f(b_1, x_2) - q(b_1, x_2)| \leq \\
& \frac{(b_1 - a_1)^2}{8} M_1 + \frac{(b_2 - a_2)^2}{8} M_2.
\end{aligned}$$

In dimension n we have by the same arguments and by the induction hypothesis

$$|E| = |f(x_1,\dots,x_n) - q(x_1,\dots,x_n)| =$$

$$\left| f(x_1,\dots,x_n) - \left(\frac{b_1 - x_1}{b_1 - a_1} q(a_1,x_2,\dots,x_n) + \frac{x_1 - a_1}{b_1 - a_1} q(b_1,x_2,\dots,x_n) \right) \right| \le$$

$$\frac{(b_1 - a_1)^2}{8} M_1 + \frac{b_1 - x_1}{b_1 - a_1} |f(a_1,x_2,\dots,x_n) - q(a_1,x_2,\dots,x_n)| +$$

$$\frac{x_1 - a_1}{b_1 - a_1} |f(b_1,x_2,\dots,x_n) - q(b_1,x_2,\dots,x_n)| \le$$

$$\frac{(b_1 - a_1)^2}{8} M_1 + \sum_{i=2}^{n} \frac{(b_i - a_i)^2}{8} M_i.$$

■

5.3.2 Reducing the size of the problem

Since the problem has an high computational cost, reducing the size of the problem is a priority. Due to the state constraints, it seems not possible to use reduced coordinates or a similar approach (see also [75, 3]). In fact, using reduced coordinates we loose every information about the real positions of the two players, so that we can not detect when they touch the boundary of the domain (and then change the dynamics consequently). We tried to introduce a third fictitious coordinate in addition to those of reduced coordinates (Section 1.4.1) in order to detect when players touch the boundary (we are not really interested to their position at every time) but also this approach failed. In conclusion, we made the problem manageable by means of the high dimensional interpolation introduced in Section 5.3.1 and taking advantage of some symmetries of the problem.

Unidimensional Tag-Chase game

Although it seems not possible to describe the game in a reduced space due to the state constraints, we can simplify the computation taking into account the symmetries of the problem. In the unidimensional Tag-Chase game we can compute the solution in just half grid and recover the entire solution in the $n \times n$ grid at the end of computation. We will see that we can reduce the size of the the two-dimensional Tag-Chase game problem from n^4 to $n^4/4$. We first explain this technique for the unidimensional Tag-Chase game. Each player can move along a line in the interval $[-x_0, x_0]^2$, therefore the game is set in the square $[-x_0, x_0]^2$.
In Fig. 5.7 we show the level sets of the solution $T = -\log(1 - v)$ in the case $V_P = 2$,

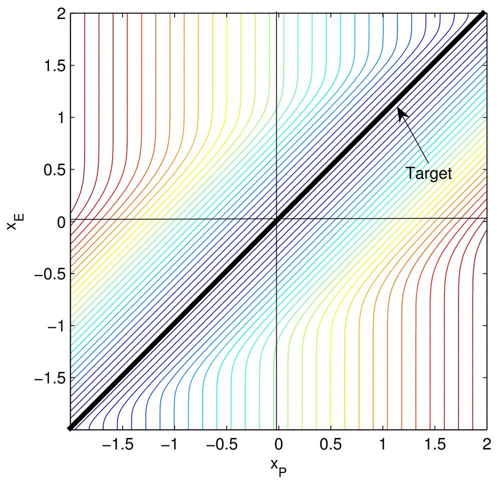


Figure 5.7: level sets of the solution $T = -\log(1-v)$ in the case $V_P = 2$, $V_E = 1$

$V_E = 1$, $x_0 = 2$. It is easy to see that

$$v(x_P, x_E) = v(-x_P, -x_E) \quad \text{for all } x_P, x_E \in [-x_0, x_0]$$

so that we can recover the entire solution either from the subsets $Q_{NW} = \{(x_P, x_E) : x_P \le x_E\}$ and $Q_W = \{(x_P, x_E) : x_P \le 0\}$. This corresponds to the fact that it is sufficient to compute the solution for all the initial positions of P and E in which P is on the left of E or P is in the left side of the domain $[-x_0, x_0]$ (see Fig. 5.8). There is an important difference between the two approaches. In fact the target

Figure 5.8: Two initial positions which correspond to the same value for v

$\mathcal{T} = \{(x_P, x_E) : x_P = x_E\}$ is entirely contained in Q_{NW} but not in Q_W. Since the target divides the domain $\Omega = [-x_0, x_0]^2$ in two parts and no characteristics can pass from one part to the other, all the optimal trajectories starting from Q_{NW} remains in Q_{NW}. This is clearly not true for Q_W. As a consequence, if we compute the solution only in Q_W this will be not correct because not all the usable part of the target is visible from the domain. Unfortunately the domain Q_{NW} has not a correspondence in the two-dimensional Tag-Chase game in which the target $\mathcal{T} = \{(x_P, y_P, x_E, y_E) : x_P = x_E,\ y_P = y_E\}$ does not divide the entire space $\Omega = ([-x_0, x_0] \times [-x_0, x_0])^2$ in two parts. In fact in this case the co-dimension of the target is strictly greater than 1. On the contrary, we will see in the next session that the domain Q_W has a natural generalization in the two-dimensional case.
For this reason it is preferable to localize the computation only in Q_W. In order to do this we adopt the following idea. First of all we choose n even. Then we compute

the approximation of v at the node (i, j), for $i = 1, \ldots, n/2$, $j = 1, \ldots, n$ via the numerical scheme (HJI$_h^k$–Ω). After every iteration we copy the line $(i = n/2,\ j = 1 : n)$ in $(i = n/2 + 1,\ j = n : 1)$ as a sort of "periodic boundary condition" for Q_W. In this way the information coming from the south-western part of the target can substitute the missing information needed by the north-western part of the domain.
When the algorithm reached the convergence we can easily recover the solution on all over the domain Ω.

Two-dimensional Tag-Chase game

As we did in the unidimensional case, we want to use the symmetries of the problem to avoid useless computation.
We assume that each player can move in a square so that the game is set in a four-dimensional hypercube. In this case we have more than one symmetry. In fact it easy to check that the following three inequalities hold (see Fig. 5.9)

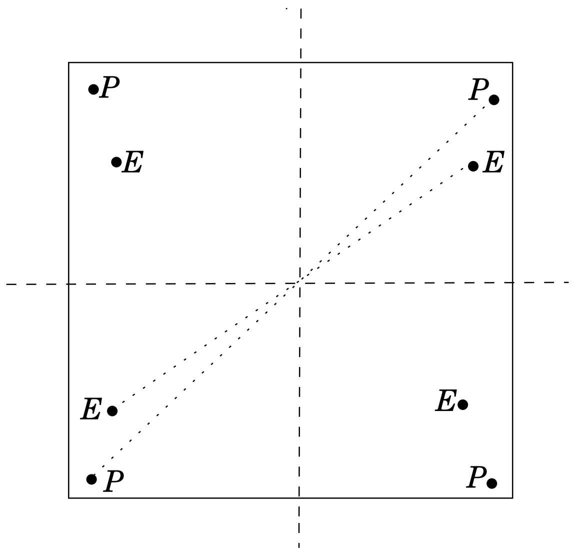


Figure 5.9: Four initial positions which correspond to the same value for v

$$v(x_P, y_P, x_E, y_E) = v(-x_P, -y_P, -x_E, -y_E)\,, \qquad \text{for all } (x_P, y_P, x_E, y_E) \in \Omega \tag{5.12}$$

$$v(x_P, y_P, x_E, y_E) = v(-x_P, y_P, -x_E, y_E) \tag{5.13}$$

$$v(x_P, y_P, x_E, y_E) = v(x_P, -y_P, x_E, -y_E). \tag{5.14}$$

We note that once we take into account the symmetry (5.12) we took into account automatically the symmetry in (5.13).
The following nested `for`'s take into account only the symmetry (5.14) and they allow to compute correctly the whole 4D matrix containing the grid nodes.

```
for i=1:n
  for j=1:n/2
    for j=1:n
      for j=1:n
        v(i,j,k,l)=SLscheme(...);
        v(i,n-j+1,k,n-l+1)=v(i,j,k,l);
```

