Big Ray Tracing and Eikonal Solver on Unstructured Grids : Application to the Computation of a Multi-valued Travel-time Field in the Marmousi Model

Remi Abgrall, Jean-David Benamou

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Big Ray Tracing and Eikonal Solver on Unstructured Grids : Application to the Computation of a Multi-valued Travel-time Field in the Marmousi Model

Rémi Abgrall , Jean-David Benamou

N° 3019

octobre 1996

THÈME 4

ISSN 0249-6399

Big Ray Tracing and Eikonal Solver on Unstructured Grids : Application to the Computation of a Multi-valued Travel-time Field in the Marmousi Model

Rémi Abgrall [*], Jean-David Benamou [†]

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Ondes

Rapport de recherche n° 3019 — octobre 1996 — 26 pages

Abstract: We present in this paper the numerical computation of the multi-valued travel-time field generated by a point source experiment in the Marmousi model. Two recently developed methods are combined to achieve this goal : a method called big ray tracing, used for the computation of multi-valued travel-time fields, and an eikonal solver designed to work on unstructured meshes.
Big ray tracing is based on a combination of ray tracing and local solutions of the eikonal equation. A classical ray tracing first 'discretizes' the phase space and defines local zones which possibly overlap where the travel-time field is multi-valued. An eikonal solver then computes the travel-time in these zones called big rays. It acts as an exact interpolator between rays associated to different branches of the travel-time field. The geometry of the big rays may be complicated and is best discretized using unstructured meshes. An eikonal solver designed to work on unstructured meshes is used.
Keywords : Hamilton-Jacobi, Calculus of Variation, Fermat Principle, Geometric al Optics, Ray Tracing, Multi-valued Travel Time Field, Viscosity Solution, FD/FE Upwind Scheme,High Frequency Asymptotic, Wave equation, Eikonal equation, Hyperbolic Equation, unstructured mesh .
AMS(MOS) subject classification : 34H05, 49S05, 65N06, 65D99, 73D99, 78-08 , 78A05.

(Résumé : tsvp)

* Université de Bordeaux I, 351 Cours de la Libération, 33 405 Talence cedex France
† INRIA, Domaine de Voluceau, B.P.105 78153 Le Chesnay Cedex, France

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : (33 1) 39 63 55 11 – Télécopie : (33 1) 39 63 53 30

Lancer de Gros Rayons et Résolution de l'Equation Eikonale sur maillage non Structures : Application au Calcul de Temps d'Arrivées Multiples dans le Modèle Marmousi

Résumé : Nous utilisons la technique de lancer de gros rayons combinée a un schéma de résolution de l'equation eikonale discretisée sur maillage non structurés. Nous calculons ainsi les temps d'arrivées multiples générés par une source ponctuelle dans un modèle de vitesse 'réaliste' : le modèle Marmousi

1 Introduction

Motivated by potential application in tomography and the computation of migration operators, several studies on the direct resolution of the eikonal equation have appeared in the geophysical literature (see e.g. [25, 15]). This technique, usually based on finite differences, allows the computation of the travel-time field on a regular grid. This desirable property is not shared by classical ray tracing which uses interpolation processes to reach the same goal. The limitation of this approach dwells in the fact that eikonal solvers only compute the earliest travel-time (usually associated to the less energetic reflections). The computation of the multi-valued travel time field (i.e. all the branches of the solution) using ray tracing, eikonal solvers or other methods, is a difficult academic problem and a challenging industrial issue. A non exhaustive bibliography on this subject follows : [27, 23, 5, 6, 4, 13, 20, 24, 12, 17] .

We present in this paper the application of two methods, recently developed, to the computation of the multi-valued travel-time field generated by a source point experiment in the two–dimensional Marmousi Model. It is namely the 'big ray tracing' method [9] combined with an eikonal solver designed to work on unstructured meshes [2]. Both methods are quickly reviewed and the result of the computation presented and commented.

A rapid outlook of the problem indicates a first difficulty : given a velocity model, there is no way of a priori determining the number of different branches of solution and their spatial location. This information is only contained in the phase space solution of the problem, i.e. the solution computed by ray tracing. A second problem is to understand the relation between the single-valued numerical solution of the eikonal equation and the multi-valued travel-time field. Since the eikonal equation is a very particular case of a Hamilton–Jacobi PDE, we discuss this problem in connection with the theory of viscosity solutions in the next section. We then detail the hybrid algorithm based on ray tracing and local resolutions of the eikonal equation in section 3. The current limitations of the method are also described here.

There is some litterature about the numerical discretisation of Hamilton–Jacobi equations on regular grids. One may cite for example [11, 7, 21]. There are two major difficulties in this problem : how to derive stable schemes (i.e. the numerical solution does not blow up) ?, how to derive schemes which ensure the convergence of the numerical solution to the viscosity solution ? Hidden behind the eikonal equation are the rays of the geometrical optics : the perturbations generated by the source point travel with a finite speed in the computational domain on the rays from the source to any current point. By taking into account this "upwinding" phenomena, it is possible to construct stable, monotone and convergent schemes. For example the Godunov or Lax–Friedrichs schemes of [21] belong to this class. Because of the potentially complicated geometry of the local domains on which these local solutions are computed, they are more accurately discretized using an unstructured mesh. It is possible to generalise the above schemes to unstructured meshes and to keep the stability and convergence properties. In section 4, we detail our Lax–Friedrichs scheme

on triangular unstructured meshes. It is a first order scheme, its second order (in space) version is described in the Appendix. We have chosen this particular scheme because its implementation is relatively easy.

2 Viscosity solutions of the eikonal equation

The theory of viscosity solutions of Hamilton-Jacobi equations ([19, 8]) provides a rigorous mathematical framework for the resolution of the eikonal equation. Eikonal numerical solvers actually belong to the 'upwind' schemes family, the solution of which converge to generalized (the gradient of the solution may be discontinuous) or 'viscosity' solutions. In what follows, u is the travel time, n is the slowness index of the domain Ω, and ∇ is the gradient with respect to the variables x and z.

The viscosity solution of the eikonal equation in a given domain Ω

$$\|\nabla u(x,z)\| = n(x,z), \quad (x,z) \in \Omega \tag{1}$$

with a forced point source S at (x_0, z_0)

$$u(x_0, z_0) = 0 \tag{2}$$

and the Soner 'discontinuous' boundary condition ([22] [26]) conveniently written as (see [8])

$$u(x,z) = +\infty, \quad (x,z) \in \partial\Omega \tag{3}$$

can be schematically characterized, for all (x, y), as the solution of a *state constrained* optimal control problem. More precisely, $u(x, y)$ is the infimum of

$$\int_{(x_0,z_0)}^{(x,z)} n(\xi(s))ds$$

over all paths ξ joining (x_0, z_0) and (x, z) and such that ξ *remains strictly in* $\overline{\Omega}$, see Figure 1. Here, s is the arc length. When Ω is the whole space, this is formally equivalent to the Fermat principle and the optimal $s \mapsto \xi(s)$ are rays which are associated to the smallest optical length, that is, the earliest travel-time.

Latest travel-times can also be characterized by a similar minimization problem. The key point is to notice that, when a ray gives a local minimum for the above optimal control problem set in the whole space, the associated value function gives a latest travel-time. So, if we consider a ray giving a latest travel-times and choose a restricted Ω such that this ray will become a global minima for the new optimal control problem (see fig. 1), the viscosity solution of the eikonal equation on this restricted domain will correspond to this latest travel time. In short, the viscosity solution always gives the earliest travel time subject to the constraint $\xi(s) \in \overline{\Omega}$ which at the equation level is enforced by the Soner boundary condition.

