DOI: 10.1016/j.cpc.2013.08.010

Computer Physics Communications 185 (2014) 274–293

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Analytical ray tracing system: Introducing art, a C-library designed for seismic applications☆

Eduardo X. Miqueles [a,*], Tiago A. Coimbra [b], Bruno D. Amaro [b], J.J.S. de Figueiredo [c]

[a] Brazilian Synchrotron Light Laboratory/CNPEM, Campinas, Brazil
[b] Department of Applied Mathematics, University of Campinas, SP, Brazil
[c] Department of Geophysics, Federal University of Pará, Pará, Brazil

ARTICLE INFO

Article history:
Received 18 April 2013
Received in revised form
12 August 2013
Accepted 16 August 2013
Available online 26 September 2013

Keywords:
Ray tracing
Seismic
Eikonal
Geophysics
Wave equation

ABSTRACT

Ray tracing technique is an important tool not only to forward but also for inverse problems in Geophysics, which most of the seismic processing steps depend on. However, implementing ray tracing codes can be very time consuming. This article presents a computer library to trace rays in 2.5D media composed by a stack of layers. The velocity profile inside each layer is such that the eikonal equation can be analytically solved. Therefore, the ray tracing within such profile is made fast and accurate. The great advantage of an analytical ray tracing library is the numerical precision of the quantities computed and the fast execution of the implemented codes. Even though ray tracing programs exist for a long time, for example the seis88 package by Červený, most of those programs use a numerical approach to compute the ray. Regardless of the fact that numerical methods can solve more general problems, the analytical ones could be part of a more sophisticated simulation process, where the ray tracing time is completely relevant. We demonstrate the feasibility of our codes using several examples (Miqueles et al., 2013) [1]. The library can also be used for other applications besides seismic, e.g., optics and tomography.

Program summary

Program title: art

Catalogue identifier: AEQK_V1_0

Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEQK_V1_0.html

Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland

Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html

No. of lines in distributed program, including test data, etc.: 149519

No. of bytes in distributed program, including test data, etc.: 2609188

Distribution format: tar.gz

Programming language: C.

Computer: Workstations and PCs.

Operating system: Linux and Windows.

RAM: $\geq$2 Mb

Classification: 2.9.

External routines: LibConfuse (http://www.nongnu.org/confuse/).
To run the examples included in the distribution file, gengetopt (http://www.gnu.org/software/gengetopt/gengetopt.html), Seismic Unix (http://www.seismicunix.com/w/Main_Page), gnuplot (http://www.gnuplot.info/) and SU.

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author. Tel.: +55 19 9746 8001.
E-mail addresses: edu.miqueles@gmail.com (E.X. Miqueles), tgo.coimbra@gmail.com (T.A. Coimbra), brunoama@gmail.com (B.D. Amaro), jadsomjose@gmail.com (J.J.S. de Figueiredo).

0010-4655/$ – see front matter
http://dx.doi.org/10.1016/j.cpc.2013.08.010

Nature of problem:
Fast ray tracing algorithms for Seismic simulation and problems related to Wave propagation and/or Optics.

Solution method:
Method of characteristics for the eikonal equation, at a layered media, with analytical velocities.

Running time:
Milliseconds to 3 min, depending on the data size

1. Introduction

The seismic image problem of determining the approximate subsurface model of the earth from a set of seismic measurements, is a well known task of the geophysics community through the years. Most of the methods used in practice assume that a priori macro-velocity model between the subsurface reflectors is approximately known. To validate the assumed reference velocity, computer simulations are made, and if the resultant experiments agree with real ones, then within a reasonable tolerance, the velocity is a good approximation.

If not, we estimate another macro-velocity and repeat this process iteratively until we get a desired accuracy. The process to validate a given reference velocity model is intrinsically related to the propagation of elastic waves along a stratified media, and such waves always concentrate a great amount of energy in a three-dimensional curve in the euclidean space, that is called *a ray*.

Hence, whenever a synthetic model of the subsurface is given, a basic step in seismic surveying is the ray tracing, which consists in the determination of the raypath along the propagation medium. Sometimes, practical experimentations have shown that there is a great velocity variation over short time intervals, so that is reasonable to think in a continuous variation of the velocity with respect to depth and distance within a seismic layer (the space bounded by subsurface reflectors). As a consequence, sometimes we can consider the velocity field formulated through an analytical expression; and with a physical meaning. Sometimes, this is exactly the case when the velocity is constant through the medium (see Fig. 1).

When properly chosen, the analytical formulation of the velocity field also allows the analytical determination of the raypath and other related quantities, either kinematic or dynamical, like the traveltime and the amplitude associated to the elastic wave. This is a good motivation to build a computer library for ray tracing. The great advantage of an analytical ray tracing library (`art` for short) is the numerical precision of the quantities computed and the fast execution of the implemented codes. Although ray tracing programs already exist for a long time, for example the Seis package revisited in [2] and the Software Norsar® [3], a different approach is commonly used, with numerical methods to compute the ray. A short article for `art` was presented in [1].

A straightforward and powerful application of ray tracing technique is the computation of seismic records associated to a common-source or common-receiver seismic event. Although this is not the primary intention of this article, we provide an easy-to-use tool for computing further quantities, e.g. the seismic trace. Indeed, since the software provides the traveltime and the reflectivity upon each interface, a simple convolution will provide an impulse.

In Section 2 we review the mathematical theory for ray tracing. In Section 3, we describe the seismic medium where the associated ray equations have an analytical solution. In Section 4 we define the C-library `art` and Sections 5 and 6 presents several examples on how this library can be used at different kinds of seismic applications.

2. Ray trace system

We now briefly summarize the mathematical theory for seismic ray tracing. For simplicity, the main equations are depicted in Table 1. For further reference, we refer to such table as the symbol **T**[1].

Seismic waves propagating through an elastic and isotropic medium obeys the elastodynamic wave equation [4] given at **T**[1].(A), where $\mathbf{u}$ is the displacement vector, ρ is the density and $\{\lambda, \mu\}$ are the Lamé parameters. Assuming a high frequency solution for **T**[1].(A), it is easy to show that propagating waves are approximately given by a superposition of two other waves, the compressional scalar wave $\theta = \nabla\cdot\mathbf{u}$ and the shear vectorial wave $\Omega = \nabla\times\mathbf{u}$, also called P and S waves respectively (Helmholtz decomposition Theorem, see [5]). The ray-theory assumes that the zeroth-order approximation **T**[1].(B) is a high-frequency solution (WKJB approximation) of **T**[1].(A), where $\mathbf{U}$ is a complex-valued vectorial *amplitude* function, F is a high-frequency signal and T is the real-valued *traveltime* function. So, to estimate the zeroth-order solution [6], we need to compute the traveltime and amplitude functions.

After several assumptions and operations with the *ansatz* **T**[1].(B) and the elastodynamic equation, it can be shown [7,4] that the traveltime obeys the so-called *eikonal equation* **T**[1].(C), where $\mathbf{p} = \nabla T$ is the *slowness vector* and v is the wave velocity. Usually, we have to solve the eikonal equation for P and S waves separately, so we denote $v = \alpha$ for the first case and $v = \beta$ for the second. For a better understanding on how both α and β are related to the Lamé parameters, see [8].

We are most concerned with pressure waves, where the displacement vector is perpendicular to the wavefront [4,9] and thus, we can write $\mathbf{U}(\mathbf{x}) = \alpha(\mathbf{x})A(\mathbf{x})\mathbf{p}$. Shear waves propagate in the parallel wavefront direction, we write $\mathbf{U}(\mathbf{x}) = B(\mathbf{x})\mathbf{e_1}(\mathbf{x}) + C(\mathbf{x})\mathbf{e_2}(\mathbf{x})$, for a particular choice of polarization vectors $\{\mathbf{e_1}, \mathbf{e_2}\}$. In both cases, we need to compute the scalar amplitude functions A, B and C. In the case of P waves, the function A obeys the *transport equation* given by **T**[1].(D). For S waves, the pair $\{B, C\}$ also obeys the same transport equation **T**[1].(D) with A replaced by $\{B, C\}$, and α replaced by β, see [4].

Once the eikonal equation is a non-linear first order partial differential equation, the solution can be computed by the method of characteristics (or Lagrange's Method, see [10,11]), which is fully described in [6,12,13]. The Hamiltonian associated to the eikonal equation **T**[1].(C) can be written in the following form

$$\mathcal{H}(\mathbf{x}, \mathbf{p}, T) = \frac{1}{n}\left[\left(\mathbf{p}^T\mathbf{p}\right)^{n/2} - \frac{1}{v(\mathbf{x})^n}\right], \qquad \mathcal{H}(\mathbf{x}, \mathbf{p}, T) = \frac{1}{2}\ln\left(v(\mathbf{x})^2\mathbf{p}^T\mathbf{p}\right), \tag{1}$$

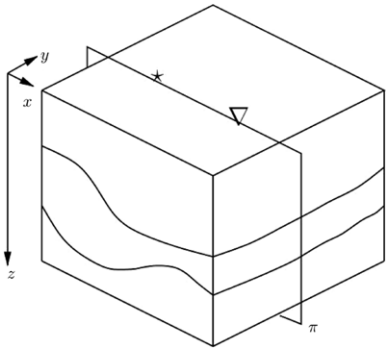

Fig. 1. Seismic profile π: isotropic 2.5D medium for ray tracing.

Table 1
Main equations for seismic ray tracing. See text for more details. SI Units are the following: $\mathbf{u}$ (km), T (sec), v (km/sec), $\mathbf{p}$ (sec/km), A (1/km). Other quantities are dimensionless.

Equation	Description
$\rho\frac{\partial^2\mathbf{u}}{\partial t^2} = (\lambda+\mu)\nabla(\nabla\cdot\mathbf{u}) + \mu\nabla^2\mathbf{u} + (\nabla\cdot\mathbf{u})\nabla\lambda + \nabla\mu\times(\nabla\times)\mathbf{u} + 2(\nabla\mu\cdot\nabla)\mathbf{u} + \mathbf{f}$	(A) Elastodynamic equation
$\mathbf{u}(\mathbf{x},t) = \mathbf{U}(\mathbf{x})F(t-T(\mathbf{x}))$	(B) Ansatz solution for (A)
$\mathbf{p}^T\mathbf{p} = \|\mathbf{p}\|^2 = \frac{1}{v(\mathbf{x})^2} = \sum_{j=1}^{3}\left(\frac{\partial T}{\partial x_j}\right)^2$	(C) Eikonal equation
$2\mathbf{p}^T\nabla A + A\nabla^2 T = 0$	(D) Transport equation
$\frac{d\mathbf{x}}{du} = \frac{\partial\mathcal{H}}{\partial\mathbf{p}}$, $\frac{d\mathbf{p}}{du} = -\frac{\partial\mathcal{H}}{\partial\mathbf{x}} - \frac{\partial\mathcal{H}}{\partial T}\mathbf{p}$, $\frac{dT}{du} = \mathbf{p}^T\frac{\partial\mathcal{H}}{\partial\mathbf{p}}$	(E) General ray equations
$A(u) = \sqrt{\frac{\rho(u_0)v(u_0)J(u_0)}{\rho(u)v(u)J(u)}}$	(F) Amplitude function along a ray
$\frac{d}{du}\begin{bmatrix}\mathbf{Q}\\ \mathbf{P}\end{bmatrix} = \begin{bmatrix}\mathbf{0} & v^{2-n}(u)\mathbf{I}\\ -\frac{\mathbf{V}(u)}{v^{n+1}(u)} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{Q}\\ \mathbf{P}\end{bmatrix}$	(G) Dynamical ray tracing system
$\frac{d\mathbf{w}^{(i)}}{du} = \mathbf{S}^{(i)}\mathbf{w}^{(i)}$, $\frac{d\mathbf{w}^{(o)}}{du} = \mathbf{S}^{(o)}\mathbf{w}^{(o)}$	(H) In-plane and out-of-plane systems
$\mathbf{S}^{(i)} = \begin{bmatrix}0 & v^{2-n}(u)\\ v(u) & 0\end{bmatrix}$	(I) In-plane matrix
$\mathbf{S}^{(o)} = \begin{bmatrix}0 & v^{2-n}(u)\\ 0 & 0\end{bmatrix}$	(J) Out-of-plane matrix
$\nu(u) = -\frac{[\mathbf{J}\mathbf{p}(u)]^T\mathbf{H}_v[\mathbf{J}\mathbf{p}(u)]}{v^{n-1}(u)}$	(K) Function $\nu = \nu(u)$
$\mathbf{w}^{(o)}(u) = \Pi^{(o)}(u)\mathbf{w}^{(o)}(u_0)$, $\Pi^{(o)}(u) = \begin{bmatrix}1 & y_n\\ 0 & 1\end{bmatrix}$	(L) Out-of-plane system solution of (H)
$y_n(u) = \int_{u_0}^{u} v^{2-n}(b)db$	(M) Function $y_n = y_n(u)$

for $0 \neq n \in \mathbb{R}$ and $n = 0$ respectively. We shall refer to the constant n as the *Hamiltonian index*. The method of characteristics states that the solution of $\mathcal{H} = 0$ is equivalent to the solution for a system of ordinary differential equations **T**[1].(E), with initial conditions $\mathbf{x}(u_0) = \mathbf{x}_0$, $\mathbf{p}(u_0) = \mathbf{p}_0$ and $T(u_0) = T_0$. In our case, this system is reduced to

$$\begin{cases} d\mathbf{x}/du = v(\mathbf{x})^{2-n}\mathbf{p},\\ d\mathbf{p}/du = \dfrac{1}{n}\nabla\left(\dfrac{1}{v(\mathbf{x})^n}\right),\\ dT/du = \dfrac{1}{v(\mathbf{x})^n}, \end{cases} \qquad \begin{cases} d\mathbf{x}/du = v(\mathbf{x})^2\mathbf{p},\\ d\mathbf{p}/du = -\nabla\left(\ln v(\mathbf{x})\right),\\ dT/du = 1 \end{cases} \tag{2}$$

for $n \neq 0$ and $n = 0$ respectively. The above system is called the *kinematic ray tracing equations*. The curve $\mathbf{x}\colon \mathbb{R} \to \mathbb{R}^3$ has the slowness vector $\mathbf{p}$ as a tangent vector and the integration parameter u varying over the domain $u \geq u_0$. Usually, we will adopt the convention $u_0 = 0$. For the case where the Hamiltonian index is zero, the integration parameter is the traveltime itself. The kinematic part of a ray is mathematically represented by the characteristic curve $\{\mathbf{x}, \mathbf{p}, T\}$ and the raypath is determined by $\mathbf{x}$.

