Numerical Methods for Traveltimes and Amplitudes

William W. Symes

The Rice Inversion Project
Department of Computational and Applied Mathematics
Rice University

Mathematical Geophysics Summer School
Stanford University, August 1999

Outline

1. Hamilton-Jacobi Theory: wavefronts and rays
2. Lagrangian computation of traveltimes: Ray Tracing
   - 2.1 Fundamental concepts and notations
   - 2.2 Rays as solutions of initial value problems: Ray Shooting
   - 2.3 Wavefront Construction
   - 2.4 Rays as solutions of 2 point boundary value problems: Ray Bending

Outline cont'd

3. Eulerian computation of traveltimes: Eikonal Solvers

   3.1 Viscosity solutions: first arrival times

   3.1 Space Marching methods: an example

   3.2 Overview of algorithms

4. Reflections

5. Amplitudes

6. Eulerian approaches to computation of multiple arrival times

Hamilton-Jacobi Theory: Wavefronts and Rays

Wavefronts and orthogonal trajectories

Recall *eikonal equation* for wave phase or traveltime $T(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^3$ in velocity field $v(\mathbf{x})$:

$$v^2 \nabla T \cdot \nabla T = 1$$

Wavefronts are the surfaces $T = \text{const.}$

Orthogonal trajectories are curves perpindicular at each point to wavefront, i.e. with velocity vector parallel to ∇T

Rays and Hamilton's Equations

Physical units $\Rightarrow$ natural ODE for orthogonal trajectories is

$$\frac{\partial \mathbf{x}}{\partial t} = v^2(\mathbf{x})\nabla T(\mathbf{x})$$

Introduce *ray slowness vector* $\mathbf{p}(t) \equiv \nabla T(\mathbf{x}(t))$, so that $(\mathbf{x}(t), \mathbf{p}(t))$ forms a trajectory in *phase space* $\mathbf{R}^6$.

Claim: provided that T is of class C^2 in its domain of definition, the phase space trajectory ("ray") $(\mathbf{x}(t), \mathbf{p}(t))$ satisfies *Hamilton's Equations:*

$$\frac{d\mathbf{x}}{dt} = \nabla_{\mathbf{p}} H, \;\; \frac{d\mathbf{p}}{dt} = -\nabla_{\mathbf{x}} H$$

in which H is the *Hamiltonian* $H(\mathbf{x}, \mathbf{p}) = \frac{1}{2} v^2(\mathbf{x})\mathbf{p} \cdot \mathbf{p}$

Proof of Claim

$$\nabla_{\mathbf{p}} H = v^2 \mathbf{p}, \ -\nabla_{\mathbf{x}} H = -\frac{1}{2}\nabla(v^2)\mathbf{p}\cdot\mathbf{p}$$

so first equation is just ODE for orthogonal trajectories introduced above.

From the definition of **p**,

$$\frac{d\mathbf{p}}{dt} = \nabla\nabla T(\mathbf{x}) \cdot \frac{d\mathbf{x}}{dt} = v^2 \nabla\nabla T(\mathbf{x}) \cdot \nabla T(\mathbf{x})$$

$$= \frac{1}{2} v^2 \nabla(\nabla T(\mathbf{x}) \cdot \nabla T(\mathbf{x})) = -\nabla(\frac{1}{2}v^2)\nabla T(\mathbf{x}) \cdot \nabla T(\mathbf{x})$$

because of the eikonal equation; the last expression is just the second Hamilton equation in view of the definition of **p**. **q.e.d.**

Observations on Hamilton's Equations

Observation 1. For *any* solution $(\mathbf{x}(t), \mathbf{p}(t))$ of Hamilton's equations, $H(\mathbf{x}(t), \mathbf{p}(t))$ is independent of t (HW Problem 1!)

Observation 2. If $v^2(\mathbf{x}_0)\mathbf{p}_0 \cdot \mathbf{p}_0 = 1$, then the solution $(\mathbf{x}(t), \mathbf{p}(t))$ of Hamilton's equations with initial data $(\mathbf{x}_0, \mathbf{p}_0)$ satisfies

$$\frac{d\mathbf{x}}{dt} \cdot \frac{d\mathbf{x}}{dt} \equiv v^2(\mathbf{x})$$

(HW Problem 2!)

The Method of Characteristics 1

Hamilton's equations are *a priori* independent of any solution of the eikonal equation, but may be used to *construct* solutions, provided that v has Lipshitz continuous partial derivatives.

Ingredients:

- $S \subset \mathbf{R}^3$ bounded smooth surface
- function $T_0 : S \to \mathbf{R}$ = Cauchy data for eikonal equation; must satisfy $v^2 \nabla_S T_0 \cdot \nabla_S T_0 < 1$ ($\nabla_S =$ gradient along S)
- projection $\Pi_{\mathbf{x}_0}$ from $\mathbf{R}^3$ onto tangent space to S at $\mathbf{x}_0 \in S$

The Method of Characteristics 2

System of 2 linear and 1 quadratic equation:

$$\Pi_{\mathbf{x}_0}\mathbf{p} = \nabla_S T_0(\mathbf{x}_0); \;\; v^2(\mathbf{x}_0)\mathbf{p}\cdot\mathbf{p} = 1$$

has two solutions at each $\mathbf{x}_0 \in S$; make consistent (smooth) choice $\mathbf{p}_0 : S \to \mathbf{R}^3$.

At each $\mathbf{x}_0 \in S$ solve Hamilton's equations with initial data $(\mathbf{x}_0, \mathbf{p}_0(\mathbf{x}_0))$. Easy excercise in Implicit Function Theorem: $(\mathbf{x}_0, t) \mapsto \mathbf{x}(t)$ is an invertible, differentiable map for $-\epsilon < t < \epsilon, \mathbf{x}_0 \in S$ provided ϵ is small enough.

The Method of Characteristics 3

That is, for each point $\mathbf{x}$ close enough to S, there is a unique choice of $t, \mathbf{x}_0$ so that the ray with initial data $(\mathbf{x}_0, \mathbf{p}_0(\mathbf{x}_0))$ passes through $\mathbf{x}$ at time t. Define $T(\mathbf{x}) = t$.

Claim: T, defined near S as above, satisfies the eikonal equation.

Proof: By construction, $\nabla T || \frac{d\mathbf{x}}{dt}$, so

$$1 = \frac{d}{dt} T(\mathbf{x}(t)) = \nabla T(\mathbf{x}(t)) \cdot \frac{d\mathbf{x}}{dt}(t) = c \frac{d\mathbf{x}}{dt}(t) \cdot \frac{d\mathbf{x}}{dt}(t)$$

where c is the ratio of the lengths of $\nabla T(\mathbf{x}(t))$ and $\frac{d\mathbf{x}}{dt}$.

Method of Characteristics 4

From Observation 2, this is

$$= cv^2(\mathbf{x}(t)) \;\Rightarrow c = v^{-2}(\mathbf{x}(t)) \;\Rightarrow$$

$$\nabla T(\mathbf{x}) \cdot \nabla T(\mathbf{x}) = v^{-4}(\mathbf{x}) \frac{d\mathbf{x}}{dt} \cdot \frac{d\mathbf{x}}{dt} = v^{-2}(\mathbf{x})$$

q.e.d

HW Problem 3: show that the solution $T(\mathbf{x})$ constructed by the method of characteristics satisfies
$\mathbf{p}(t) = \nabla T(\mathbf{x}(t))$ as expected.

Observations on the M. of C.

- provides a *local* construction of solutions to the eikonal from Cauchy data on a smooth surface - requires that v have Lipschitz derivatives (so must be modified for less regular v, eg. with discontinuities)
- *fails* where rays cross - because then definition of T becomes ambiguous. Apparently T may be *multivalued* in general.

Another form of the Ray Equations

$$\frac{d\mathbf{x}}{dt} = v^2(\mathbf{x})\mathbf{p}, \;\; \frac{d\mathbf{p}}{dt} = -v(\mathbf{x})\,\nabla v(\mathbf{x})\,\mathbf{p}\cdot\mathbf{p}$$

Reserve the term *ray* for solutions $(\mathbf{x}(t), \mathbf{p}(t))$ of this system satisfying the addtional condition $\mathbf{p}\cdot\mathbf{p} = v^{-2}(\mathbf{x})$ - only these solutions have expected relation with eikonal $(\mathbf{p} = \nabla T(\mathbf{x}))$

For rays, second equation is equivalent to

$$\frac{d\mathbf{p}}{dt} = -\nabla \log v(\mathbf{x})$$

A second order ray equation

Differentiate the first ray equation and eliminate $\frac{d\mathbf{p}}{dt}$ using second to get

$$\frac{d^2\mathbf{x}}{dt^2} = 2\left(\nabla v(\mathbf{x}) \cdot \frac{d\mathbf{x}}{dt}\right)\frac{d\mathbf{x}}{dt} - v(\mathbf{x})\nabla v(\mathbf{x})$$

First term: acceleration *parallel* to instantaneous direction of ray. Second term: acceleration *perpindicular* to ray

Simple consequences ray equations

- rays bend toward *decreasing velocity*
- If $v =$ const (indepedent of $\mathbf{x}$), then $\mathbf{p} =$ const along each ray, i.e. ray doesn't change direction, i.e. ray is *straight line traversed at constant speed.*
- If v is independent of some space coordinate, say x, then corresponding component of $\mathbf{p}$ is constant along ray.

Example: Constant Velocity

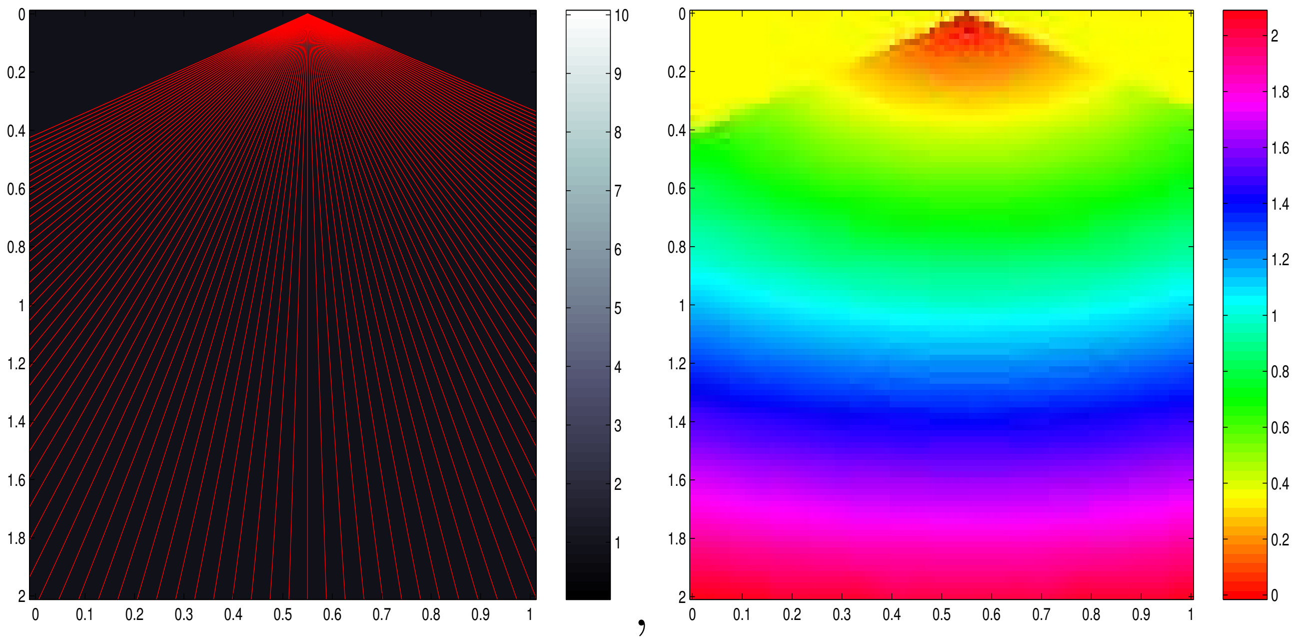


Figure 1: Left: rays from x_s=0.55, 101 equally spaced angles; Right: interpolation from irregularly sampled rays onto regularly sampled grid - time defualted to 0 in region of no coverage

Yet another equivalent form...

