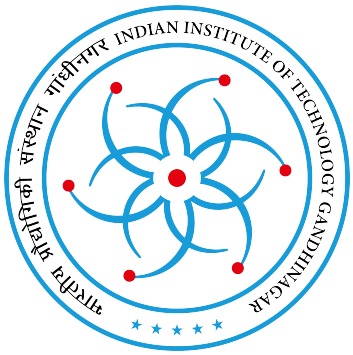
Project Report

October 31, 2025



EH 611

Near Surface Geophysics
Prof. Utsav Mannu

INDIAN INSTITUTE OF TECHNOLOGY GANDHINAGAR
Palaj, Gandhinagar - 382355

Travel-Time Tomography: Inversion of First-Arrival Data to Retrieve Velocity-Depth Models

---

Submitted by

Guntas Singh Saran (22110089)
Computer Science and Engineering
Jinil Patel (22110184)
Computer Science and Engineering
Pranav Patil (22110199)
Computer Science and Engineering
Deepak Soni (22110068)
Mechanical Engineering

Abstract

Travel-time tomography is a fundamental tool for estimating subsurface velocity structure from first-arrival (refraction) seismic data. This report presents the theoretical framework and numerical implementation of travel-time tomography using PyGIMLi [5]. We describe the continuous and discretized forward problems solved via Dijkstra ray tracing, the inverse problem formulation based on damped least-squares optimization with Tikhonov regularization, and iterative Gauss-Newton inversion. We demonstrate the methodology on both synthetic layered models with controlled topography and the `Koenigsee` field dataset (714 picks from 63 sensors). The code is available at `https://github.com/guntas-13/travel-time-tomography/`

Introduction

Problem statement

Given travel-time data consisting of first-arrival picks d_j recorded for a set of source-receiver pairs, the goal of travel-time tomography is to estimate a subsurface velocity model $v(\mathbf{x})$ (or equivalently slowness $s(\mathbf{x}) = 1/v(\mathbf{x})$) that explains the observed arrival times.

Travel-time tomography is applied across scales: from engineering near-surface surveys (bedrock mapping, groundwater exploration) to deep crustal imaging and exploration seismology. While alternative methods like MASW [4], ambient-noise tomography [1] offer complementary information, travel-time tomography remains the preferred choice when high spatial resolution, localized imaging, and rapid turnaround are priorities. Further, standard reference Earth models like IASP91 [2], AK135 [3] though provide average P- and S-wave velocities as functions of depth but they are constructed from decades of *global* seismic data..

Theory and Methods

Continuous forward problem

In the ray-theory (high frequency) limit, the travel time T between source $\mathbf{x}_s$ and receiver $\mathbf{x}_r$ is

$$T(\mathbf{x}_s, \mathbf{x}_r; s) = \int_{\Gamma(\mathbf{x}_s, \mathbf{x}_r)} s(\mathbf{x})\, \mathrm{d}l, \tag{1}$$

where $s(\mathbf{x}) = 1/v(\mathbf{x})$ is slowness and $\Gamma(\mathbf{x}_s, \mathbf{x}_r)$ denotes the ray path connecting source and receiver determined by the Fermat principle (shortest travel time path).

Discretization and forward operator

Partition the computational domain into N_m discrete cells (e.g., triangular mesh) and assume slowness is piecewise constant within each cell: $s(\mathbf{x}) \approx s_i$ for $\mathbf{x}$ in cell i. For data index j (one source–receiver pair) the forward problem becomes

$$T_j(\mathbf{s}) = \sum_{i=1}^{N_m} L_{ji}\, s_i, \tag{2}$$

where L_{ji} is the length of the ray for data j inside cell i. The forward mapping $\mathcal{F} : \mathbb{R}^{N_m} \to \mathbb{R}^{N_d}$ maps the model vector $\mathbf{s}$ to predicted travel times $\mathbf{t} = \mathcal{F}(\mathbf{s})$.
In implementations like pyGIMLi's [5] Dijkstra-based modelling, the continuous domain is further represented as a graph: mesh nodes form vertices and edges connect neighbouring nodes. Dijkstra's algorithm yields the shortest-path distances across that graph; for travel-time tomography, this approximates the ray path and yields the discrete forward operator and path-length sensitivity.

