Analysis and approximation of some PDE models for 3D vision and image segmentation

M. Falcone

Dipartimento di Matematica
SAPIENZA, Università di Roma

Lecture: Fast Marching Methods
Napoli, maggio 2018

Outline of the course

- Lecture 1: An introduction to some classical models in image processing
- Lecture 2: The level set-method and some basics on viscosity solutions
- Lecture 3: Finite difference and semi-Lagrangian schemes
- Lecture 4: Fast Marching methods

Outline of this lecture

- Fast Marching schemes
- Fast Sweeping
- Other acceleration techniques

Outline

Fast Marching Method

In this process we do not want loose accuracy and we would like to keep the derivative of u big enough (ideally $|Du(x)| = 1$) to get an accurate representation of the front.

The Fast Marching method has been conceived to speed up the computations and save CPU time.

The crucial point is to concentrate the computational effort around the front at every iteration avoiding useless computations.

Questions

In order to set up and analyze the FMM we have to answer several questions:

- what is the initial configuration of the front in our numerical scheme?
- what is the position of the front at every iteration?
- which nodes will be necessary to compute the front configuration at the following time step?
- does the procedure converge to the correct viscosity solution?
- how many operations will be needed to get the right solution?

Front Propagation Problem

The main idea of Fast Marching method is based on the front propagation point of view.

Let $\partial\Omega$ be a closed curve (the front) in $\mathbb{R}^2$ and suppose that each of its point moves in the normal direction with speed $c(x)$.

We will use the equivalence between the evolutive and the stationary problem (minimum time).

Front Propagation Problem

Then, the evolution of the front at every time is given by the level sets of the function $T(x)$ solution of the

Eikonal equation

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

In this interpretation, $T(x)$ is the arrival time of the front at x.

The FD discretization

Let us write equation the eikonal equation as

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

The standard up-wind first order FD approximation is

$$\left(\max\left\{ \max\left\{ \frac{T_{i,j} - T_{i-1,j}}{\Delta x}, 0 \right\}, -\min\left\{ \frac{T_{i+1,j} - T_{i,j}}{\Delta x}, 0 \right\} \right\} \right)^2 +$$
$$+ \left(\max\left\{ \max\left\{ \frac{T_{i,j} - T_{i,j-1}}{\Delta y}, 0 \right\}, -\min\left\{ \frac{T_{i,j+1} - T_{i,j}}{\Delta y}, 0 \right\} \right\} \right)^2 = \frac{1}{c_{i,j}^2}$$

where, as usual, $T_{i,j} = T(x_i, y_j)$.

The Fast Marching method

The FM method was introduced by J. A. Sethian in 1996. It is an acceleration method for the classical iterative FD scheme for the eikonal equation.

Main Idea (Tsitsiklis (1995), Sethian (1996))

Processing the nodes in a special ordering one can compute the solution in just 1 iteration.
This special ordering corresponds to the increasing values of T.

The FMM is able to find the ordering corresponding to the increasing values of (the unknown) T, while computing. This is done introducing a *NARROW BAND* which locates the front.

The Fast Marching method

The FM method was introduced by J. A. Sethian in 1996. It is an acceleration method for the classical iterative FD scheme for the eikonal equation.

Main Idea (Tsitsiklis (1995), Sethian (1996))

Processing the nodes in a special ordering one can compute the solution in just 1 iteration.
This special ordering corresponds to the increasing values of T.

The FMM is able to find the ordering corresponding to the increasing values of (the unknown) T, while computing. This is done introducing a *NARROW BAND* which locates the front.

The Fast Marching method

The FM method was introduced by J. A. Sethian in 1996. It is an acceleration method for the classical iterative FD scheme for the eikonal equation.

Main Idea (Tsitsiklis (1995), Sethian (1996))

Processing the nodes in a special ordering one can compute the solution in just 1 iteration.
This special ordering corresponds to the increasing values of T.

The FMM is able to find the ordering corresponding to the increasing values of (the unknown) T, while computing. This is done introducing a *NARROW BAND* which locates the front.