Finally, to compute the solution of equation **T**[1].(D), we emphasize that along a ray, function A can be computed by **T**[1].(F), where $v(u_0) = v(\mathbf{x}(u_0))$ and J is the ray Jacobian related to a usual orthonomic ray system parameterization, see [6]. The *dynamical ray tracing system* is mainly concerned with the computation of the ray Jacobian, in order to evaluate **T**[1].(F). [4,6,14] shows that, introducing a properly change of variables, we obtain the so-called *dynamical ray tracing system* **T**[1].(G), where matrices $\{\mathbf{Q}, \mathbf{P}\}$ represent the Jacobian in a new system of coordinates centered at the ray, with $\mathbf{I}$ the identity in $\mathbb{R}^{2\times 2}$ and $\mathbf{V}$ the Hessian matrix in this new coordinate system. For a 3D medium,[1] that differential equation can be decoupled into two other ones, the *in-plane* and *out-of-plane* dynamical systems. Therefore, the solution can be made easier. We write those decoupled systems as in **T**[1].(H) with matrices $\mathbf{S}^{(i)}$ and $\mathbf{S}^{(o)}$ given by **T**[1].(I) and **T**[1].(J) respectively. Function ν, introduced within those matrices, is an appropriate real mapping depending on the Hessian matrix of the velocity

[1] Actually, 2.5D.

Table 2
Analytical velocity fields.

Id.	(n, u)	$\Phi(v)$
ART_VCONST	$(1, s)$	$v(\mathbf{x}) = c$
ART_VAFF	$(-1, \xi)$	$v(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + c$
ART_VASS, ART_VQSS	$(2, \sigma)$	$\frac{1}{v(\mathbf{x})^2} = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{a}^T\mathbf{x} + c$
ART_VCGL	$(0, T)$	$\ln v(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + c$

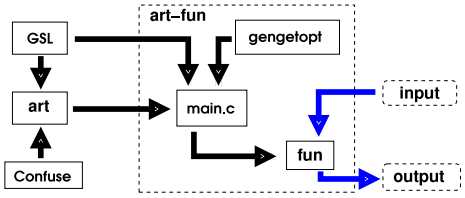

Fig. 2. Fluxogram for `art`.

field and $v(u) = v(\mathbf{x}(u))$. It follows from all these properties that $J(u) = \mathbf{w}_1^{(i)}(u)\mathbf{w}_1^{(o)}(u)$ is the Jacobian along the ray, see [4]. Also, other quantities like geometrical spreading and ray-centered-amplitude can be obtained from the solution of both dynamical systems.

3. Seismic analytical medium

We want to determine the kinematic part of the ray through the solution of one the ray systems in (2), i.e., quantities $\{\mathbf{x}, \mathbf{p}, T\}$. For this, we are motivated to choose velocity fields that enable analytical solutions. In this sense, we pick a particular Hamiltonian index to make easier the ray equations, and for our purposes, the best choices for n are in the set $\{-1, 0, 1, 2\}$.

For further reference, we consider that velocities can be written in the general form

$$\Phi(v(\mathbf{x})) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{a}^T\mathbf{x} + c \tag{3}$$

where Φ assume the value of one of the functions $\{v, v^{-2}, \ln v\}$, as illustrated by Table 2. By a suitable choice of the parameters $\mathbf{A}$, $\mathbf{a}$ and c we can always ensure that $\Phi(v(\mathbf{x})) > 0$ for all $\mathbf{x}$. To distinguish the velocities, we established the following identification: `vass` stands for affine square of slowness, `vqss` for quadratic square of slowness, `vcgl` for constant gradient of logarithmic velocity, `vaff` for affine velocity and `vconst` for constant velocity.

Since we are mainly interested in rays that are confined in a vertical plane along the seismic profile, we can assume that parameters in Eq. (3) are two-dimensional, i.e., $\mathbf{A} \in \mathbb{R}^{2\times 2}$ and $\mathbf{a} \in \mathbb{R}^2$, where the position vector $\mathbf{x}$ has first component related to horizontal distance and second component related to depth.

Function ν appearing in the in-plane dynamical matrix $\mathbf{S}^{(i)}$ of equation $\mathbf{T}[1].(\mathrm{I})$ – see Table 1 – is defined by equation $\mathbf{T}[1].(\mathrm{K})$, where $\mathbf{J}$ is a two-dimensional symplectic matrix [15] and $\mathbf{H}_v$ is the Hessian matrix of the velocity field in Cartesian coordinates. The proof of $\mathbf{T}[1].(\mathrm{K})$ is given in [7]. The out-of-plane dynamical system $\mathbf{T}[1].(\mathrm{H})$ always have an analytical solution, which is given by $\mathbf{T}[1].(\mathrm{L})$. The in-plane solution $\mathbf{w}^{(i)}(u) = \Pi^{(i)}(u)\mathbf{w}^{(i)}(u_0)$ is usually computed by a numerical method. $\Pi^{(i)}$ and $\Pi^{(o)}$ are called in-plane and out-of-plane propagator matrices, respectively.

Note that the solution of the kinematic ray tracing system (2) can be written in the general form

$$\mathbf{x}(u) = \mathbf{x}_0 + \mathbf{d}_n(u), \qquad T(u) = T_0 + \mathbf{t}_n(u), \qquad \mathbf{p}(u) = \mathbf{p}_0 + \mathbf{s}_n(u) \tag{4}$$

where $\mathbf{d}_n$, $\mathbf{s}_n$ and $\mathbf{t}_n$ vary according to the velocity field (i.e., according to the index n). The analytical equations for $\{\mathbf{d}_n, \mathbf{t}_n, \mathbf{s}_n\}$ are given in Appendix A.1, where we assume a continuous velocity field $v = v(\mathbf{x})$ without boundaries. Therefore, the ray starts at a free source point and goes through the medium without any restriction. Problems related to incidence/transmission at a boundary are treated in the following sections.

4. The C-library `art`

As other free libraries, like `gsl` [16], our library provides many 'ray tracing functions', so the user can be free to implement his/her own codes (in standard C-language), which is basically the idea of `art`. To introduce users with `art`, we also provide programs that already contain some standard routines in seismic ray tracing, like two-point ray tracing, one-way ray tracing, common-shot experiments, and others. These functions are contained within `art-fun`. Also, the library depends on other free softwares, the `libConfuse` [17] and `gengetopt` [18], as shown in the Fluxogram of Fig. 2. The library is in continuous expansion and we expect it to increase and improve. In the next section we present the usage of `art-fun/`.

The 2.5D synthetic medium for ray tracing considered by the library is presented in Fig. 3(a). The main goal of the library is the determination of a ray in this kind of medium, starting at a source point (usually at the surface) and arriving at a receiver or at an interface. Since the medium is composed by a stack of layers, where each layer has one of the analytical velocities presented in the previous section, the first step in tracing a ray through the profile, is the computation of incidence points along a sequence of layers. Computationally, the union of incidence points fully determines the raypath since we know exactly (or analytically) the behavior of the curve between such extreme points. The sequence of layers (for the raypath) is given a priori, and is known as the *ray code*. Fig. 3(b) shows an example of two rays and their respective ray codes. Finally, each interface is represented by a cubic spline and we assume that at least three knots can always be determined, even in the case of planar interfaces.

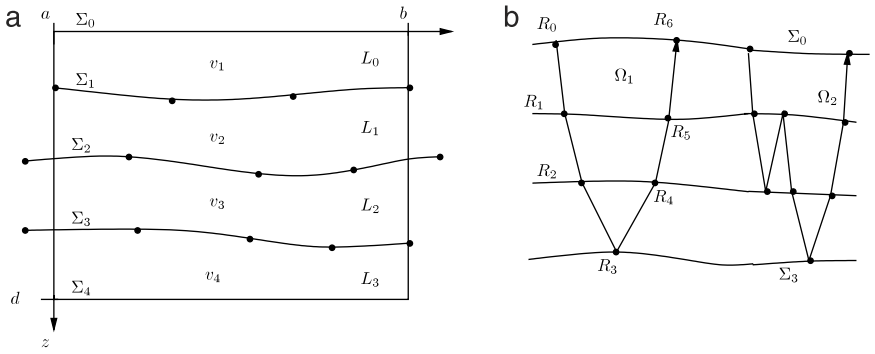

Fig. 3. Ray tracing profile. (a) Stack of layers, each one bounded by structural interfaces, represented by cubic splines. (b) Two rays in a three-layered profile, with ray codes {0, 1, 2, 2, 1, 0} (left) and {0, 1, 1, 1, 2, 2, 1, 0} (right).

In a synthetic profile like the one described in Fig. 3, the library allows the following two basic operations: (i) *one-way ray tracing*—given a point source and an initial slowness direction, we compute the ray through the profile, following a ray code sequence and (ii) *two-point ray tracing*—compute the ray through the profile starting at the source and finishing at a given receiver, following a ray code sequence. Other operations are also allowed in the library, but are nothing more than applications of these two basic procedures.

Before starting the explanation for the state-of-the-art of `art`, there are some basic rules that we must keep in mind whenever we want to trace a ray.

(a) The library can only perform a ray tracing if analytical velocities are defined within each layer, but the choice of parameters **A**, **a** and c in Eq. (3), is of entire user's responsibility. In the case of $\Phi(v(\mathbf{x})) \leq 0$ for some $\mathbf{x}$ within a layer, there is a great probability to fail the ray tracing.
(b) Rays are traced only within a seismic *box* $[a, b] \times [0, d]$, as illustrated by Fig. 3(a). If any ray point lies outside the box, the ray tracing is considered unfeasible.
(c) For each spline (interface), the first and last knots cannot be within the box, or the library will produce an error.
(d) Since four types of waves are produced whenever a ray strikes an interface, a segment of the raypath can represent either pressure waves (P waves) or shear waves (S waves). Hence, the ray code includes the wave type associated to each segment of the path. In a profile with N layers, a ray code sequence takes the form $r_c = \{(l_0, w_0), (l_1, w_1), \ldots, (l_{M-1}, w_{M-1})\}$ where $0 \leq l_k \leq N$ indicates the layer number and w_k indicates the wave type, for example $w_k = 1$ for S waves and $w_k = 0$ for P waves. This means that the ray must traverse the profile according to $l_0 \to l_1 \to \cdots \to l_{M-1}$ where each layer is crossed only once at a time. In Fig. 3(b) there are two examples of ray code sequences, note that for the ray displayed at left, the sequence {0, 1, 2, 1, 0} *it is not* a correct ray code and may produce an error.
(e) If a given ray code has length M then the number of incidence points will always be $M + 1$. This is illustrated in Fig. 3(b).
(f) Given a source point upon an interface, the initial slowness vector could not have the same direction of the surface tangential vector at that point. Whenever this happens (within a default tolerance) an error is produced and the ray tracing is considered unfeasible. As a consequence, `art` does not consider Rayleigh waves.
(g) Two-point ray tracing can only be performed with the receiver placed at an interface, but the source point could be placed anywhere inside the seismic box. On the other hand, one-way ray tracing always end at an interface.
(h) The library do not trace turning rays (within a given layer).
(i) Earth's surface is always represented by the plane $z = 0$, i.e., the library does not take into account ground topography.
(j) Layers are considered as disjoint sets. This means that they do not cross each other, like salt domes and more complex seismic reflectors.

4.1. Implementation of `art`

In this section we briefly summarize the implementation of point-interface ray tracing, since this is the most basic step to compute a single ray. Next we show how the two-point ray tracing is done, using a slight modification of the bisection method.

We adopt the following convention: given a ray code $r_c = \{l_0, l_1, \ldots, l_{M-1}\}$, the ray Ω is computationally equivalent to the array of extreme points $\Omega \equiv \{R_0, R_1, \ldots, R_M\}$, where R_k belongs to the layer number l_k. The point R_0 is the source point and each R_k is either an incidence point or a transmission point.

4.1.1. Point to interface

Given a ray code $r_c(\Omega)$ and an angle $\theta \in [-\pi, \pi]$, the initial slowness vector direction is given by $\mathbf{p}_0 = (1/v_0)(\sin\theta, \cos\theta)$ where v_0 is the velocity at the initial point $\mathbf{x}_0 \equiv R_0$. Starting at the point R_0 we must determine the intersection between the curve $\mathbf{x} = \mathbf{x}(u)$ in Eq. (4) (lying within layer number l_0) with the interface Σ where R_1 must be placed. Then, a new slowness direction is obtained using Snell's law. Next, we use the same reasoning to find R_2 on a new interface. This procedure continues until we had successfully computed the desired number of extreme points inside the seismic box.

So, the geometric problem here is just one: find the intersection of two curves, one representing the raypath, given by (4), and the other representing an interface. Let S be an initial point, we want to find $R \in \Sigma$ so that the curve $\mathbf{x} = \mathbf{x}(u)$ lies within the layer, as illustrated by Fig. 4(a). Let $\mathcal{S}$ be the spline function representing the interface Σ. The task can be stated as a root-finding problem, i.e., we want to find u (integration parameter) such that $\eta(u) = \mathbf{x}_2(u) - \mathcal{S}(\mathbf{x}_1(u)) = 0$ with $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)^T$ the position vector (4), see Fig. 4(b). We use the

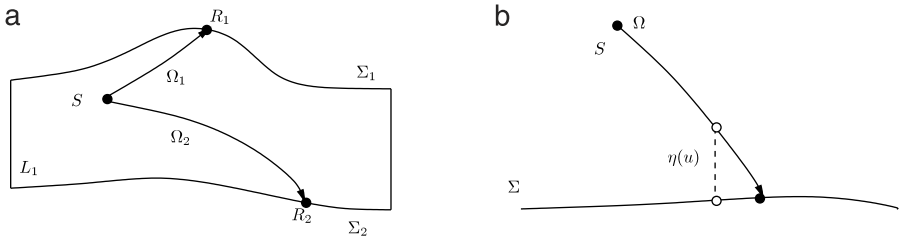

Fig. 4. Point-interface problem. (a) A ray $\Omega_j \in L_1$ starts from point S and strikes an interface at point $R_j \in \Sigma_j$, $j = 1, 2$. (b) Distance $\eta = \eta(u)$ to be minimized.