Jacobian / Sensitivity

The Jacobian J (sensitivity matrix) has entries

$$J_{ji} = \frac{\partial T_j}{\partial s_i} = L_{ji}, \tag{3}$$

i.e., each column corresponds to the sensitivity of all travel times to the slowness of a single cell, and each row is the length vector of a ray through the cells.

Inverse problem and objective function

The inverse problem is posed as a regularized nonlinear least-squares problem. Using data vector $\mathbf{d}$ and predicted data $\mathcal{F}(\mathbf{m})$ (where $\mathbf{m}$ is the parameter vector - often slowness or the logarithm of slowness), the objective is

$$\Phi(\mathbf{m}) = \|W_d(\mathcal{F}(\mathbf{m}) - \mathbf{d})\|_2^2 + \lambda \|W_m(\mathbf{m} - \mathbf{m}_0)\|_2^2, \tag{4}$$

where W_d is the data weighting (often inverse measurement standard deviations), W_m is the regularization operator (e.g., discrete derivative / roughness), λ is the regularization parameter, and $\mathbf{m}_0$ a reference model.

Gauss–Newton linearization

Linearize the forward operator around the current iterate $\mathbf{m}^k$:

$$\mathcal{F}(\mathbf{m}^k + \Delta\mathbf{m}) \approx \mathcal{F}(\mathbf{m}^k) + J^k \Delta\mathbf{m}, \tag{5}$$

and minimize the quadratic approximation of the objective. Setting the gradient to zero yields the Gauss–Newton normal equations:

$$((J^k)^T W_d^T W_d J^k + \lambda W_m^T W_m)\, \Delta\mathbf{m} = (J^k)^T W_d^T W_d(\mathbf{d} - \mathcal{F}(\mathbf{m}^k)) - \lambda W_m^T W_m(\mathbf{m}^k - \mathbf{m}_0). \tag{6}$$

After solving for $\Delta\mathbf{m}$, update $\mathbf{m}^{k+1} = \mathbf{m}^k + \Delta\mathbf{m}$ (or with a step-length/line search). Iterations continue until convergence criteria are met (chi-square target, small model updates, or max iterations).

Model transforms and positivity constraints

It is common to invert in a transformed domain to enforce positivity and better numerical scaling. Typical transforms are logarithmic: if $\mathbf{p} = \log(\mathbf{s})$ then inversion is carried out in $\mathbf{p}$ and mapping back uses $\mathbf{s} = \exp(\mathbf{p})$. Transforms modify Jacobian via chain rule: if $s_i = \exp(p_i)$ then

$$\frac{\partial T_j}{\partial p_i} = \frac{\partial T_j}{\partial s_i}\frac{\partial s_i}{\partial p_i} = L_{ji}\exp(p_i).$$

pyGIMLi [5] provides transformation classes to handle these consistently.