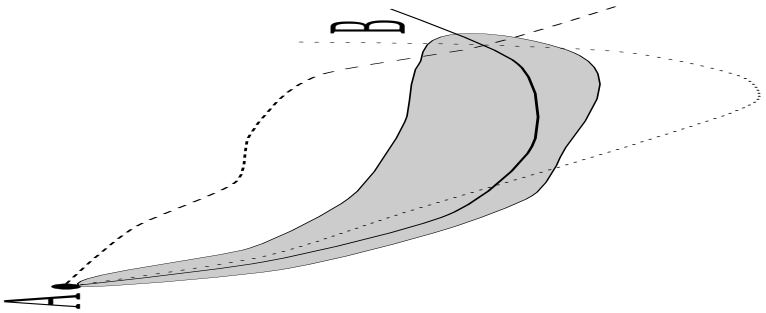

Figure 1: Three rays are crossing. We suppose that the dotted rays give earliest arrival times at the crossing B from the source A, the other (plain line in the colored domain) a later arrival time. The latter ray (a local minimum in all space) will still be a global minimum in the colored domain and therefore the viscosity solution of the eikonal equation with Soner boundary condition in this domain gives the later arrival time at the crossing B.

In [9], where these ideas are developed, a method called 'big ray tracing' is proposed to select a family of local domains, the big rays, such that the associated viscosity solutions give in each big ray a different branch of the multi-valued travel time field. We face here again the first difficulty mentioned in the introduction : each branch of the eikonal multi-valued solution is associated to 'local' families of rays. A natural idea is to perform a ray tracing as a first step to analyze the problem and select these local domains. This is explained in the next section.

3 The Big Ray tracing algorithm

We propose the following automatic procedure:

- **Ray discretization :** Trace a given number of rays with initial shooting angles (see figure 2) regularly discretizing an initial opening angle corresponding to a fan of rays (with a Runge Kutta algorithm for instance). Rays are numbered following these initial directions (say from left to right on figure 2).

- **Big rays generation (see also appendix A):** We now define a 'big ray' as the envelope of three successively shot rays. If the rays are numbered from 1 to N following the order with which they are shot. Big rays are the envelopes of rays (1,2,3) then (3,4,5) and so forth. angle (see also appendix A). See figure 3, 5, 6, 8 and 9 to see actual big rays and figure 10 to see all the big rays covering the domain. We detail this construction in appendix A.

- **Eikonal resolution (see also appendix B and C):** Compute the viscosity solution of the eikonal equation in each big ray. We describe our method of resolution in the next section. See figure 4 and 10 to see contour lines of the solution for two big rays.
- **Multi-valued travel time analysis :** The travel-time field is given by the superposition of all local travel-time fields computed in the big rays. When big rays intersect, this step gives a multi-valued travel time field. See figures 12 and the following.

For an (incomplete) mathematical analysis of this method see [9] where we discuss, in particular, the consistency of the method (its ability to compute the multi-valued travel time field).

The consistency is limited by the ray discretization step. First, simply because we can miss multi-valued travel time produced by fine local heterogeneities contained in one big ray. If rays are potentially crossing inside a big ray, the eikonal solver will only compute the earliest travel-time out of these two rays.

The second inconsistency is linked to the generation of the big rays. It may happened near caustics or focal points that the big rays generation strategy leaves out only a portion of a ray. It will exit and reenter the big ray. Then this ray does not satisfy the state constraint 'ξ remains strictly in $\overline{\Omega}$'. The viscosity solution has therefore no theoretical reason to represent the travel-time associated to this ray once it reenters the domain. A preliminary study [18] however suggest that the viscosity solution is a good approximation of these travel-times, that is, converges to the correct solution as we decrease the initial aperture of the big ray around the direction of this particular ray.

The last problem is the possible presence of conjugate points (focal or caustic points) on the rays. After these points there are no guarantee that the ray is a local minimum. It more likely corresponds to a saddle point for the optimal control problem discuss in section 2. Because of the big ray discretization we avoid most of these rays which fall in the category (just described) of rays which do not satisfy the state constraint. We are currently trying to address these points more rigorously.

4 Eikonal solver on unstructured meshes

The big rays may have complicated geometries. We developed an automatic procedure to mesh them using an unstructured mesh, see appendix A.

We now describe the eikonal numerical solver working on these unstructured meshes and the practical implementation of the Soner boundary condition.

We are implementing an Hamilton-Jacobi solver designed to work on unstructured meshes ([2]). First, instead of solving the steady eikonal equation (1–2–3), we solve its unsteady version on a period of (pseudo-)time $t \in]0, T_f[$ to reach a steady state solution of the eikonal

equation

$$u_t(t,x,z) + \|\nabla u(t,x,z)\| - n(x,z) = 0, \quad (x,z) \in \Omega \tag{4}$$

We consider the following point source boundary condition

$$u(t,x_0,z_0) = 0, \quad \text{for all } t \geq 0 \tag{5}$$

and the Soner boundary condition

$$u(t,x,z) = +\infty, \quad \text{for all } (x,y) \in \partial\Omega \text{ and } (x,z) \neq (x_0,y_0). \tag{6}$$

In (6), $\partial\Omega$ denotes the boundary of Ω.

We need to pick up a sufficiently large T_f to reach the steady-state solution of (4–5–6). We typically take $T_f >$ diameter of $\Omega \times \max_{(x,z)\in\Omega} n(x,z)$.

We now focus on the numerical scheme proposed in [2] to solve the unsteady eikonal equation (4–5–6). The computational domain is first discretized by mean of a triangulation which nodes are denoted by $M_i, i = 1,\ldots,n_s$ (with coordinate (x_i,y_i)) and the triangles are denoted by $T_j, j = 1,\ldots,n_T$. The approximation of u at node M_i and (pseudo-)time $t_n = n\Delta t$ is denoted by u_i^n. The relation between u_i^{n+1} and u_i^n is

$$\begin{cases} u_i^{n+1} = 0 & \text{if } M_i = S = \text{ source point} \\ u_i^{n+1} = u_i^n - \Delta t\,(\mathcal{H}_i^n - n(x_i,y_i)) & \text{else} \end{cases} \tag{7}$$

The Soner boundary condition is *automatically* enforced by the "upwind" properties of the numerical (Lax–Friedrichs) Hamiltonian $\mathcal{H}_i^n$ (the 'upwind' approximation of $\|\nabla u(t,x,y)\|$). This is explained in appendix B where $\mathcal{H}_i^n$ is precisely defined.

This solver has been shown to be convergent (in [2, 1]) provided the time step satisfies

$$\frac{\Delta t}{\rho_i} \leq \frac{1}{2} \tag{8}$$

where ρ_i is the radius of the largest circle of center M_i and contained in the union of the triangles listed in N_i (see figure 18). The "1/2" condition can be relaxed by 1 in practice.

As previously mentioned, the sequence $(u_i^n)_i$ converges to an approximation of the solution of the eikonal equation with the Soner boundary conditions. The difference between the *exact* solution $u(x,z,t)$ and the numerical solution u_i^n is expressed by the following proposition ([2]) :

There exists a constant C which depends only on $n(x,z)$ (the slowness) and α the smallest of all the angles of the triangles T such that, for any $t \in [n\Delta t,(n+1)\Delta t[$, we have :

$$|u(M_i,t) - u_i^n| \leq C\,\sqrt{\rho}. \tag{9}$$

where ρ is the size of the largest triangle.