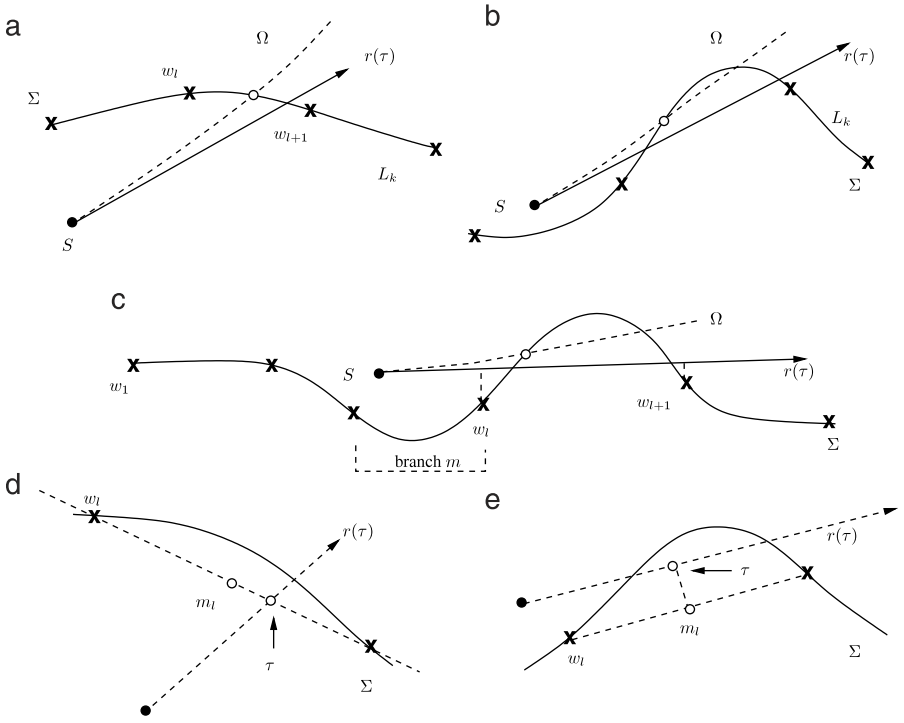

Fig. 5. Newton initial guess. (a) Exist $l \in \mathbb{N}$ such that $\mathbf{p}_0 \in C_l$. (b) Do not exist $l \in \mathbb{N}$ such that $\mathbf{p}_0 \in C_l$. (c) Exist $l \in \mathbb{N}$ such that the average of distances to $\mathbf{r}$ is minimum (d) and (e) Intersection parameter τ (see text for details).

Newton–Raphson method to compute the optimal u^* so that $R \equiv \mathbf{x}(u^*)$. Since $\eta'(u) = (\mathbf{x}_1'(u), \mathbf{x}_2'(u)) \cdot (-\mathcal{S}'(\mathbf{x}_1(u)), 1)$, we have a typical iteration procedure:

$$u^{i+1} = u^i - v^{n-2}(u^i)\frac{\eta(u^i)}{\mathbf{p}(u^i)^T\mathbf{n}(u^i)}, \quad i = 0, 1, 2, \ldots$$

where $\mathbf{n}(u) = (-\mathcal{S}(\mathbf{x}_1(u)), 1)^T$ is the normal vector to the interface at point $\mathbf{x}_1(u)$. Assuming that there are *no turning rays*, it follows that $(\mathbf{p}^T\mathbf{n})(u) \neq 0$ and the above iterative method is always well posed. To obtain quadratic convergence $\lim_{i\to\infty} u^i = u^*$, we need to find an initial guess u_0 such that $\mathbf{x}(u_0) \approx R$.

Finding an initial guess

The spline $\mathcal{S}$ is represented by control points $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_l, \ldots\}$, $\mathbf{w}_l \in \mathbb{R}^2$. If we assume that the ray has a low-curvature, the first strategy to compute the nearest point $\tilde{R}$ to R is the following: find two control points $\mathbf{w}_l$, $\mathbf{w}_{l+1}$ so that the straight line spanned by the initial slowness vector $\mathbf{p}_0$ intersects the straight line between both points. This can be restated using the idea of a cone: given the initial point S, build the positive cone[2] $C_l \equiv \mathbf{w}_l \to S \to \mathbf{w}_{l+1}$ (with S as the vertex), and find an index l so that $\mathbf{p}_0 \in C_l$. Fig. 5(a) shows an example where exists the requested index, whereas in Fig. 5(b) the index is empty. Let $\mathbf{r}(\tau) = S + \tau\mathbf{p}_0$ be the straight line spanned by the initial slowness vector. The intersection of $\mathbf{r}$ with the line segment between $\mathbf{w}_l$ and $\mathbf{w}_{l+1}$ occurs when (see Fig. 5(d))

$$\tau = \frac{\mathbf{q}^T(\mathbf{w}_l - S)}{\mathbf{q}^T\mathbf{p}_0} \tag{5}$$

with $\mathbf{q} = (\delta, -1)$ and $\delta = (\mathbf{w}_{l+1,2} - \mathbf{w}_{l,2})/(\mathbf{w}_{l+1,1} - \mathbf{w}_{l,1})$. We refer to this idea as *cone strategy*.

Proof for (5). The line through points $\{\mathbf{w}_l, \mathbf{w}_{l+1}\}$ is given by $z = \mathbf{w}_{l,2} + \delta(x - \mathbf{w}_{l,1})$. Replacing $z = S_2 + \tau\mathbf{p}_{0,2}$ and $x = S_1 + \tau\mathbf{p}_{0,1}$, we obtain (5) after some cumbersome calculations. ■

[2] Mathematically: $C_l = \{q(\mathbf{w}_{l+1} - S) + c(\mathbf{w}_l - S) : q, c \in \mathbb{R}_+\}$.

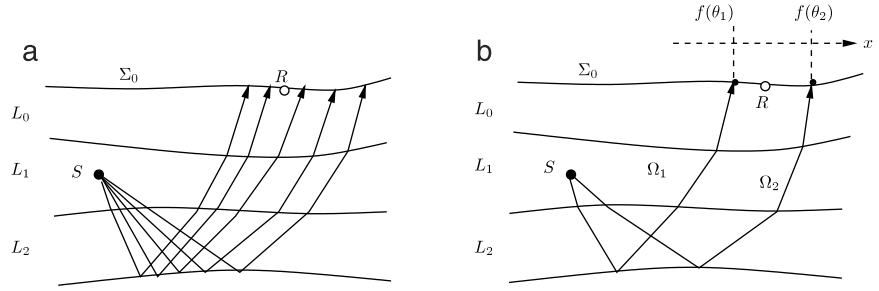

Fig. 6. (a) Fan of rays starting at source point S and arriving at receiver interface, following ray code $r_c = \{1, 2, 2, 1, 0\}$. (b) Two auxiliary rays Ω_1 and Ω_2 surrounding the receiver point.

By the ray equations, $dT/du = 1/v^n$ or $ds = du/v^n$ from what follows that $u \approx v^{n-1}(\tau)s$. Hence, after a reparametrization of $\mathbf{r}$ by arclength s, we arrive at that $\tau = sv_0$, taking the initial guess u_0 as

$$u_0 \approx v^{n-1}(\tau)s = v^{n-1}(\mathbf{r}(\tau))\frac{\tau}{v_0}. \tag{6}$$

Since the cone strategy eventually fails, as in Fig. 5(c), we need to develop a second strategy, which we call *minimum distance*. The idea is also simple, we search for a pair of points $\mathbf{w}_l, \mathbf{w}_{l+1}$ such that the average of the distance from $\mathbf{r}$ to each one of these points is minimum. In other words, if d_l is the distance from $\mathbf{w}_l$ to the line $\mathbf{r}$ then we search for a $l \in \mathbb{Z}$ such that $(d_l + d_{l+1})/2$ is minimum. Once determined the index l, we choose τ such that $\mathbf{r}(\tau) - \mathbf{m}_l \perp \mathbf{p}_0$ where $\mathbf{m}_l = (\mathbf{w}_l + \mathbf{w}_{l+1})/2$ (see Fig. 5(e)), i.e., $\tau = \frac{\mathbf{p}_0^T(\mathbf{m}_l - S)}{\|\mathbf{p}_0\|^2} = \mathbf{p}_0^T(\mathbf{m}_l - S)v_0^2$, and take the initial guess u_0 as in Eq. (6). Here, it only makes sense to search an index $l \geq m$ where m is the spline branch where the source point S lies, and whenever the slowness vector points toward right. Analogously, we search for $l \leq m$ whenever the slowness vector points toward left.

4.1.2. Two-point ray tracing

Given a pair of fixed points $S, R \in \mathbb{R}^2$, we are searching for the set of rays connecting S and R; this is called a *two-point ray tracing problem*. For the sake of completeness, we present the implemented method, which is similar to the bisection algorithm. It is an adaptation of the classical *shooting algorithm* due to Červený and Pšenčik [19–21]. We always assume that the receiver R lies at an interface, whereas the source S is placed anywhere inside the seismic box. A ray code $r_c(\Omega)$ is given a priori.

Let $f\colon \theta \mapsto R(\theta)_1$ the real-valued function that maps an angle θ to the abscissa of an end ray-point $R(\theta)$, that we denote as $R(\theta)_1$. The mathematical problem here, is to find an angle θ^* such that $f(\theta^*) = R_1$, i.e., a one-dimensional root finding problem. The method starts searching two rays $\Omega_1(\theta_1)$ and $\Omega_2(\theta_2)$ so that their end points $R(\theta_1)_1, R(\theta_2)_2$ satisfy the condition $f(\theta_1) \leq R_1 \leq f(\theta_2)$ as illustrated in Fig. 6(b). These rays are chosen among a fan of rays starting at the source point (following the ray code) reaching the receiver interface, see Fig. 6(a). Next, a linear interpolation creates a new angle estimate $\theta^{(1)} = (R_1 - a)/b$, where $a = (\theta_2 f(\theta_1) - \theta_1 f(\theta_2))/(\theta_2 - \theta_1)$ and $b = (f(\theta_2) - f(\theta_1))/(\theta_2 - \theta_1)$, and tracing the ray $\Omega(\theta^{(1)})$ to obtain a new pair of rays surrounding the receiver. This process is repeated until we get a desired accuracy for $|f(\theta^{(i)}) - R_1|$. This generates a sequence of take-off angles $\{\theta^{(i)}\}$ that converges to the requested angle θ^*. The proof is similar to the bisection root-finding method (see [22]), in this case the ray $\Omega(\theta^*)$ connects the source and the receiver point. This algorithm is applied to each pair of rays enclosing the receiver point, as depicted in Fig. 6(b), a phenomenon that may occur due to caustic interfaces.

4.2. The ray: computational aspects

We have seen that the ray is computationally equivalent to an array of extreme points $\{R_0, R_1, \ldots, R_N\}$ since we know exactly the raypath behavior between any pair of points R_k and R_{k+1}. Starting at R_k we have determined an optimal integration parameter $u^{(k)}$ such that $\mathbf{x}(u^{(k)}) \equiv R_{k+1}$, where $\mathbf{x} = \mathbf{x}(u)$ is the raypath, given in (4), according to the layer velocity field. Fig. 7 shows a physical ray and its computational representation. Within each layer, the local integration parameter varies in the interval $[0, u^{(k)}]$. To unify the integration parameter along the raypath, we have chosen the traveltime to be the global integration parameter, as illustrated by Fig. 7.

If the ray Ω has $N + 1$ extreme points, and $\tau \in [0, \tau_N]$ is the traveltime varying over the ray, i.e., we have $\tau_N = T(u^{(N)})$ with T being the traveltime function within the last branch. We want to know the exact value of a certain kinematic/dynamic quantity at τ, e.g. the position vector. To solve this problem, first we find the ray-branch corresponding to the input traveltime τ, say k, and second, using the traveltime function T at the branch k, we find a parameter u^*, the solution of $T(u^*) - \tau = 0$.

This is a one-dimensional root-finding problem, with unique solution[3] and once again, through the usage of the Newton–Raphson method, we iterate:

$$u_{i+1} = u_i - v^n(u_i)[T(u_i) - \tau], \quad i = 0, 1, 2, \ldots$$

with initial guess u_0 being the solution of $P(u_0) - \tau = 0$. Here, P is an interpolating parabola of function T through the points $\{0, u^{(k+1)}/2, u^{(k+1)}\}$. This initial guess ensures quadratic convergence $u_i \to u^*$ since the traveltime function behaves like a hyperbola (see [8]), and a parabola usually fit reasonably well. This one-dimensional root finding problem is used to compute the reflectivity and the acoustic impedance as a function of the traveltime along the ray. It should be noted that most of the computational complexities in our ray tracing

[3] Since the traveltime is always a continuous and increasing function.

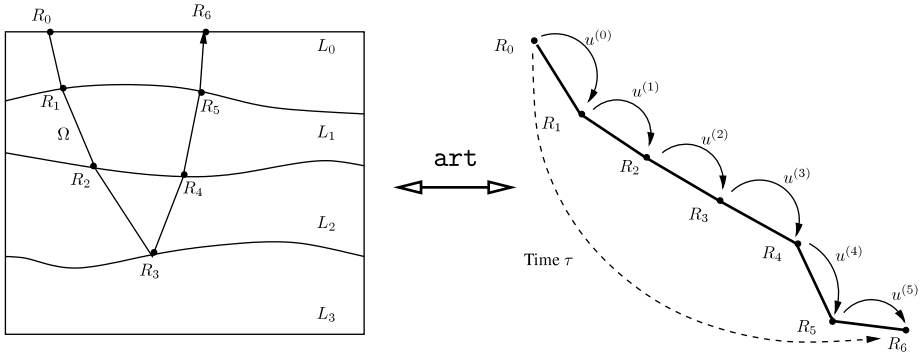

Fig. 7. Physical and computational representations of a ray Ω in `art`. The traveltime varies continuously along the raypath while the local integration parameter varies from 0 to $u^{(k)}$ within each ray-branch.

Table 3
Dynamic ray tracing equations. See text for more details.

Equation	Description
$\mathbf{w}_I^{(*)}(R_{k+1}) = \Pi^{(*)}(R_{k+1}, R_k)\mathbf{w}_T^{(*)}(R_k)$	(A) Incidence solution
$\mathbf{w}_T^{(*)}(R_{k+1}) = \Pi_T^{(*)}(R_{k+1})\mathbf{w}_I^{(*)}(R_{k+1})$	(B) Transmitted solution
$\mathcal{L}(R_k) = \mathcal{L}^{(i)}(R_k)\mathcal{L}^{(o)}(R_k)$	(C) Standard incident geometrical spreading
$\mathcal{L}(R_k, R_0) = \mathcal{L}^{(i)}(R_k, R_0)\mathcal{L}^{(o)}(R_k, R_0)$	(D) Relative incident geometrical spreading
$A_I(R_k) = e^{-i\frac{\pi}{2}\kappa(R_k)}C(R_k)d(R_k)$	(E) Incident amplitude
$A_T(R_k) = \mathcal{C}(R_k)A_I(R_k)$	(F) Transmitted amplitude
$d(R_k) = \sqrt{\frac{V(R_k)}{L(R_k)}}\frac{v(R_0)}{\mathcal{L}^{(i)}(R_k,R_0)\mathcal{L}^{(o)}(R_k,R_0)}\sqrt{\frac{v_T(R_0)\rho_T(R_0)}{v_I(R_k)\rho_I(R_k)}}$	(G): Used in (E)
$V(R_k) = \prod_{m=1}^{k}\frac{v_T(R_m)\rho_T(R_m)}{v_I(R_m)\rho_I(R_m)}$	(H): Used in (G)
$L(R_k) = \prod_{m=1}^{k}\frac{\cos\alpha_I(R_m)}{\cos\alpha_T(R_m)}$	(I): Used in (G)
$C(R_k) = \prod_{m=1}^{k-1}\mathcal{C}(R_m)$	(J): Used in (E)

approach come from the optimization technique used to solve the root-finding problem for intersection between ray and interface, or even the evaluation along the ray.