In terms of Slowness field $u(\mathbf{x}) = 1/v(\mathbf{x})$, arc length along ray s:

$$\frac{d}{ds}\left(u(\mathbf{x})\frac{d\mathbf{x}}{ds}\right) = \nabla u(\mathbf{x})$$

Derivation from Hamiltonian form: HW Problem 4

Follows from this: if $\mathbf{x}(s)$ is space component of ray, parametrized by arc length, and $\mathbf{x}_s = \mathbf{x}(s_0), \mathbf{x}_r = \mathbf{x}(s_1)$, then

$$T(\mathbf{x}_r) - T(\mathbf{x}_s) = \int_{s_0}^{s_1} u(\mathbf{x}(s))\, ds$$

(derivation: HW Problem 5)

Regularity and Fermat

Standard Thm on local existence, uniqueness of rays: v must have Lipshitz continuous derivatives.

Extension possible to less regular velocity fields (eg. piecewise smooth) *via* relation to *Fermat's Principle*: (space component of) ray makes the traveltime integral

$$\int u\, ds$$

stationary amongst all C^0 paths connecting $\mathbf{x}_s$ to $\mathbf{x}_r$.

Note relaxed regularity of path in statement of Fermat: can also relax regularity of v (eg. to piecewise C^1_{Lip})

Still another form

2D case (3D similar): $\mathbf{x} = (z, x)$

Because $H = \frac{1}{2}$ on any ray, can write $\mathbf{p} = v^{-1}(\mathbf{x})(\cos\theta, \sin\theta)$ and describe ray in terms of z, x, θ:

$$\frac{dz}{dt} = v(z, x)\cos\theta$$

$$\frac{dx}{dt} = v(z, x)\sin\theta$$

$$\frac{d\theta}{dt} = \frac{\partial v}{\partial z}\sin\theta - \frac{\partial v}{\partial x}\cos\theta$$

This form of ray equations particularly convenient for computation.

Two Points of View

Traveltime is:

- a quantity which evolves along rays
- a solution to the eikonal equation

Corresponding families of numerical methods:

- Lagrangian: solve the ray equations numerically ("ray tracing")
- Eulerian: solve the eikonal equation numerically ("eikonal solvers")

Lagrangian computation of traveltimes:

Ray Tracing = Method of Characteristics

Problem Statement

Given:

- source point x_s, receiver points $x_r \in R$
- velocity field $v(\mathbf{x})$

Find:

- traveltime(s) from $\mathbf{x}_s$ to each $\mathbf{x}_r$, or equivalently a ray $(\mathbf{x}, \mathbf{p})$ and a time t for which $\mathbf{x}(0) = \mathbf{x}_s, \mathbf{x}(t) = \mathbf{x}_r$ - then t is (a) traveltime from x_s to x_r
- variant # 1: find the *smallest* such traveltime
- variant # 2: find *all* such times

Lagrangian Method 1: Shooting

Idea: sample a circle (sphere in 3D) of $\mathbf{p}$ of radius $v^{-1}(\mathbf{x}_s)$; for each sample $\mathbf{p}_s$, approximate a ray by solving the initial value problem

$$\frac{d\mathbf{x}}{dt} = v^2(\mathbf{x})\mathbf{p}, \;\; \frac{d\mathbf{p}}{dt} = -\nabla \log v(\mathbf{x})$$

$$\mathbf{x}(0) = \mathbf{x}_s, \;\; \mathbf{p}(0) = \mathbf{p}_s$$

numerically, using either general- or special-purpose solver.

Must *interpolate* times on rays to output points somehow.

Lagrangian Method 1: Shooting

Example implementation: `ray.m`, a Matlab ray tracer

Principle: apply general purpose ODE/IVP solver to ray equations

Inputs: source and receiver coordinates, a function `vel.m` returning the velocity and its partial derivatives *at any point in space* (gridded velocity $\Rightarrow$ interpolation)

Outputs: graphical representation of rays, interpolated times at receiver points

ODE solver: `ode113` (Matlab variable order nonstiff solver - could use others); interpolation is linear, using `griddata`

Example: Constant Velocity

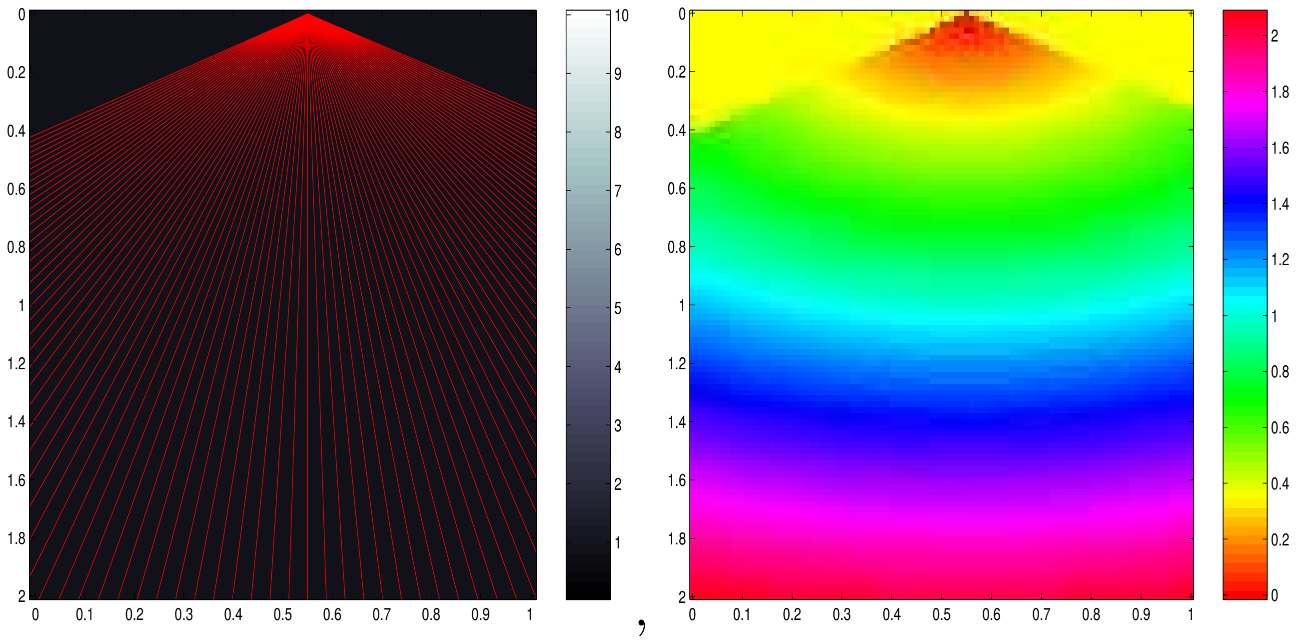


Figure 2: Left: rays from x_s=0.55, 101 equally spaced angles; Right: interpolation from irregularly sampled rays onto regularly sampled grid - time arbitrary in region of no coverage

Example: Linear Velocity

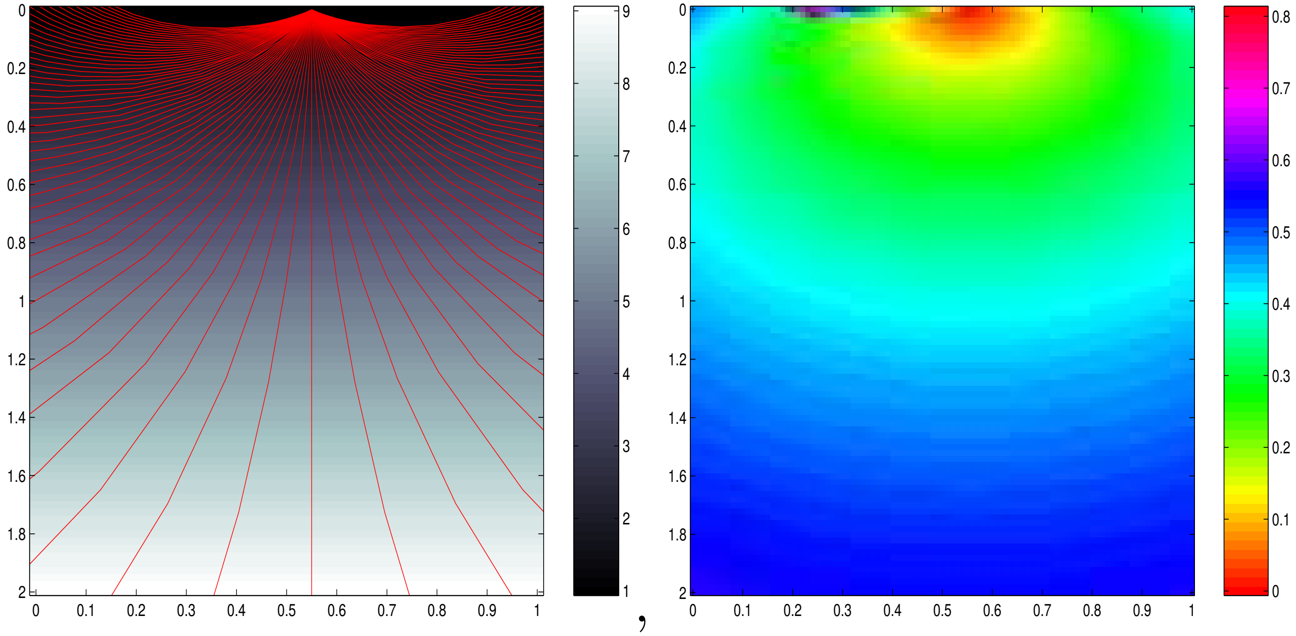


Figure 3: $v = 1 + 4z$. Left: rays from x_s=0.55, 101 equally spaced angles; Right: interpolation from irregularly sampled rays onto regularly sampled grid - time arbitrary in region of no coverage

Special ray equation solvers

Many based on the observation:

- in linear velocity, rays are arcs of circles

(Proof: HW Problem 6)

Favorite model parametrization: $v(\mathbf{x})$ piecewise linear. Fermat $\rightarrow$ C^0, piecewise C^2 "generalized rays".

$\Rightarrow$ exact (generalized) solution of ray equations in piecewise linear velocity field: link circular arcs by Snell's law at boundaries (eg. Chander, 1975) $\Rightarrow$ much more efficient than general purpose IVP solver *for these special models*

Variants: approximate circular arcs by quadratics in t, lose transcendentals (eg. Langan et al. 1985)

Effects of Ray Bending

Recall: rays bend towards slower regions

- example `lens1`: a fast region in center - rays bend away
- example `lens2`: a slow region in center - rays bend toward center, begin to cross
- example `lens3`: an even slower region - rays bent more strongly $\Rightarrow$ 0 - 3 rays pass over each pt $\Rightarrow$ interpolation very confused
- example `lens4`: several strong lenses - complex ray pattern, interpolation is hopeless.

