ChatGPT

Travel-Time Seismic Tomography Project Proposal

Overview. Seismic travel-time tomography uses observed wave arrival times to reconstruct subsurface velocity structure [1] [2]. In practice one starts with an assumed velocity model and iteratively adjusts it so that computed travel times match the observed data [2]. The output is a seismic velocity model (1D, 2D or 3D) whose anomalies can be interpreted as changes in rock type, temperature, or fluid content [3] [4]. For example, a recent Alpine study used ~162,000 P-wave travel-time picks (331 earthquakes at ~600 stations) to invert for a 3D velocity model with ~25 km lateral resolution [4] [5]. In our project we will design a simpler, tractable version of such an inversion (start with 1D then extend to 2D) and use it to infer layer velocities and major anomalies from travel-time data.

1D Tomography (Layered-Earth Assumption). In 1D tomography we assume the Earth is composed of horizontal layers with constant (but unknown) seismic velocity in each layer. Typical data are first-arrival travel times from a downhole source (vertical seismic profile) or from surface shots recorded along a line of geophones. The travel time T to a receiver at distance x depends on the layer thicknesses and velocities via straight-ray or head-wave formulas. Inversion linearizes these formulas: for example, in refraction profiles the intercept time and apparent slope give layer velocities [6]. In practice one might fix the number of layers and use least-squares or grid-search to solve for velocities (and sometimes layer depths) that best fit the observed times. Smoothness or monotonicity constraints (e.g. increasing velocity with depth) are often applied to stabilize the inversion. Well-known 1D methods like the DeltatV algorithm assume smoothly increasing velocity with depth and invert stacked common-midpoint travel times without explicitly picking refraction segments [6].

- *Tools:* ObsPy's TauPyModel can compute theoretical travel times through standard 1D models (e.g. IASP91) [7]. Pyrocko's **Cake** tool solves ray theory in layered ("cake") models, producing arrival times and ray paths for any specified set of layers [8]. These can generate synthetic data or predict times for a given velocity model. For inversion, one can use NumPy/SciPy solvers or pyGIMLi's `TravelTimeManager`, which can handle layered refraction inversion [9] [10]. For example, pyGIMLi's refraction tutorial shows how to define a sloping three-layer model, compute its travel times, add noise, and then invert back to recover the layer velocities [9].

- *Assumptions:* The 1D approach neglects any lateral variation: it assumes horizontal layering, straight-ray paths (or critical refractions) and simple geometry. In noisy data or dipping layers, the inversion may be non-unique, so it often requires a good initial guess or regularization. Nonetheless, 1D inversion can quickly recover a velocity-vs-depth profile. The result may reveal, for instance, a low-velocity sedimentary layer or a high-velocity basement layer in the subsurface.

2D Tomography (Grid-Based Inversion)

Assumptions and Principles. 2D travel-time tomography relaxes the layered assumption and allows velocity to vary laterally. We set up a 2D grid (mesh) covering the study area, with slowness (inverse velocity) as the parameter in each cell. Data come from many source–receiver ray paths: for example, local earthquake (or explosion) events recorded at a seismic array, or crosshole surveys between boreholes. Under the straight-ray approximation, each travel path is a straight line through the grid, so

the travel time is the line integral of slowness along that path. In more advanced schemes, rays bend in high-contrast models (requiring eikonal or shooting methods), but we can start with simple straight-ray or fast-marching (Eikonal) predictions.

To invert, we linearize: travel-time residuals ΔT are related to slowness perturbations Δs by a sensitivity matrix **G** (path lengths in cells) such that ΔT = G Δs. We solve this (typically damped) least-squares problem, often iteratively: update the model, recompute G (if rays bend), and repeat until fit converges. Regularization (smoothing, damping, or a priori models) is applied to mitigate non-uniqueness [5].

- *Tools:* Libraries like **pyGIMLi** and **SimPEG** have tomography modules. For example, SimPEG's 2D straight-ray example builds a TensorMesh, defines a background velocity with a low-velocity block, and then uses an integral-based solver to predict travel times [11]. PyGIMLi's `TravelTimeManager` and `TravelTimeDijkstraModelling` can perform ray tracing on a 2D mesh. Pyrocko also offers an eikonal solver (`pyrocko.modelling.eikonal`) for first-arrival times in heterogeneous media (see Fig.* below). Open-source tools like **Rayfract** or simple finite-difference eikonal solvers (e.g. from Pyrocko) can also compute travel times on 2D velocity models.