The Fast Marching method

Just the nodes in the NB are computed at each step.

Local scheme

The computation of T in every node of the NB is performed using the standard FD-discretization mentioned above (but other choices are possible)

Adaptive Narrow Band

Only one node of the NB is accepted at every iteration. It is the node where we have the minimal value. The NB is up-dated with the new neighbors of the accepted node. The algorithm stops when all the nodes have been accepted.

The Fast Marching method

Just the nodes in the NB are computed at each step.

Local scheme

The computation of T in every node of the NB is performed using the standard FD-discretization mentioned above (but other choices are possible)

Adaptive Narrow Band

Only one node of the NB is accepted at every iteration. It is the node where we have the minimal value. The NB is up-dated with the new neighbors of the accepted node. The algorithm stops when all the nodes have been accepted.

The Fast Marching method

Just the nodes in the NB are computed at each step.

Local scheme

The computation of T in every node of the NB is performed using the standard FD-discretization mentioned above (but other choices are possible)

Adaptive Narrow Band

Only one node of the NB is accepted at every iteration. It is the node where we have the minimal value. The NB is up-dated with the new neighbors of the accepted node. The algorithm stops when all the nodes have been accepted.

Front Representation on the Grid

Let us consider the simple case of a structured uniform grid in $\mathbb{R}^2$,

$$Z \equiv \{x_{ij} : x_{ij} = (x_i, y_j), x_i = i\Delta x, y_j = j\Delta y\}$$

we compute on $Q \cap Z$ where $\Omega_0 \subset Q$.
We set

- $T_{ij} = 0$ for every $x_{ij} \in \Omega_0$
- $T_{ij} = +\infty$ elsewhere

The Basic Local Rule

Let us consider the simple case of a structured uniform grid in $\mathbb{R}^2$,

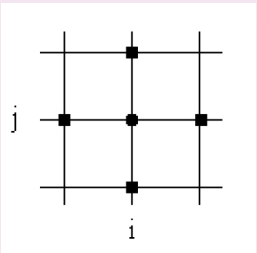
$$\left(\max\left\{ \max\left\{ \frac{T_{i,j} - T_{i-1,j}}{\Delta x}, 0 \right\}, -\min\left\{ \frac{T_{i+1,j} - T_{i,j}}{\Delta x}, 0 \right\} \right\} \right)^2 +$$

$$+ \left(\max\left\{ \max\left\{ \frac{T_{i,j} - T_{i,j-1}}{\Delta y}, 0 \right\}, -\min\left\{ \frac{T_{i,j+1} - T_{i,j}}{\Delta y}, 0 \right\} \right\} \right)^2 = \frac{1}{c_{i,j}^2}$$

The Basic Local Rule

The points involved in this formula are the stencil of the scheme and they are the "first neighbors" of the node where we are computing, i.e.