Example: A Fast Lens

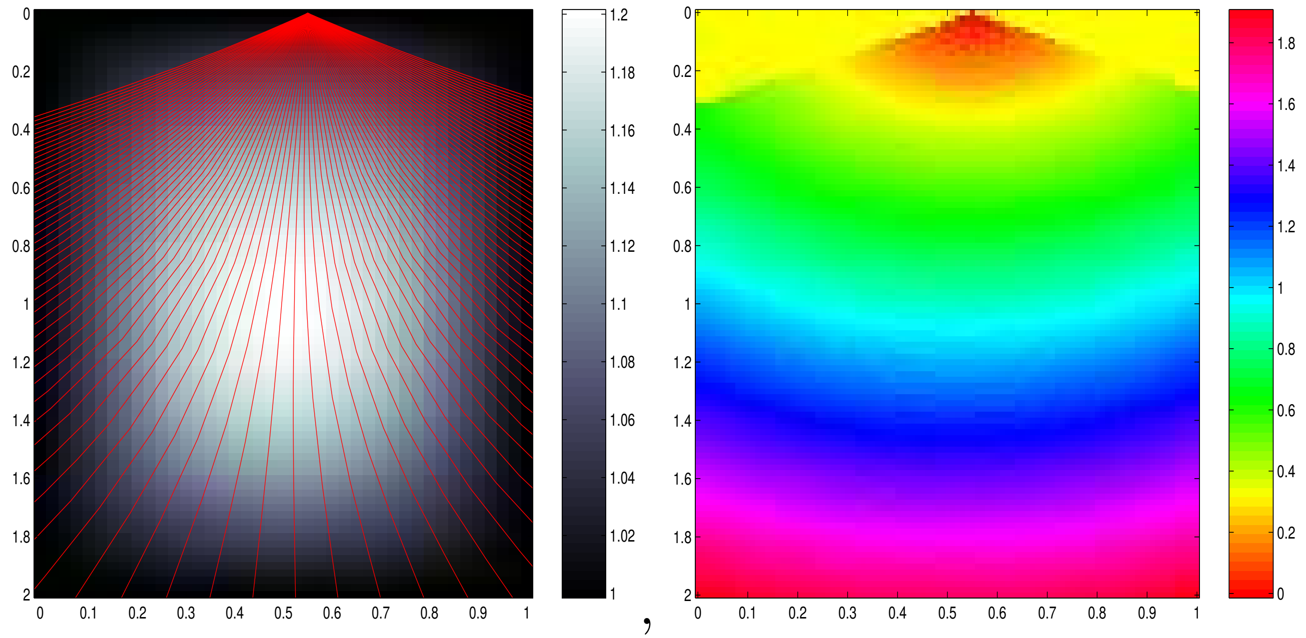


Figure 4: $v = 1 + d(f_x \sin \pi x + f_z \cos \pi z)$, $d = 0.2, f_x = 1, f_z = 0.5$. Left: rays from x_s=0.55, 101 equally spaced angles; Right: interpolation from irregularly sampled rays onto regularly sampled grid

Example: A Slow Lens

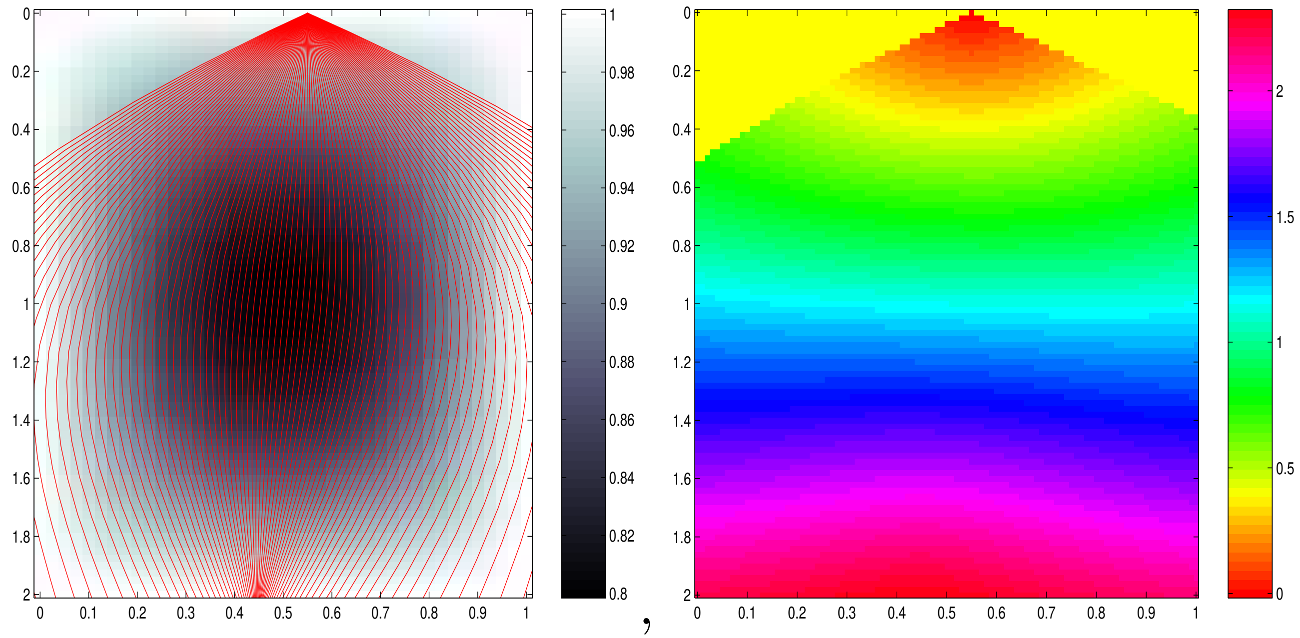


Figure 5: $v = 1 + d(f_x \sin \pi x + f_z \cos \pi z)$, $d = -0.2, f_x = 1, f_z = 0.5$. Left: rays from x_s=0.55, 101 equally spaced angles; Right: interpolation from irregularly sampled rays onto regularly sampled grid

Example: A Slower Lens

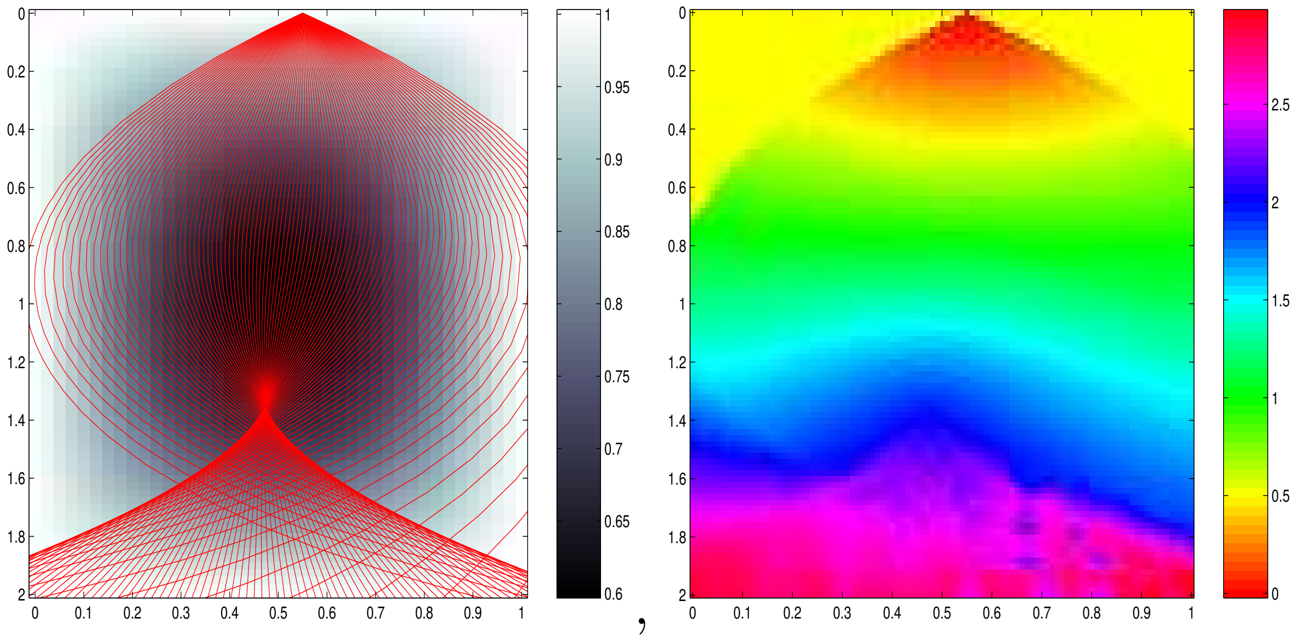


Figure 6: $v = 1 + d(f_x \sin \pi x + f_z \cos \pi z)$, $d = -0.4, f_x = 1, f_z = 0.5$. Left: rays from x_s=0.55, 101 equally spaced angles; Right: interpolation from irregularly sampled rays onto regularly sampled grid

Example: A Complex Lens

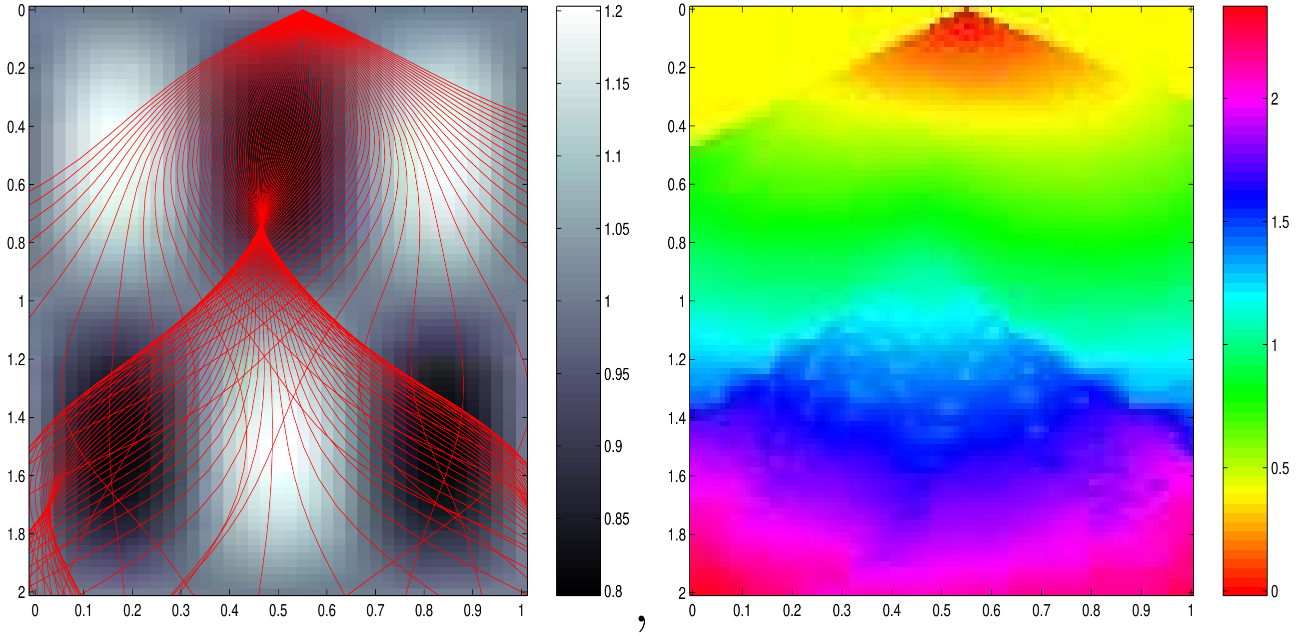


Figure 7: $v = 1 + d(f_x \sin \pi x + f_z \cos \pi z)$, $d = -0.4, f_x = 3, f_z = 1$. Left: rays from x_s=0.55, 101 equally spaced angles; Right: interpolation from irregularly sampled rays onto regularly sampled grid

Ray Shooting Issues: Shadow Zones

- no rays $\Rightarrow$ no data for interpolation
- possibile solution: shoot more rays
- appropriate selection of rays not evident *a priori*, must be discovered dynamically.
- **This is a limitation of ray shooting.**

Ray Shooting Issues: Multiple Arrivals

- generally, several rays pass over each point at sufficient distance from source
- $\Rightarrow$ traveltime is multivalued
- Cannot interpolate to give a sensible single-valued output
- a solution (variant # 1): discard all arrivals but one
- another solution (variant # 2): interpolate a multivalued function (!).
- **This a feature of the traveltime field, not of any particular computational method.**

Wave kinematics are multivalued!