4.3. Dynamic ray tracing

For simplicity, the main equations for the dynamic ray tracing are summarized in Table 3, that we denote by **T**[3], presented at the end of this section.

Many quantities related to dynamic tracing depend on the solution of equations **T**[1].(H)—see Table 1. Since the out-of-plane system have an analytical solution, the only numerical task here is the computation of the in-plane solution. We solve this system by the Runge–Kutta–Fehlberg (4,5) method, using standard GSL procedures [16]. Once we have both solutions, and the kinematic ray tracing complete, everything remains analytical again (e.g. the geometrical spreading and amplitude values) because we know the optimal integration parameter that allows us to leave an extreme point and reach the next one. For completeness, we list below the dynamical expressions used in the library. A more detailed treatment is given in [4].

Consider the following notation for the in-plane and out-of-plane propagator matrices: $\Pi^{(*)}(u^{(k)}, 0) = \Pi^{(*)}(R_{k+1}, R_k)$, for $* = \{i, o\}$ respectively, i.e. the propagator matrix from an extreme point R_k to R_{k+1}. Assuming that we know a transmitted dynamic solution $\mathbf{w}_T^{(*)}$ at R_k, the incident and transmitted solutions at R_{k+1} are given in **T**[3].(A) and **T**[3].(B) respectively, where $\Pi_T^{(*)}$ is the propagator matrix across an interface.[4] Here, $\Pi_T^{(o)} = \mathbf{I}$ is the two-dimensional identity matrix, and $\Pi_T^{(i)}$ have a more complicated expression, described in [4]. The in-plane propagator matrix can be written as $\Pi^{(i)} = [\mathbf{w}_1^{(i)}\ \mathbf{w}_2^{(i)}]$ where $\mathbf{w}_j^{(i)}$ is a solution of equation **T**[1].(H) – see Table 1 – with normalized plane wavefront initial condition $(1, 0)^T$ for $j = 1$ and the normalized point source initial condition $(0, 1)^T$ for $j = 2$.

The standard and relative incident geometrical spreadings at an extreme point R_k are given in **T**[3].(C) and **T**[3].(D) respectively, where $\mathcal{L}^{(*)}(R_k) = \sqrt{|\mathbf{w}_{I,1}^{(*)}(R_k)|}$ and $\mathcal{L}^{(*)}(R_k, R_0) = \sqrt{|\Pi_{I,01}^{(*)}(R_k, R_0)|}$. Also, the same applies for the transmitted geometrical spreading. Here, the propagator matrix $\Pi^{(*)}(R_k, R_0)$ is computed by the chain-rule property for propagator matrices.

Finally, the vectorial-complex valued amplitude **U** in equation **T**[1].(B) – see Table 1 – is computed by

$$\mathbf{U}_j(R_k) = \begin{bmatrix} A_j(R_k)\mathbf{e}_{11} \\ 0 \\ A_j(R_k)\mathbf{e}_{13} \end{bmatrix}, \qquad \mathbf{U}_j(R_k) = \begin{bmatrix} A_j(R_k)\mathbf{e}_{13} \\ 0 \\ -A_j(R_k)\mathbf{e}_{11} \end{bmatrix}, \qquad \mathbf{U}_j(R_k) = \begin{bmatrix} 0 \\ A_j(R_k) \\ 0 \end{bmatrix} \tag{7}$$

for P, SV and SH waves respectively, incident at point R_k with $j = I$ and transmitted for $j = T$. Here, $\mathbf{e}_1 = v_j(R_k)\mathbf{p}_j(R_k)$ is the polarization vector at point R_k and A is the solution of the transport equation (see Table 1, Eq. (F)). The equations are depicted from **T**[3].(E) to **T**[3].(J).

[4] Subscript T stands for transmitted data and I for incident data.

Numbers α_j and ρ_j are related to the incident/transmitted angle and density respectively, and also κ is the KMAH index (counting the number of caustic points along the ray). The function $\mathcal{C}$ is the reflection/transmission coefficient at R_m following Snell's law; see [4,6].

5. Using `art-fun/`

In this section we describe some technical details on how using `art`, assuming that the theoretical basis is sufficiently clear from the previous sections. The library is written in C, under the GNU Scientific Library standards.

Once a synthetic model (the seismic profile) has been proposed, the first step is the construction of a description file that specifies the subsurface under investigation, as illustrated in File 1. The description file is self-explained, composed from three types of sections: *box*, *surface* and *layer*. The layer section contains the following subsections: `id` is the layer number, `type` is the string of velocity type (as convention in Table 2), `Pvelocity` defines the analytical coefficients associated to P waves (analogous to `Svelocity` for S waves), `interface` defines the control points associated to the spline representation of an interface and `density` is a positive number representing the constant density within layer. Variable `coeff` is an array with coefficients $\{\mathbf{A}, \mathbf{a}, c\}$ of Eq. (3),

$$\texttt{coeff} = \{c\}, \qquad \texttt{coeff} = \{\mathbf{a}_1, \mathbf{a}_2, c\}, \qquad \texttt{coeff} = \{\mathbf{A}_{11}, \mathbf{A}_{12}, \mathbf{A}_{21}, \mathbf{A}_{22}, \mathbf{a}_1, \mathbf{a}_2, c\}$$

for `type=ART_VCONST`, `type=ART_VAFF`, `ART_VASS`, `ART_VCGL` and `type=ART_VQSS` respectively. If subsection `Svelocity` is empty, we use the approximation $v_S \approx v_P/\sqrt{3}$ (see [8]) for P wave velocity v_P and S wave velocity v_S. Finally, interface points are given in `x`= $\{x_1, x_2, \ldots, x_l, \ldots\}$ and `z`= $\{z_1, z_2, \ldots, z_l, \ldots\}$ so that (x_l, z_l) is a spline control point. By default, if `id` $= k$ then the interface defined within the layer section is the interface number $k + 1$, see Fig. 3. Also, the box constraint $0 \leq z \leq d$, implies that the last layer section has the interface defined by `z` $= \{d, d, \ldots, d, \ldots\}$.

```
box{
  xaxis = { a, b }
  zaxis = { 0, d }
}

surface{
   x = { x1, x2, x3 }
   z = { 0 , 0 , 0  }
}

layer{

  id = ...
  type = ...

  Pvelocity{
     coeff = { ... }
  }
  Svelocity{
     coeff = { ... }
  }
  interface {
     x = { ... }
     z = { ... }
  }
  density = ...
}
```

File 1: General art description file, containing three global sections *box*, *surface* and *layers*. The layer section appears many times as the number of layers.

We remark that not all the functions implemented in `art` are implemented in `art-fun/`, although future versions of latter may include a specific example program for each `art` function. Also, `art-fun/` can be used for a first contact with the library. More optimized codes can be written in C language using `art` functions, although this includes additional effort which includes: learning the documentation of `art`, writing a C code, properly linkage of libraries for compilation and finally execution.

The functions distributed within `art-fun/` are listed below. Routines ending with `*-view` or `*-print` were implemented to view and print the results generated from `art-fun/` functions. Also, every function on the first column below has a brief help guide which can be seen using the command line `art-` *name-of-function* `-h`.

```
art-csrays          art-profile          art-rayset-path      art-twop-view
art-csrays-print    art-profile-print    art-rayset-print     art-vrms
art-csrays-view     art-profile-view     art-rayset-view      art-vrms-print
art-fan             art-ray-oneway       art-ray-view         art-vrms-view
art-fan-path        art-ray-oneway-path  art-twop
art-fan-print       art-ray-print        art-twop-path
art-fan-view        art-rayset           art-twop-print
```

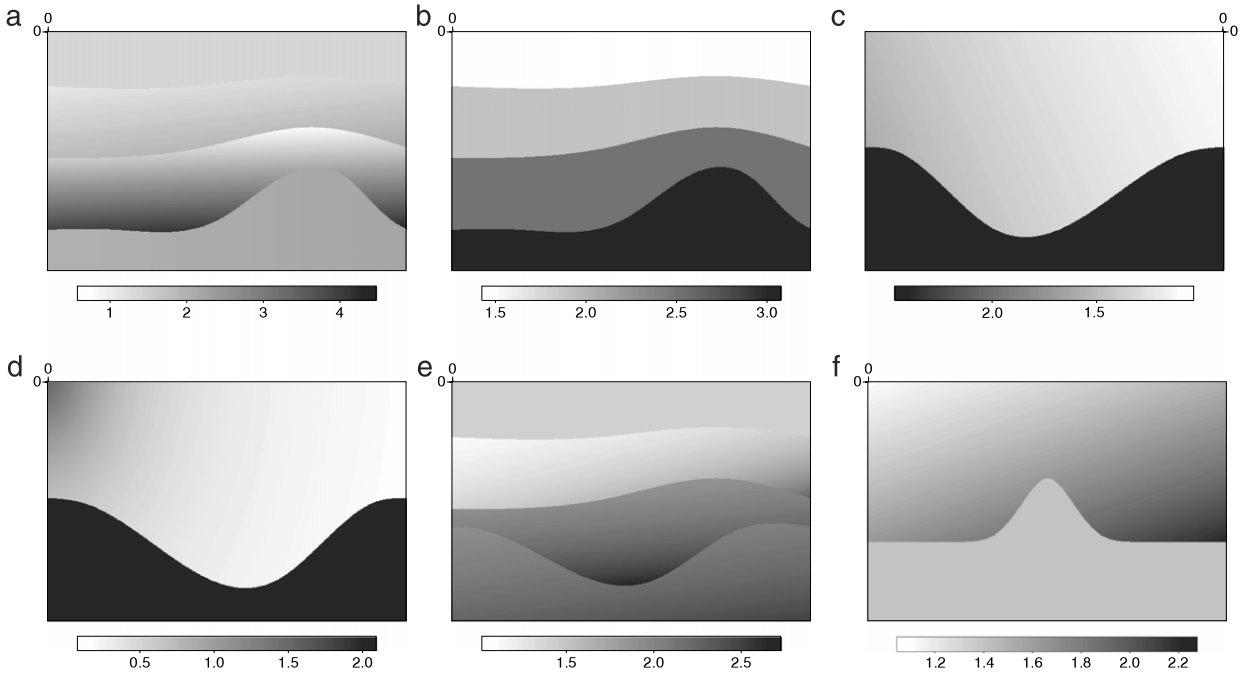

Fig. 8. Example of six seismic models used for simple test, distributed within `art-fun/`. Each figure was generated with the routine `art-profile-print`.

As for `*-view` and `*-print` functions, we remark that typing his name on command line we get a help guide, as shown below:

```
$ art-ray-print

Usage: art-ray-print [-a] [-b] [<name>]
Options
    [-a] seismic model file
    [-b] raypath file
    [<name>] output postscript
```

6. Practical examples

As shown in Fig. 8, we present six seismic profiles, each one with an analytical velocity model within layers, as those presented in Table 2. We remark that, seismic models are described through File 1. Six velocity models are distributed within `raft-fun/` and are denoted by `M00-4L-aff.cfg`, `M01-4L-const.cfg`, `M03-2L-qss.cfg`, `M04-4L-ass.cfg` and `M05-2L-cgl-cfg`, each one associated to Fig. 8(a)–(e), respectively.

6.1. One-way ray tracing

Let us consider the velocity model from Fig. 8(a). Our goal is a ray tracing starting from surface at point $x = 1$, reflecting at the third interface (separating third and fourth layers), and with initial angle $\theta = 25^\circ$. Next, we want to export the output data in order to print a postscript image of the raypath. This is done in the following command lines, respectively

```
$ art-ray-oneway -m M00-4L-aff.cfg -s 1.0 -a 25
                 -i 3 -n 20 -o model.dat > ray.dat 2> report.txt

$ art-ray-print model.dat ray.dat rayM{00}_{2}5.ps
```
(8)

In the first line above, we use file `M00-4L-aff.cfg` as the description file for the velocity model. The function `art-ray-oneway` exports a matrix with columns representing each interface; such a matrix is printed within file `model.dat`.[5] The output `ray.dat` is a two-column matrix with the (x, z) coordinates of the raypath. File `report.txt` (on standard error output) contains a complete report of the ray through the profile. A complete example of `report.txt` is shown in Appendix B. Finally, the raypath postscript image for the above example is depicted in Fig. 9(a). Fig. 9(b) represents the same ray tracing, but with initial angle equal to $\theta = 0^\circ$. This is done changing the flag `-a 25` on command line (8) to `-a 0`.

Let us give another example using routine `art-ray-oneway` and extracting some information from the report file. We are able to trace many rays (creating a fan of rays) starting from source point $x = 1$, with initial angle $-50^\circ \leq \theta \leq 50^\circ$ and also reflecting at the third interface. This is done in Script 1 (see Appendix B). Fig. 9(c) was generated with the command line `art-ray-print model.dat allrays.dat myfig.ps`. Fig. 9(d) shows the traveltime curve obtained from the report file for each successfully traced ray. We remark that this curve can be easily used to obtain a seismic record associated to the common-source event of Fig. 9(c).

[5] If `Grace` is installed, the command line `xmgrace -nxy model.dat` displays the seismic profile.