$$N_{FD}(x_{ij}) = \{x_{i+1,j}, x_{i,j-1}, x_{i-1,j}, x_{i,j+1}\}$$



Partitioning the nodes

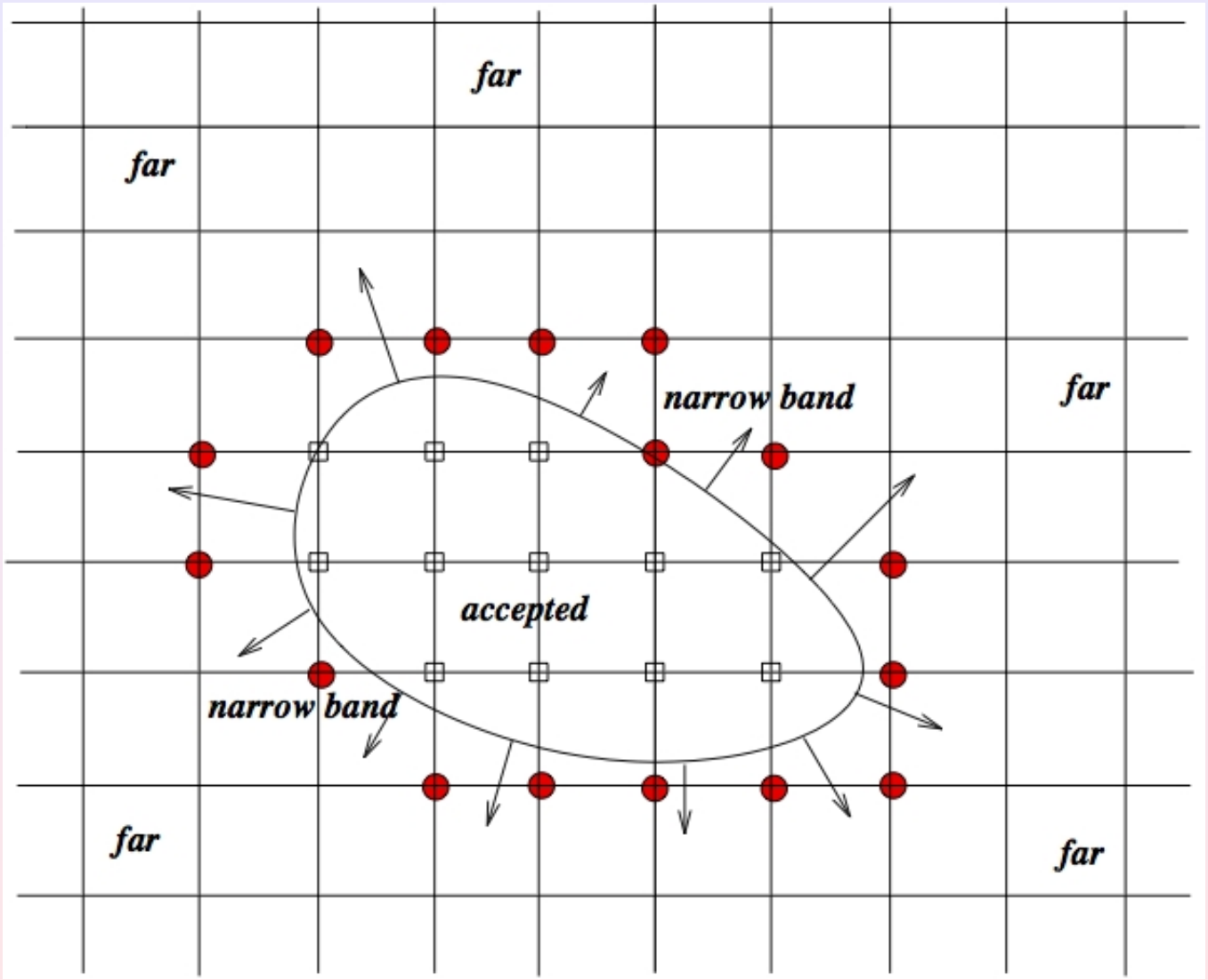
At every iteration (but the last one) we will have three sets of nodes:

- the Accepted nodes, where the values has been already computed and fixed
- the Narrow Band nodes, where the algorithm is computing
- the Far nodes, where the algorithm will compute in the next iterations

We will denote by $A(k)$, $NB(k)$ and $F(k)$ these subsets at the k-th iteration.

Definition

At the iteration k the $NB(k)$ is the set of nodes which are first neighbors of the nodes in the Accepted region of the previous iteration, i.e. $A(k-1)$.


far
far
narrow band
far
accepted
narrow band
far
far

Initialization

Let us consider the simple case of a structured uniform grid in $\mathbb{R}^2$,

$$Z \equiv \{x_{ij} : x_{ij} = (x_i, y_j), x_i = i\Delta x, y_j = j\Delta y\}$$

we compute on $Q \cap Z$ where $\Omega_0 \subset Q$.
We set

- $T_{ij} = 0$ for every $x_{ij} \in \Omega_0$
- $T_{ij} = +\infty$ elsewhere

FMM Algorithm

The algorithm step-by-step, initialization

1. The nodes belonging to the initial front Γ_0 are located and labeled as *Accepted*. They form the set $\widetilde{\Gamma}_0$. The value of T of these nodes is set to 0.
2. $NB(1)$ is defined as the set of the nodes belonging to $N_{FD}(\widetilde{\Gamma}_0)$, external to Γ_0.
3. Set $T_{i,j} = +\infty$ for any $(i,j) \in NB(1)$.
4. The remaining nodes are labeled as *Far*, their value is set to $T = +\infty$.