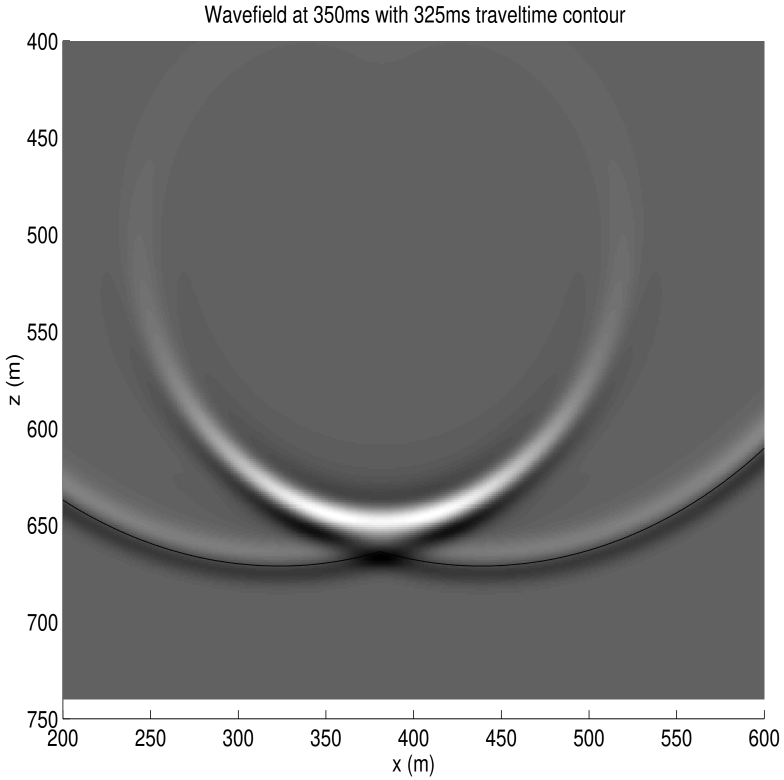


Figure 8: Accurate finite difference solution of wave equation with low velocity lens, showing presence of multiple arrivals

Lagrangian Method 2: Wavefront Construction

Vinje et al (1993): advance all rays of interest by uniform steps in time (thus implicitly advancing *wavefront* = constant time surface). When rays grow too far apart, interpolate new rays. When rays converge, delete rays. Thus maintain more or less uniform ray density.

- some use in exploration geophysics
- can either keep least-time ray (variant # 1) or all rays (variant # 2)
- complex, dynamic data structure $\Rightarrow$ relatively expensive

Lagrangian Method 3: Two Point Ray Tracing

Idea: to find time for wave to travel from x_s to x_r, solve the BVP

$$\frac{d\mathbf{x}}{dt} = v^2(\mathbf{x})\mathbf{p},\ \ \frac{d\mathbf{p}}{dt} = -\nabla \log v(\mathbf{x})$$

$$\mathbf{x}(0) = \mathbf{x}_s;\ \ \mathbf{p}(0) \cdot \mathbf{p}(0) = v^{-2}(x_s);\ \ \mathbf{x}(t_r) = \mathbf{x}_r;$$

Unknowns are trajectory $\mathbf{x}(t), \mathbf{p}(t)$, and arrival time t_r.

Lagrangian Method 3: Two Point Ray Tracing

Three general approaches:

A. Shooting: regard $\mathbf{x}(t_r) = F_{\text{shooting}}[\mathbf{x}(0), \mathbf{p}(0), t_r]$ as function of $\mathbf{x}(0), \mathbf{p}(0), t_r$ (with constraint that $\mathbf{p}(0) \cdot \mathbf{p}(0) = v^{-2}(\mathbf{p}_s)$,; solve the system

$$F_{\text{shooting}}[\mathbf{x}(0), \mathbf{p}(0), t_r] = \mathbf{x}_r$$

B. Bending: regard the BVP itself as a large nonlinear system, solve somehow

C. Continuation: solve BVP in "simpler" velocity field (eg. $v_0 =$ const.), evolve along a 1 param. family of models $v_0 \to v$. (Keller & Perozzi, 1983)

Lagrangian Method 3B: Ray Bending

Solve discretized BVP as nonlinear system for $\mathbf{x}(t), \mathbf{p}(t)$, and t_r [or equivalent system].

Two general approaches:

- *ad hoc* discretization, system solver (eg. Newton with constant damping) (Julian & Gubbins 1977, Farra 1992,...)
- "standard NA technology" for 2 pt bdry problems: variable order, variable step discretization, quasiNewton method with globalization (linesearch or trust region) (Pereyra et al 1980).

Ray Bending in PW Constant Velocity

Perhaps the most popular algorithm of all!

Velocity, slowness = piecewise constant on rectangular grid

Rays = straight lines inside cells, bend according to Snell at cell boundaries $\Rightarrow$ satisfy Fermat

+: Very easy to implement - leads to ray path matrix model for linearized tomography $Ms = t$, cf. Berryman lectures

-: Leads to nonconvergent tomographic results under grid refinement, see Delprat-Jannaud and Lailly 1993.

Multiple Arrivals impede Ray Bending

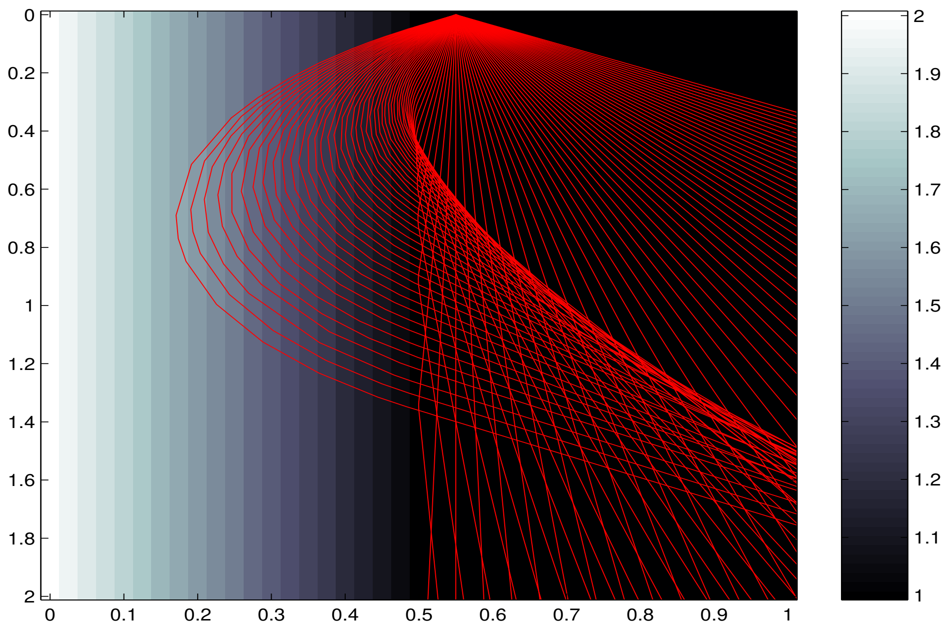


Figure 9: Rays in piecewise linear velocity: $v(z,x) = 2 - 2x$ if $x < 0.5$, $v(z,x) = 1$ else. For any point (z_r, x_r) in $x > 0.5$, beginning bending with the straight line from $\mathbf{x}_s$ **guarantees** that the curved rays through the same point will **never** be found! (see Farra 1992, Snieder & Spencer 1993)

Comparison

- All 2 pt ray tracing methods approximate solution by (i) discretization, (ii) iteration (Newton,...)
- relatively expensive compared to ray shooting + interpolation, maybe more accurate

$\Rightarrow$ *roughly,*

Shooting + interpolation for large number of output pts with modest accuracy, **2 pt ray tracing** for small number of output pts with low $\rightarrow$ high accuracy

NB: *many caveats - choice must be problem driven*

Eulerian computation of traveltimes: Eikonal Solvers

Eikonal Equation

$$|\nabla T(\mathbf{x})| = u(\mathbf{x}) = \frac{1}{v(\mathbf{x})}$$

Requires side conditions to determine solution uniquely:

- give $T(\mathbf{x})$ along smooth hypersurface
- point source (limit of small spheres):

$$|T(\mathbf{x}) - T(\mathbf{x}_s) - u(\mathbf{x}_s)\|\mathbf{x} - \mathbf{x}_s\|| = o(\|\mathbf{x} - \mathbf{x}_s\|)$$

Local vs. global solutions

For smooth $u(\mathbf{x})$, Method of Characteristics (= ray tracing) gives smooth solution near datum set (hypersurface, source point)

Problem: like many nonlinear PDEs, global smooth solutions do not exist: method of characteristics breaks down where characteristics = rays **cross**

Another gloss: implicit in statement of eikonal equation is single-valuedness of $T(\mathbf{x})$. However we already know that traveltime is generally multivalued at sufficient distance from point source or other initial value surface.

Viscosity solutions

Since smooth solutions do not exist (globally), must generalize concept of solution.

Viscosity solutions = C^0, piecewise C^1 fcns satisfying eikonal equation whereever possible + add'l conditions. Existence, uniqueness for $v \in C^0$: Lions 1982 (def'ns, proofs).

Related idea: *entropy solutions* of conservation laws

Lions 1982: The viscosity solution of the point source problem is the first arrival time field.

$\Rightarrow$ gridded eikonal solvers compute first arrival time.

Other Space Marching Methods

Large literature - see reference list. *Very* small selection:

- unidirectional Godunov schemes
- Vidale's scheme (1988)
- Sethian's "Fast Marching" method (1996)

Unidirectional Space Marching

Idea: view eikonal equation as evolution equation in a space variable, at least locally (perhaps different space variables in different places): for example, if you choose $+z$, get

$$\frac{\partial T}{\partial z} = \sqrt{u^2 - \left(\frac{\partial T}{\partial x}\right)^2}$$

(2D version - 3D version similar).