- *Data:* In a 2D survey, data might be picks of P-wave arrivals at many stations from multiple events. Public catalogs like the USGS ComCat include phase picks and hypocenters for millions of quakes [12], which can serve as travel-time data. Experimental arrays (e.g. the AlpArray network) often publish travel-time datasets used in tomography [4]. If real data are limited, one can generate synthetic data: define a velocity model, simulate travel times (e.g. with TauP or eikonal), add noise, and then invert. For teaching purposes, synthetic models (two-layer, block in homogeneous background, etc.) and synthetic travel times are commonly used.

- *Expected Outcome:* The inversion yields a 2D velocity/slowness map. High-velocity anomalies may correspond to dense (cooler) rock or unfractured material, while low-velocity zones may indicate hot materials (magma) or fluids. In practice, recovered features are interpreted in geological terms. The resulting model is compared against the misfit and resolution analyses. Iterative refinement and alternative parameterizations (e.g. adaptive mesh, joint P/S inversion) are possible extensions.

Figure: Synthetic P-wave travel-time contours in a 5-layer model (velocities increasing with depth) computed by Pyrocko's eikonal solver [13].

Data Sources and Software Tools

Public Datasets: Many seismic phase datasets are publicly available. For example, the USGS ComCat provides extensive earthquake catalogs including P- and S-wave picks [12]. Global and regional networks (e.g. IRIS GSN, GEONET, European Seismic Network) routinely release event catalogs and phase data. Some research projects (e.g. Alpine, Himalayan tomography) have shared processed travel-time residuals. If field data are not accessible, synthetic data can be generated from known models (e.g. using TauP or Pyrocko) for methodology testing.

Software Packages: We will use Python-based tools, for example: - **ObsPy (TauP)** – for computing theoretical travel times in 1D Earth models or matching phases [7]. This helps validate our forward model or pick data quality.
- **Pyrocko (Cake, Eikonal)** – "Cake" computes arrivals and rays in 1D layered models [8]; its eikonal module can compute first-arrival travel-time fields in 2D media (Fig. above) [13].
- **pyGIMLi** – offers mesh-based tomography and inversion routines. Its `TravelTimeManager` can set up 2D tomography problems, and examples exist for refraction and crosshole surveys [9] [10].

- **SimPEG** – provides tutorials (straight-ray tomography) and inversion tools that we can adapt [11].
- **Generic tools** – NumPy/SciPy for linear algebra (least-squares solvers), Matplotlib for plotting, and possibly Jupyter for interactive development.

Each of these tools has documentation and examples. We will review tutorials (e.g. pyGIMLi's refraction inversion example [9] or SimPEG's tomography tutorial) to guide our implementation. The workflow will be implemented step-by-step in code, testing on synthetic data before applying to any real dataset.

Proposed Project Workflow

1. **Select Dataset:** Choose or generate travel-time data. For a 1D test, we might simulate first-arrival times from a two-layer model (with added noise). For 2D, options include a subset of real event–station data (e.g. a small local earthquake catalog) or a synthetic grid of shots/receivers.
2. **Initial Model:** Define a starting velocity model. In 1D this might be a single-layer or smoothly varying profile; in 2D a homogeneous or gently graded model. PyGIMLi's `createGradientModel2D` or simple maps can set this up.
3. **Forward Modeling:** Compute predicted travel times in the initial model. For 1D, use analytic formulas or a ray solver (ObsPy/Pyrocko). For 2D, use straight-ray integration or eikonal solver to get travel times for all source–receiver pairs. Compare predicted times to observed picks to form residuals.
4. **Linear Inversion Setup:** Build the tomography matrix **G**. For straight-ray 2D, **G** contains path lengths of each ray through each cell. For 1D, set up equations that relate travel time differences to layer slownesses. Assemble the data vector (observed – predicted times) and **G**.
5. **Solve Inverse:** Use least-squares (e.g. SciPy's `lstsq` or pyGIMLi/SimPEG solvers) with damping/smoothing regularization. Solve for slowness corrections. Update the velocity model.
6. **Iteration (if needed):** If rays bend or if solution not converged, re-compute forward travel times with the new model and iterate the inversion until residuals are minimized.
7. **Analysis:** Evaluate the fit (e.g. root-mean-square misfit) and resolution (checkerboard or diagonal resolution). Check that the solution is stable and geologically plausible.
8. **Interpretation:** Interpret velocity anomalies. In the 1D case, the results are layer velocities and thicknesses. In 2D, we get a velocity map – e.g. a low-velocity anomaly might indicate a fluid reservoir or magma chamber, high-velocity could indicate solid bedrock. Cross-validate with known geology or any borehole data if available.