FMM Algorithm

The algorithm step-by-step, main cycle

Repeat

1. Compute $T_{i,j}$ by the FD scheme on $NB(k)$
2. Find the minimum value of $T_{i,j}$ in $NB(k)$.
3. Label (i,j) as Accepted, i.e. $A(k+1) = A(k) \cup \{x_{ij}\}$
4. Remove (i,j) from $NB(k)$.
5. Up-date the $NB(k)$ to $NB(k+1)$adding the first neighbors of the NEW accepted node.
6. Set k=k+1

Until ALL the nodes have been accepted.

FMM Algorithm

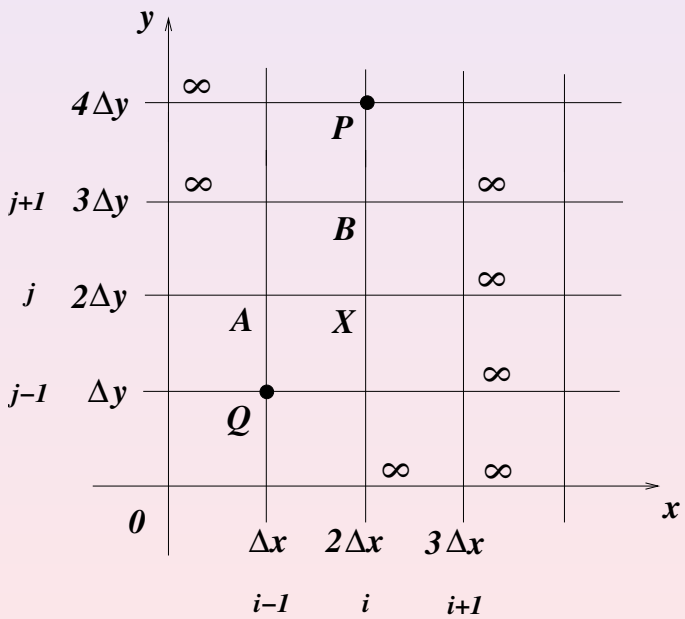
On a finite grid this give the solution after a finite number of operations, the solution coincides with the numerical solution of the global scheme.

How much it costs ?

We have to compute the solution at every point at most 4 times and we have to search for the minimum in NB at every iteration. Using a heap-sort method this search costs $O(\ln(N_{nb}))$. The global cost is dominated by $O(N \ln(N))$ (N represents the total number of nodes in the grid).

Problems with FM technique

However, there are examples which shows that the FM technique is not always suited to approximate the solution of our model problem. In fact it can produce complex solutions if no compatibility condition is introduced between the velocity and Δx !



Convergence of FM method

Theorem

Let the following assumptions hold true:

- $c \in Lip(\mathbb{R}^n) \cap L^\infty(\mathbb{R}^n)$
- $c(x) > 0$
- $\Delta x \leq (\sqrt{2}-1)\frac{c_{min}}{L_c}$ where $c_{min} = \min_{\mathbb{R}^n \setminus \Omega} c(x)$ and L_c is the Lipschitz constant of c.

Then, FD-FM method computes a *real* approximate solution of the eikonal equation. Moreover, this solution is exactly the same solution of the FD classical iterative scheme.

Min-Heap Data Structure

At the k-th iteration we must compute the minimum value on the nodes belonging to the $NB(k)$.
This can be done in many ways but can be rather expensive.
An efficient procedure consists in the construction of a "complete binary tree" with the property that the value at any given node is less or equal to the values of its children.
This structure must be up-dated at every iteration (moving the values up and down along the tree).
Since this keeps the minimum value always at the root of the tree we just pay the price of the up-date which is $O(\ln H)$ if there are H nodes in the heap (delicate implementation).