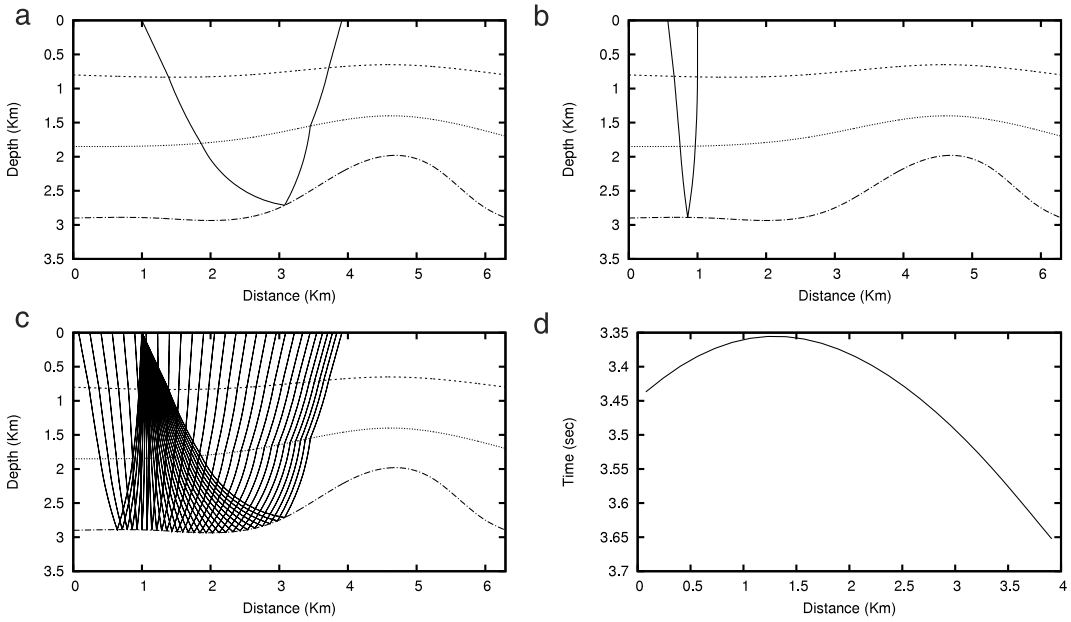

Fig. 9. (a) and (b): Example of a *one-ray tracing* using routine `art-ray-oneway` with initial angles $\theta = 25^\circ$ and $\theta = 15^\circ$ respectively. (c) and (d): Output from Script 1 (see text for details).

6.2. One-way ray tracing: given a ray code

The routine `art-ray-oneway-path` is very similar to `art-ray-oneway`, although the first compute the ray following a given path, or a *ray code*, as described in Section 4.1. A typical execution follows:

```
$ art-ray-oneway-path -m M01-4L-const.cfg -p path.dat -s 1.0 -a 25
                      -n 20 -o model.dat > ray.dat 2> report.txt

$ art-ray-print model.dat ray.dat rayM00_path.ps
```
(9)

The flag `-p` determines a file for the raycode `path.dat`. For example (9), it is a file with a single line given by `10 0 1 1 1 2 2 2 2 1 0`; with first position indicating the path length, followed by the sequence of layers for wave propagation. It is important to note that multiple reflections, on seismic surveys, can be modeled with the usage of such a ray tracing. The file `report.txt` has the same pattern described in Appendix B. The output file `ray.dat` for a successful ray tracing is exported to a postscript image using routine `art-ray-print`, as depicted in Fig. 10(a).

6.3. One-way ray tracing: source to interface

This routine, called `art-ray-s2i`, determines the ray following a given path and starting at a source point, which can be placed anywhere inside the seismic box. A typical execution follows, using the seismic model description file `M01-4L-const.cfg`:

```
$ art-ray-s2i -m M01-4L-const.cfg -x 1 -z 1.5 -a 140
              -p path.dat -n 20 -o model.dat > ray.dat 2>report.txt

$ art-ray-print model.dat ray.dat rayM00_s2i.ps
```
(10)

In the above command line, flag `-a 140` indicates an initial angle $\theta = 140^\circ$ for the wave propagation. Flags `-x 1` and `-z 1.5` set the source position $x = 1$ and $z = 1.5$ within the profile. The above example uses the same ray code given in the command line (9) (given through the file `path.dat`). The exported postscript image for this example is presented in Fig. 10(b).

Script 2 creates a fan of rays starting at the source position $(x, z) = (1, 2.5)$ varying the angle from $\theta = 110^\circ$ to $\theta = 170^\circ$ equispaced with $\Delta\theta = 3^\circ$. Gathering only the feasible rays within the seismic box, the resulting ray tracing is presented in Fig. 10(c). Using the same source point and changing the ray code (through the command line `echo "7 2 2 2 2 2 1 0" >path.dat`), we obtain the ray tracing shown in Fig. 10(d).

6.4. One-way ray tracing: free wave

The library is also capable to trace rays starting at the source point and traveling through an unbounded media, until the wavefront reaches a given traveltime. We call this experiment as a "free ray tracing" since there is any kind of reflection/transmission at a boundary. To specify the unbounded media, we choose a particular layer from a given seismic profile, which is described within a file `*.cfg`. The routine that computes 'free rays' is called `art-ray-free`. A typical example follows:

```
$ art-ray-free -m M04-4L-ass.cfg -x 1.5 -z 1.5
               -a 5 -t 2.5 -l 1 -n 200 > ray.dat 2>report.dat

$ art-ray-free-print ray.dat image.ps
```
(11)

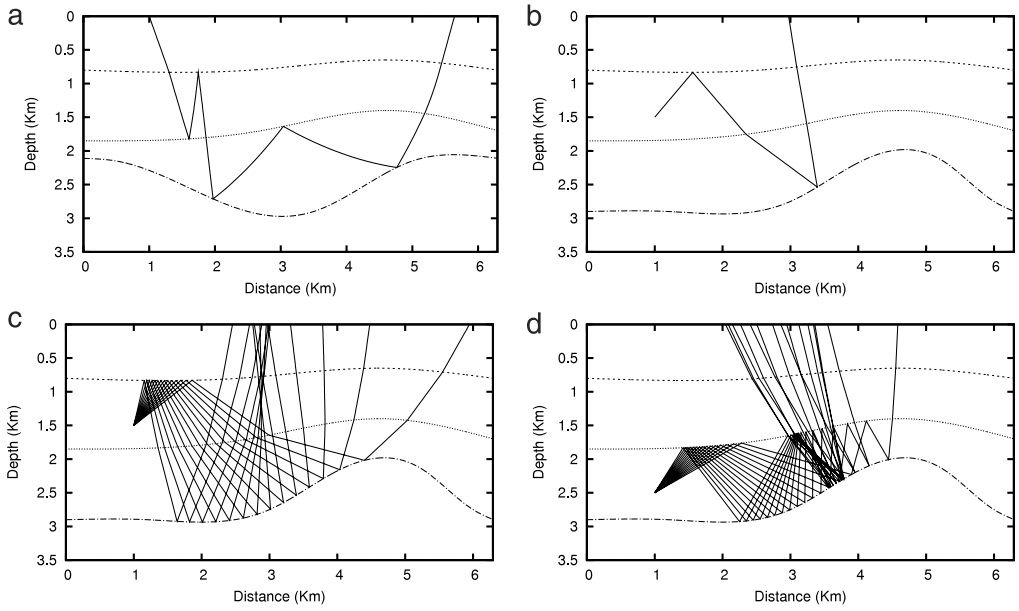

Fig. 10. (a) Example of *one-way-tracing* with initial angle $\theta = 25^\circ$ and following the ray code {0, 1, 1, 1, 2, 2, 2, 1, 0}, using routine `art-ray-oneway-path`. (b) Example of *source to interface* ray tracing, with initial angle $\theta = 140^\circ$ and following the ray code {1, 1, 2, 2, 1, 0}. (c) and (d) Examples from Script 2, see text for more details.

In the above command line, we ask for a ray starting at the source point $(x, z) = (1.5, 1.5)$ and traveling through the first layer (flag `-l 1`) from description file `M04*.cfg` (see Fig. 8(d)), reaching the traveltime $T = 2.5$ (flag `-t 2.5`), starting with initial angle $\theta = 5^\circ$ and exporting 200 points upon the curve. The standard error output `report.dat` gives the final point of the curve. The postscript image generated with routine `art-ray-free-print` is shown in Fig. 11(a). Using model `M00*.cfg` (see Fig. 8(a)) and a simple loop, as presented in (12), we create a fan of rays starting at the same source point and reaching the same traveltime.

```
$ for ((j=0;j<=360;j=j+14)); do
      art-ray-free -m M00-4L-aff.cfg -x 1 -z 3 -t 1.5 \
                   -n 1000 -a $j -l 1 \
                    >> ray.dat 2>> wfront.dat;
  done
```
(12)

The resulting file `ray.dat`, a stacking of feasible rays, is used to generate the image depicted in Fig. 11(b). We remark that the first layer has constant velocity. Changing the layer flag to `-l 2`, i.e., using the second layer, we obtain the result from Fig. 11(c) (with affine velocity). The standard error output `wfront.dat` from routine `art-ray-free`, see command line (11), gives the final point from the curve. Such a point belongs to the wavefront traveling through the medium. Routine `art-ray-wavefront-print` generates a postscript image, as shown in (13). The resulting image is presented in Fig. 11(d).

```
$ art-ray-wavefront-print ray.dat wfront.dat 1 3 image.ps
```
(13)

6.5. Fan of rays

In this example, we consider the model `M05*.cfg` (see Fig. 8(f)), with an anticlinal reflector on the first interface. The command line (14) shows a construction of fan of rays reflecting at this anticlinal reflector using routine `art-fan`. The source is positioned at $x = 1$ and the fan has depth-angle aperture of 45°. The routine is attempting to trace 30 rays within this aperture.

```
$ art-fan -m M05-2L-cgl.cfg -s 1.0 -r 30 -i 1 -a 45
          -n 20 -o model.dat > rays.dat 2>report.txt;
```
(14)

Fig. 12(a) was generated with routine `art-fan-print` using the output from the command line above. Changing flag `-s 1.0` to `-s 5`, means that we are changing the source point to $x = 5$. The result is presented in Fig. 12(b). Also, routine `art-fan-path` computes a fan of rays following a given path (or a ray code) (see functions `art-fan*` distributed within `art-fun/`).

6.6. Two-point ray tracing

We now consider the two-point ray tracing, which is implemented on routines `art-twop` and `art-twop-path`. For illustration, we use the velocity model of Fig. 8(c)—described within file `M02*.cfg`. The command line (15) executes a two-point ray tracing between the source point at $x = 1.6$ and the receiver at $x = 4$, reflecting at a synclinal reflector.

```
$ art-twop -m M02-2L-ass.cfg -s 1.6 -r 4 -i 1 -n 20
           -o model.dat > rays.dat 2> report.txt
```
(15)

Plotting the output file `rays.dat`, using routine `art-twop-print`, gives us the postscript image shown in Fig. 12(c). Also, the file `report.txt` generated from the standard error output, displays the number of rays connecting the source and the receiver and the most useful incident information at the receiver, e.g., the traveltime, the slowness vector and the vectorial amplitude. This report is presented below:

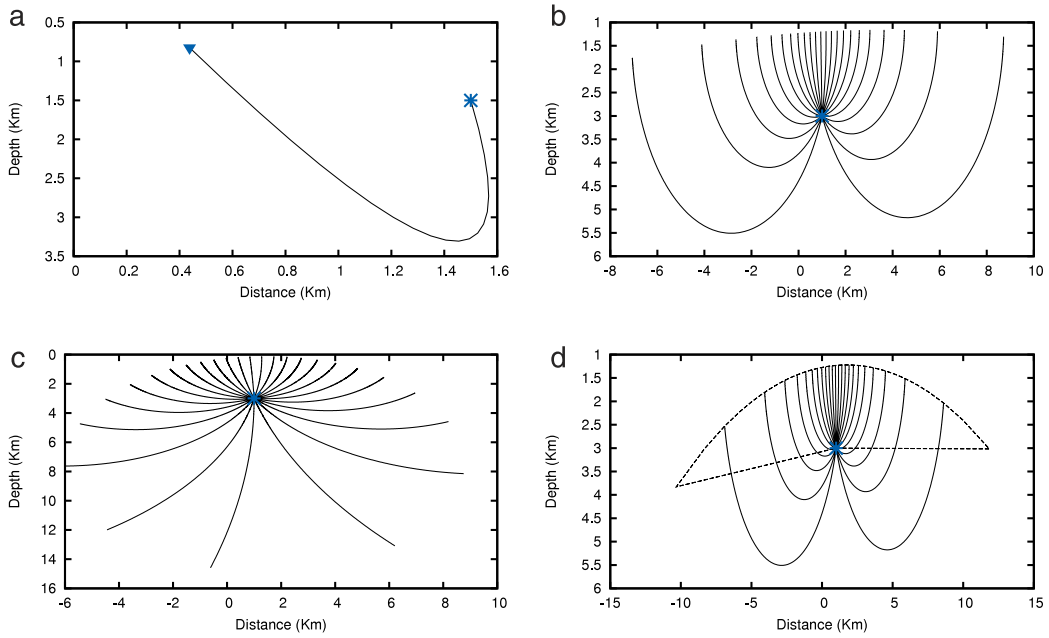

Fig. 11. (a) and (b) Ray and a family of rays at an unbounded media with affine velocity; see text for details. (c) Family of rays at an unbounded media with constant velocity. (d) Wavefront construction using ray tracing and routine `art-ray-free`.

