PestoSeis: A Python package for educational seismology

13 July 2021

Summary

PestoSeis is a Python package aiming at educational seismology. It contains functions to solve two-dimensional seismic problems in terms of traveltimes, rays and acoustic and elastic wave propagation. Moreover, a set of functions performing basic seismic processing for exploration geophysics is provided. The entire package is written in the Python language to allow users to explore the code for understanding the numerical algorithms and to minimize dependencies and simplify the installation process. A set of illustrative examples covering all included algorithms is provided.

Statement of Need

PestoSeis is a numerical laboratory to learn about seismology, written in the Python language. One of the design goals is simplicity and easiness of use. It contains a set of functions related to different aspects of seismology, ranging from simple straight ray computations to a full-waveform elastic solver for two-dimensional (2-D) problems. It thus collects several routines under a single package allowing the user to explore different topics in seismology with a hands-on approach. Moreover, a complete set of examples demonstrating the functionality of the code is provided, with the double aim of illustrating some seismological problems and how to utilize the code.

The entire code is written in Python to avoid the need of linking to a low-level language and to allow the user to directly explore the code in order to understand how the algorithms work. Moreover, the depency on other packages is kept to the minimum to simplify the installation process and, again, understanding of the source code from the user perspective. PestoSeis can thus be used, for instance, in a course on seismology, where a set of exercises can be designed to help understanding certain topics in a practical way, allowing the students to calculate results that could not be obtained by hand calculations.

`PestoSeis` addresses the following set of 2-D seismic problems: - computation of traveltimes for given source and receiver positions, provided a layered or grid-based velocity model, assuming straight rays; - computation of traveltimes for given source and receiver positions, provided a grid-based veloity model, using a finite-difference method (fast marching); - backtracing rays from the result of traveltime calculations in heterogeneous models; - performing simple linearized ray tomography using a least squares approach; - computing acoustic or elastic seismic wave propagation using a finite-difference method; - performing some basic seismic processing for exploration seismology - plotting various input and output data related to the problems above.

This package allows anybody interested in playing around or gaining a more profound understanding of certain aspects of seismology to do that in a simple yet rigorous way.

Package Content

Logically, the `PestoSeis` is split into tree categories:

1. Traveltimes and rays using `ttimerays`;
2. Seismic wave propagation using `seismicwaves2D`;
3. Seismic processing using `to be added`.

Each one of these categories considers a specific use case and provides a variety of functions tailored to solve problems within the considered application.

Traveltimes and rays

`PestoSeis` provides code to compute rays in heterogeneous media as:

1. Bent rays using `pestoseis.ttimerays.traceallrays()`;
2. Straight rays using `pestoseis.ttimerays.traceall_straight_rays()`;
3. Additionally, in the special case of a horizontally layered medium, the function `pestoseis.ttimerays.tracerayhorlay()` applies Snell's law to compute ray paths, traveltimes and the distance covered by the ray.

The functions provided in `pestoseis.ttimerays` allow the user to compute rays and traveltimes for a 2D velocity model, which is discretized on a rectilinear grid constructed from the user input using the function `pestoseis.ttimerays.setupgrid()`. Figure 1 shows an example with computed traveltimes, bent ray paths and straight ray paths through a speed-of-sound model mimicking human breast tissue. The computed traveltimes can then be used to set up a tomographic problem, which may for instance be solved with a simple linear inversion under Gaussian assumption (least squares approach) using `pestoseis.ttimerays.lininv()`. The available functions allow the user to set up and solve a simple 2D tomographic problem, thus they aim at providing a conceptual guide of how travel time tomography is performed in practice. Since the main focus lies on the forward computation of travel times, the set of functions `pestoseis.ttimerays.lininv()` performing the linear inversion is kept on a very basic level on purpose.