Gauss-Seidel acceleration

The standard iterative method is

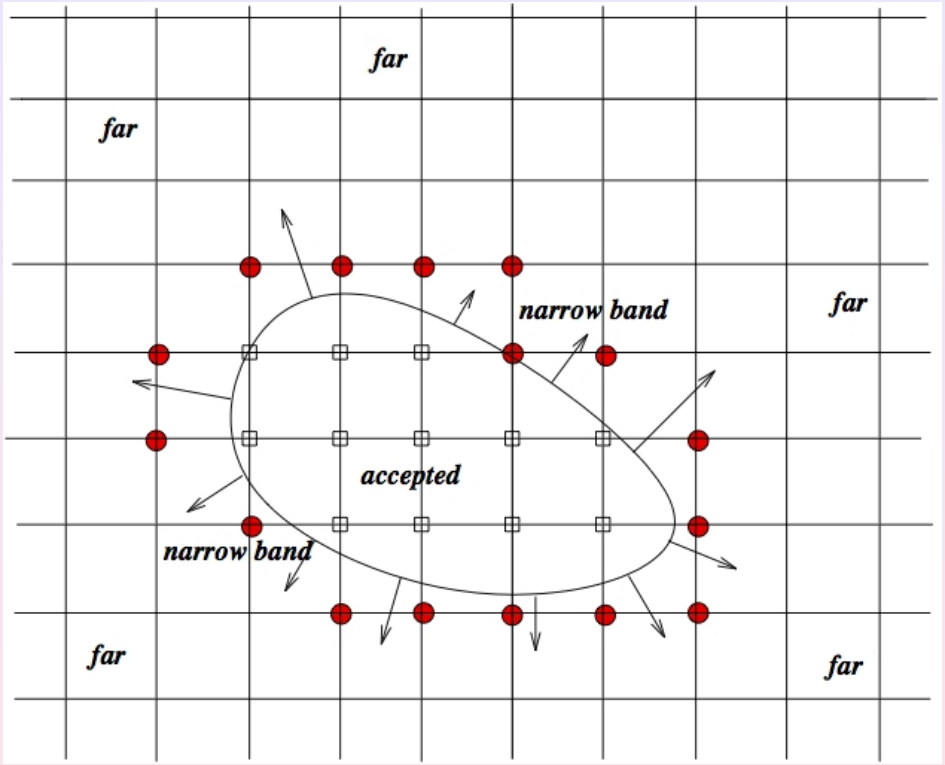
$$T^{k+1} = S(T^k)$$

for a given T_0. The S operator is defined locally by our rule ad will be applied sequentially on the nodes $NB(k)$.
We can accelerate convergence up-dating the values of T as far as they are computed, i.e. in the 1 dimensional case

$$T_i^{k+1} = S(T_{i-1}^{k+1}, T_i^k), \text{ for any} k, i$$

However, the improvement obtained by this "re-alimentation" algorithm is very limited.

The sets of nodes for the SL-FM



The NB_{SL} includes MORE nodes (NSEW and the diagonals for every A node).

Convergence for the FM-SL scheme

Theorem (Cristiani-F, SIAM J. Num. Anal., 2007)

Let the following assumptions hold true:

- $c \in Lip(\mathbb{R}^n) \cap L^\infty(\mathbb{R}^n)$
- $c(x) \geq 0$

Then, SL-FM method computes an approximate solution of eikonal equation. Moreover, this solution is exactly the same solution of SL classical iterative scheme.

Remark

NO conditions on c and/or Δx are required!

SL-FMM Algorithm

On a finite grid this give the solution after a finite number of operations, the solution coincides with the numerical solution of the global scheme.

How much it costs ?

We have to compute the solution at every point at most 8 times and we have to search for the minimum in NB at every iteration. Using a heap-sort method this search costs $O(\ln(N_{nb}))$. The global cost is dominated by $O(N \ln(N))$ (N represents the total number of nodes in the grid).