Now we are ready to make use of symmetry (5.12) by means of the technique introduce for the unidimensional Tag-Chase game. We compute just half matrix corresponding to the indexes $i = 1, \ldots, n/2$ and after every iteration we copy the submatrix $(i = n/2,\ j = 1 : n,\ k = 1 : n,\ l = 1 : n)$ in $(i = n/2 + 1,\ j = n : 1,\ k = n : 1,\ l = n : 1)$ as boundary condition.
At the end of computation we easily recover the solution in the whole domain.

Remark 5.11 *In order to reduce the computational effort we also tried to adapt the fast search of the minimum we developed in Section 3.2.1. Unfortunately this seems to not be convenient in the case of games. In fact, in the 1-player case we have four cases corresponding to four orthants and every case is very simple to be solved while in the 2-player case we have sixteen cases each of which is divided in other subcases.*

Remark 5.12 *We ran a Fast Sweeping version of the one-dimensional Tag-Chase game. We noticed that no improvements about the number of iterations is achieved. This is probably due to the presence of state constraints so that the information first propagates from the target and then it comes back after hitting the boundary.*

5.4 Numerical experiments

In this section we solve equation (HJI-Ω) by the semi-Lagrangian numerical scheme (HJI$_h^k$–Ω) we developed in previous sections. We only deal with two-dimensional constrained Tag-Chase game (see Section 1.4.1 for the definition of this game). We consider the case $V_P > V_E$ as well as $V_P = V_E$ and $V_P < V_E$. Note that these two last cases appear for the first time in a numerical test. Although the equation related to the Tag-Chase game (1.51) is simpler than that of a generic game, we will use the scheme for a generic dynamics $f(x, a, b)$ (without splitting *max* and *min* operators) due to Remark 1.41.
The code is written in C++ and its parallel version has been obtained by means of OpenMP directives. The algorithm ran on a PC equipped with a processor Intel Pentium dual core 2×2.80 GHz, 1 GB RAM and on an IBM system p5 575 equipped with 8 processors Power5 at 1.9 GHz and 32 GB RAM located at CASPUR[1].

Notations and choice of parameters

We denote by n the number of nodes in each dimension. This number is clearly associated to the space step Δx (or k).
We denote by n_c the number of admissible directions/controls for each player *i.e.* we

[1]Consorzio interuniversitario per le Applicazioni di Supercalcolo per Università e Ricerca, `www.caspur.it`.

discretize the unit ball $B(0,1)$ with n_c points. We restrict the discretization to the boundary $\partial B(0,1)$ and in some cases we add the central point (in this case we denote the number of directions by $n_c^- +1$ where $n_c^- = n_c - 1$).
We always use a uniform structured grid with four-dimensional cells of volume Δx^4 and we choose the (fictitious) time step h such that $\|hf(x,a,b)\| \leq \Delta x$ for all x, a, b (so that the interpolation is made in the neighboring cells of the considered point).
We introduce the following stopping criterion for the fixed point iterations (1.61),

$$\|V^{(n+1)} - V^{(n)}\|_\infty \leq \varepsilon\,, \qquad \varepsilon > 0.$$

We remark that the quality of the approximate solution depends on h, k and also (strictly) on the ratio h/k (see [6]).
The real game is played in a square $[-2,2]^2$ so the problem is set in $\overline{\Omega} = [-2,2]^4$.
The target is 3 pixels thick.
Once we computed the approximate solution, we recover the optimal trajectories (see Section 1.2.3). At this stage we have to choose the time step Δt in order to discretize the dynamical system by Euler scheme. It should be noted that this parameter can in general different from the (fictitious) time step h chosen for the computation of the value function and this is true also for the number of controls n_c. Moreover, computing optimal trajectory requires the evaluation of the argminmax which is done again choosing a value for h, and this value can be in principle different from that used in first computation.
We denote by $v(x_P, y_P, x_E, y_E)$ the approximate value function and by $T(x_P, y_P, x_E, y_E) = -\ln(1 - v(x_P, y_P, x_E, y_E))$.

5.4.1 Case $V_P > V_E$

The case $V_P > V_E$ is the classical one and it was already studied by Alziary de Roquefort [3]. In this case the value function v is continuous and all theoretical results we presented in this chapter hold true.

Test 1
We choose $\varepsilon = 10^{-3}$, $V_P = 2$, $V_E = 1$, $n = 50$, $n_c = 48 + 1$. Convergence was reached in 85 iterations. The CPU time (IBM - 8 procs) was 17h 36m 16s, the wallclock time was 2h 47m 37s. Fig. 5.10 shows the value function $T(0, 0, x_E, y_E)$ and its level sets (we fix the Pursuer's position at the origin). It is immediately seen that if the distance between P and E is greater than $V_P - V_E = 1$ then the state constraints have a great influence on the solution. Moreover it is clear that the presence of state constraints gives an advantage to the Pursuer.
Fig. 5.11 shows four optimal trajectories corresponding to the starting points

$$\begin{cases} P = (-1,0) \\ E = (0,0) \end{cases} \qquad \begin{cases} P = (-2,-2) \\ E = (1,0.7) \end{cases} \qquad \begin{cases} P = (-1.8,-1.8) \\ E = (0.5,-1.6) \end{cases} \qquad \begin{cases} P = (-1.8,-1.8) \\ E = (0.5,-1.8) \end{cases}$$

We always denote by black crosses the Pursuer's position and by red squares the Evader's position.