Modelling a Synthetic 2D Profile

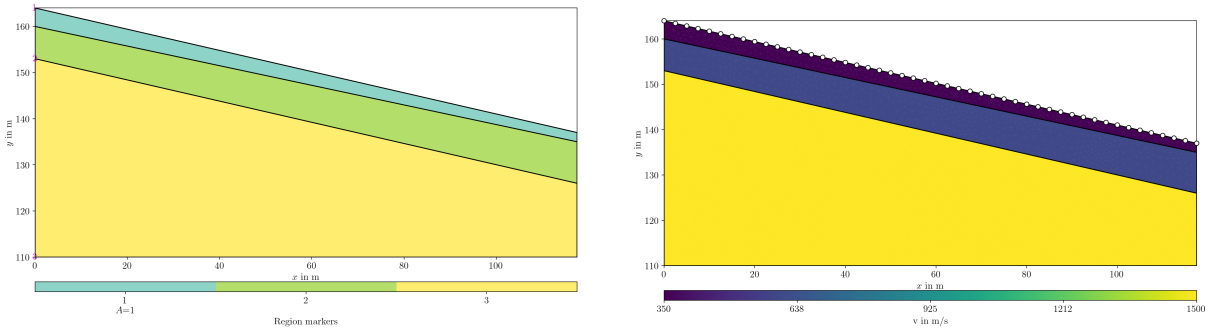


Figure 1: (a.) Sloping Layered geometry (b.) Geophone and shot positions on the sloped surface. White circles indicate receiver positions, and source positions are at equivalent coordinates

Objective

Recover a layered 2D velocity model from first-arrival (refraction) picks along a surface profile.

True model and parameterization

- Three-layer sloping model defined by polygons coordinates (m):

```
layer1 = [[0.0,164], [117.5,137], [117.5,135], [0.0,160]]  (marker=1, v=350 m/s)
layer2 = [[0.0,153], [0.0,160], [117.5,135], [117.5,126]]  (marker=2, v=600 m/s)
layer3 = [[0.0,110], [0.0,153], [117.5,126], [117.5,110]]  (marker=3, v=1500 m/s)
```

- **Layer 1 (350 m/s)**: Unconsolidated sediments, weathered zone, or saturated clay
- **Layer 2 (600 m/s)**: Partially consolidated sediments or fractured rock
- **Layer 3 (1500 m/s)**: Consolidated bedrock, dense limestone, or crystalline basement

Geophone Deployment & Shot Configuration

A linear array of $N_{\text{geo}} = 48$ geophone stations is deployed along the topographic surface. The horizontal positions are uniformly distributed:

$$x_i = \frac{i}{N_{\text{geo}} - 1} \times 117.5\,\text{m}, \quad i = 0, 1, \ldots, 47 \tag{7}$$

The vertical positions are adapted to follow the sloped surface:

$$y_i = m_s \cdot x_i + y_{\text{intercept}} = 0.23 \cdot x_i + 137 \tag{8}$$

Shots (seismic sources) are deployed with a fixed shot distance interval:

$$\Delta x_{\text{shot}} = 3\,\text{m} \tag{9}$$

This creates a reciprocal refraction dataset where shots and receivers occupy the same surface locations. The resulting measurement scheme consists of:

- Total number of shots: $N_{\text{shots}} \approx 40$
- Total number of receivers per shot: $N_{\text{geo}} = 48$
- Total number of ray paths: ≈ 1920

Synthetic data generation (forward modelling)

- Forward simulation call: `tt.simulate(slowness=1.0/vp, scheme=scheme, mesh=mesh, noiseLevel=0.001, noiseAbs=0.001, seed=1337)`.
- Noise: relative noise = 0.1% (`noiseLevel=0.001`), absolute noise = 0.001 s, seed fixed for reproducibility.

Dijkstra Ray-Tracing Operator

PyGIMLi creates a **TravelTimeDijkstraModelling** forward operator takes a velocity model $\mathbf{m}$, converts velocity to slowness: $s = 1/v$, traces rays from each shot to each receiver using Dijkstra's algorithm, returns predicted travel times $\mathbf{d}_{\text{pred}} = \mathbf{f}(\mathbf{m})$

How Dijkstra Ray Tracing Works

Dijkstra's algorithm solves the **eikonal equation** numerically:

$$|\nabla T(\mathbf{r})|^2 = \frac{1}{v^2(\mathbf{r})} \tag{10}$$

Algorithm steps:

1. Initialize travel-time field T to infinity everywhere except at the source: $T(\text{source}) = 0$
2. Build a directed graph where:
   - Nodes = mesh vertices
   - Edges connect adjacent nodes
   - Edge weight = distance $/v_{\text{avg}}$ (slowness-weighted distance)
3. Use Dijkstra's shortest-path algorithm to compute minimum travel times to all receivers

Computational complexity: $O(n \log n)$ where n is the number of mesh nodes

First-break Travel Times

Each colored curve corresponds to a different shot location along the survey line, and the horizontal axis shows the position of geophones (receivers) in meters. The vertical axis records the measured arrival (first-break) travel times in seconds for each shot-receiver pair.