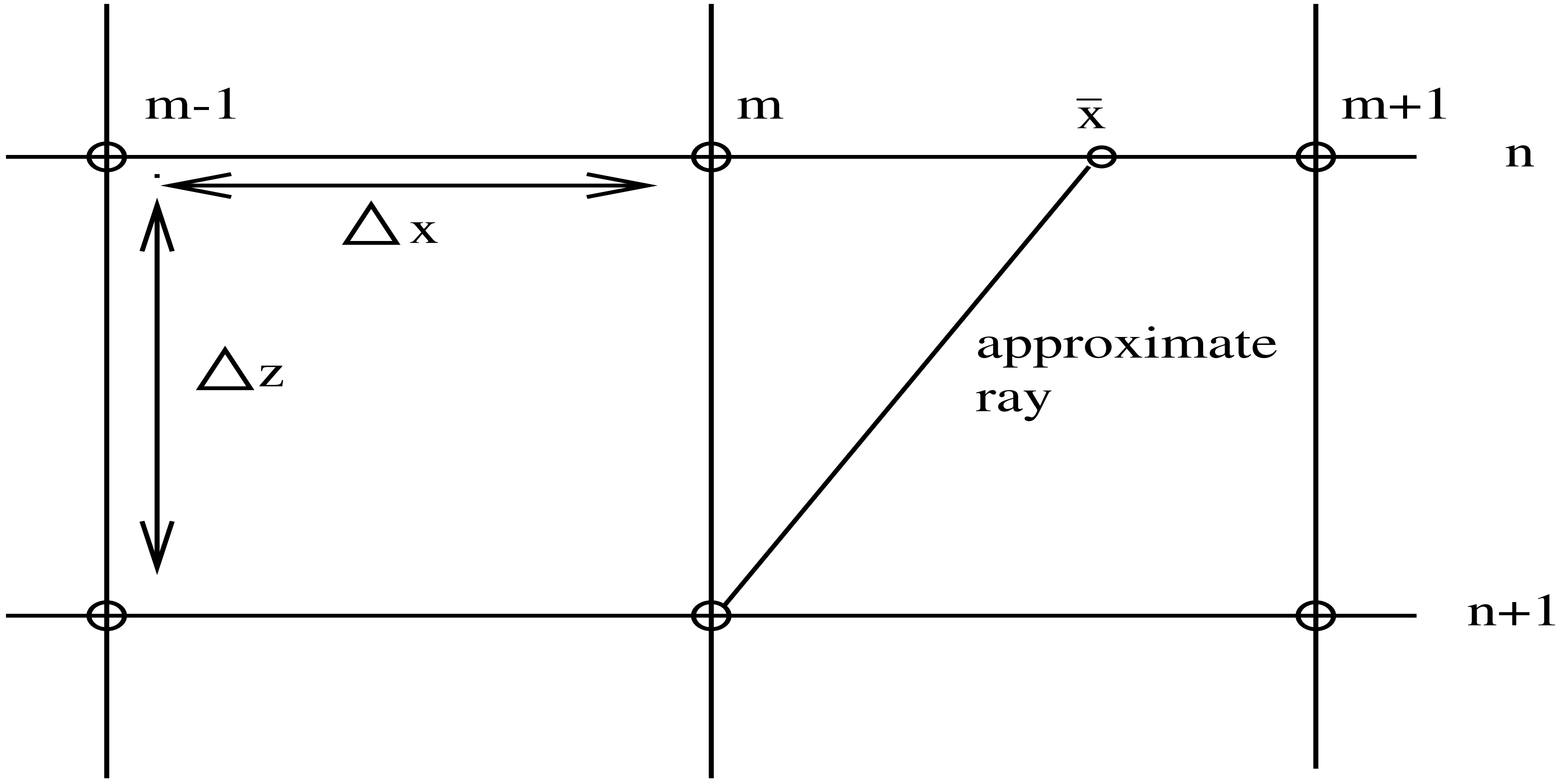
Simple Difference Scheme

Aim: produce approximations to traveltime $T(\mathbf{x})$ on a *regular grid* $(m\Delta x, n\Delta z)$:

$$T_m^n \simeq T(m\Delta x, n\Delta z)$$

Principle: use connection between rays, eikonal to extrapolate times from depth level n to depth level $n+1$ by approximate backwards ray tracing

Geometry of Simple Scheme



Approximations

Locally, approximate

- slowness u as constant $= u_m^n = u(m\Delta x, n\Delta z)$
- time on level n as piecewise linear
- rays as straight lines

Ray to the target point $(m\Delta x, (n+1)\Delta z)$ intersects level n at $(\bar{x}, n\Delta z)$

T_m^{n+1} = time at intersection point $(\bar{x}, n\Delta z)$ + time along ray segment

Time at intersection point

Three possibilities:

R: ray comes from right: $m\Delta x < \bar{x} < (m+1)\Delta x$

L: ray comes from left: $(m-1)\Delta x < \bar{x} < m\Delta x$

O: ray comes from outside $[(m-1)\Delta x, (m+1)\Delta x]$

Assume first - second similar, third handled at end

Then $T(\bar{x}, n\Delta z) \simeq \bar{T} = T_m^n + (\bar{x} - m\Delta x)D_x^+ T_m^n$, where

$$D_x^{\pm} T_m^n = \pm \frac{T_{m\pm 1}^n - T_m^n}{\Delta x},$$

Time along Ray Segment

Locally const slowness approximation $\Rightarrow$ time increment along ray segment = slowness $\times$ length

$$= u_m^n \sqrt{(\bar{x} - m\Delta x)^2 + \Delta z^2}$$

Thus

$$T_m^{n+1} = T_m^n + (\bar{x} - m\Delta x) D_x^+ T_m^n + u_m^n \sqrt{(\bar{x} - m\Delta x)^2 + \Delta z^2}$$

Determining $\bar{x}$

Key ingredients:

1. value of Hamiltonian: $\mathbf{p} \cdot \mathbf{p} = u^2$
2. first ray equation $\dot{\mathbf{x}} = v^2 \mathbf{p}$
3. basic Hamilton-Jacobi equation: $\nabla T = \mathbf{p}$

Write $\mathbf{p} = (q, p)$.

1. $\Rightarrow q = \sqrt{u^2 - p^2}$

2. $\Rightarrow$ slope of ray $= \frac{\dot{x}}{\dot{z}} = \frac{p}{q} = \frac{p}{\sqrt{u^2 - p^2}}$

Determining $\bar{x}$

Therefore

$$\bar{x} - m\Delta x = \frac{\dot{x}}{\dot{z}}\Delta z = \frac{p}{\sqrt{(u^n_m)^2 - p^2}}\Delta z$$

Get two conclusions from 3:

(1) From 1. get $p = D^+_x T^n_m$. Since $q = u^2\dot{z} > 0, p = u^2\dot{x} < 0$ for ray as in diagram, assumption (R) above is possible only if $D^+_x T^n_m < 0$

(2)

$$T^{n+1}_m = T^n_m + \frac{p^2\Delta Z}{\sqrt{(u^n_m)^2 - p^2}} + u^n_m \Delta z \sqrt{\frac{p^2}{(u^n_m)^2 - p^2} + 1}$$

Synthesis

A little bit of algebra (HW Problem 7), plus $p = D_x^+ T_m^n$, gives

$$T_m^{n+1} = T_m^n + \Delta z \sqrt{(u_m^n)^2 - (D_x^+ T_m^n)^2}$$

in case (R) i.e. when $D_x^+ T_m^n < 0$.

Similarly in case (L) ($D_x^- T_m^n > 0$)

$$T_m^{n+1} = T_m^n + \Delta z \sqrt{(u_m^n)^2 - (D_x^- T_m^n)^2}$$

Choices

Note that one, both, or neither of (R) or (L) could occur.

- if exactly one occurs, then preceding slide gives formula for T_m^{n+1}
- if *both* occur, then select smaller value of T_m^{n+1} from above two formulae $\Rightarrow$ *this scheme will compute the first arrival time = viscosity solution!*
- if *neither* occur, because $D_x^- T_m^n \leq 0 \leq D_x^+ T_m^n$, then rays are *diverging*; compute time increment using vertical ray,

$$T_m^{n+1} = T_m^n + \Delta z u_m^n$$

Convenient form

HW Problem 8: show that the above four cases can be subsumed in a single update formula:

$$T_m^{n+1} = T_m^n +$$

$$\Delta z \sqrt{(u_m^n)^2 - \max((\max(0, D_x^- T_m^n))^2, (\min(0, D_x^+ T_m^n))^2)}$$

= *Godunov scheme* (eg. Osher and Sethian 1988)

Properties:

- formally first order accurate for z-marching eikonal
- "upwind": difference in direction from which ray comes

Practicalities

Still dangling: case (O), in which rays are converging but from outside the interval $[(m-1)\Delta x, (m+1)\Delta x]$.

If this occurs, algorithm becomes unstable! Example of Courant-Friedrichs-Lewy instability of difference schemes.

Remedy: legislate such steep rays out of existence!

Gray and May 1994: modify eikonal to *paraxial eikonal*

$$\frac{\partial T}{\partial z} = \sqrt{\max\left(u^2 - \left(\frac{\partial T}{\partial x}\right)^2, u^2 \cos^2 \theta\right)}$$

Correctly computes first arrivals along rays making angle $< \theta$ with z-axis.

Matlab 1st Order Godunov

Program `eik.m`

Input parameters: function returning velocity field and partial derivatives as in `ray.m`, location of source, computational rectangle, Δx, aperture limit θ - Δz computed internally to satisfy CFL stability criterion for Godunov scheme, paraxial eikonal

Output: traveltime field throughout computational rectangle, displayed in color contour plot.

Examples

- A slower lens (**`lens3`**): note that eikonal solver returns much smoother traveltime field estimate than shooting + interpolation. Eikonal gives first arrival time, lower part returned by shooting is not *any* traveltime. Note transition locus from left-arriving rays to right-arriving rays: this is analogous to *shock* in fluid mechanics.

- A piecewise linear velocity (**`badbend`**): eikonal solver returns complete field (though incorrect for points outside aperture because of paraxial modification). No shadow zone!