Test 2
The second test is just to compare the CPU time on the two architectures we have at our

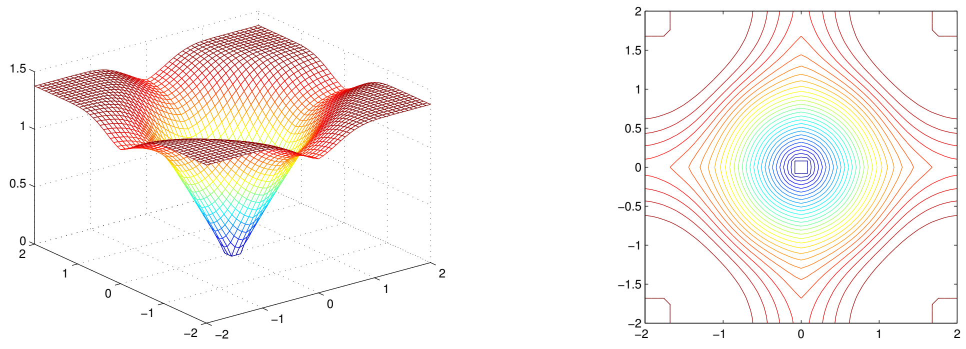


Figure 5.10: Test 1: value function $T(0, 0, x_E, y_E)$ (left) and its level sets (right)

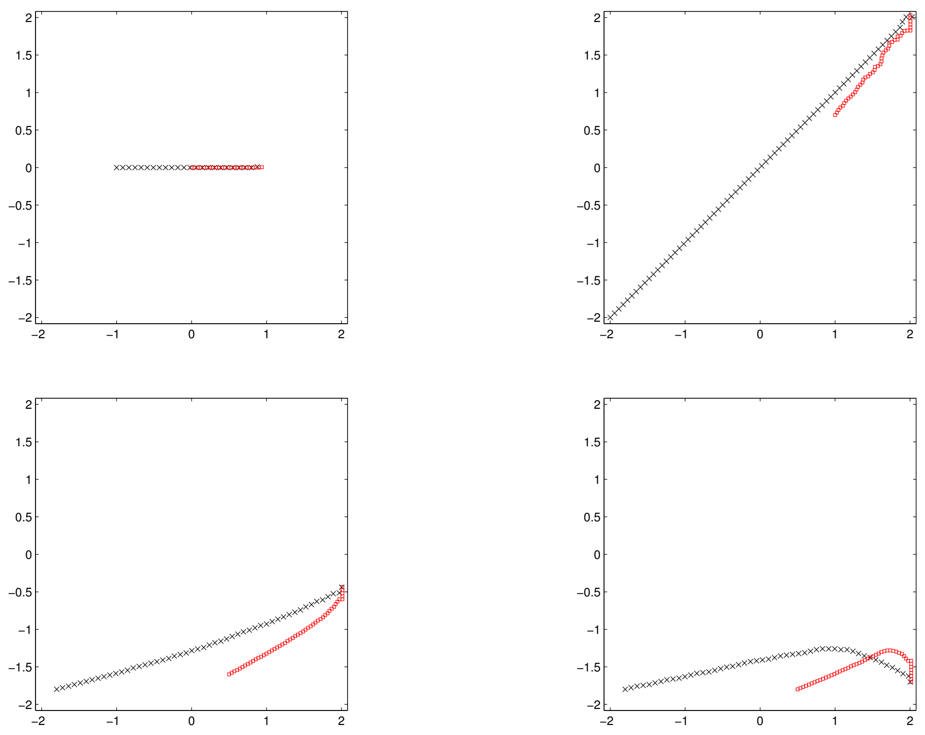


Figure 5.11: optimal trajectories for Test 1. Black crosses: Pursuer. Red squares: Evader

disposition. It is interesting to test the new dual core processors in order to understand how much they can be useful in parallel scientific computing. They are indeed conceived mainly to deal with distributed computing or simply multitasking. The performances of the parallel code are measured in terms of two well known parameters, the *speed-up* and the *efficiency*. Let T_{ser} and T_{par} be the times corresponding respectively to the execution

architecture	wallclock time	speed-up	efficiency
IBM serial	26m 47s	-	-
IBM 2 procs	14m 19s	1.87	0.93
IBM 4 procs	8m 09s	3.29	0.82
IBM 8 procs	4m 09s	6.45	0.81
PC dual core, serial	1h 08m 44s	-	-
PC dual core, parallel	34m 51s	1.97	0.99

Table 5.1: CPU time for Test 2

of the serial and parallel algorithms over n_p processors. We define

$$\textit{speed-up} := \frac{T_{ser}}{T_{par}} \qquad \text{and} \qquad \textit{efficiency} := \frac{\textit{speed-up}}{n_p}.$$

Note that an ideal parallel algorithm would have *speed-up* $= n_p$ and *efficiency* $= 1$. Table 5.1 shows the wallclock time, the *speed-up* and the *efficiency* for the following choice of parameters: $\varepsilon = 10^{-5}$, $V_P = 2$, $V_E = 1$, $n = 26$, $n_c = 36 + 1$.

Test 3

In this test the domain has a square hole in the center. The side of the square is 1.06. We choose $\varepsilon = 10^{-4}$, $V_P = 2$, $V_E = 1$, $n = 50$, $n_c = 48 + 1$. Convergence was reached in 109 iterations. The CPU time (IBM - 8 procs) was 1d 00h 34m 18s, the wallclock time was 3h 54m 30s. Fig. 5.12 shows the value function $T(-1.5, -1.5, x_E, y_E)$.

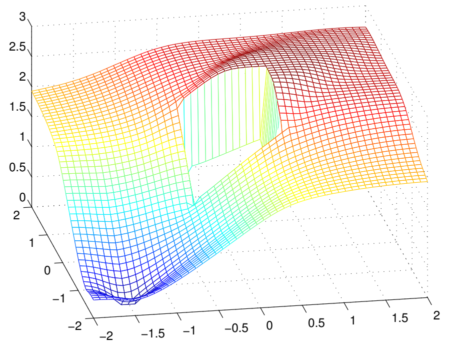


Figure 5.12: Test 3: value function $T(-1.5, -1.5, x_E, y_E)$

Fig. 5.13 shows two optimal trajectories corresponding to the starting points

$$\begin{cases} P = (-1.9, -1.9) \\ E = (1.9, 1.9) \end{cases} \qquad \begin{cases} P = (-1.9, 0) \\ E = (1, 0) \end{cases}$$

It interesting to note that in both cases the Evader waits until the Pursuer decides if he wants to skirt around the obstacle clockwise or counterclockwise. After that, the Evader goes in the opposite direction. If both players touch the obstacle, they run around it until

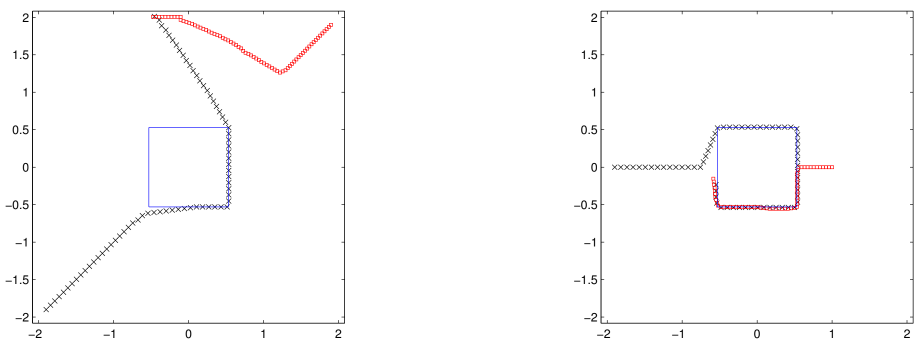


Figure 5.13: optimal trajectories for Test 3. Black crosses: Pursuer. Red squares: Evader

the capture occurs.