This inequality is very pessimistic because in its proof, we have to take into account all the possible singularities of the solution. In practice, we observe for points M_i strictly contained inside the domain :

$$|u(M_i,t) - u_i^n| \leq C\ \rho.$$

for the first order scheme and

$$|u(M_i,t) - u_i^n| \leq C\ \rho^2$$

for the second order scheme described in appendix C.

Finally, we observed that the scheme needs $O(ns)$ pseudo-time iteration steps to converge to the steady state solution. The information indeed comes from the source and travels from node to node with a speed limited by the CFL condition (8). This local stability condition therefore enforces a minimum number of (pseudo-)time step iterations to reach the steady-state at $t = T_f$. We are developing an implicit version of this scheme to improve its computational performances.

5 Point source simulation in the Marmousi model and numerical comments

We present the application of this method to the computation of the multi-valued travel-time field in a smoothed 122×384 points Marmousi model (24 meters samples). The horizontal axis x ranges from 0 to $9192m$, the vertical axis z from 0 to $-2904m$. The source is located at $x = 6000m$ and $z = 2800m$.

The objective of this simulation is to show the ability of this algorithm to compute an accurate realistic multi-valued travel time field. We therefore choose to build a large number of big rays, use a reasonably fine mesh and a second order eikonal solver. The number of rays shot, and consequently of big rays built, does not directly penalize the computational cost of the method. Shooting rays (here 150) is indeed not very expensive and one can consider that (at fixed minimum mesh size) the number of discretization points generated in all the big rays only increases with the number of big rays when multi-valued-ness occurs in a region. It is therefore necessary to compute the multi-valued solution. The computational cost of the eikonal solver is linear with respect to the number of points in the considered domain. So, the computational cost does not directly depends on the number of big rays once the ray discretization is fine enough to catch the multi-valued solution. The real extra cost induced by refining the ray discretization is the extra number of mesh generations needed.

We now comment the figures displaying the results at the end of this paper.

- Figure 2 : Left: a smoothed Marmousi model (velocity), the black cross indicates the point source. Right: 150 rays shot in this model, the initial directions regularly discretize the cone $(-90^o/90^o)$. Rays are numbered following these initial directions (from left to right).
- Figure 3 : Left: rays 7 8 9 . Right: the corresponding big ray and its mesh (7968 triangles).
- Figure 4 : Contour line of the eikonal solution (every $0.1s$) in the above big ray of figure 3.
- Figure 5 : Left: rays (71,72,73). Right: rays (73,74,75) (the x scale is larger, the rays are very close).
- Figure 6 : Corresponding big rays and meshes (around 4500 triangles). Note that, as rays 73 and 74 intersect, these big rays overlap.
- Figure 7 : Zoom of the meshes of figure 5.
- Figure 8 : Left: rays (113,114,115). Right: rays (115,116,117).
- Figure 9 : Corresponding big rays and meshes (around 8000 triangles). They overlap.
- Figure 10 : Contour line of the eikonal solution (every $0.1s$) in the big rays of figure 9. The zone where the big rays overlap gives two different travel times.
- Figure 11 : All the big rays (74) and their meshes displayed together. Zones where they overlap are dense and are the locus of multi-valued-ness.
- Figure 12 : Traces of the travel times at the surface grid points of the model (x versus travel-time). Left : a ray tracing solution (given in [16]). It is obtained using a shooting and a two point ray tracing method. Right : solution obtained by considering the travel time computed in all the big rays at the surface ($z = 0$) and interpolated on a 384 regular grid.
- Figures 13,14,15,16,17 : Comparison with the snap shots (on the left) of a wave equation resolution in the same model. The wave equation (in the time domain) is solved using a second-order finite-difference scheme with higher order absorbing boundary conditions [10]. The source is localized in space and a $23Hz$ Gaussian derivative in time.

  On the right all the big rays solutions have been interpolated on a 384 by 122 grid. this gives a multi-valued travel-time field on a regular grid. We then select the 'front' given by the travel times matching the time of the corresponding snap shot with a tolerance of 0.005 seconds. Even though the wave equation solution is band limited in frequency and our solution represent the high frequency asymptotic the kinematics are in very good accordance.

General comments on the numerical results

Figure 12 shows that we are able to recover the three principal zones of multi-valued-ness at the surface. There are actually two superposed triplication in the middle one. All these triplications can be anticipated from the ray tracing of figure 2 where three focal zones can be identified. Note also that the ray tracing solution has difficulties in zones of geometrical spreading on the furthermost right and left side of the figure.

Even though the wave equation solution is band limited in frequency and our solution represents the kinematics (phase) of the high frequency approximation of the wave equation, the results are in very good accordance (figure 13,14,15,16,17). The leading edges of the waves match our travel time contours.

Note that we recovered a distinct second front on the right of the model in a region of somewhat complicated geometrical spreading. As can be seen on the figure 2 this front certainly corresponds to 'S' shaped rays (only one in the ray tracing) which are reflected twice by two different heterogeneities.

Finally, lest us mention that the eikonal solver failed to converge on two very thin rays. This explains the inconsistent black spots on figures 14.

6 Conclusion

We want first to emphasize that the method does not rely on any a priori (qualitative or quantitative) information on the solution. All steps of the algorithm can be completed on any (smooth) velocity model. Our big ray strategy adapt itself to the nature of the solution. In a zone with a low density of rays, big rays will be big and generally the solution is single valued, the eikonal solver then acts as an 'exact' interpolator. In a zone where a lot of rays are crossing, several big rays will superpose, selecting the different 'waves', or families of rays, which interact in this area. We automatically restrict our resolution to the support of the solution in phase space. It guarantees in particular that there are no useless computations. The big ray method is a natural way of domain decompose (if needed) a big problem into smaller subproblem which can be solved in parallel.

The main theoretical and numerical difficulty lies in the construction of the big rays. We currently rely on the information given by the state constrained optimal control. It provides a vague definition of a big ray as a domain which, for every point, strictly contains only one of the rays arriving at that point. By no mean are we claiming that the current strategy for building the big rays is optimal. A better algorithm should certainly incorporate the computation of the amplitudes. The geometry of caustics would then be known (where the amplitude blows up). Incorporating this information in the definition of the big rays may be useful to remove some of the current difficulties.

The 3-D extension of this method is in perspective. The Soner boundary condition will remain the same. The eikonal solver can be generalized to 3-D. The main difficulty will

again be the construction and meshing of the big rays.

There are several directions which can be pursued to speed up the method. First, an implicit version of this scheme can be implemented which would allow to pick up larger time step. An obvious remark is also the underlying parallel feature of the algorithm : the generation of each big ray and the resolution of the eikonal equation can be done separately, hence in parallel.

Acknowledgement Most of this research has been conducted while the first author was in residence at the Courant institute of Mathematical Science, New York, on leave from INRIA Sophia Antipolis. He was partialy supported by DOE grant DE–FG02–88ER25053.