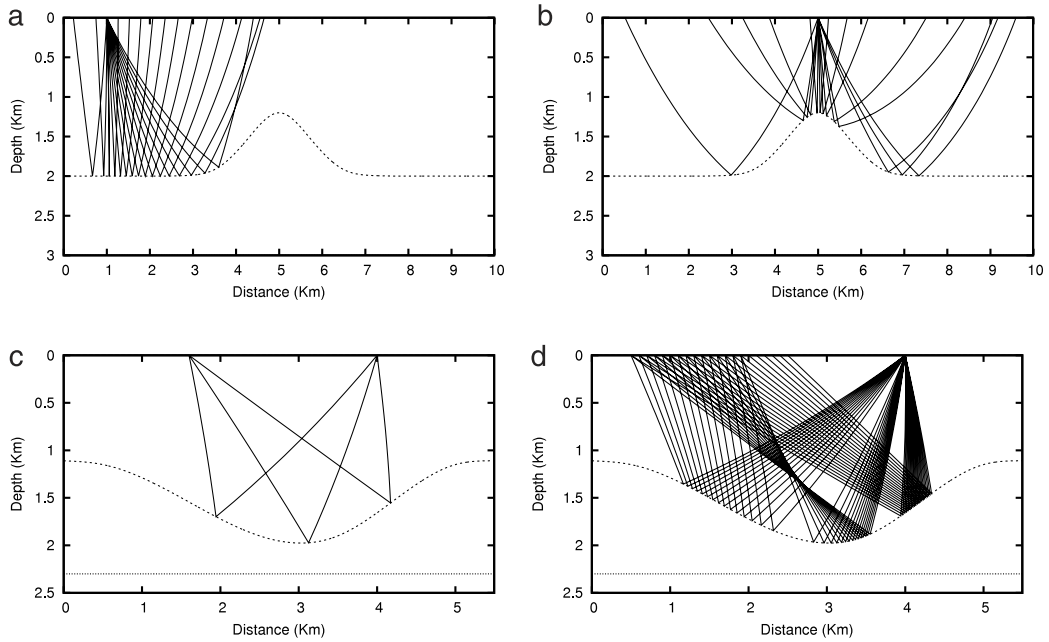

Fig. 12. (a) and (b): Example of routine `art-fan` to create a fan of rays. See text for more details. (c) Two-point ray tracing using routine `art-twop`: result from the command line (15). (d) Output rays from Script 4 (see Appendix B).

```
Number of rays: 3

-- Ray 1 --:
Time=3.435063, p=(0.522485,-0.525010)
Amplitude A = (0.061460, 0.000000)
Initial angle between source and receiver: 13.579396 deg
-- Ray 2 --:
Time=3.523274, p=(0.242478,-0.699879)
Amplitude A = (0.032317, 0.000000)
Initial angle between source and receiver: 39.092164 deg
-- Ray 3 --:
```

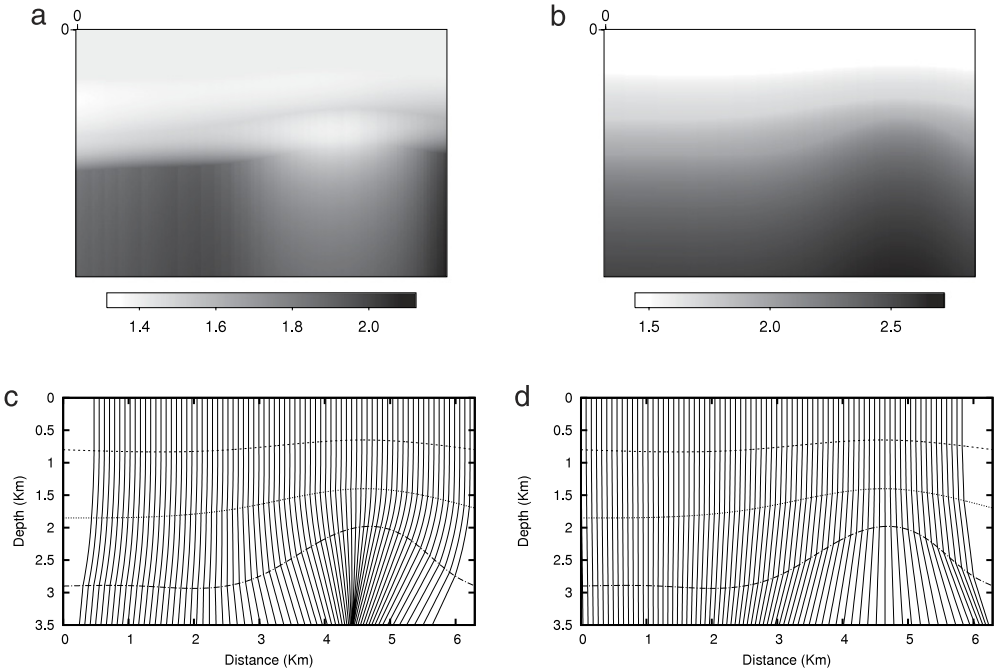

Fig. 13. (a) and (b): RMS velocity field. (c) and (d): Image rays for RMS velocity (a) and (b) respectively.

```
Time=3.410444, p=(-0.116572,-0.731463)
Amplitude A = (0.016891, 0.000000)
Initial angle between source and receiver: 58.500262 deg
```

We remark that the amplitude function reported above is represented as a complex number $(\Re(A), \Im\mathrm{m}(A))$.

To illustrate an application of this ray tracing – with routine `art-twop` – and the report file above, we propose Script 4 (see Appendix B) to reunite all the feasible rays between twenty source points and a fixed receiver. The source position varies uniformly in the interval $[0.5, 2.5]$. The report file for each two-point ray tracing gives us the traveltime between the source and the receiver. The results are presented in File 3 (see Appendix B): the first column representing the source location, the second column for the number of rays connecting the source and the receiver; third and fourth columns giving the multi-valued traveltime for each traced rays. The rays obtained with Script 4 are depicted in Fig. 12(d).

Routine `art-twop-path` – distributed within `art-fun/` – is also similar to `art-twop`, computing Fermat rays following a given ray code sequence.

6.7. Root mean square velocity

For a given a velocity model $v = v(\mathbf{x})$, the *Root mean square* velocity (RMS) is denoted by v_{rms} and defined by the squared average of $v^2(\mathbf{x})$ along *image rays*. These are rays starting from the surface with initial angle $\theta = 0^\circ$ and traveling through the seismic profile. Mathematically, the RMS velocity is defined by $v^2_{\mathrm{rms}}(\mathbf{x}) = \frac{1}{\ell}\int_\Omega v^2(\mathbf{x})d\tau$ with $\ell = \int_\Omega d\tau$, Ω an image ray and τ the traveltime along the ray. Needless to say that $\mathbf{x} = \mathbf{x}(\tau)$ is the raypath of Ω. To compute the RMS velocity, we use routine `art-vrms`. The command line (16) illustrates the calculation for the RMS velocity model from Fig. 8(a). The mean velocity is presented in Fig. 13(a). The same applies for Fig. 13(b) that corresponds to the seismic model from Fig. 8(b). Also, the corresponding image rays for these examples are presented in Fig. 13(c) and (d) respectively.

```
$ art-vrms -m M00-4L-aff.cfg -r 80 -s 0.01 -t 3 -n 20
           -o model.dat > vrms.dat 2> irays.dat;

$ art-vrms-print model.dat irays.dat vrms.dat VRMS.eps IRAYS.eps
```
(16)

6.8. Common-shot rays

Common-shot experiments are widely used in seismic survey. We provide the routine `art-csrays` to reunite all the feasible rays obtained for such an experiment. As an example, we have computed the common-shot rays for each velocity model from Fig. 8. The source is positioned at $x = 1$ and 50 receivers were equally distributed at the surface. The command line is given below, and the results are depicted in Fig. 14 following the same label from Fig. 8, i.e., Fig. 14(a) for the CS-rays corresponding to Fig. 8(a), and henceforth:

```
$ art-csrays -m M00-4L-aff.cfg -r 50 -s 1 -n 20
             -o model.dat > rays.dat
```
(17)

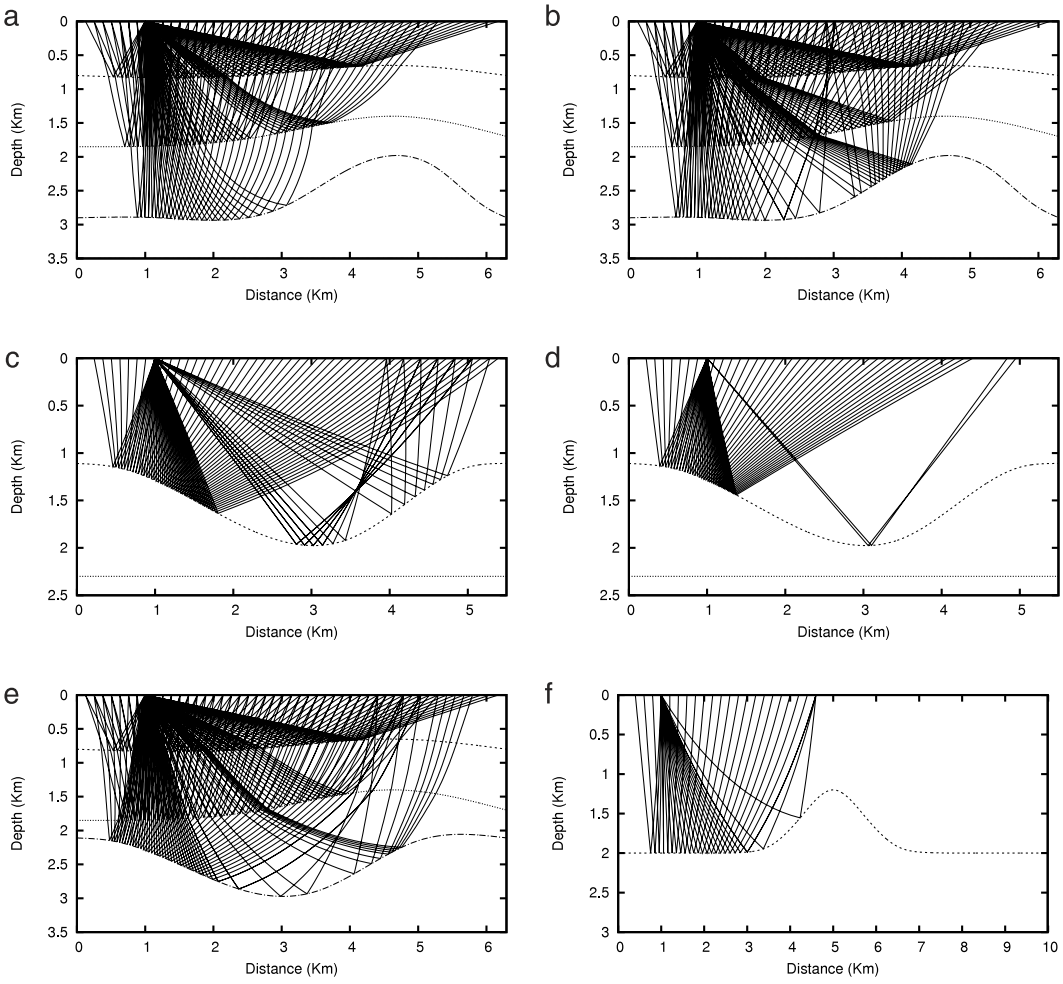

Fig. 14. Common-shot rays using routine `art-csrays`. See text for more details.

6.9. A complete example: caustic rays

A seismic profile with four layers, each one with constant velocity; being the first and the second interface a concave/convex reflector. From first to fourth layer, the velocities were set to 4 $\frac{km}{s}$, 1.5 $\frac{km}{s}$, 4 $\frac{km}{s}$ and 1.7 $\frac{km}{s}$ respectively. Density is one through all the profiles. The velocity model is presented in Fig. 15(a).

Caustic rays are very important in dynamical ray tracing due to the intersection of many wavefronts at the same traveltime; each wavefront being represented by a single ray. This phenomenon causes a singularity point, where the Jacobian vanishes and the amplitude goes to infinity. The ray tracing library was tested at a profile with lens shape, to generate a caustic point situation. The rays obtained are presented in Fig. 15(b), where we clearly see the caustic point. The traveltime curve versus offset (for every successful ray tracing) is presented in Fig. 15(c). Fig. 15(d) presents the multivalued amplitude function versus offset. Most of the seismic methods are based on a smooth velocity model, neglecting the strong discontinuity for the velocity at an interface. Sometimes, the smooth model may cause a caustic point, and a subsequent loss of amplitude information along the ray. This problem cannot occur with the analytical ray tracing since the amplitude values depend only on the Jacobian of extreme points (lying at the interface).

The following script sets the construction of a ray set on the lens shape profile. All rays are reflecting on the third interface, with source placed at $x = 0.5$. The position of each end point and also the amplitude value are extracted from the report file `report.txt` (see File 2 at Appendix B). For this particular example, the amplitude arriving at the surface is a pure imaginary number. Indeed, the Jacobian vanishes from the first interface to surface, causing a phase shift (KMAH index equal to 1). We remark that, Fig. 15(d) represents the absolute value of the amplitude function.

```
#!/bin/bash
#
#

for ((j=-40;j<40;j=j+3)); do

    art-ray-oneway -m modeloII.cfg -s 0.5 -a $j -i 3 -n 20 -o model.dat > ray.dat 2> report.txt;

    status=$(grep "Error:" report.txt);
```

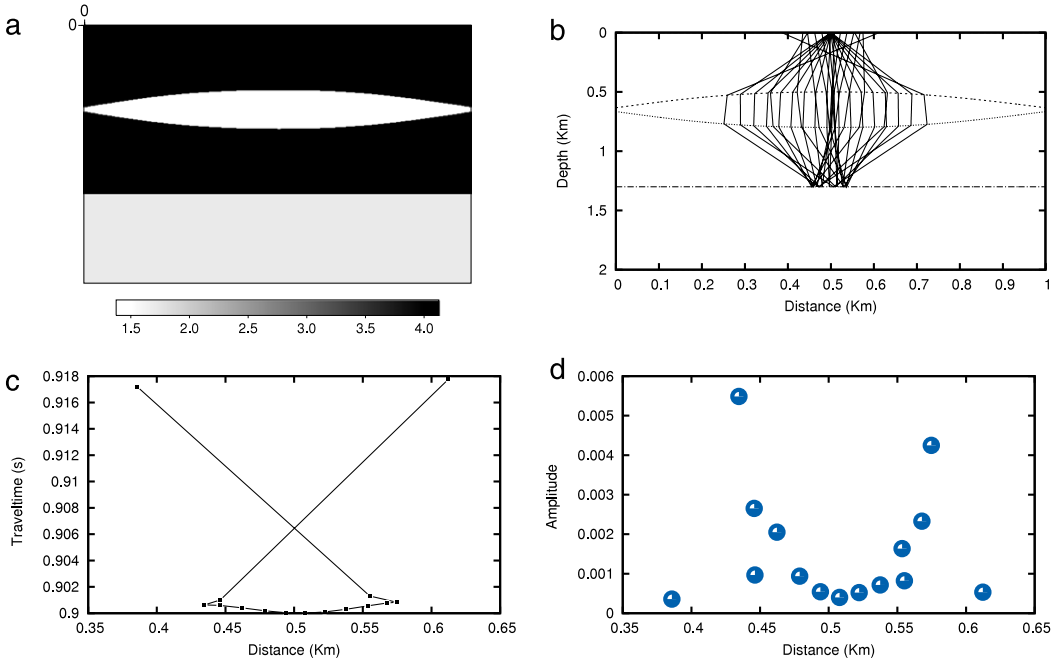

Fig. 15. (a) Velocity profile. (b) Caustic rays. (c) Traveltime curve. (d) Amplitude value for successfully computed caustic rays.

```
    if [ ! "$status" ]; then

      cat ray.dat >> allrays.dat;

      T=$(grep 'Traveltime' report.txt | awk '{print $2}' | tail -1);
      x=$(grep 'Position' report.txt | awk '{print $2}' | tail -1);
      A=$(grep "Amplitude" report.txt | tail -1 | awk '{print $7}');

      echo $x $T >> traveltime.dat     -
      echo $x $A >> amplitude.dat

    fi
done
```

7. Conclusions

As a typical inverse problem, the seismic imaging problem can be solved through several direct problems. Most of these direct problems are intrinsically related to ray tracing [23–28]. Computing rays in an analytical medium, as presented here, is a fast option for inversion techniques. Execution time usually varies from milliseconds to 3 min, depending on the data size. Our computational tool can be used for several seismic procedures, e.g. Kirchhoff modeling. Though the usage of `art` is mainly designed for seismic problems, the library also can be used to different problems related to optics, tomography and wave propagation in general.