Convergence

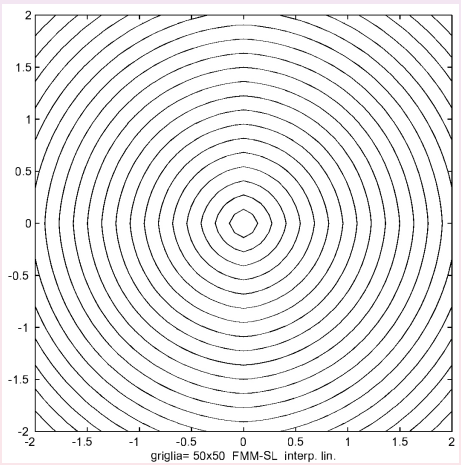
WARNING

The convergence of FD-FM and SL-FM comes from the equivalence of the FM approximate solutions with respect to the fixed point solutions (which have been shown to converge to the correct viscosity solution).

TEST 1. Numerical comparison

Distance from the origin. $\Omega = \{(0,0)\}, \quad c(x,y) \equiv 1$

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



TEST 1. Numerical comparison

Distance from the origin. $\Omega = \{(0,0)\}, \quad c(x,y) \equiv 1$

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

method	Δx	L^∞ error	L^1 error	CPU time (sec)
SL (46 its.)	0.08	0.0329	0.3757	8.4
FM-FD	0.08	0.0875	0.7807	0.5
FM-SL	0.08	0.0329	0.3757	0.7

TEST 1. Numerical comparison

Distance from the origin. $\Omega = \{(0,0)\}, \quad c(x,y) \equiv 1$

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

method	Δx	L^∞ error	L^1 error	CPU time (sec)
SL (46 its.)	0.08	**0.0329**	**0.3757**	**8.4**
FM-FD	0.08	0.0875	0.7807	0.5
FM-SL	0.08	**0.0329**	**0.3757**	**0.7**

TEST 1. Numerical comparison

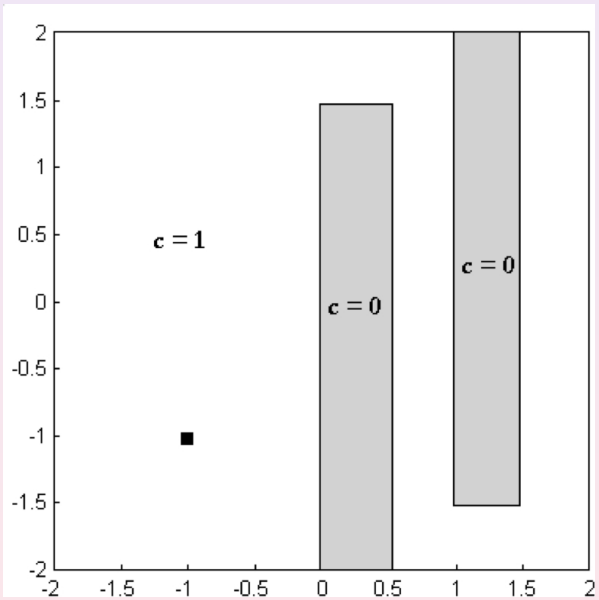
Distance from the origin. $\Omega = \{(0,0)\}, \quad c(x,y) \equiv 1$

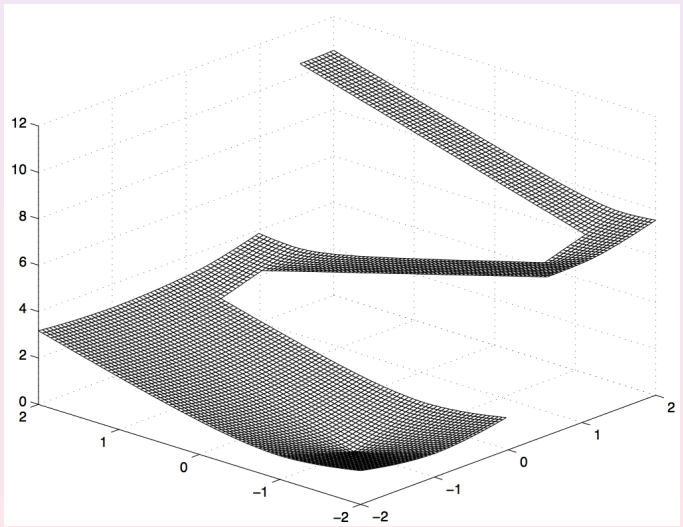
Exact solution: $T(x,y) = \sqrt{(x^2+y^2)}$