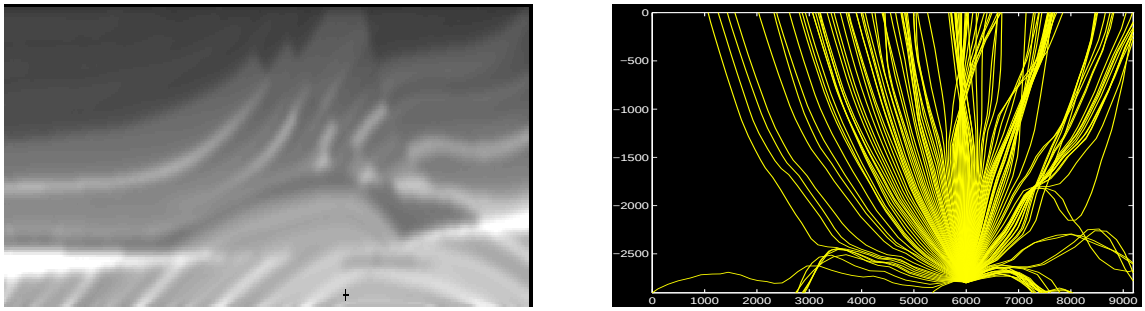

Figure 2:

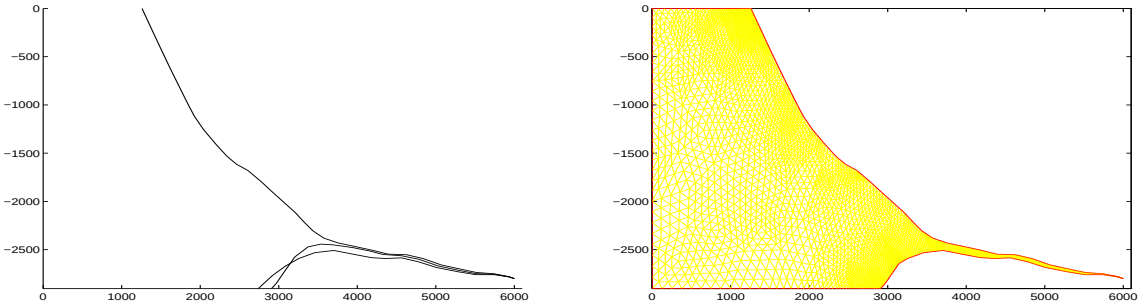

Figure 3:

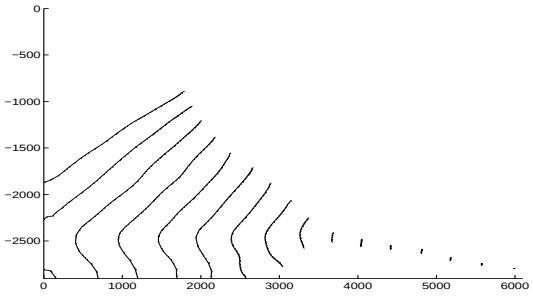

Figure 4:

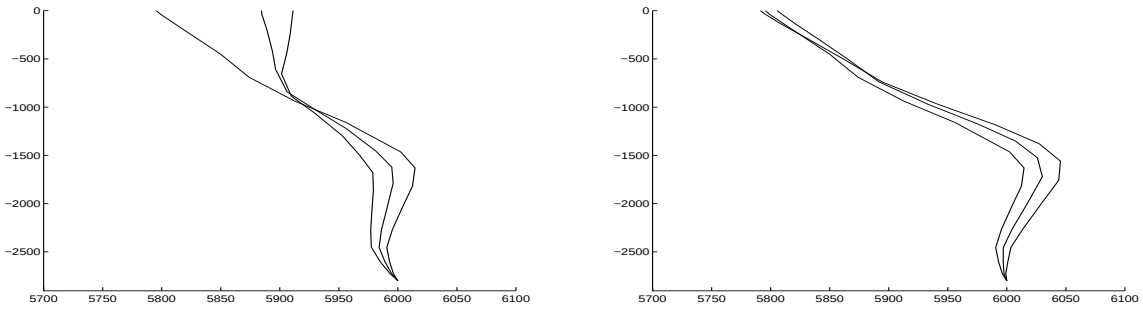

Figure 5:

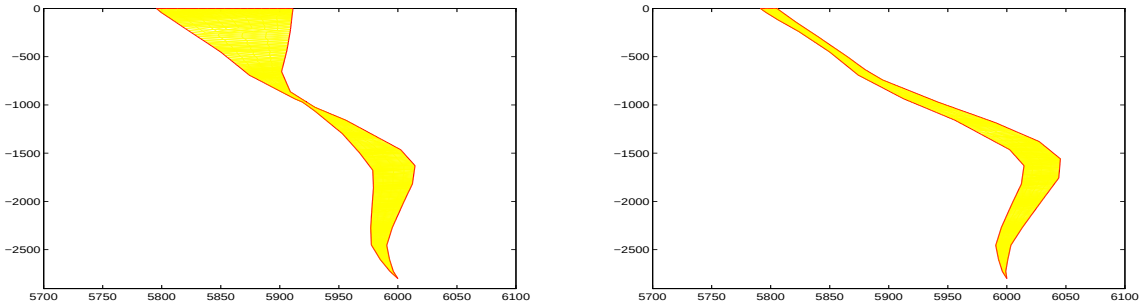

Figure 6:

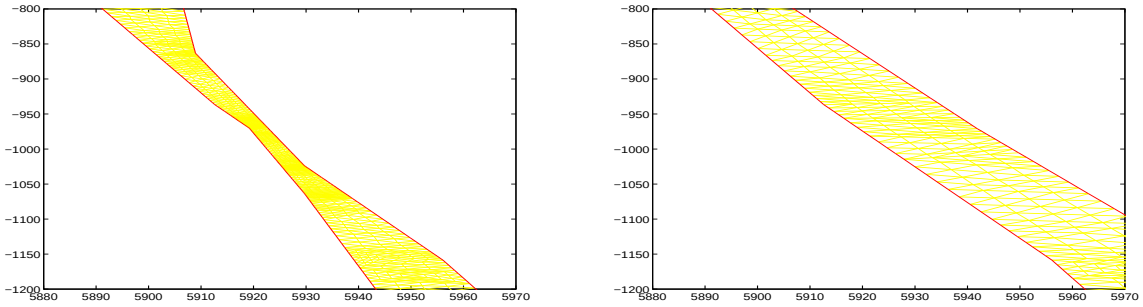

Figure 7:

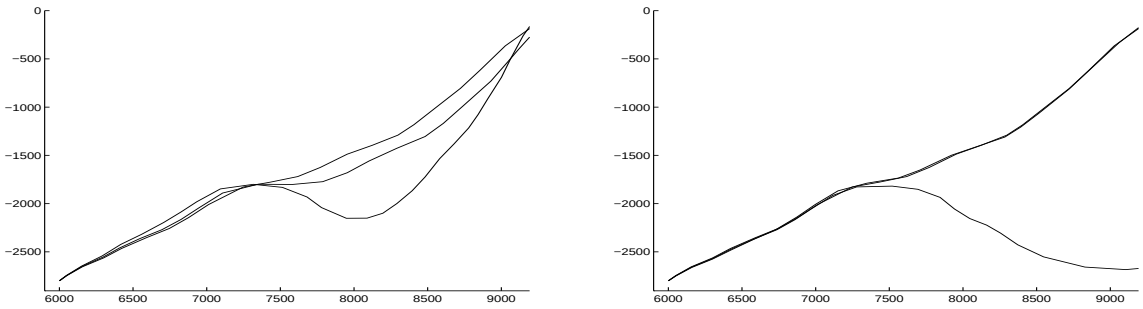

Figure 8:

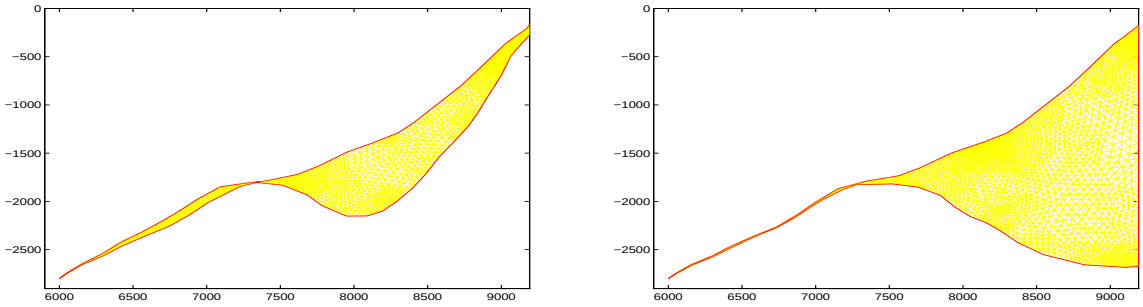

Figure 9:

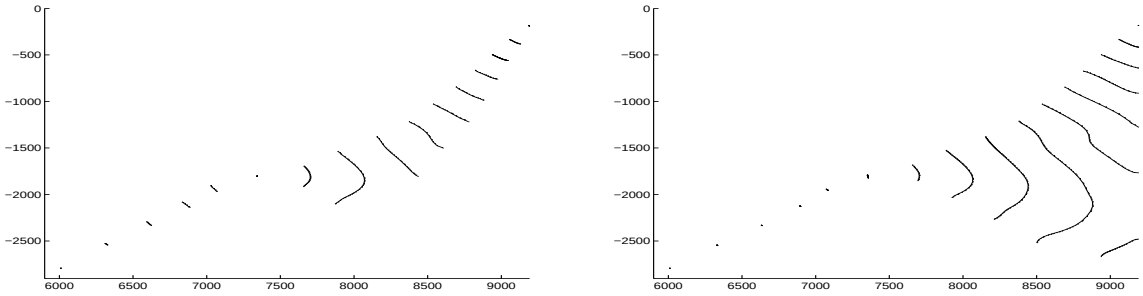

Figure 10:

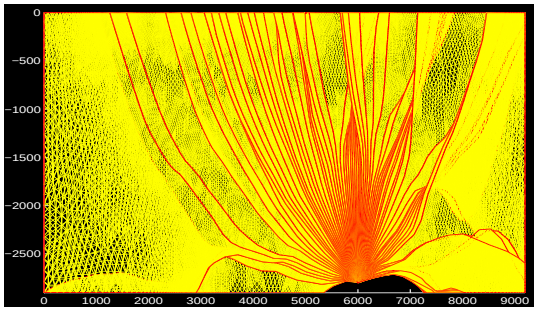

Figure 11:

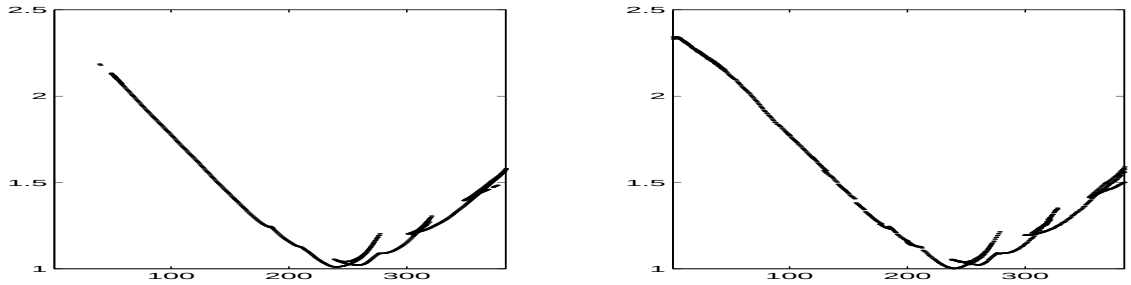

Figure 12:

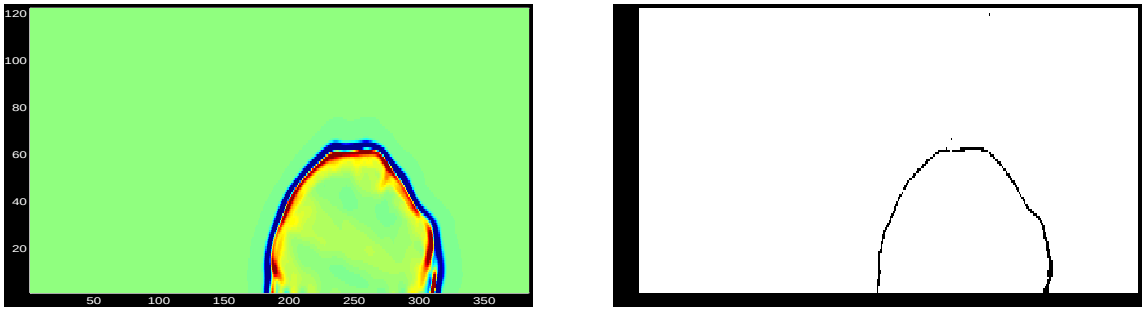

Figure 13:

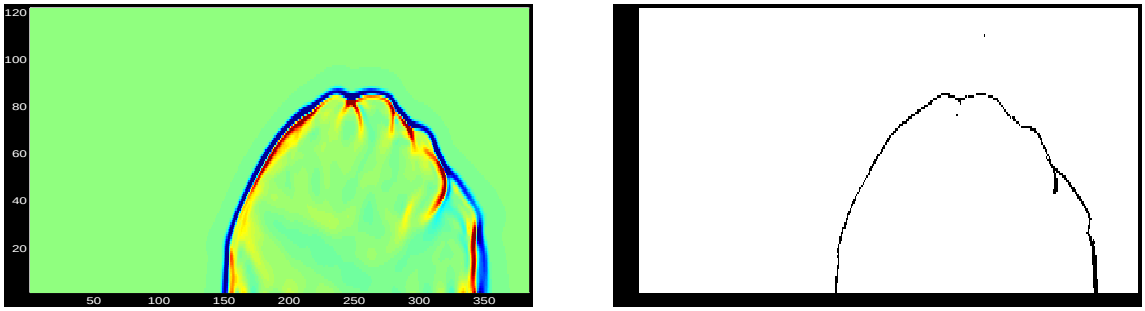

Figure 14:

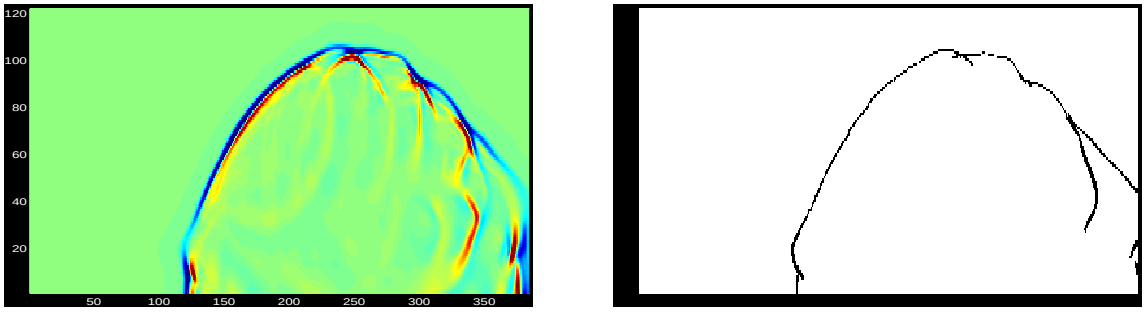

Figure 15:

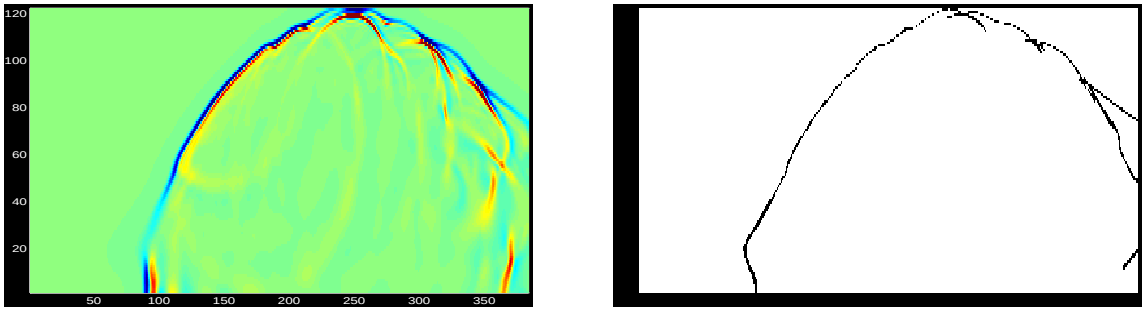

Figure 16:

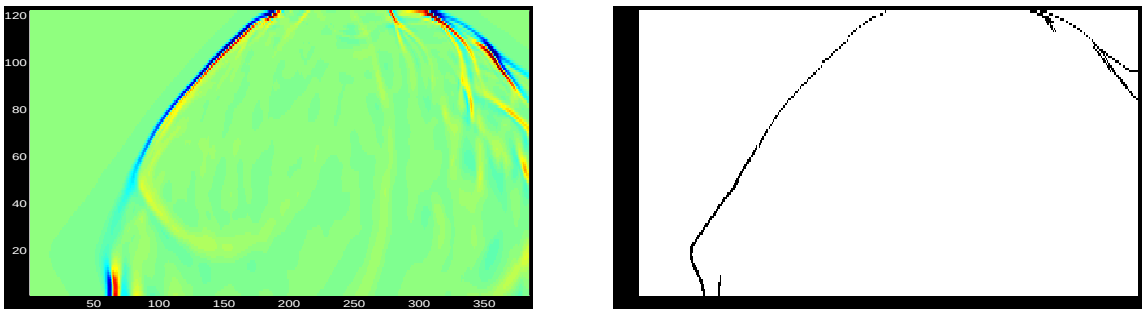

Figure 17:

A The big rays

Big rays generation First a ray tracing is performed in the model form the source point. We trace nr rays with initial direction discretizing an initial opening angle $[-\theta, \theta]$. The initial direction of the i^{th} ray is $\dot{X}_i(0) = (\cos(\theta_i), \sin(\theta_i))$ with $\theta_i = -\theta + i \times \frac{2\times\theta}{nr}$. We then have nr rays labeled R_i for $i = 1, \ldots, nr$. Each ray is a set of point defining a broken line from the source to the point where the ray exit the global computational domain.

Each big ray is defined as the envelope of three successive rays. More precisely, it is the smallest closed and connected domain containing these three rays and bounded by the boundary of the global computation domain. The chances of encountering a true focal point common to these three rays is very small. It therefore guarantees that the big ray will keep a minimum thickness. We build $int(\frac{nr+1}{2})$ big rays labeled Br_j, for $j = 1, \ldots, int(\frac{nr+1}{2})$. Big ray Br_j is built using rays three rays R_i, $i = 2 * (j-1) + k$ and $k = 1, 2, 3$.

The big ray is an approximation of the envelope of all the rays shot in the initial cone of directions defined by the first and third considered ray. It should hope be a domain for which all these rays are turned into global minima for the *state constrained* optimal control problem stated in section 2. The viscosity solution then gives the corresponding travel-time.

This strategy aims at decomposing the phase space. This can be seen near the source point where there is no multivaluedness. The big rays have no overlap. Every potential ray shot in the initial opening angle of the experiment 'lives' only in one big ray. When multivaluedness occurs, the different branches of the multi-valued solutions are split and live in different big rays which overlap in the physical space. Zones where big rays overlap correspond to the physical domain where multi-valued travel occurs.

Meshing of the big rays Meshing these big rays was done using Matlab PDE toolbox [14]. We wrote a script automatically taking into account the different cases when rays are crossing or not.

We logically compute all the zones in the big rays bounded by rays or segments of rays. Then we get rid of all the points on these rays which are strictly contained in the big ray. Finally we mesh the big ray. The mesher is based on a Delaunay algorithm.

In the case where the three rays are close and therefore the big ray thin, the meshing may need very small triangles. In order to avoid situations where the big ray would be only one triangle thick, we systematically refine the mesh by splitting every triangle into four triangles. This is done twice.

B Description of the eikonal solver

The aim of this section is to provide a detailled formulation of the scheme. We are given a triangulation of a domain Ω. The source term S belongs to Ω. The nodes are listed $M_i, i = 1, \ldots, n_s$, the triangles are $T_j, j = 1, \ldots, n_t$.

Hence the data structure is given by a set of nodes given by their coordinate $(x(i), y(i), i = 1, \ldots, n_s$, a set of triangles given by the index in the previous list of their nodes $(nu(j,1), nu(j,2), n(j,3), j = 1, \ldots, n_t)$ and a set of boundary type $(\log(i), i = 1, \ldots, n_s)$ determining whether the point is in the interior, on the boundary or a source point.

We define a pointer between the nodes $\{M_i\}$ and the set of triangles $N_i = \{T_j^1, \ldots, T_j^{k_i}\}$ of all the triangles that have M_i as vertex. For each node, we compute the interior radius ρ_i of the molecule $\mathcal{M}_\rangle = \cup_{|\in\mathcal{N}_\rangle} \mathcal{T}_|$ (see figure 18). In practice, we take the maximum on $T \in N_i$ of the distance between M_i and the side of T opposite to M_i.

Now, given the data u_i, we consider its piecewise linear interpolation still denoted by u, $u(M_i) = u_i$. In each triangle T with the nodes $M_{i_1}, M_{i_2}, M_{i_3}$, the restriction of u in T is the only linear polynomial

$$u_{|T} = u_{i_1}\Lambda_1^T + u_{i_1}\Lambda_2^T + u_{i_1}\Lambda_3^T$$

where for $k = 1, 2, 3$, $\Lambda_k^T(M_{i_l}) = 1$ if $M_{i_l} = M_{i_k}$ and 0 else. The Λ_k^T are nothing but than the barycentric coordinates with respect to triangle T as in standard finite element methods. The gradient of u is constant T and given by

$$\nabla_T u = u_{i_1}\nabla\Lambda_1^T + u_{i_1}\nabla\Lambda_2^T + u_{i_1}\nabla\Lambda_3^T.$$

The terms $\nabla\Lambda_l^T, l = 1, \ldots 3$ can be recomputed.