Test 4
In this test the domain has a circular hole in the center. The radius r of the circle is $7\Delta x$. We choose $\varepsilon = 10^{-4}$, $V_P = 2$, $V_E = 1$, $n = 50$, $n_c = 48 + 1$. Convergence was reached in 108 iterations. The CPU time (IBM - 8 procs) was 1d 17h 27m 43s, the wallclock time was 6h 39m 00s. Note that handling with a circular obstacle inside the domain of computation is not easy as in the previous test where the boundary of the obstacle matches with the lattice. We adopt the following procedure. First of all, we define the radius r of the circle as a multiple of the space step Δx. Then, at every node $(P{=}(i,j), E{=}(k,l))$, we compute the distance d_{PO} (resp., d_{EO}) between P (resp., E) and the center of the domain. Let us focus on P, E being treated in the same way. If $r \leq d_{PO} < r + \Delta x$ then we say that P in on the "numerical boundary" of the circle. The exterior normal vector $\eta(i,j)$ to the (numerical) boundary of the circle is simply given by the coordinates of the node (i,j), so that we can easily compute the scalar product $\eta \cdot a$ where a is the desired direction of P. If the scalar product is negative, we label the direction a as *not admissible.*
Fig. 5.14 shows two optimal trajectories corresponding to the starting points

$$\begin{cases} P = (-1.9, -1.9) \\ E = (1.9, 1.9) \end{cases} \qquad \begin{cases} P = (-0.6, 0) \\ E = (1, 0.4) \end{cases}$$

The behavior of the optimal trajectories is similar to the previous Test.

5.4.2 Case $V_P = V_E$

When $V_P = V_E$ the value function v is discontinuous on $\partial\mathcal{T}$. In this case no convergence results are known, nevertheless the numerical scheme seems to work very well. We remind that results in Section 5.2 guarantee that $v < 1$ (the capture always occurs). This is confirmed by the following test.

Test 5
We choose $\varepsilon = 10^{-3}$, $V_P = 1$, $V_E = 1$, $n = 50$, $n_c = 36$. Convergence was reached in 66 iterations. Fig. 5.15 shows the value function $T(0, 0, x_E, y_E)$ and its level sets. Fig.

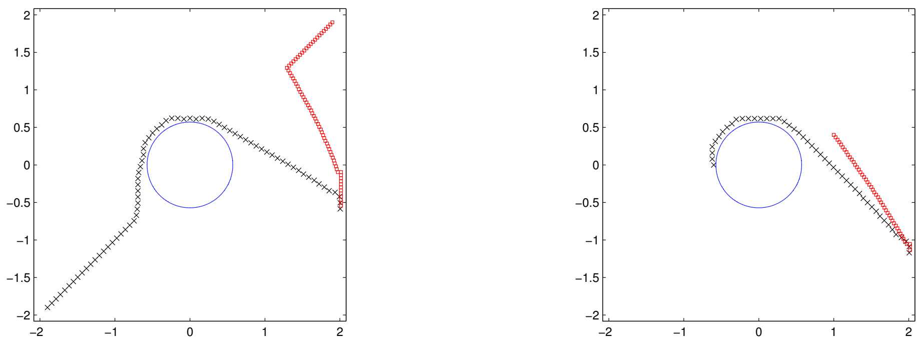


Figure 5.14: optimal trajectories for Test 4. Black crosses: Pursuer. Red squares: Evader

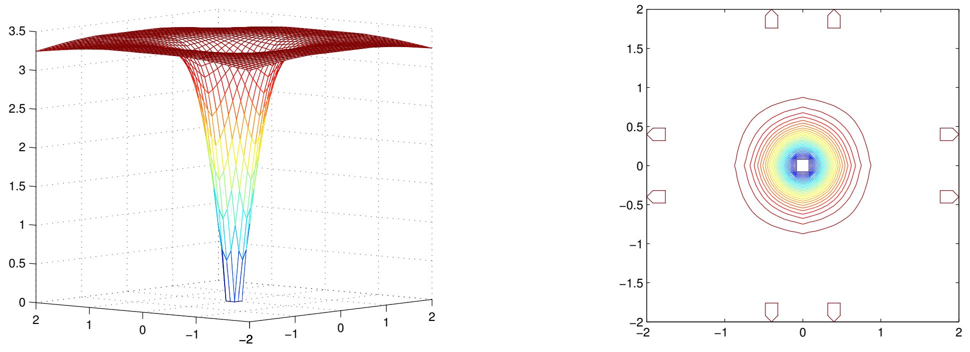


Figure 5.15: Test 5: value function $T(0, 0, x_E, y_E)$ (left) and its level sets (right)

5.16 shows the value function $T(1.15, 1.15, x_E, y_E)$ and its level sets. Fig. 5.17 shows four

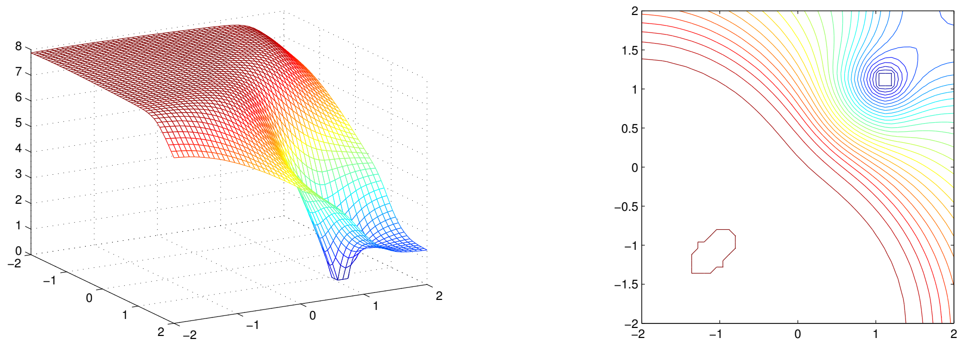


Figure 5.16: Test 5: value function $T(1.15, 1.15, x_E, y_E)$ (left) and its level sets (right)

optimal trajectories corresponding to the starting points

$$\begin{cases} P=(0,1) \\ E=(0,0) \end{cases} \quad \begin{cases} P=(1,1.5) \\ E=(-0.5,0) \end{cases} \quad \begin{cases} P=(1.3,1.8) \\ E=(0,0) \end{cases} \quad \begin{cases} P=(-1.9,-1.9) \\ E=(-1.7,-1.9) \end{cases}$$