Example: a slower lens

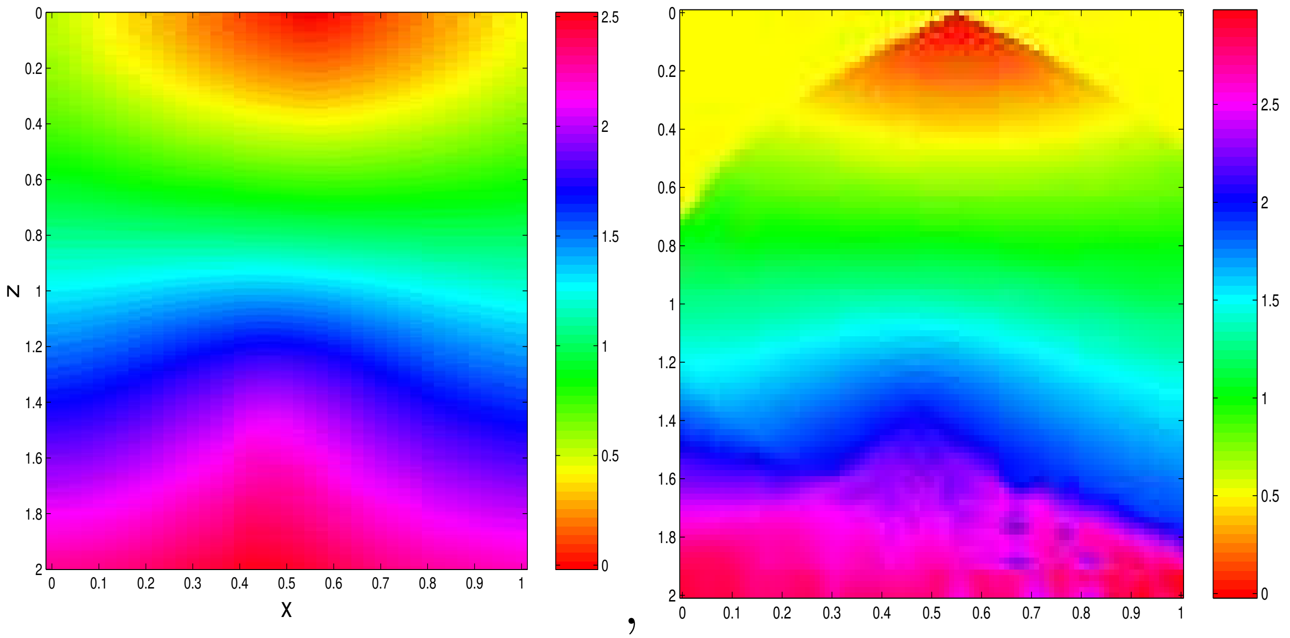
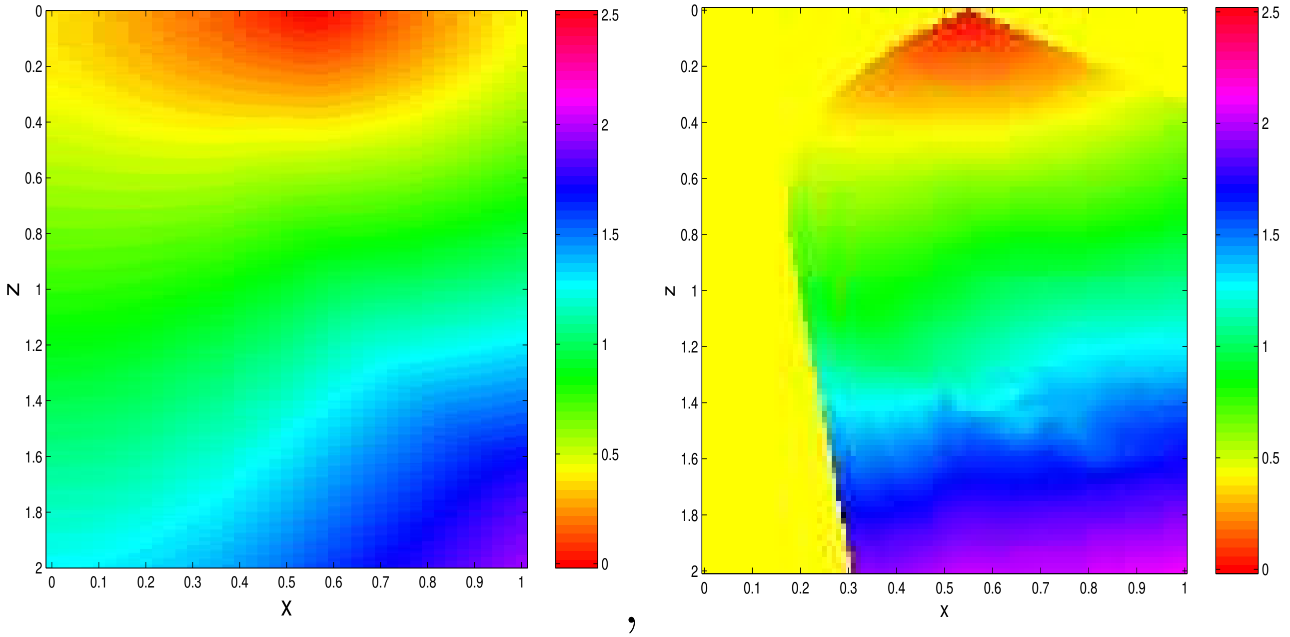


Figure 10: $v = 1+d(f_x \sin \pi x + f_z \cos \pi z)$, $d = -0.4$, $f_x = 1$, $f_z = 0.5$. Left: eikonal solver; Right: shooting + interpolation

Example: piecewise linear velocity



,

Figure 11: piecewise linear velocity: $v(z, x) = 2 - 2x$ if $x < 0.5$, $v(z, x) = 1$ else. Left: eikonal solver; Right: shooting + interpolation. Note that times are similar where only one ray passes; also absence of shadow zone in eikonal result.

Extensions, performance

- "**E**ssentially **N**on**O**scillatory" order > 1 extensions of Godunov & similar schemes: Osher and Sethian 1988, Osher and Shu 1991
- *much* more efficient than 1st order, needed for amplitude computation
- cost $= O(N)$, $N = n_x n_y ... =$ number of gridpoints
- limited by paraxial assumption: correctly computes times along first-arriving rays making angle $< \theta$ with vertical - OK for some apps, not for others

Vidale's Scheme

Vidale, 1988: first generally reliable solver, got the field going. Features:

- *implicit* upwind scheme
- space marching in all directions - 4 in 2D, 6 in 3D
- second order

Vidale's scheme

Concept: given times T on a square array, extrapolate to neighboring rows and columns of grid cells. [Array is initially a point.]

- find minima of T on sides of box
- first: at minima, gradient of T should be normal to box so extrapolate T linearly: if grid step is Δ, then $T_{\text{next}} = T_{\min} + u\Delta$

Vidale's scheme

- in other grid cells with corners A, B, C, and D, solve centered difference approximation on diagonal grid

$$(T_A - T_C)^2 + (T_B - T_D)^2 = (\Delta x^2 + \Delta z^2)\frac{u_A^2 + u_B^2 + u_C^2 + u_D^2}{4}$$

  to compute T at 4th corner given T at other three

- start at cells next to mins where 3 values are known, work out from mins - in effect, with proper ordering system is *block triangular*

NB: implicit and upwind

Local Geometry of Vidale scheme

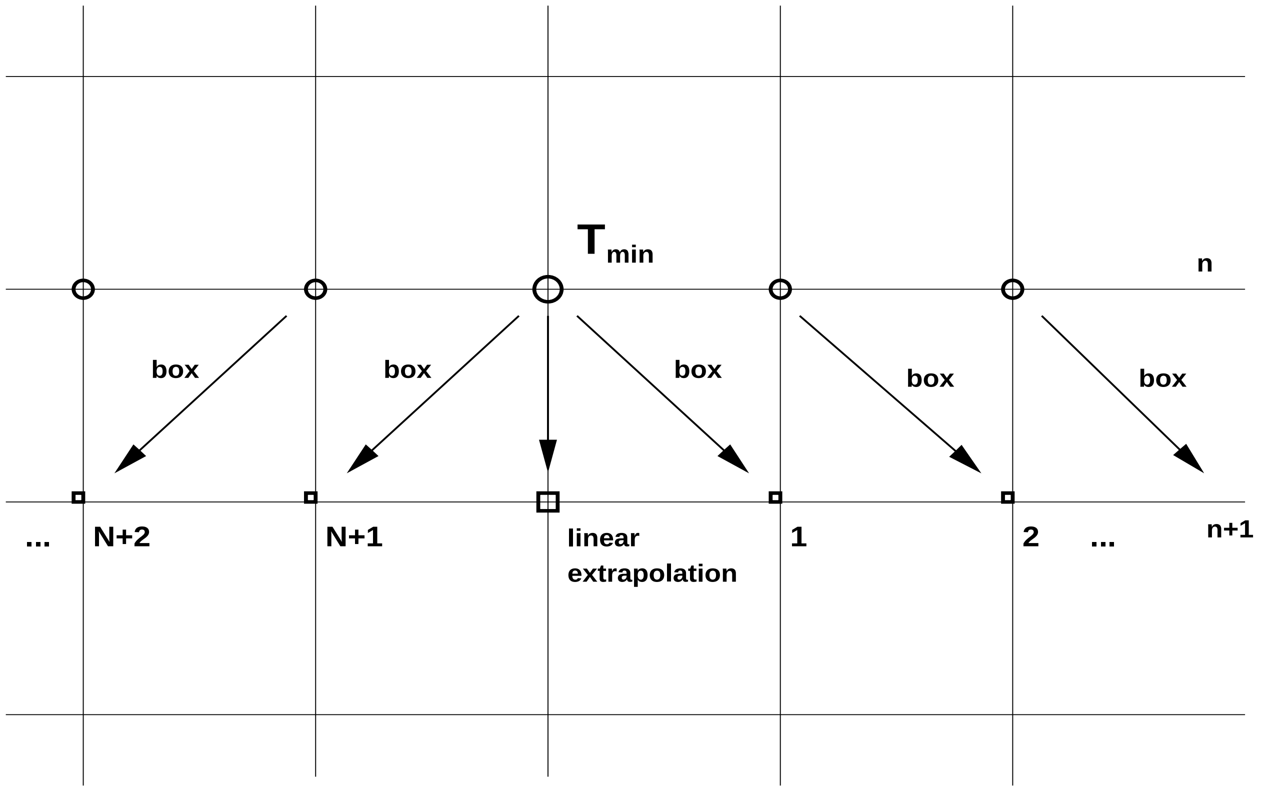


Figure 12: Extrapolation along normal at minimum time point performed first (large circle → large box), then box scheme used to update other points in order indicated

Box scheme

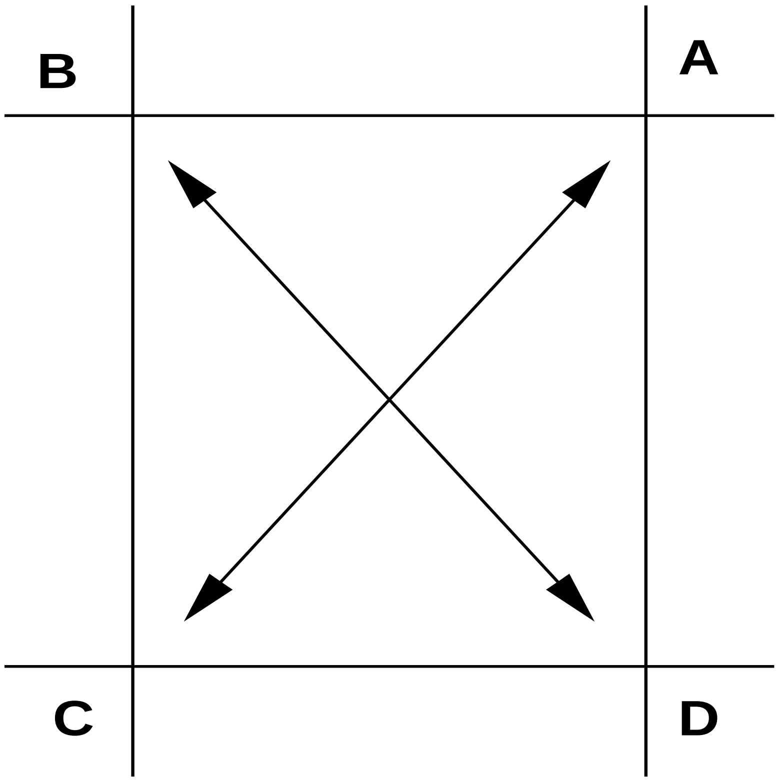


Figure 13: Given T_Z, T_B, T_C, for example, find T_D by solving $(T_A - T_C)^2 + (T_B - T_D)^2 = (\Delta x^2 + \Delta z^2) u_{\mathrm{avg}}$

Vidale's scheme

Advantages:

- propagates times in all directions, extends to 3D
- good accuracy; cost $= O(N)$, $N = n_x n_y ... =$ number of gridpoints

Disadvantages:

- stops (system inconsistent) when ray carrying first arrival is tangent to side of expanding box
- requires finding minima on sides

Sethian's Fast Marching Method

- Sethian 1996, Sethian & Popovici 1999, Sethian 1999
- for many references, seismic & nonseismic applications, and demos:

  `www.math.berkeley.edu/~sethian`

Based on upwind *discrete eikonal*:

$$\begin{array}{l}(\max(D_x^- T_m^n, -D_x^+ T_m^n, 0))^2 + \\ (\max(D_z^- T_m^n, -D_z^+ T_m^n, 0))^2\end{array} = (u_m^n)^2$$

Fast Marching - Initialization

Presume T given on initial subgrid ($\sim$ initial value surface - may be single point) - forms initial set of *accepted points*

Assign $T = \infty$ at all other *inactive points*

NB: At a gridpoint next to one or more with finite T values, *only* differences with known values contribute to LHS!