Then, for each node M_i belonging to T, we consider the angle at M_i, namely θ_i^T, and we define

$$\Theta_i = \sum_{T\in N_i} \theta_i^T.$$

Of course $\Theta_i = 2\pi$, except when $\log(M_i) > 0$, i.e. M_i is on the boundary. In this case N_i only contains triangles which are *inside* the domain. This is important with regard to the Soner boundary condition as we will see below.

We can now define the numerical Hamiltonian

$$\mathcal{H}(\mathcal{M}_\rangle, \sqcap) := \mathcal{H}(\nabla_{\mathcal{T}_\infty}\sqcap, \ldots, \nabla_{\mathcal{T}_{\#\mathcal{N}_\rangle}}\sqcap) = ||\mathcal{U}_\rangle|| - \frac{\infty}{\times_\rangle}\sum_{|\in\mathcal{N}_\rangle} \alpha_\rangle^{\mathcal{T}}\nabla_{\mathcal{T}}\sqcap \,.\, \vec{\backslash}_{\mathcal{T}},$$

$$U_i = \frac{\sum_{T\in N_i} \theta_i^T \nabla_T u}{\Theta_i}$$

$$\alpha_i^T = \tan\frac{\theta_i^T}{2}$$

$$\vec{n}_T = \frac{1}{2}\left(\overrightarrow{M_iM'} + \overrightarrow{M_iM''}\right).$$

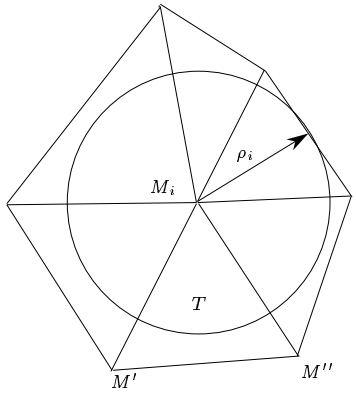

Figure 18: Geometrical elements for the Hamiltonian

The points $\{M_i, M', M''\}$ are the three vertices of T, see Figure 18.

The algorithm is the following. We initialize $u_i^0 = 0$ and set a stopping criterion called Res to a very large value and assign a threshold to a small one (typically 10^{-3}). The loop in time is

```
until ( Res < Tol) do                                  ! until convergence
   for i = 1, ..., ns do                               ! Loop on mesh points
    If(M_i = S) then                                   ! Special treatement for the source point
      u_i^{n+1} = 0
      else
      Compute Δt_i                                     ! Compute the local time step, see below
      u_i^{n+1} = u_i^n − Δt_i (H(M_i, u^n) − n(x_i, y_i))   ! Evolution step
      Res = max(Res, H(M_i, u^n))                      ! Compute the residual for the convergence test
    end if
  end do
end until
```

We just have to specify how to compute the local time step Δt_i. The terminology "local" comes from a Computational Fluid Dynamic technique devoted to the calculation of steady flows. In these calculations, as it is the case here, the time should be seen as an iteration parameter and is completely artificial. Hence, the best time step should be the one which maximizes the convergence speed of the scheme. Given two different mesh points, the "time steps" for these two points have no reason to be equal since they are chosen to guaranty the stability of the scheme via relation (10)

$$\frac{\Delta t_i}{\rho_i} = CFL \leq \frac{1}{2} \tag{10}$$

and the Courant Friedrisch Lewy number, CFL is uniform throughout the domain. In practice the relation (10) can be relaxed to $CFL \leq 1$. In (10), ρ_i is the diameter of the molecule $\mathcal{M}_\rangle$ and is computed as suggested above.

At steady state, the solution depends on a local domain in the "upwind" direction given by the rays of the geometrical optics. This is why the Lax-Friedrich scheme has two terms. The dissipation $(\frac{1}{\Theta_i}\sum_{j\in N_i}\alpha_i^T\nabla_T u)$ introduced in the numerical Hamiltonian is the quantity needed to balance the 'destabilizing' effect of the 'average' U_i. This last term is 'destabilizing' because part of its support is generally outside the "upwind" domain of dependence of the solution. In that sense, the Lax Friedrichs scheme is not *stricto sensu* an upwind scheme, as would be the Godunov scheme for example, but the amount of dissipation is tuned to damp the 'destabilizing' effect of U_i.

With the Soner boundary condition the rays are implicitly assumed to cut the boundary or be tangent to it. In that case the automatic restriction to the computational domain in the definition of the numerical Hamiltonian takes into account the absence of "upwind" contribution from the outside. The Soner boundary condition is *automatically* taken into account [1].

Remark 1 *The expression of the numerical Hamiltonian can be simplified provided the data structure is slightly modified. Now, for each node M_i, together with N_i, we consider S_i the list of edges $[M_i, M_k]$ where M_i is one end point. It can be seen that*

$$\mathcal{H}(\mathcal{M}_\rangle, \sqcap) = ||\mathcal{U}_\rangle|| - \sum_{[\mathcal{M}_\rangle, \mathcal{M}_\parallel]\in\mathcal{S}_\rangle} \alpha_\rangle^\parallel \left(\sqcap_\parallel - \sqcap_\rangle\right).$$

The value α_i^k is defined as follows: $[M_i, M_k]$ is the common edge between two triangles T and T' (in N_i), and

$$\alpha_i^k = \frac{\alpha_i^T + \alpha_i^{T'}}{2}.$$

C A second order scheme

This scheme is derived from the first order one. The main loop is replaced by

until ($Res < Tol$) do
  for $i = 1, ..., ns$ do
    If($M_i = S$) then
      $u_i^{n+1} = 0$
      else
      $\Delta t_i =$
      $u_i^{n+1/2} = u_i^n - \frac{\Delta t_i}{2}\left(\mathcal{H}_{\rangle}^{*,\backslash} - \backslash(\S_{\rangle}, \dagger_{\rangle})\right)$
      $u_i^{n+1} = u_i^n - \Delta t_i\left(\mathcal{H}_{\rangle}^{*,\backslash+\infty/\in} - \backslash(\S_{\rangle}, \dagger_{\rangle})\right)$
      $Res = \max(Res, H(M_i, u^n))$
    end if
  end do
end until

What is left is to define $\mathcal{H}_{\rangle}^{*,\backslash}$ and $\mathcal{H}_{\rangle}^{*,\backslash+\infty/\in}$. We set

$$\mathcal{H}_{\rangle}^{*,\backslash} = \mathcal{H}(\widetilde{\nabla_{\mathcal{T}_\infty}\sqcap}\backslash, \ldots, \widetilde{\nabla_{\mathcal{T}_{\#\mathcal{N}_{\rangle}}\infty}\sqcap}\backslash$$
$$\mathcal{H}_{\rangle}^{*,\backslash+\infty/\in} = \mathcal{H}(\widetilde{\nabla_{\mathcal{T}_\infty}\sqcap\backslash+\infty/\in}, \ldots, \widetilde{\nabla_{\mathcal{T}_{\#\mathcal{N}_{\rangle}}\infty}\sqcap}\backslash+\infty/\in).$$

In other words, we use the *same* numerical Hamiltonian as for the first order scheme with different values of the gradient of u in the triangles of N_i evaluated at the times $t_n = n\Delta t$ and $t_{n+1/2} = (n + 1/2)\Delta t$. This allows more modularity, and the only remaining thing is to define these new gradients.