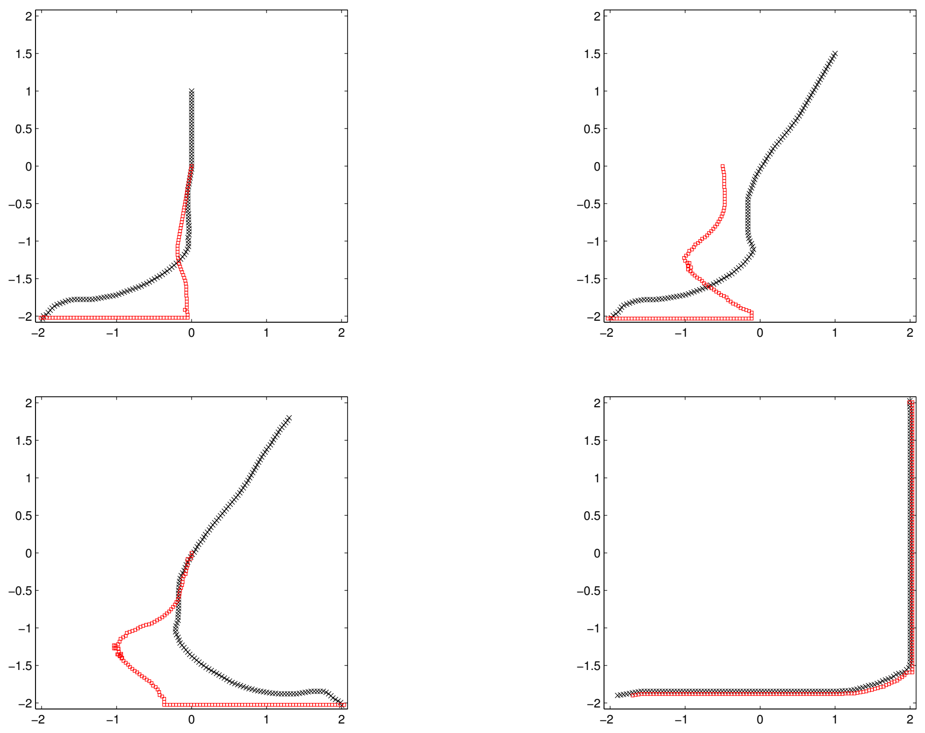


Figure 5.17: optimal trajectories for Test 5

Test 6

In this test the domain has a circular hole in the center. The radius of the circle is $7\Delta x$. Since the domain is no more convex, we have no guarantee that the time of capture is finite. Numerical results show that the value function v is equal to 1 in a large part of the domain.

It is well known that it is not possible to recover the optimal trajectories whenever $v = 1$ $(T = \infty)$ since from that regions capture never happens. Indeed, if $V_P \leq V_E$ the approximate solution shows a bizarre behavior. Even if $v < 1$, in some cases the computed optimal trajectories tend to stable trajectories such that P never reaches E. Although this is due to some numerical error, these trajectories are extremely realistic so they give to us a guess about the optimal strategies of the players in the case E wins. In this Test (and others below) we show this behavior.

We choose $\varepsilon = 10^{-4}$, $V_P = 1$, $V_E = 1$, $n = 50$, $n_c = 48 + 1$. Convergence was reached in 94 iterations. The CPU time (IBM - 8 procs) was 1d 12h 05m 22s. Fig. 5.18 shows one

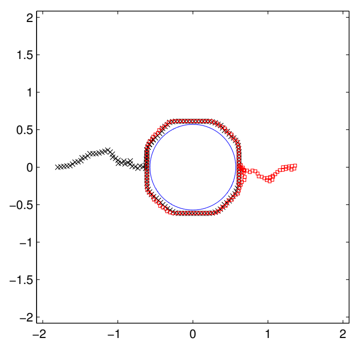


Figure 5.18: optimal trajectory for Test 6

optimal trajectory corresponding to the starting point

$$\begin{cases} P = (-1.8, 0) \\ E = (1.2, 0) \end{cases}$$

In this example, the capture did not happen after 150 time steps. The asymptotic behavior of the trajectory is stable since once the two players reached the internal circle, they run around it forever. It should be noted that, at the beginning of the game, E leaves the time goes by in order to touch the boundary of the circle exactly when P touches it.
This strange behavior urges us to invent some method to compute rigorously the trajectories corresponding to the E's win, in order to confirm our guess. Maybe we can do it considering the time-dependent problem (so that we work in $\mathbb{R}^5$ as Alziary de Roquefort does [3]). This allows to choose a time-dependent velocity $V_E(t)$ such that it is very fast for $0 \leq t < \bar{t}$ (capture impossible) and very slow for $t > \bar{t}$ (capture unavoidable). For such a velocity we have $v < 1$ so we can compute optimal trajectories but, for $0 \leq t < \bar{t}$, E will attempt to maintain a trajectory such that capture does not happen.

5.4.3 Case $V_P < V_E$

If $V_P < V_E$ the value function v is discontinuous on $\partial\mathcal{T}$. Moreover, we have no guarantee that the time of capture is finite. Numerical results show that the value function v is equal to 1 in a large part of the domain.

Test 7
We choose $\varepsilon = 10^{-3}$, $V_P = 1$, $V_E = 1.25$, $n = 50$, $n_c = 48 + 1$. Convergence was reached in 53 iterations. The CPU time (IBM - 8 procs) was 12h 43m 02s, the wallclock time was 2h 18h 06s.
Fig. 5.19 shows the value function $T(-1, -1, x_E, y_E)$ and its level sets.
Fig. 5.20 shows two optimal trajectories corresponding to the starting points

$$\begin{cases} P = (-1, -1) \\ E = (-1, 1) \end{cases} \qquad \begin{cases} P = (-1, -1) \\ E = (-0.5, -0.5) \end{cases}$$

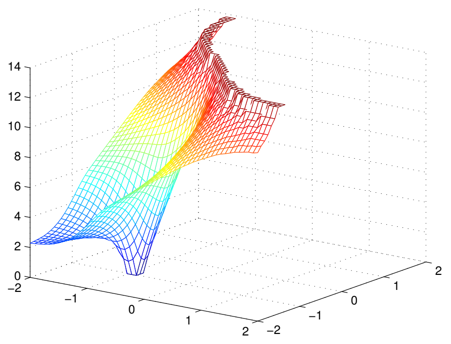
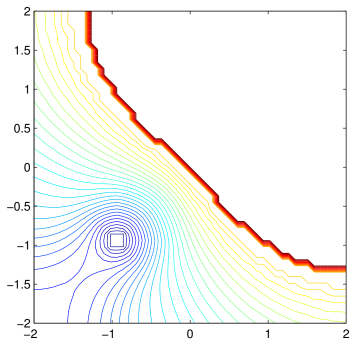



Figure 5.19: Test 7: value function $T(-1,-1,x_E,y_E)$ (left) and its level sets (right)

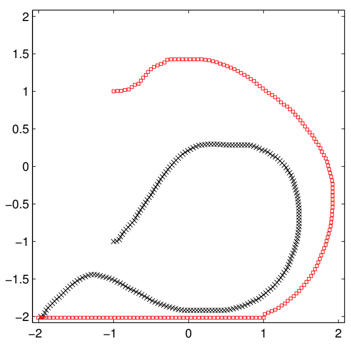
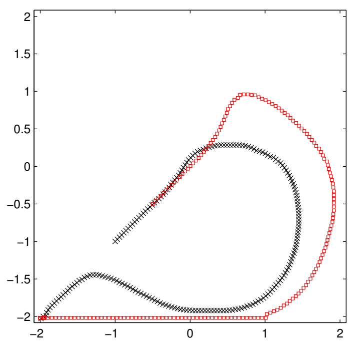



Figure 5.20: optimal trajectories for Test 7

Note that the Pursuer approaches the corner in which capture occurs along the diagonal of the square in order to block off the Evader's escape.