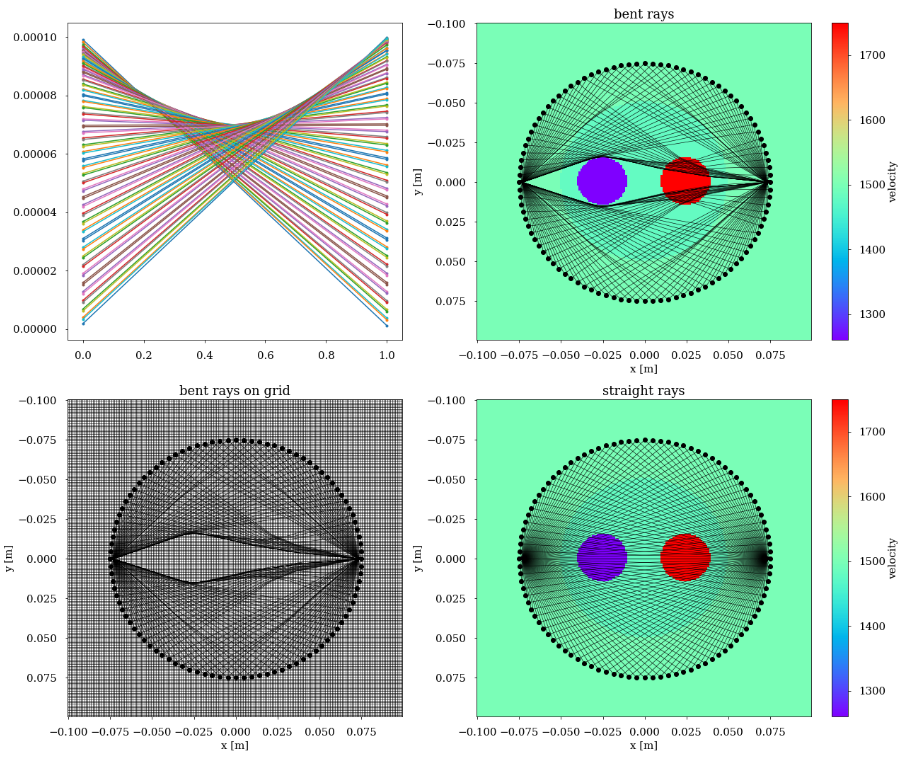

Figure 1: Visualization of computed travel times and rays (bent rays and straight rays) using the functions in`pestoseis.ttimerays`. In this example, we consider a numerical breast phantom as used in medical ultrasound to study the ray paths through the medium and subsequently compute the according travel times from an array of sources and receivers surrounding the breast phantom.

Seismic wave propagation

`PestoSeis` provides code to compute wave propagation in 2D for:

1. Acoustic mediums solving the acoustic wave equation in 2D;
2. Elastic mediums solving the elastic wave equation in 2D.

The functions `pestoseis.seismicwaves2d.solveacoustic2D()` and `pestoseis.seismicwaves2d.solveelastic2D()` solve the acoustic and elastic wave equation in 2D using finite differences on a staggered grid in both space and time. Staggered grids generally allow for a better accuracy with only a small computational overhead. In order to compute wave propagation in either an acoustic or an elastic medium, the user needs to specify the parameters of the grid used to construct the velocity models, the source-receiver geometry and a source time function with dominant frequency. Furthermore, boundary conditions need to be set (`PestoSeis` provides implementation of reflecting boundary conditions, a Gauss taper or perfectly matched layers) with the option to specify a free surface. The function returns a seismogram recorded at the receivers as well as a set of snapshots of the wavefield. Additionally, the functions `pestoseis.seismicwaves2d.animateacousticwaves()` and `pestoseis.seismicwaves2d.animateelasticwaves()` collects the snapshots and saves them to a `.mp4` movie that illustrates the propagation of the seismic waves through the medium at all simulation time steps. The functions provided aim to equip the user with the possibility to quickly set up and visualize small scale 2D simulations.

Seismic processing

`PestoSeis` provides code to set up and perform simple, small-scale seismic reflection experiments and to process the resulting data with common practice methods such as for instance arranging the data in shot gathers, generating a wiggle plot of the shot gathers, normal moveout correction, correcting for geometrical spreading or applying Automatic Gain Control to a shot gather. Furthermore, some functionalities to process the data in the wavenumber-frequency domain are provided as well. All functions concerned with generating and processing seismic reflection data are part of `pestoseis.reflectionseismo`. Again, the main focus of the provided functions is to equip the user with some fast and simple tools to get familiar and experiment with common practice processing methods.

Tutorials

There exist a number of tutorials that provide examples on how to use the functions within `PestoSeis`. These tutorials showcase different numerical scenarios and can be used to get started with `PestoSeis`.

==

Example from JOSS below

==

Statement of need

`Gala` is an Astropy-affiliated Python package for galactic dynamics. Python enables wrapping low-level languages (e.g., C) for speed without losing flexibility or ease-of-use in the user-interface. The API for `Gala` was designed to provide a class-based and user-friendly interface to fast (C or Cython-optimized) implementations of common operations such as gravitational potential and force evaluation, orbit integration, dynamical transformations, and chaos indicators for nonlinear dynamics. `Gala` also relies heavily on and interfaces well with the implementations of physical units and astronomical coordinate systems in the `Astropy` package [@astropy] (`astropy.units` and `astropy.coordinates`).

`Gala` was designed to be used by both astronomical researchers and by students in courses on gravitational dynamics or astronomy. It has already been used in a number of scientific publications [@Pearson:2017] and has also been used in graduate courses on Galactic dynamics to, e.g., provide interactive visualizations of textbook material [@Binney:2008]. The combination of speed, design, and support for Astropy functionality in `Gala` will enable exciting scientific explorations of forthcoming data releases from the *Gaia* mission [@gaia] by students and experts alike.

Mathematics

Single dollars ($) are required for inline mathematics e.g. $f(x) = e^{\pi/x}$

Double dollars make self-standing equations:

$$\Theta(x) = \left\{\begin{array}{l} 0\textrm{ if } x < 0\cr 1\textrm{ else} \end{array}\right.$$

You can also use plain LaTeXfor equations

$$\hat f(\omega) = \int_{-\infty}^{\infty} f(x) e^{i\omega x} dx \tag{1}$$

and refer to Equation 1 from text.

Citations

Citations to entries in paper.bib should be in rMarkdown format.

If you want to cite a software repository URL (e.g. something on GitHub without a preferred citation) then you can do it with the example BibTeX entry below for @fidgit.

For a quick reference, the following citation commands can be used: - `@author:2001` -> "Author et al. (2001)" - `[@author:2001]` -> "(Author et al., 2001)" - `[@author1:2001; @author2:2001]` -> "(Author1 et al., 2001; Author2 et al., 2002)"

Figures

Acknowledgements

We acknowledge contributions from Brigitta Sipocz, Syrtis Major, and Semyeong Oh, and support from Kathryn Johnston during the genesis of this project.

References