Solve discrete eikonal to assign values to all *neighbors* of initial (accepted) points: these form initial set of *active points* (no longer inactive)

Fast Marching - Propagation

Repeat until all points are accepted:

1. Select active point with *smallest* value of T, move it to accepted set
2. Solve discrete eikonal to assign T values to all neighbors of accepted points, move these to active set (actually needs to be done only with neighbors of point selected in 1.)

That's it!

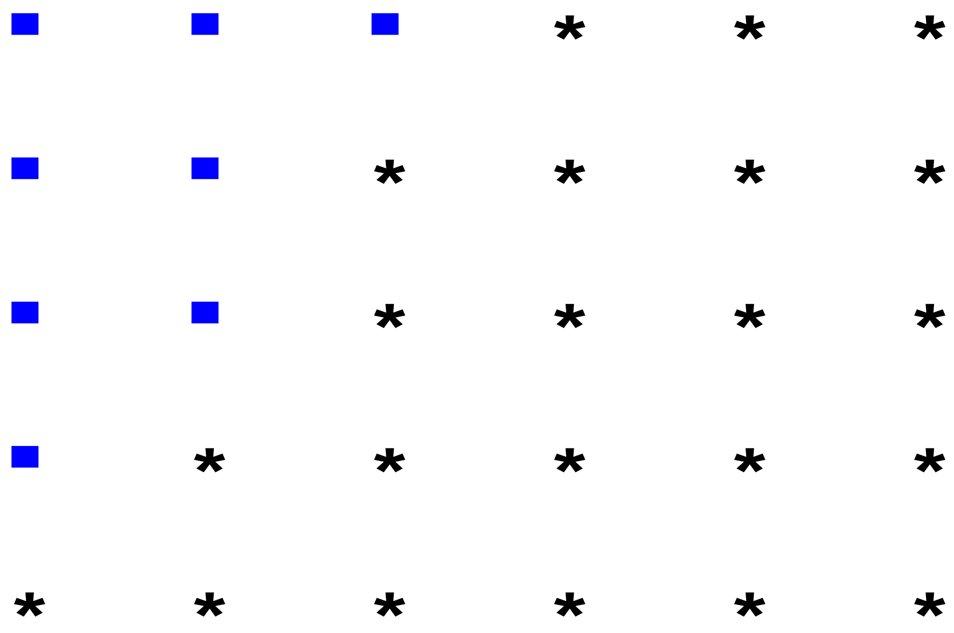


Figure 14: Fast Marching Preinitialization: initial data forms initial *accepted point set* = blue squares, all other points are *inactive*, assigned value $T = \infty$ = asterisks

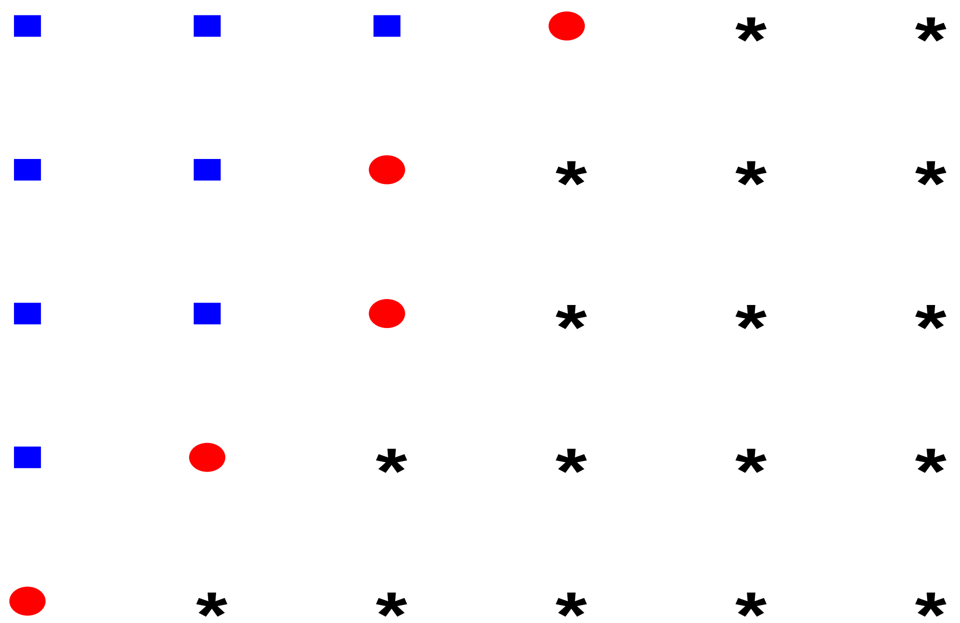


Figure 15: Fast Marching Initialization: assign tentative T values to all neighbors of accepted points using discrete eikonal; these form initial *active point set* = red circles

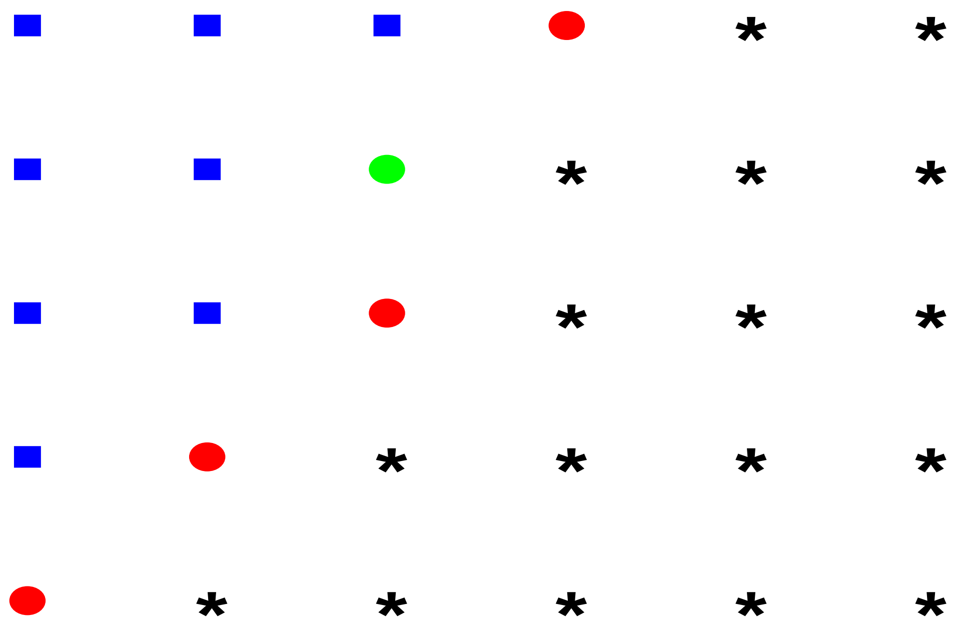


Figure 16: Fast Marching Propagation, step 1A: select active point having *smallest value of* T = green circle

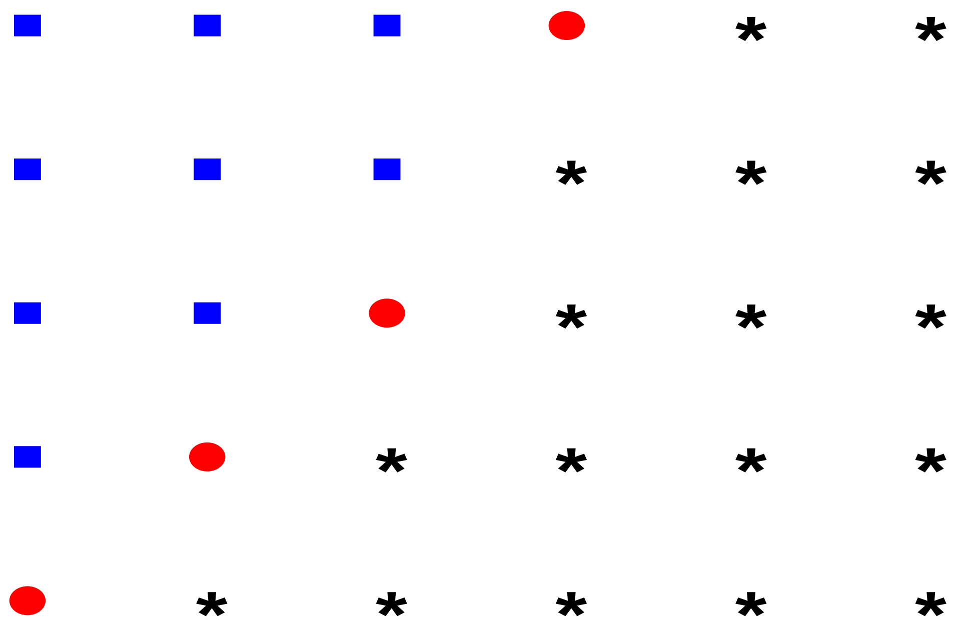


Figure 17: Fast Marching Propagation, step 1B: move active point with smallest time to accepted point set