Test 8
We choose $\varepsilon = 10^{-4}$, $V_P = 1$, $V_E = 1.5$, $n = 50$, $n_c = 36$. Convergence was reached in 65 iterations. The CPU time (IBM - 8 procs) was 15h 48m 46s, the wallclock time was 2h 30m 19s.
Fig. 5.21 shows two optimal trajectories corresponding to the starting points

$$\begin{cases} P = (0.5, 0.5) \\ E = (1.5, 1.5) \end{cases} \qquad \begin{cases} P = (0, -0.8) \\ E = (-0.3, -1.3) \end{cases}$$

In the example on the left, E makes believe he wants to be caught in the upper-left corner but after a while he turns on the right towards the upper-right corner. In the example on the right the capture did not happen after 2000 time steps (see Test 6) and the asymptotic behavior of the trajectories is quite stable. Moreover, we note that the ratio between the two radii of the circles are about 1.5 as the ratio between the velocities of the two players (and then they complete a rotation in the same time). This result must be compared with

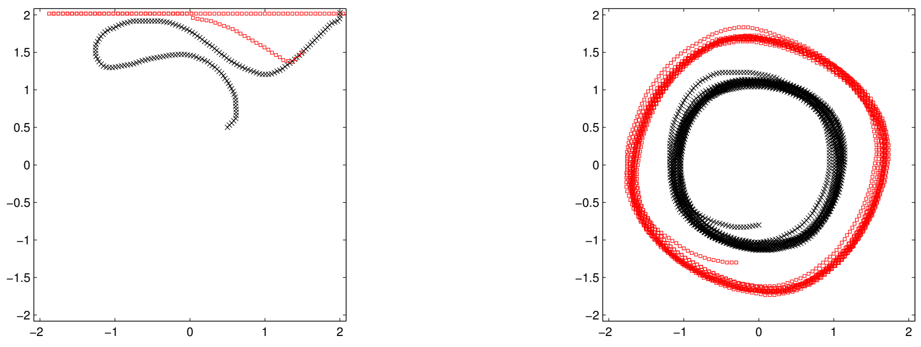


Figure 5.21: optimal trajectories for Test 8

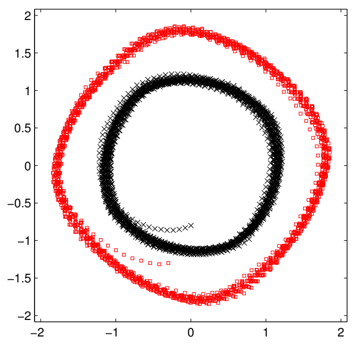


Figure 5.22: optimal trajectory for Test 9

the following test in which the velocities are different but the ratio is again 1.5.

Test 9
We choose $\varepsilon = 10^{-3}$, $V_P = 2$, $V_E = 3$, $n = 50$, $n_c = 48 + 1$. Convergence was reached in 44 iterations. The CPU time (IBM - 8 procs) was 9h 26m 42s, the wallclock time was 1h 29m 37s. Fig. 5.22 shows the optimal trajectory corresponding to the starting point

$$\begin{cases} P = (0, -0.8) \\ E = (-0.3, -1.3) \end{cases}$$

The behavior of the asymptotic trajectory are very similar to the previous one.

Bibliography

[1] O. Alvarez, P. Hoch, Y. Le Bouar, R. Monneau, *Dislocation dynamics: short time existence and uniqueness of the solution*, Archive for Rational Mechanics and Analysis, **181** (2006), 449-504.

[2] B. Alziary de Roquefort, *Jeux différentiels et approximation numérique de fonctions valeur,* 1^{re} *partie: étude théorique*, RAIRO Modél. Math. Anal. Numér., **25** (1991), 517-533.

[3] B. Alziary de Roquefort, *Jeux différentiels et approximation numérique de fonctions valeur,* 2^{e} *partie: étude numérique*, RAIRO Modél. Math. Anal. Numér., **25** (1991), 535-560.

[4] J. P. Aubin, *Viability theory*, Birkhäuser, 1992.

[5] M. Bardi, *Some applications of viscosity solutions to optimal control and differential games*, in I. Capuzzo Dolcetta, P.-L. Lions (editors), "Viscosity solutions and applications (Montecatini Terme, 1995)", Lecture Notes in Mathematics **1660**, 44-97, Springer, Berlin, 1997.

[6] M. Bardi, S. Bottacin, M. Falcone, *Convergence of Discrete Schemes for Discontinuous Value Functions of Pursuit-Evasion Games*, in G. J. Olsder (editor), "New trends in Dynamic Games and Applications", Annals of the International Society of Dynamic Games, **3**, 273-304, Birkhäuser, 1995.

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

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

[9] M. Bardi, M. Falcone, P. Soravia, *Numerical methods for Pursuit-Evasion games via viscosity solutions*, in M. Bardi, T. Parthasarathy and T. E. S. Raghavan (editors), "Stochastic and differential games: theory and numerical methods", Annals of the International Society of Dynamic Games **4**, 105-175, Birkhäuser, 1999.

[10] M. Bardi, M. Falcone, P. Soravia, *Fully discrete schemes for the value function of pursuit-evasion games*, in T. Başar and A. Haurie (editors), "Advances in Dynamic Games and Applications", Annals of the International Society of Dynamic Games **1**, 89-105, Birkhäuser, 1994.

[11] M. Bardi, S. Koike, P. Soravia, *Pursuit-evasion games with state constraints: dynamic programming and discrete-time approximations*, Discrete Contin. Dynam. Systems, **6** (2000), 361-380.

[12] M. Bardi, P. Soravia, *A PDE framework for games of pursuit-evasion type*, in T. Başar, P. Bernhard (editors), "Differential Games and Applications", Lecture Notes in Control and Information Sciences **119**, 62-71, Springer-Verlag, 1989.

[13] M. Bardi, P. Soravia, *Hamilton-Jacobi equations with a singular boundary condition on a free boundary and applications to differential games*, Trans. Amer. Math. Soc., **325** (1991), 205-229.

[14] M. Bardi, P. Soravia, *Approximation of differential games of pursuit-evasion by discrete-time games*, in R. P. Hamalainen, H. K. Ehtamo (editors), "Differential Games. Developments in Modelling and Computations", Lecture Notes in Control and Information Sciences **156**, 131-143, Springer-Verlag, 1991.

[15] G. Barles, *Solutions de viscosité des equations de Hamilton-Jacobi*, Springer, Berlin, 1994.

[16] R. Bellman, *Introduction to the mathematical theory of control processes*, Vol. 2. Academic Press, New York, 1971.

[17] J. V. Breakwell, *Time-optimal pursuit inside a circle*, Differential Games and Applications (Sophia-Antipolis, 1988), Lecture Notes in Control and Inform. Sci. **119**, 72-85, Springer, Berlin, 1989.

[18] F. Camilli, M. Falcone, *Approximation of optimal control problems with state constraints: estimates and applications*, in B. S. Mordukhovic, H. J. Sussman (editors), "Nonsmooth analysis and geometric methods in deterministic optimal control", IMA Volumes in Applied Mathematics **78**, 23-57, Springer Verlag, 1996.

[19] F. Camilli, A. Siconolfi, *Discontinuous solutions of a Hamilton-Jacobi equation with infinite speed of propagation*, SIAM Journal on Mathematical Analysis , **28** (1997), 1420-1445.

[20] P. Cardaliaguet, *A differential game with two players and one target*, SIAM J. Control Optim., **34** (1996), 1441-1460.

[21] P. Cardaliaguet, M. Quincampoix, P. Saint-Pierre, *Differential games with state-constraints*, Annals of the International Society of Dynamic Games, St. Petersburg 2002.

[22] P. Cardaliaguet, M. Quincampoix, P. Saint-Pierre, *Pursuit differential games with state constraints*, SIAM J. Control Optim., **39** (2001), 1615-1632.

[23] P. Cardaliaguet, M. Quincampoix, P. Saint-Pierre, *Numerical schemes for discontinuous value functions of optimal control. Set-valued analysis in control theory*, Set-Valued Anal., **8** (2000), 111-126.