method	Δx	L^∞ error	L^1 error	CPU time (sec)
SL (46 its.)	0.08	0.0329	0.3757	8.4
FM-FD	0.08	**0.0875**	**0.7807**	**0.5**
FM-SL	0.08	**0.0329**	**0.3757**	**0.7**

TEST 2. Minimum time problem

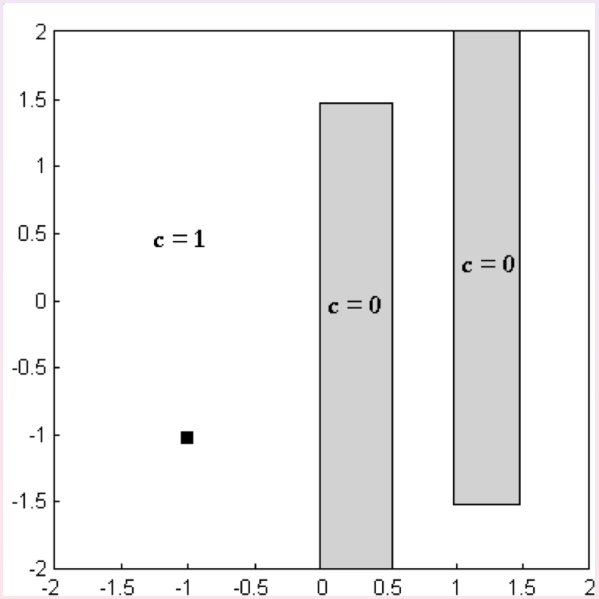
Presence of obstacles

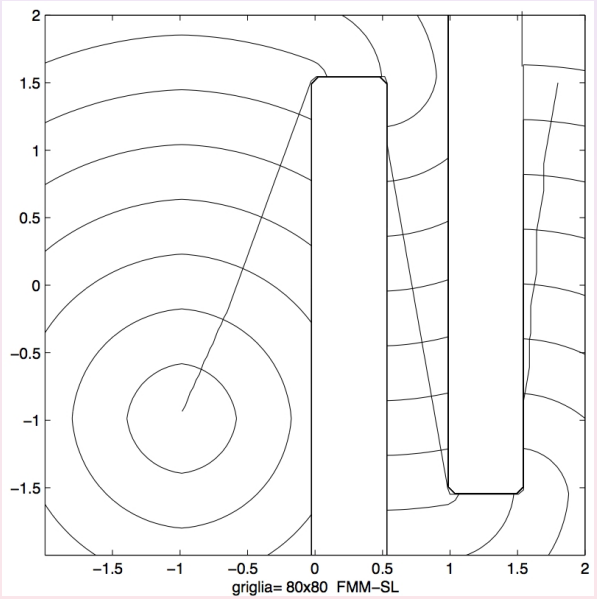




TEST 2. Minimum time problem

Presence of obstacles





Trasparent boundary conditions

In the SL method it is rather easy to implement "'state constraint" boundary conditions. This allows to deal with obstacles, O.
In fact, the algorithm look for a minimum