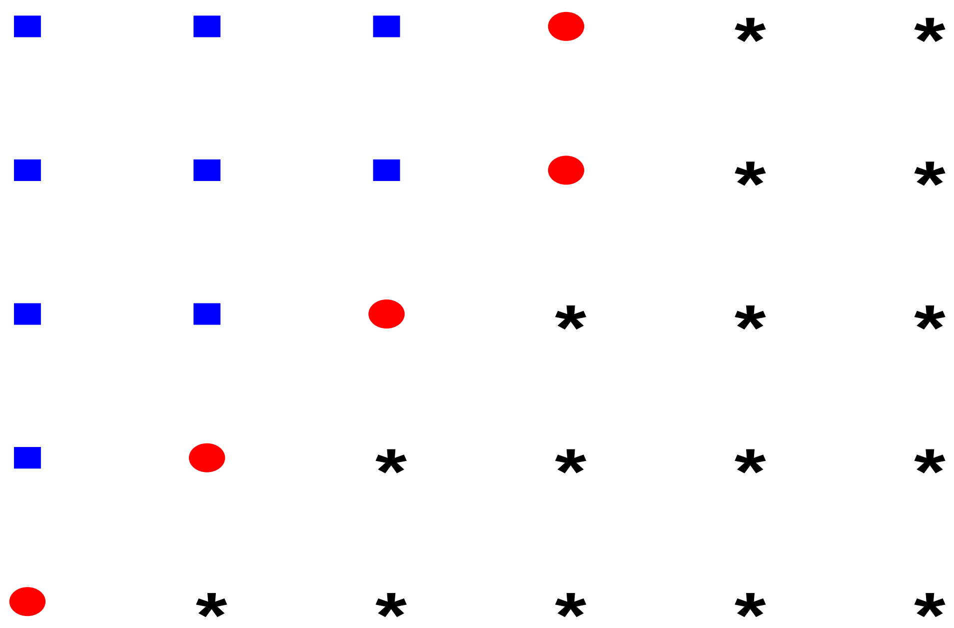


Figure 18: Fast Marching Propagation, step 2: identify all neighbors of accepted points, assign T values using discrete eikonal, move to active set

Fast Marching - Implementation

- sorting step 1. done once per gridpoint - with *heap sort*, cost is logarithmic $\Rightarrow$ total cost $= O(N \log N)$, $N = n_x n_y ...$ total number of gridpoints
- higher order variants more efficient for some seismic applications - see Sethian 1999

Reflections

Reflections

Weak form of wave equation + asymptotic analysis (or physics) leads to *Snell's law for reflected waves*: the time of arrival for a wave reflected from an interface S is the sum of times along *incident* and *reflected* rays. These rays meet at a point on the surface, making equal angles with its normal.

Incident ray $(\mathbf{x}_d, \mathbf{p}_d)$ connects source $\mathbf{x}_s$ to $\mathbf{x}_* \in S$ in time t_d; reflected ray $(\mathbf{x}_u, \mathbf{p}_u)$ connects $\mathbf{x}_*$ to receiver $\mathbf{x}_r$ in time t_u; time of arrival $t = t_u + t_d$.

Reflections

Equal angles law: bisector of $\mathbf{p}_d, \mathbf{p}_u$ parallel to surface normal ν_S. Since $\mathbf{p}_d(t_d), \mathbf{p}_u(t_u)$ have same length (eikonal equation!),

$$(\mathbf{p}_d(t_d) + \mathbf{p}_u(t_u)) \times \nu_S(\mathbf{x}_*) = 0$$

This is a system of equations! Either

- trace candidate incident and reflected rays, or
- use eikonal solver to get $\mathbf{p}_u = \nabla T_u$ etc.

In either case find reflection point $\mathbf{x}_*$ etc. by solving equal angles condition.

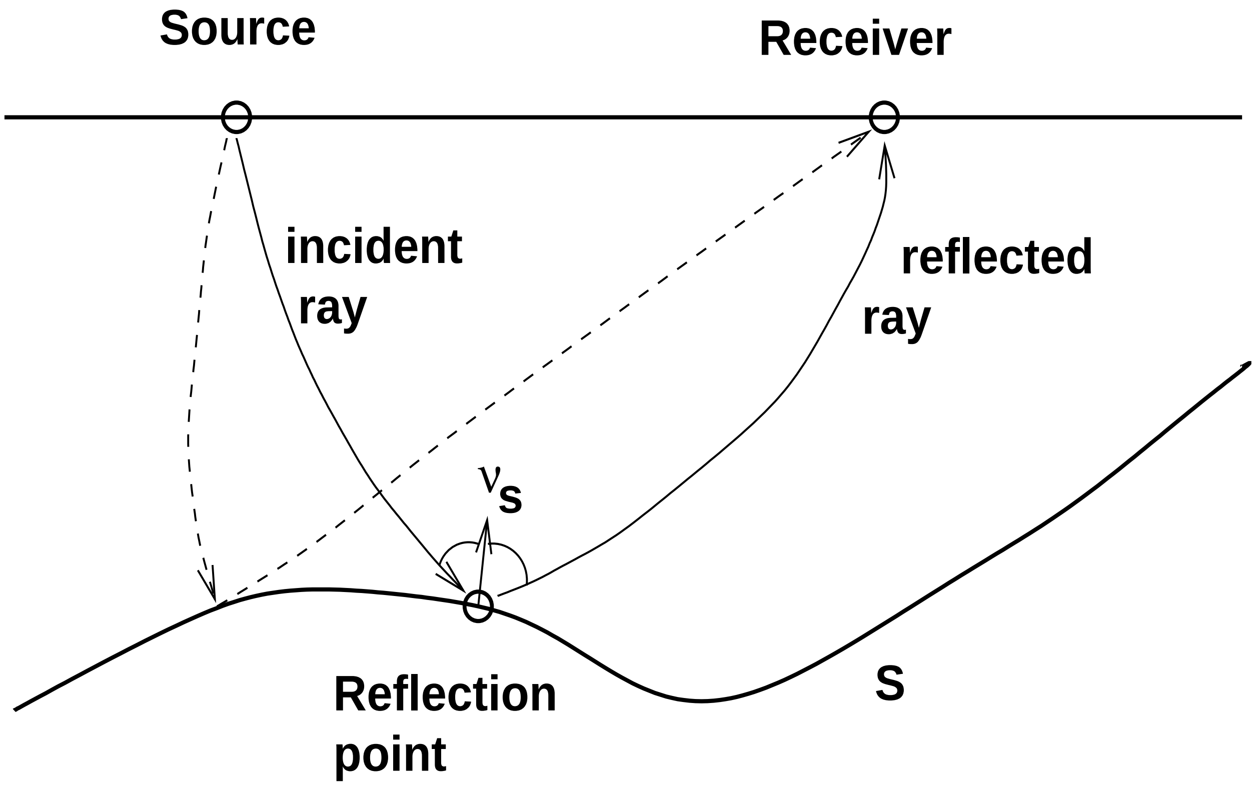


Figure 19: Trace rays (or compute traveltimes) from source and receiver to points on the surface, and find those where the equal angles condition holds (there may be more than one - is there here?)

Amplitudes

Amplitudes 1

Recall from first lecture: to leading order in frequency, wave amplitude A satisfies the *transport equation*

$$2\nabla T \cdot \nabla A + (\nabla^2 T)A = 0$$

[**NB:** this A is the *exponential* of the A in Berryman's lecture.]

Given known T, this is an *advection equation* for A. Just as for eikonal, two choices:

- method of characteristics
- finite difference method on a grid

Amplitudes 2

- method of characteristics is *trivial* for linear scalar equations: along a ray,

$$\frac{dA}{dt} = -v^{-2}(\nabla^2 T)A$$

solve by quadrature

- gridded solver: use equivalent form

$$\nabla \cdot (A^2 \nabla T) = 0$$

plus divergence theorem (Courant & Hilbert 1962 Ch. 6, Friedlander 1958 Ch.1)

Amplitudes 3

The divergence theorem gives

$$0 = \int_{\partial R} \nu \cdot (A^2 \nabla T)$$

for any volume $R \in \mathbf{R}^n$, ($n = 2$ or 3).

Take for R a *ray tube* (see figure); then

$$0 = \int_{D_1} \frac{A^2}{v} - \int_{D_0} \frac{A^2}{v}$$

Shrinking the caps D_0 and D_1 about points $\mathbf{x}_0$ and $\mathbf{x}_1$ on the axial ray, in limit get

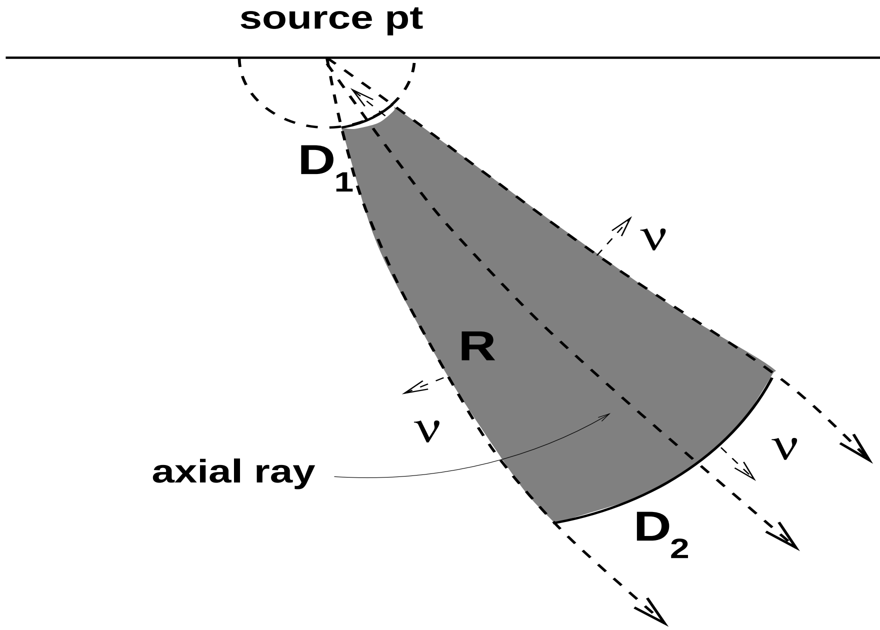


Figure 20: Ray tube construction: caps D_i are perp. to ray family, so normals on caps are parallel to rays; sides consist of rays (dashed lines) , so normals on sides are perp. to rays.

Amplitudes 4

$$\frac{A^2(\mathbf{x}_1)}{v(\mathbf{x}_1)} = \sigma \frac{A^2(\mathbf{x}_0)}{v(\mathbf{x}_0)}$$

where σ is the ratio of area element on D_0 to area element on D_1 (transverse Jacobian) on the two caps at the axial ray.

$$A(\mathbf{x}_1) = A(\mathbf{x}_0)\sqrt{\sigma \frac{v(\mathbf{x}_1)}{v(\mathbf{x}_0)}}$$

Therefore to compute amplitudes need only compute the area element σ.

Amplitudes 5

As in figure, make D_0 part of small circle (2D!), assume velocity constant between there and source point. Coordinate on circle is θ = ray takeoff angle (at source). Since caps are perp. to the unit vector $v^{-1}d\mathbf{x}/dt$,

$$\frac{1}{\sigma} = v\left|\frac{\partial(z,x)}{\partial(t,\theta)}\right| = v\left|\frac{\partial(t,\theta)}{\partial(z,x)}\right|^{-1}$$

$$= v(\nabla T \times \nabla\theta)^{-}1$$

Amplitudes 6

Upshot: for point source in 2D (similar formula in 3D),

$$A = \frac{v}{2\pi\sqrt{2}}\sqrt{\nabla T \times \nabla\theta}$$

Since take-off angle θ is constant along rays,

$$\nabla T \cdot \nabla\theta = 0$$

which is simpler than transport equation for A.

Amplitudes 7

Some important observations:

- when rays converge (*caustic*), area element ratio blows up $\Rightarrow$ this simple version of geometric optics is limited to regions of smooth single arrivals - else must use global G. O. ("Maslov Theory" - see Duistermaat 1973).
- Numerically: must have at least 1st order convergent $\theta \Rightarrow$ must use at least 3rd order scheme for T!
- Also: since T is singular at a point source, in order to obtain (effective) high order convergence may need adaptive gridding.