[24] P. Cardaliaguet, M. Quincampoix, P. Saint-Pierre, *Set valued numerical analysis for optimal control and differential games*, in M. Bardi, T. Parthasarathy and T. E. S. Raghavan (editors), "Stochastic and differential games: theory and numerical methods", Annals of the International Society of Dynamic Games **4**, 177-247, Birkhäuser, 1999.

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

[26] E. Carlini, M. Falcone, R. Ferretti, *An efficient algorithm for Hamilton-Jacobi equations in high dimension*, Comput. Visual. Sci., **7** (2004), 15-29.

[27] E. Carlini, M. Falcone, N. Forcadel, R. Monneau, *Convergence of a generalized Fast Marching method for a non-convex eikonal equation*, Preprint, 2006, Cermics, École Nationale des Ponts et Chaussées - Cité Descartes - Champs sur Marne 77455 Marne la Vallée Cedex 2, France.

[28] D. L. Chopp, *Some improvements of the fast marching method*, SIAM J. Sci. Comput., **23** (2001), 230-244.

[29] F. Courteille, *Vision monoculaire: contributions théoriques et application à la numérisation des documents*, Ph.D. thesis, Université Paul Sabatier, Toulouse, France, 2006.

[30] F. Courteille, A. Crouzil, J.-D. Durou, P. Gurdjos, *Shape from shading en conditions réalistes d'acquisition photographique*, in Actes du $14^{ème}$ Congrès Francophone de Reconnaissance des Formes et Intelligence Artificielle (volume II), 925-934, Toulouse, France, 2004.

[31] F. Courteille, A. Crouzil, J.-D. Durou, P. Gurdjos, *Towards shape from shading under realistic photographic conditions*, in Proceedings of the 17^{th} International Conference on Pattern Recognition (volume II), 277-280, Cambridge, UK, 2004.

[32] F. Courteille, A. Crouzil, J.-D. Durou, P. Gurdjos, *Shape from shading for the digitization of curved documents*, Machine Vision and Applications, 2006 (to appear).

[33] M. G. Crandall, P.-L. Lions, *Two approximations of solutions of Hamilton-Jacobi equations*, Math. Comp. **43** (1984), 1-19.

[34] M. G. Crandall, P.-L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., **277** (1983), 1-42.

[35] E. Cristiani, *Algoritmi veloci per l'evoluzione dei fronti*, Tesi di Laurea, Dipartimento di Matematica, Università di Roma "La Sapienza", Italy, 2003.

[36] E. Cristiani, J.-D. Durou, M. Falcone, *The convex-concave ambiguity in Perspective Shape-from-Shading models*, submitted to Proceedings of SSVM 2007.

[37] E. Cristiani, M. Falcone, *A fully-discrete scheme for the value function of differential games with state constraints*, submitted to Proceedings of the 12^{th} International Symposium on Dynamic Games and Applications, Sophia Antipolis, 2006.

[38] E. Cristiani, M. Falcone, *Fast semi-Lagrangian schemes for the Eikonal equation and applications*, preprint, Dipartimento di Matematica, 2005. Submitted to SIAM J. Numer. Anal., under revision. Preprint server: `http://cpde.iac.rm.cnr.it/`.

[39] E. Cristiani, M. Falcone, *A Fast Marching Method for Pursuit-Evasion Games*, 6-pages abstract published electronically in Proceedings of SIMAI 2006, Baia Samuele, Ragusa (Italy), May 22-26, 2006.

[40] E. Cristiani, M. Falcone, A. Seghini, *Numerical solution of the perspective Shape from Shading problem*, in Proceedings of "Control Systems: Theory, Numerics and Applications" PoS (CSTNA2005) 008, `http://pos.sissa.it/`.

[41] P. Danielsson, *Euclidean distance mapping*, Computer Graphics and Image Processing, **14** (1980), 227-248.

[42] M. C. Delfour, J.-P. Zolésio, *Oriented distance function and its evolution equation for initial sets with thin boundary*, SIAM J. Control Optim., **42** (2004), 2286-2304 .

[43] J.-D. Durou, M. Falcone, M. Sagona, *A survey of numerical methods for Shape from Shading*, Rapport IRIT 2004-2-R, Université Paul Sabatier, Toulouse, France, 2004. Submitted to Computer Vision and Image Understanding, under revision.

[44] R. J. Elliott, N. J. Kalton, *The existence of value in differential games*, Mem. Amer. Math. Soc. **126**, 1972.

[45] L. C. Evans, *Partial differential equations*, Graduate Studies in Mathematics **19**, American Mathematical Society, 1998.

[46] L. C. Evans, P. E. Souganidis, *Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations*, Indiana University Mathematics Journal, **33** (1984), 773-797.

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

[48] M. Falcone, *The minimum time problem and its applications to front propagation*, in A. Visintin e G. Buttazzo (editors), "Motion by mean curvature and related topics", De Gruyter Verlag, Berlino, 1994.

[49] M. Falcone, *Numerical solution of dynamic programming equations*, Appendix A in [7].

[50] M. Falcone, *Some remarks on the synthesis of feedback controls via numerical methods*, in J. L. Menaldi, E. Rofman, A. Sulem (editors), "Optimal Control and Partial Differential Equations", IOS Press, 2001, 456-465.

[51] M. Falcone, R. Ferretti, *Discrete time high-order schemes for viscosity solutions of Hamilton-Jacobi-Bellman equations*, Numerische Mathematik, **67** (1994), 315-344.

[52] M. Falcone, R. Ferretti, *Semi-Lagrangian schemes for Hamilton-Jacobi equations, discrete representation formulae and Godunov methods*, Journal of Computational Physics, **175** (2002), 559-575.

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

[54] R. Fedkiw, S. Osher, *Level set methods and dynamic implicit surfaces*, Applied Mathematical Sciences **153**, Springer-Verlag, New York, 2003.

[55] J. Gomes, O. Faugeras, *Reconciling distance functions and level sets*, J. Visual Com. and Image Representation, **11** (2000), 209-223.

[56] B. K. P. Horn, *Shape from Shading: a method for obtaining the shape of a smooth opaque object from one view*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 1970.

[57] B. K. P. Horn, M. J. Brooks (editors), *Shape from Shading*, MIT Press, 1989.

[58] R. Isaacs, *Differential Games*, John Wiley and Sons, 1965.

[59] H. Ishii, S. Koike, *A new formulation of state constraint problems for first-order PDEs*, SIAM J. Control Optim., **34** (1996), 554-571.

[60] H. Ishii, M. Ramaswamy, *Uniqueness results for a class of Hamilton-Jacobi equations with singular coefficients*, Communications in Partial Differential Equations, **20** (1995), 2187-2213.

[61] C.-Y. Kao, S. Osher, J. Qian, *Lax-Friedrichs sweeping scheme for static Hamilton-Jacobi equations*, J. Comput. Phys., **196** (2004), 367-391.

[62] C.-Y. Kao, S. Osher, Y.-H. Tsai, *Fast sweeping methods for static Hamilton-Jacobi equations*, SIAM J. Numer. Anal., **42** (2005), 2612-2632 (electronic).

[63] S. Kim, *An O(N) level set method for eikonal equations*, SIAM J. Sci. Comput., **22** (2001), 2178-2193.