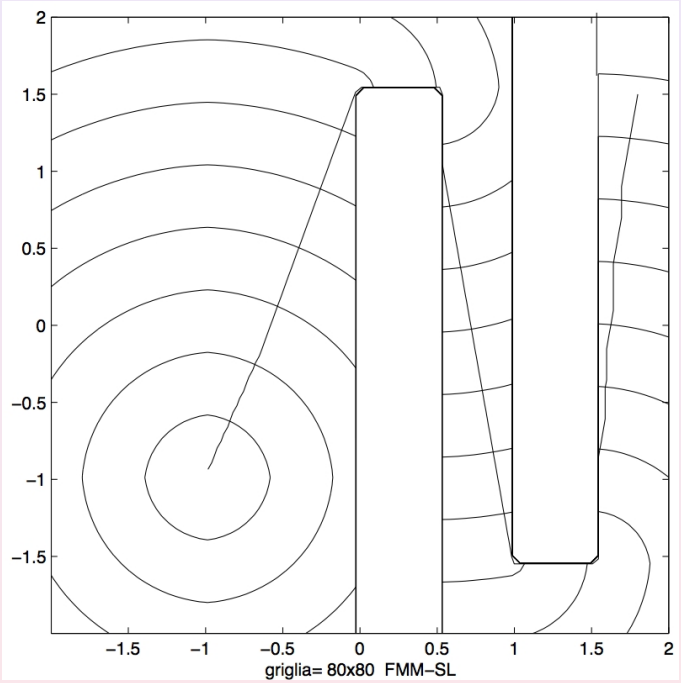
$$v_h(x_{ij}) = \min_{a \in B(0,1)} \{e^{-h} v_h(x_{ij} - hc(x_{ij})a)\} + 1 - e^{-h}$$

for $x_{ij} \in \Omega_0^C \cap \mathbb{Z} \setminus O$ so it is sufficient to impose on the boundaries ∂Q and ∂O a value $\widehat{v}$

$$\widehat{v} > \max_{x \in Q \setminus O} v_h(x)$$

Trasparent boundary conditions

The results are quite accurate



Outline

Sweeping

It has been observed that for the eikonal equation there is a simple way to speed-up the computations instead of looking for a fixed point iteration and search for the information driven by the characteristics at every node.
The sweeping method sweeps the informations along the directions of the axis until the solution does not change

- The first iteration sweeps to the North every point
- The second iteration sweeps to the West every point
- The third iteration sweeps to the South every point
- The forth iteration sweeps to the Est every point

Sweeping

This algorithm continues until convergence.
In fact, for the eikonal equation only 8 iterations are needed, so the cost is very limited.
The sweeping method has been applied to more general Hamilton-Jacobi equations, f.e. to Bellman equations. The Lax-Friedrichs scheme or the Godunov scheme have been used as local rules [Kao-Osher-Tsai (2005)].

Sweeping

Advantages

The methods is rather simple to implement and does not require dynamic structures
Convergence is obtained in finite number of iterations (for the eikonal equation)

Disadvantages

It is difficult to determine its complexity It does not converge in a finite number of iterations for more general equations

Group Marching Method (GMM)

The idea is to select a group of points at every iteration and make them advance at the same time [Kim (2001)].
This requires a correction-by-iteration strategy but avoids the search for the minimum value in the Narrow Band. Every iteration cost a bit more than the standard FMM.
This algorithm continues untill all the nodes have been Accepted.

Group Marching

Advantages

It has as an $O(N)$ complexity
Convergence is obtained in finite number of iterations (for the eikonal equation)

Disadvantages

Specifically designed for the eikonal equation
Difficult to apply for more general equations

Basic References

Many informations regarding the schemes and a variety of fields of application can be found in the books:

J.A. Sethian, Level Set Method. Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science, Cambridge Monographs on Applied and Computational Mathematics, vol. 3, Cambridge University Press, Cambridge, 1996.

S. Osher, R.P. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer–Verlag, New York, 2003.

Basic references

FAST MARCHING

J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Tran. Automatic. Control, **40** (1995), 1528-1538.

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

J.A. Sethian, *Fast Marching Methods*, SIAM Review, No.2 **41**, 199–235 (1999)

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

Basic references

FAST MARCHING ctnd

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

E. Carlini, M. Falcone, N. Forcadel, R. Monneau, *Convergence of a Generalized Fast Marching Method for an eikonal equation with a velocity changing sign*, SIAM J.Num.Anal., 2009