Acknowledgments

The authors would like to thank the reviewers for their comments and critical contribution to this work.
The second author was supported by Brazil/CNPq (National Council for Scientific and Technological Development).

Appendix A. Analytical expressions

A.1. Ray tracing functions {**d**, **t**, **s**}

(A) Constant velocity: Setting $\mathbf{A} \in \mathbb{R}^{2\times 2}$ and $\mathbf{a} \in \mathbb{R}^2$ as the zero matrix and vector in (3), we obtain a constant velocity field $v(\mathbf{x}) = c$. The kinematic solution (4), for $n = 1$, consider the following functions

$$\mathbf{d}_1(s) = c\, s\, \mathbf{p}_0, \qquad \mathbf{t}_1(s) = s/c, \qquad \mathbf{s}_1(s) = \mathbf{0} \tag{A.1}$$

where $u \equiv s$ denotes arclength along the raypath. Since the Hessian matrix of $v = v(\mathbf{x})$ is zero, it follows that $\nu(u) = 0$ vanishes for $u \geq 0$ and the in-plane and transverse dynamical systems are equal, with analytical solution given by $\mathbf{T}[1].(\mathrm{L})$, where $y_1(s) = cs$.

(B) Affine velocity: Setting $\mathbf{A} = \mathbf{0}$ in Eq. (3) we obtain the affine velocity field $v(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + c$. The kinematic solution (4), for $n = -1$, consider the following functions: $\mathbf{d}_{-1}(\xi) = \frac{\mathbf{p}_0 - \kappa\mathbf{a}}{\|\mathbf{a}\|^3}\left\{\sin\arctan\frac{\xi-\kappa}{\sqrt{\sigma}} + \sin\arctan\frac{\kappa}{\sqrt{\sigma}}\right\} + \frac{\mathbf{a}}{\|\mathbf{a}\|^3}\left\{\frac{1}{\sqrt{\sigma+(\xi-\kappa)^2}} - \frac{\|\mathbf{a}\|}{\|\mathbf{p}_0\|}\right\}$, traveltime $\mathbf{t}_{-1}(\xi) = \ln\left\{\frac{\sec\arctan\left(\frac{\xi-\kappa}{\sqrt{\sigma}}\right)+\frac{\xi-\kappa}{\sqrt{\sigma}}}{\sec\arctan\left(\frac{-\kappa}{\sqrt{\sigma}}\right)+\frac{-\kappa}{\sqrt{\sigma}}}\right\}$ and $\mathbf{s}_{-1}(\xi) = -\xi\mathbf{a}$. Here $u \equiv \xi$ is the integration parameter, $\kappa = \mathbf{p}_0^T\mathbf{a}/\|\mathbf{a}\|^2$ and $\sigma = \|\mathbf{p}_0\|^2/\|\mathbf{a}\|^2 - \kappa^2$ are positive constants.[6] Since the Hessian matrix of the velocity field is zero, the dynamic function ν vanishes for $\xi \geq 0$ and in-plane/out-of-plane solutions are equal to **T**[1].(L), with $y_{-1}(\xi) = \frac{\sin\arctan\frac{\xi-\kappa}{\sqrt{\sigma}} + \sin\arctan\frac{\kappa}{\sqrt{\sigma}}}{\|\mathbf{a}\|^3\sigma}$.

(C) Affine square of slowness: Setting $\mathbf{A} = \mathbf{0}$ in Eq. (3) we obtain the affine square of slowness velocity field $1/v(\mathbf{x})^2 = \mathbf{a}^T\mathbf{x} + c$. The kinematic solution (4), for $n = 2$, consider the following functions: $\mathbf{d}_2(\sigma) = \sigma\mathbf{p}_0 + \sigma^2\frac{\mathbf{a}}{4}$, traveltime $\mathbf{t}_2(\sigma) = \frac{\sigma}{v(\mathbf{x}_0)^2} + \frac{\mathbf{a}^T\mathbf{p}_0}{2}\sigma^2 + \frac{\mathbf{a}^T\mathbf{a}}{12}\sigma^3$ and $\mathbf{s}_2(\sigma) = \sigma\frac{\mathbf{a}}{2}$. Here, $u \equiv \sigma$ is the integration parameter. Since $\Phi(v) = v^{-2}$, it follows from the chain-rule that the Hessian matrix of v is given by $\mathbf{H}_v = \frac{3}{4}v(\mathbf{x})^5\mathbf{a}\mathbf{a}^T$. Hence, the dynamical functions ν and y_2 are given by $\nu(\sigma) = -\frac{3}{4}v^4(\sigma)\mathbf{p}(\sigma)^T(\mathbf{J}^T\mathbf{a})(\mathbf{J}^T\mathbf{a})^T\mathbf{p}(\sigma)$ and $y_2(\sigma) = \sigma$.

(D) Logarithmic velocity: Setting $\mathbf{A} = \mathbf{0}$ in Eq. (3) we obtain the logarithmic velocity field $\ln v(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + c$. The kinematic solution (4), for $n = 0$ and traveltime as the integration parameter ($u \equiv T$), consider the following functions: $\mathbf{d}_0(T) = \frac{\mathbf{p}_0 - \kappa\mathbf{a}}{\sqrt{a\sigma}}f_1(T) - \frac{\mathbf{a}}{2a}f_2(T)$, traveltime $\mathbf{s}_0(T) = -T\mathbf{a}$, where $\kappa = \mathbf{a}^T\mathbf{p}_0/\|\mathbf{a}\|^2$, $a = \|\mathbf{a}\|^2$, $\sigma = \|\mathbf{p}_0\|^2 - (\mathbf{a}^T\mathbf{p}_0)^2/\|\mathbf{a}\|^2$. Also $f_1(T) = \arctan\left(T\sqrt{\frac{a}{\sigma}}\right) + \arctan\left(\kappa\sqrt{\frac{a}{\sigma}}\right)$ and $f_2(T) = \ln\left[\frac{(T-\kappa)^2 - (\sigma/a)}{\kappa^2 - (\sigma/a)}\right]$.

Since $\Phi(v) = \ln v$ it follows that the Hessian matrix of v is given by $\mathbf{H}_v = v(\mathbf{x})\mathbf{a}\mathbf{a}^T$ and the dynamical functions ν and y_0 are given by

$$y_0(T) = \frac{1}{\sqrt{a\sigma}}\left\{\arctan\left[(T-\kappa)\sqrt{\frac{a}{\sigma}}\right] + \arctan\left[\kappa\sqrt{\frac{a}{\sigma}}\right]\right\}$$

and $\nu(T) = -v^2(T)v^4(T)\mathbf{p}(T)^T(\mathbf{J}^T\mathbf{a})(\mathbf{J}^T\mathbf{a})^T\mathbf{p}(T)$.

(E) Quadratic square of slowness: Here $\Phi(v) = v^{-2}$, and we assume that **A** is a two-dimensional symmetric matrix in the general velocity model (3). The kinematic ray equation (2) in this case, for $n = 2$ and $u \equiv \sigma$, reduces to $\frac{d\mathbf{x}}{d\sigma} = \mathbf{p}$, $\frac{d\mathbf{p}}{d\sigma} = \frac{1}{2}\nabla\Phi(v)$ and $\frac{dT}{d\sigma} = \Phi(v)$. Since $\nabla\Phi(v) = 2\mathbf{A}\mathbf{x} + \mathbf{a}$ we obtain a second order differential equation[7] for the slowness vector, i.e. $\ddot{\mathbf{p}}(\sigma) = \mathbf{A}\mathbf{p}$ whose fundamental solutions are $\mathbf{p}_1 = e^{\sigma\sqrt{\mathbf{A}}}\mathbf{p}_0$ and $\mathbf{p}_2 = e^{-\sigma\sqrt{\mathbf{A}}}\mathbf{p}_0$. Such solutions are presented in Appendix A.2, using a spectral factorization of matrix **A**. It follows that $\mathbf{p}(\sigma) = \xi\,\mathbf{p}_1(\sigma) + \eta\,\mathbf{p}_2(\sigma)$, $\mathbf{x}(\sigma) = \mathbf{x}_0 + \mathbf{d}_2(\sigma)$ and $T(\sigma) = T_0 + \mathbf{t}_2(\sigma)$. Here, $\eta = \mathbf{p}_0^T(\dot{\mathbf{p}}(0) - \dot{\mathbf{p}}_1(0))\mathbf{p}_0^T(\dot{\mathbf{p}}_2(0) - \dot{\mathbf{p}}_1(0))$, $\xi = 1 - \eta$, $\dot{\mathbf{p}}(0) = \mathbf{A}\mathbf{x}_0 + \mathbf{a}/2$, $\mathbf{d}_2(\sigma) = \xi\,\mathbf{f}(\sigma) + \eta\,\mathbf{g}(\sigma)$, traveltime

$$\begin{aligned}\mathbf{t}_2(\sigma) = {} & \frac{\sigma}{v(\mathbf{x}_0)^2} + 2\xi(\mathbf{A}\mathbf{F}(\sigma))^T\mathbf{x}_0 + 2\eta(\mathbf{A}\mathbf{G}(\sigma))^T\mathbf{x}_0 + 2\xi\eta\int_0^\sigma[\mathbf{A}\mathbf{g}(w)]^T\mathbf{f}(w)dw \\ & + \mathbf{a}^T[\xi\mathbf{F}(\sigma) + \eta\mathbf{G}(\sigma)] + \xi^2\int_0^\sigma[\mathbf{A}\mathbf{f}(w)]^T\mathbf{f}(w)dw + \eta^2\int_0^\sigma[\mathbf{A}\mathbf{g}(w)]^T\mathbf{g}(w)dw.\end{aligned}$$

We use $\mathbf{f} = \int\mathbf{p}_1$, $\mathbf{g} = \int\mathbf{p}_2$, $\mathbf{F} = \int\mathbf{f}$ and $\mathbf{G} = \int\mathbf{g}$. Functions **f**, **g**, **F** and **G** are presented in Appendix A.2. Although analytic expressions can be written for those integrals, they are computed numerically. Finally, since the Hessian matrix of v is $\mathbf{H}_v = -v^3(\mathbf{x})\mathbf{A} + \frac{3}{4}v^5(\mathbf{x})\mathbf{g}(\mathbf{x})\mathbf{g}(\mathbf{x})^T$ with $\mathbf{g}(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{a}$, the dynamical function ν is given by

$$\nu(\sigma) = v^2(u)[\mathbf{J}\mathbf{p}]^T\mathbf{A}[\mathbf{J}\mathbf{p}]|_\sigma - \frac{3}{4}v^4\mathbf{p}[\mathbf{J}^T\mathbf{g}][\mathbf{J}^T\mathbf{g}]^T\mathbf{p}|_\sigma \tag{A.2}$$

and $y_2(\sigma) = \sigma$.

A.2. Quadratic square of slowness

The fundamental solution of the differential equation $d^2\mathbf{p}/d\sigma^2 = \mathbf{A}\mathbf{p}$, with $\mathbf{A} \in \mathbb{R}^{2\times2}$ a symmetric matrix, is given by $\mathbf{p}(\sigma) = e^{\pm\sigma\sqrt{\mathbf{A}}}\mathbf{p}_0$, see [29]. Since **A** has a spectral factorization and therefore a square root, we list below all possible solutions, according to the sign of the matrix eigenvalues. Assume λ_1, λ_2 eigenvalues of **A**:

(i) $\lambda_j > 0$, $j = 1, 2 : \mathbf{p}_\pm(\sigma) = \mathbf{S}\,e^{\pm\sigma\sqrt{\Lambda}}\mathbf{S}^{-1}\mathbf{p}_0$

(ii) $\lambda_j < 0$, $j = 1, 2 : \mathbf{p}_1(\sigma) = \mathbf{S}\,\cos(\sigma\sqrt{\Lambda})\mathbf{S}^{-1}\mathbf{p}_0$ and $\mathbf{p}_2(\sigma) = \mathbf{S}\,\sin(\sigma\sqrt{\Lambda})\mathbf{S}^{-1}\mathbf{p}_0$

(iii) $\lambda_1 > 0, \lambda_2 < 0 : \mathbf{p}_j(\sigma) = \mathbf{S}\Lambda_j(\sigma)\mathbf{S}^{-1}\mathbf{p}_0$ with $\Lambda_1(\sigma) = \mathrm{diag}\{e^{\sigma\sqrt{\lambda_1}}, \cos(\sigma\sqrt{-\lambda_2})\}$ and $\Lambda_2(\sigma) = \mathrm{diag}\{0, \sin(\sigma\sqrt{-\lambda_2})\}$

(iv) $\lambda_j = 0, \lambda_k > 0 : \mathbf{p}_\pm(\sigma) = \frac{1}{\lambda_k}\mathbf{X}\mathbf{A}\mathbf{X}^{-1}\mathbf{p}_0 + \frac{1}{\lambda_k}e^{\pm\sigma\sqrt{\lambda_k}}\mathbf{A}\mathbf{p}_0$

(v) $\lambda_j = 0, \lambda_k < 0 : \mathbf{p}_1(\sigma) = \frac{1}{\lambda_k}\mathbf{X}\mathbf{A}\mathbf{X}^{-1}\mathbf{p}_0 + \frac{1}{\lambda_k}\cos(\sigma\sqrt{-\lambda_k})\mathbf{A}\mathbf{y_0}$ and $\mathbf{p}_2(\sigma) = \frac{1}{\lambda_k}\sin(\sigma\sqrt{-\lambda_k})\mathbf{A}\mathbf{y_0}$.

[6] It follows by the Cauchy–Schwarz inequality that $\sigma \geq 0$.

[7] We denote $\dot{\mathbf{p}}$ as the derivative of vector **p**.