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

[65] R. Kimmel, J. A. Sethian, *Optimal algorithm for shape from shading and path planning*, Journal of Mathematical Imaging and Vision, **14** (2001), 237-244.

[66] S. Koike, *On the state constraint problem for differential games*, Indiana University Mathematics Journal, **44** (1995), 467-487.

[67] P.-L. Lions, *Generalized solutions of Hamilton-Jacobi equations*, Research notes in mathematics **69**, Pitman, Boston, Mass.-London, 1982.

[68] P.-L. Lions, E. Rouy, A. Tourin, *Shape from shading, viscosity solution and edges*, Numerische Mathematik, **64** (1993), 323-353.

[69] B. O'Neill, *Elementary differential geometry*, Academic Press, New York and London, 1966.

[70] T. Okatani, K. Deguchi, *Reconstructing Shape from Shading with a point light source at the projection center: shape reconstruction from an endoscope image*, in Proceedings of the 13^{th} International Conference on Pattern Recognition (volume I), 830-834, Wien, Austria, 1996.

[71] S. Osher, R. Fedkiw, *Level sets and dynamic implicit surfaces*, Springer Verlag, 2002.

[72] S. Osher, J. A. Sethian, *Fronts propagation with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., **79** (1988), 12-49.

[73] V. S. Patsko, V. L. Turova, *Numerical study of differential games with the homicidal chauffeur dynamics*, Scientific Report of the Russian Academy of Science (Ural Branch, Ekaterinburg).

[74] H. J. Pesch, *Solving optimal control and Pursuit-Evasion game problems of high complexity*, Computational Optimal Control (Munich, 1992) 43-61, Internat. Ser. Numer. Math. **115**, Birkhäuser, 1994.

[75] H. J. Pesch, I. Gabler, S. Miesbach, M. H. Breitner, *Synthesis of optimal strategies for differential games by neural networks*, in G. J. Olsder (editor), "New trends in Dynamic Games and Applications", Annals of the International Society of Dynamic Games **3**, 111-141, Birkhäuser, 1995.

[76] E. Prados, *Application of the theory of the viscosity solutions to the Shape From Shading problem*, Ph.D. thesis, Univ. of Nice - Sophia Antipolis, France, 2004.

[77] E. Prados, O. Faugeras, *"Perspective Shape from Shading" and viscosity solutions*, in Proceedings of the 9^{th} IEEE International Conference on Computer Vision (volume II), 826-831, Nice, France, 2003.

[78] E. Prados, O. Faugeras, F. Camilli, *Shape from Shading: a well-posed problem?*, Rapport de Recherche n. 5297. INRIA Sophia Antipolis, 2004.

[79] E. Prados, S. Soatto, *Méthode de "Fast Marching" générique pour le "Shape from Shading"*, in Actes du $15^{ème}$ Congrès de Reconnaissance des Formes et Intelligence Artificielle, Tours, France, 2006.

[80] J. Qian, Y.-T. Zhang, H. Zhao, *Fast sweeping methods for eikonal equations on triangular meshes*, UCLA CAM05-07, under revision.

[81] E. Rouy, A. Tourin, *A viscosity solutions approach to Shape-from-Shading*, SIAM J. Numer. Anal., **29** (1992), 867-884.

[82] M. Sagona, *Numerical methods for degenerate eikonal type equation and applications*, Ph.D. thesis, Dipartimento di Matematica, Università di Napoli "Federico II", Italy, 2001.

[83] J. A. Sethian, *Fast marching methods*, SIAM Rev. **41** (1999), 199-235.

[84] 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, 1999.

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

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

[87] P. Soravia, *Estimates of convergence of fully discrete schemes for the Isaacs equation of pursuit-evasion differential games via maximum principle*, SIAM J. Control Optim., **36** (1998), 1-11.

[88] A. Tankus, N. Sochen, Y. Yeshurun, *A new perspective [on] Shape-from-Shading*, in Proceedings of the 9^{th} IEEE International Conference on Computer Vision (volume II), 862-869, Nice, France, 2003.

[89] A. Tankus, N. Sochen, Y. Yeshurun, *Perspective Shape-from-Shading by Fast Marching*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (volume I), 43-49, Washington, D.C., USA, 2004.

[90] A. Tankus, N. Sochen, Y. Yeshurun, *Reconstruction of medical images by perspective Shape-from-Shading*, in Proceedings of the 17^{th} International Conference on Pattern Recognition (volume III), 778-781, Cambridge, UK, 2004.

[91] A. M. Tarasyev, A. A. Uspenskiy, V. N. Ushakov, *Approximation schemes and finite-difference operators for constructing generalized solutions of Hamilton-Jacobi equations*, J. Comput. Systems Sci. Internat. **33** (1995), 127-139.

[92] M. M. Tidball, E. Altman, *Approximations in dynamic zero-sum games. I*, SIAM J. Control Optim., **34** (1996), 311-328.

[93] M. M. Tidball, R. L. V. Gonzalez, *Zero sum differential games with stopping times: some results about their numerical resolution*, in T. Başar and A. Haurie (editors), "Advances in Dynamic Games and Applications", Annals of the International Society of Dynamic Games **1**, 106-124, Birkhäuser, 1994.

[94] M. M. Tidball, O. Pourtallier, E. Altman, *Approximations in dynamic zero-sum games. II*, SIAM J. Control Optim., **35** (1997), 2101-2117.

[95] C. Truini, *Approssimazione numerica del problema controllistico di tempo minimo e applicazione all'evoluzione dei fronti*, Master Thesis, Dipartimento di Matematica, Roma, Italy, 1995.

[96] 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** (2003), 673-694.

[97] J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Tran. Automatic. Control, **40** (1995), 1528-1538.

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

[99] S. Y. Yuen, Y. Y. Tsui, Y. W. Leung, R. M. M. Chen, *Fast Marching method for Shape from Shading under perspective projection*, in J. J. Villanueva (editor), Proceedings of Visualization, Imaging and Image Processing, Marbella, Spain, 2002.

[100] H. Zhao, *A fast sweeping method for eikonal equations*, Math. Comp., **74** (2005), 603-627.

List of publications

[1] E. Cristiani, M. Falcone, *A fully-discrete scheme for the value function of differential games with state constraints*, submitted to Proceedings of the 12^{th} International Symposium on Dynamic Games and Applications, Sophia Antipolis, 2006.

[2] E. Cristiani, J.-D. Durou, M. Falcone, *The convex-concave ambiguity in Perspective Shape-from-Shading models*, submitted to Proceedings of SSVM 2007.

[3] E. Cristiani, M. Falcone, *A Fast Marching Method for Pursuit-Evasion Games*, 6-pages abstract published electronically in Proceedings of SIMAI 2006, Baia Samuele, Ragusa (Italy), May 22-26, 2006.

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

[5] E. Cristiani, M. Falcone, *Fast semi-Lagrangian schemes for the Eikonal equation and applications*, preprint, Dipartimento di Matematica, 2005. Submitted to SIAM J. Numer. Anal., under revision. Preprint server: `http://cpde.iac.rm.cnr.it/`.

[6] E. Cristiani, M. Falcone, A. Seghini, *Numerical solution of the perspective Shape from Shading problem*, in Proceedings of "Control Systems: Theory, Numerics and Applications" PoS (CSTNA2005) 008, `http://pos.sissa.it/`.