This is done via the Essentially Non Oscillatory (ENO) technique [21]. Instead of interpolating u by a piecewise linear polynomial, we interpolate u *in each triangle* T by a quadratic polynomial. However, since singularities can be expected (the solution is not everywhere continuously differentiable in general), we cannot use a fixed stencil to compute this quadratic polynomial : it can be shown that such a scheme would be unstable. Since their location is not known *a priori*, an adaptive strategy is employed to avoid the potential singularities.

For any triangle T, we consider the family of four stencils as on Figure 19 They are constructed as follow, with the constraint that any such family should contain the three vertices of T. Any triangle T has 3 edges. If it is not too close from the boundary, then each edge of T belongs to 2 triangles : T and T_1 or T_2 or T_3 as on Figure 19. Consider, for $i = 1, .., 3$, each construction with T and T_i. T_i has 3 edges, then it is possible in general to find 2 new triangles, named T_i' and T_i''. The family $T, T_i, T_i', T_i''\}$ has 6 distinct nodes which constitute the stencils $\mathcal{S}_{\rangle}$ (for $i = 1, .., 3$). We construct a fourth stencil, $\mathcal{S}_{\rfloor\rceil\backslash\sqcup\nabla\dashv\updownarrow}$ as on Figure 19. Of course, if we are too close to the boundary, this procedure breaks down. This case happens only when the triangle T has at least one vertex on the boundary. In this

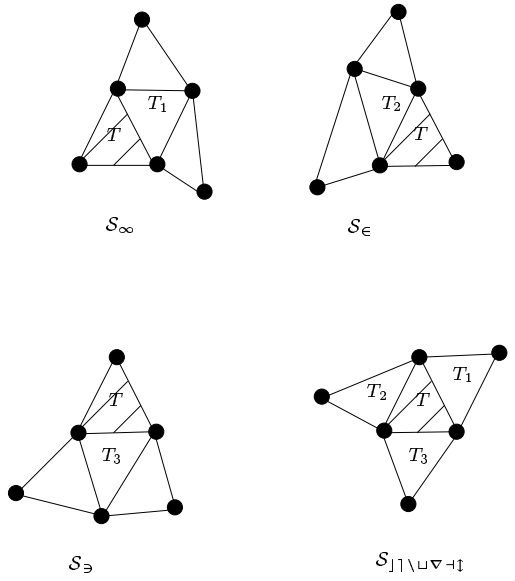

Figure 19: The four potential stencils

case, we switch back to the first order scheme.

Now taking any stencil, with its 6 vertices, it is possible in general to compute a polynomial of degree 2, so that we have 4 polynomials P in hand for each triangle. It interpolate the data at these 6 points. If (a,b) are the coordinate of any point of the stencil, we write

$$P(x,y) = a_0 + a_1(x-a) + a_2(y-b) + a_3(x-a)^2 + a_4(x-a)(y-a) + a_5(y-a)^2.$$

Among the 4 polynomials, we select the one, P^T_{min} which minimizes

$$|a_3| + |a_4| + |a_5|.$$

This ensures that we have picked up the smoothest quadratic approximation of the gradient. Then the argument $\widetilde{\nabla_{T_l} u^n}, T_l \in N_i$ is

$$\widetilde{\nabla_{T_l} u^n} = \nabla P^T_{min}(M_i).$$

More details on the calculation of P_{min} can be found in [3].

References

[1] R. Abgrall. Numerical approximation of the boundary condition for first order hamilton–jacobi equations. Technical report, Université de Bordeaux I. Work in preparation.

[2] R. Abgrall. Numerical Discretization of First Order Hamilton Jacobi Equations on Triangular Meshes. *Comm. in Pure and Applied Math.*, 1996. to appear.

[3] R. Abgrall and Th. Sonar. On the use of Muehlbach expansions in the recovery step of ENO methods. *Numer. Math.*, 1996. To appear, also DLR Report, August 1995, DLR, Bunsenstr. 10, D-37073 Goettingen, Germany.

[4] P. Podvin an Lecompte I. Finite-difference computation of traveltimes in very contrasted models: a massively parallel approach and its associated tools. *Geophys. J. Int.*, 105:271–284, 1991.

[5] O. Runborg B. Engquist. Multi-phase computation in geometrical optics. Technical report, Nada KTH, 1995.

[6] S. Osher B. Engquist, E. Fatemi. Numerical resolution of the high frequency asymptotic expansion of the scalar wave equation. *J. Comp. Physics*, 120:145–155, 1995.

[7] M. Bardi and S. Osher. The nonconvex multi-dimensional Riemann problem for Hamilton-Jacobi equations. *SIAM J. Math. Anal.*, 22(2):344–351, March 1991.

[8] G. Barles. *Solutions de viscosité des équations de Hamilton–Jacobi.* Number 17 in Mathématiques et Applications. Springer Verlag, 1994.

[9] J.-D. Benamou. Big ray tracing : Multivalued traveltime field computation using viscosity solution of the eikonal equation. Technical Report 2688, INRIA, 1996. To appear in *Journ. Comp. Physics.*

[10] F. Collino. High order absorbing boundary conditions for wave propagation models: Straight line boundary and corner cases. pages 161–171. SIAM, 1993.

[11] M.G. Crandall and P.L. Lions. Two approximations of solutions of Hamilton–Jacobi equations. *Math. Comp.*, 43(167), July 1984.

[12] A. Hanyga and J. Pajchel. Point to curve ray tracing in complex geological models. *Geophysical Prospecting*, 43:859–872, 1995.

[13] Lecompte I. Hybrid modelling with ray tracing and finite-difference. 1996.

[14] The Math Works Inc. *Partial Differential Equation TOOLBOX : User's guide.* 1995.

[15] W. W. Symes J. Van Trier. Upwind finite-difference calculation of traveltimes. *Geophysics*, 56:812–821, 1991.

[16] L. Klimes. Travel times in the inria marmousi models. *Private communication, short note*, 1996.

[17] L. Klimes, M. Kvasnicka, and V. Cerveny. Grid computations of rays and travel times. in : Seismic waves in complex 3-d structures, report 1. *Tech. Report, Department of Geophysics, Charles University Prague*, pages 85–114, 1994.

[18] A.P.E. Ten Kroode and R. Rietdyk. 1996.

[19] P.L. Lions. *Generalized solutions of Hamilton-Jacobi equations.* Pitman, 1982.

[20] T.J. Moser. Shortest path calculation of seismic rays. *Geophysics*, 56:59–67, 1991.

[21] S. Osher and C. W. Shu. High–order essentially nonoscillatory schemes for Hamilton–Jacobi equations. *SIAM J. Numer. Anal.*, 28(4), 1991.

[22] H. M. Soner. Optimal control problems with state space constraints. *SIAM J. Control and Optimisation*, 24:552–561, 1986.

[23] W. Symes. A slowness matching algorithm for multiple traveltimes. *preprint*, 1996.

[24] H. Gjoystdal V. Vinje, E. Iversen. Traveltime and amplitude estimation using wavefront construction. *Geophysics*, 58:1157–1166, 1993.

[25] J. Vidale. Finite-difference calculation of traveltimes. *Bull. Seis. Soc. Am.*, 78:2062–2076, 1988.

[26] H. M. Soner W. H. Fleming. *Controled Markov Processes and Viscosity Solutions*. Springer-Verlag, 1993.

[27] L.Corrias Y. Brenier. A kinetic formulation for multi-branch entropy solutions of scalar conservation laws. *Ann. IHP Analyse non-linéaire*, 1996. to appear.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399