Table A.4
Functions {**f**, **g**, **F**, **G**} for the quadratic square of slowness. See Appendix A.1(E).

$$\text{(iv)}\begin{cases}\mathbf{f}=\left\{\dfrac{\sigma}{\lambda_k}\mathbf{D}+\dfrac{e^{\sigma\sqrt{\lambda_k}}-1}{\lambda_k^{3/2}}\mathbf{A}\right\}\mathbf{p}_0\\ \mathbf{F}=\dfrac{\sigma^2}{2\lambda_k}\mathbf{D}\mathbf{p}_0+\left\{\dfrac{\exp(\sigma\sqrt{\lambda_k})-1}{\lambda_k^2}-\dfrac{\sigma}{\lambda_k^{3/2}}\right\}\mathbf{A}\mathbf{p}_0\\ \mathbf{g}=\left\{\dfrac{\sigma}{\lambda_k}\mathbf{D}+\dfrac{1-e^{-\sigma\sqrt{\lambda_k}}}{\lambda_k^{3/2}}\mathbf{A}\right\}\mathbf{p}_0\\ \mathbf{G}=\dfrac{\sigma^2}{2\lambda_k}\mathbf{D}\mathbf{p}_0+\left\{\dfrac{\exp(-\sigma\sqrt{\lambda_k})-1}{\lambda_k^2}+\dfrac{\sigma}{\lambda_k^{3/2}}\right\}\mathbf{A}\mathbf{p}_0\end{cases}\qquad \text{(i)}\begin{cases}\mathbf{f}=\mathbf{B}^{-1}[\exp\sigma\sqrt{\mathbf{A}}-\mathbf{I}]\mathbf{p}_0\\ \mathbf{F}=\mathbf{B}^{-1}[\mathbf{f}(\sigma)-\sigma\mathbf{p}_0]\\ \mathbf{g}=\mathbf{B}^{-1}[\mathbf{I}-\exp(-\sigma\sqrt{\mathbf{A}})]\mathbf{p}_0\\ \mathbf{G}=-\mathbf{B}^{-1}[\mathbf{g}(\sigma)-\sigma\mathbf{p}_0]\end{cases}$$

$$\text{(iii)}\begin{cases}\mathbf{f}=\mathbf{S}\text{diag}\left\{\dfrac{e^{\sigma\sqrt{\lambda_1}}-1}{\sqrt{\lambda_1}},\dfrac{\sin\sigma\sqrt{-\lambda_2}}{\sqrt{-\lambda_2}}\right\}\mathbf{S}^{-1}\mathbf{p}_0\\ \mathbf{F}=\mathbf{S}\text{diag}\left\{\dfrac{e^{\sigma\sqrt{\lambda_1}}-1-\sigma}{\lambda_1},\dfrac{\cos\sigma\sqrt{-\lambda_2}-1}{\lambda_2}\right\}\mathbf{S}^{-1}\mathbf{p}_0\\ \mathbf{g}=\mathbf{S}\text{diag}\left\{0,\dfrac{1-\cos\sigma\sqrt{-\lambda_2}-1}{\sqrt{-\lambda_2}}\right\}\mathbf{S}^{-1}\mathbf{p}_0\\ \mathbf{G}=\mathbf{S}\text{diag}\left\{0,\dfrac{\sigma}{\sqrt{-\lambda_2}}-\dfrac{\sin\sigma\sqrt{-\lambda_2}}{\lambda_2}\right\}\mathbf{S}^{-1}\mathbf{p}_0\end{cases}\qquad \text{(ii)}\begin{cases}\mathbf{f}=\mathbf{S}\Lambda^{-1/2}\sin(\sigma\sqrt{\Lambda})\mathbf{S}^{-1}\mathbf{p}_0\\ \mathbf{F}=\mathbf{S}\Lambda^{-1}[\mathbf{I}-\cos(\sigma\sqrt{\Lambda})]\mathbf{S}^{-1}\mathbf{p}_0\\ \mathbf{g}=\mathbf{S}\Lambda^{-1/2}[\mathbf{I}-\cos(\sigma\sqrt{\Lambda})]\mathbf{S}^{-1}\mathbf{p}_0\\ \mathbf{G}=\mathbf{S}\Lambda^{-1}[\sigma\sqrt{\Lambda}-\sin(\sigma\sqrt{\Lambda})]\mathbf{S}^{-1}\mathbf{p}_0\end{cases}$$

$$\text{(v)}\begin{cases}\mathbf{f}=\left\{\dfrac{\sigma}{\lambda_k}\mathbf{D}+\dfrac{\cos\sigma\sqrt{-\lambda_k}-1}{\lambda_k\sqrt{-\lambda_k}}\mathbf{A}\right\}\mathbf{p}_0\\ \mathbf{F}=\dfrac{\sigma^2}{2\lambda_k}\mathbf{D}\mathbf{p}_0-\left\{\dfrac{\sin(\sigma\sqrt{-\lambda_k})}{\lambda_k^2}+\dfrac{\sigma}{\lambda_k\sqrt{-\lambda_k}}\right\}\mathbf{A}\mathbf{p}_0\\ \mathbf{g}=\left\{\dfrac{-\sin\sigma\sqrt{-\lambda_k}}{\lambda_k\sqrt{-\lambda_k}}\right\}\mathbf{A}\mathbf{p}_0\\ \mathbf{G}=\left\{\dfrac{1-\cos(\sigma\sqrt{-\lambda_k})}{\lambda_k^2}\right\}\mathbf{A}\mathbf{p}_0\end{cases}$$

where **S** and Λ are such that $\mathbf{A}=\mathbf{S}\Lambda\mathbf{S}^{-1}$ and $\mathbf{X}=\mathbf{S}\mathbf{P}\mathbf{S}^{-1}$ with **P** a two-dimensional permutation matrix. Functions **f**, **g**, **F** and **G** introduced in Appendix A.1.(E) are listed below, according to the sign of the matrix eigenvalues shown above. Considering the notation $\mathbf{D}=\mathbf{X}\mathbf{A}\mathbf{X}^{-1}$ and $\mathbf{B}=\sqrt{\mathbf{A}}$, functions {**f**, **g**, **F**, **G**} are depicted in Table A.4.

Appendix B. Scripts and files

We present here a collection of bash script files used for simple test of `art-fun/` routines presented in Section 5.

Script 1 Example of a bash script using routine `art-ray-oneway` and the ray report.

```
#!/bin/bash

for ((j=-50;j<50;j=j+1)); do
    art-ray-oneway -m M00-4L-aff.cfg -s 1.0 -a $j -i 3
                   -n 20 -o model.dat > ray.dat 2> report.txt;

    cat ray.dat >> allrays.dat;

    T=$(grep 'Traveltime' report.txt | awk '{print $2}' | tail -1);
    x=$(grep 'Position' report.txt | awk '{print $2}' | tail -1);

    echo $x $T >> traveltime.dat
done

sed "s/(//g" traveltime.dat > null.dat
sed "s/,//g" null.dat > traveltime.dat
rm ray.dat report.txt;
```

Script 2 Example of a bash script using routine `art-ray-s2i`.

```
#!/bin/bash

echo "10 0 1 1 1 2 2 2 2 1" > path.dat

for ((j=110;j<170;j=j+3)); do

    art-ray-s2i -m M01-4L-const.cfg -x 1 -z 1.5 -a $j
                -p path.dat -n 20 -o model.dat > ray.dat 2>report.txt

    status=$(grep "Error:" report.txt);

    if [ ! "$status" ]; then
      cat ray.dat >> allrays.dat;
      printf "\n" >> allrays.dat;
    fi
done

rm ray.dat report.txt;
```

Script 3 Example of a bash script using routine `art-fan` and the ray report.

```
#!/bin/bash

rm Nrays.dat;

for ((j=0;j<=100;j++)); do

    S=$(echo "1 + 0.07*$j" | bc);

    art-fan -m M05-2L-cgl.cfg -s $S -r 30 -i 1 -n 20
            -o model.dat > rays.dat 2>report.txt;

    R=$(awk '{print $5}' report.txt);

    echo $S $R >> Nrays.dat
done;
```

Script 4 Example of a bash script using routine `art-twop` and the ray report

```
#!/bin/bash

for ((j=0;j<=20;j++)); do

    S=$(echo "0.5 + 0.1*$j" | bc);

    art-twop -m M02-2L-ass.cfg -s $S -r 4 -i 1 -n 20
             -o model.dat >> rays.dat 2> report.txt

    E=$(grep 'empty ray set' report.txt);

    if [ $? -eq 0 ]; then
       N="0"; T="0";
    else
       N=$(grep "Number of rays" report.txt | awk '{print $4}');
       T=$(grep 'Time' report.txt | awk '{print $1}' |
                sed "s/Time=//g" | sed "s/,//g");
    fi

    echo -e "$S\t$N\t" $T
done
```

File 2 A typical report generated from routine `art-ray`, see command line (8), is depicted in File 2. The report displays the kinematic and dynamical quantities at each extreme point R_k that lies at an interface. The "ray-report" for some extreme point `R[k]` is presented. Needless to say that such a report contains the same information for each extreme point along the ray, i.e, `R[0]`, `R[1]`, `R[2]`, ..., `R[N]`. Information of extreme point R_k extracted from the "ray-report" exported by routine `art-ray`.

```
Point R[5]

Position: (3.721358, 0.694336)
Traveltime: 3.138275
Coefficient Reflection: (0.068771, 0.000000)
Coefficient Transmission: (1.068771, 0.000000)
In-plane propagator matrix across interface: [0.985385 0.000000; -0.042030 1.014831]
Relative Inplane Geometrical Spreading: 3.035499
Relative Out-of-plane Geometrical Spreading: 3.035499
Incident quantities at the point:
  Slowness vector: (0.180659, -0.787343)
  Incidence angle: 0.311962
  Velocity: 1.237924
  Amplitude (complex): -0.000192 + 0.000000 i        0.000192
  Vector Q: (-4.425952, 8.389156)
  Vector P: (1.000000, 1.000000)
  Inplane Geometrical Spreading: 2.103795
  Out-of-plane Geometrical Spreading: 2.896404
  Geometrical Spreading: 6.093439
  In-plane propagator matrix: [1.132956 -9.214257; 0.216105 -0.874922]
  Out-of-plane propagator matrix: [1.000000 11.744819; 0.000000 1.000000]
Transmitted quantities at the point:
  Slowness vector: (0.189198, -0.688773)
  Incidence angle: 0.354488
  Velocity: 1.400000
  Amplitude (complex): -0.000204 + 0.000000 i        0.000204
  Vector Q: (-0.371902, 0.714286)
  Vector P: (1.000000, 1.000000)
  Inplane Geometrical Spreading: 2.088365
  Out-of-plane Geometrical Spreading: 2.896404
  Geometrical Spreading: 6.048749
  In-plane propagator matrix: [1.116399 -9.079595; 0.171692 -0.500624]
  Out-of-plane propagator matrix: [1.000000 11.744819; 0.000000 1.000000]
```

File 3 Output from Script 4. See text, Section 6 for details.

```
.5 1  4.245209
.6 1  4.166297
.7 2  3.673226 4.087915
.8 2  3.638657 4.010091
.9 2  3.606489 3.932850
1.0 2  3.576595 3.856214
1.1 2  3.548828 3.780207
1.2 2  3.523020 3.704853
1.3 2  3.498981 3.630174
1.4 2  3.476488 3.556196
1.5 2  3.455286 3.482943
1.6 2  3.435063 3.410444
1.7 2  3.415417 3.338726
1.8 2  3.395781 3.267820
1.9 2  3.375168 3.197760
2.0 1  3.128579
2.1 1  3.060315
2.2 0  0
2.3 0  0
2.4 0  0
2.5 0  0
```

References

[1] E.X. Miqueles, J.J.S. de Figueiredo, T.A. Coimbra, J. Phys.: Conf. Ser. 410 (2013) 012005.
[2] V. Červený, I. Psencik, Ray-theory amplitudes and synthetic seismograms in 2-D inhomogeneous isotropic layered structures. Program packages SEIS seismic waves in complex 3-D structures, Report 12, Dep. Geophys., Charles Univ., Prague, 2002, p. 53.
[3] Norsar, http://www.norsar.no/norsar/home/.
[4] V. Červený, Seismic Ray Theory, Cambridge Univ. Press, 2001.
[5] M.M. Popov, Wave Motion 4 (1982) 85.
[6] N. Bleistein, Mathematical Methods for Wave Phenomena, Academic Press, 1984.
[7] E.X. Miqueles, Seismic modelling in analytical media, Master's, Federal University of Parana, UFPR, 2006 (in portuguese).
[8] R. Sheriff, Exploration Seismology, Vol 1—History, Theory, & Data Acquisition, Cambridge University Press, 1982.
[9] M.M. Popov, Ray Theory and Gaussian Beam Method for Geophysicists, in: Lecture Notes, University of Bahia, Salvador, ISBN: 85-232-0256-0, 2002, p. 172. EDUFBA.
[10] N. Bleistein, J. Cohen, H. Jaramillo, Geophysics 64 (1) (1999) 112.
[11] N. Bleistein, J.K. Cohen, J.W. Stockwell Jr., Mathematical Methods of Seismic Imaging, Migration and Inversion, Springer-Verlag, New York, 2001.
[12] R. Courant, D. Hilbert, Methods of Mathematical Physics, Vol. II, Partial Differential Equations, Interscience, Cambridge, 1962.
[13] L.C. Evans, Partial Differential Equations, second ed., in: Graduate Studies in Mathematics, vol. 19, American Mathematical Society, 2010, p. 749.
[14] V. Červený, Determination of second derivates of travel-time field by dinamic ray tracing, SEP-28, 1981, p. 31.
[15] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., in: John Hopkins Studies in Mathematical Sciences, 1996.
[16] GSL: GNU Scientific Library, http://www.gnu.org/software/gsl/.
[17] LibConfuse, http://www.nongnu.org/confuse/.
[18] Gengetopt, http://www.gnu.org/software/gengetopt/.
[19] V. Červený, I. Pšenčik, in: E.R. Endghal (Ed.), Numerical modelling of seismic wave fields in 2-D laterally varying layered structures by the Ray method, Doc. of Earthquake Algorithms, Report SE-35, World Data Center A for Solid Earth Geophysics, Boulder, p. 36.
[20] V. Červený, M.M. Popov, I. Pšenčik, Geophys. J. R. Astron. Soc. 70 (1982) 109.
[21] P. Hubral, T. Krey, Interval Velocities from Seismic Reflection Time Measurements, in: SEG Monograph Series, vol. 3, 1980.
[22] R.L. Burden, J.D. Faires, Numerical Analysis, Brooks/Cole, 2000.
[23] M. Tygel, J. Schleicher, P. Hubral, Geophysics 61 (2013) 759.
[24] P. Hubral, J. Schleicher, M. Tygel, Geophysics 61 (1996) 742.
[25] E. Iversen, Geophysics 71 (2006) 117.
[26] B.D. Amaro, J. Schleicher, A. Novais, J.C. Costa, Stabilized Least-Squares Imaging Conditions for Common-Shot Wave-Equation Migration, WIT Consortium, Hamburg, Germany, 2011.
[27] T.A. Coimbra, A. Novais, J. Schleicher, Stud. Geophys. Geod. (Praha) 56 (1) (2012) 65.
[28] T.A. Coimbra, J.J.S. de Figueiredo, J. Schleicher, A. Novais, J.C. Costa, Geophysics 78 (3) (2013) S125.
[29] M.W. Hirsch, S. Smale, Differential Equations, Dynamical Systems and Linear Algebra, Academic Press, 1974.