This workflow can be documented in code (Jupyter notebook) and the methodology will be explained step-by-step. For example, PyGIMLi's "Koenigsee" refraction example shows a similar inversion process on field data [10].

Target Applications

This project is fundamentally a methodological exercise, but we should tie it to an application. For 1D tomography, one could target a vertical seismic profile in a borehole: the inversion would yield a velocity-depth curve useful for local geology or reservoir characterization. For 2D tomography, a classic application is imaging beneath a volcano or fault zone: for instance, identifying a low-velocity magma chamber or a fractured zone. If real data are available (e.g. a small network of seismic stations around a fault), we could apply the 2D inversion to reveal such structures. At minimum, we will validate the method on synthetic models with known anomalies (e.g. a buried low-velocity block). The final report will discuss how the inferred velocity features relate to subsurface properties (rock type, fluids, temperature) [3].

Subsurface Properties Inferred. The primary outcome is the seismic velocity distribution. Velocity contrasts often correlate with rock composition or state: e.g. sedimentary layers (low velocity) vs crystalline basement (high velocity), or hot fluid-saturated zones (slow). Seismic tomography results can therefore suggest layer boundaries, presence of fluids, or thermal anomalies. As noted, features in the velocity model "may be interpreted as structural, thermal, or compositional variations" [3]. We will discuss these interpretations relative to any known geology of the study area.

References: This proposal draws on standard tomography theory [1] [2], existing software examples [9] [13] [11], and published case studies [4] [5]. All cited sources are open-access or documentation (ObsPy/Pyrocko/pyGIMLi) that explain the methods and tools we plan to use.

---

[1] [2] [3] Seismic tomography - Wikipedia
https://en.wikipedia.org/wiki/Seismic_tomography

[4] [5] SE - Imaging structure and geometry of slabs in the greater Alpine area – a P-wave travel-time tomography using AlpArray Seismic Network data
https://se.copernicus.org/articles/12/2671/2021/

[6] Delta-t-V 1D seismic refraction inversion method : Theory
https://rayfract.com/pub/deltatv.pdf

[7] Seismic data process via ObsPy — CUSeisTut
https://cuseistut.readthedocs.io/en/latest/obspy-2/index.html

[8] Cake manual — Pyrocko v2025.01.21 Manual
https://pyrocko.org/docs/current/apps/cake/manual.html

[9] 2D Refraction modelling and inversion — pyGIMLi - Geophysical Inversion and Modelling Library
https://www.pygimli.org/_examples_auto/2_seismics/plot_01_refraction_manager.html

[10] pygimli.physics.traveltime — pyGIMLi - Geophysical Inversion and Modelling Library
https://www.pygimli.org/pygimliapi/_generated/pygimli.physics.traveltime.html

[11] Forward Simulation for Straight Ray Tomography in 2D — SimPEG 0.22.2 documentation
https://docs.simpeg.xyz/v0.22.2/content/tutorials/12-seismic/plot_fwd_1_tomography_2D.html

[12] ANSS Comprehensive Earthquake Catalog (ComCat) Documentation
https://earthquake.usgs.gov/data/comcat/

[13] Traveltime calculation and raytracing — Pyrocko v2025.01.21 Manual
https://pyrocko.org/docs/current/library/examples/cake_raytracing.html