- **X-Axis** (`x (m)`): Geophone position along the survey line on the surface.
- **Y-Axis** (`Traveltime (s)`): Recorded time for the first seismic arrival at each receiver, with time increasing downward.
- **Multiple Curves/Colors**: Each colored line traces first-arrival times for a different shot position; the colors distinguish between different shots (sources).
- **Crossover Point**: The inflection or bend in each curve marks the transition from direct arrivals to refracted arrivals—a key diagnostic for determining subsurface velocities and layer depths.

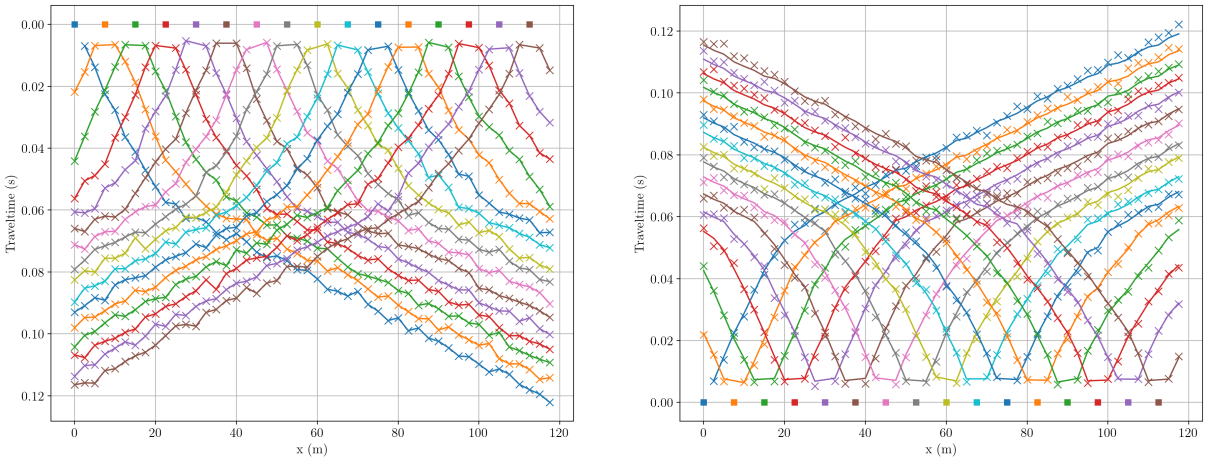


Figure 2: (a.) Synthetic travel-time data simulated through the forward velocity model. The figure shows the first-break arrival times (in milliseconds) organized by shot-receiver pairs using a refraction tomography acquisition geometry. The squares represent the shot positions and the crosses represent the corresponding x location of geophone. (b.) The corresponding predicted arrivals calculated using the inverted velocity model through forward ray tracing indicating residual misfit.

Inversion Problem

The inverse problem seeks to recover the velocity model $\mathbf{m}$ from observed travel times $\mathbf{d}_{\text{obs}}$ by minimizing the misfit functional:

$$\chi^2(\mathbf{m}) = \sum_{i=1}^{N_{\text{data}}} \left(\frac{T_i(\mathbf{m}) - d_i^{\text{obs}}}{\sigma_i} \right)^2 + \lambda R(\mathbf{m}) \tag{11}$$

where:

- $T_i(\mathbf{m})$ = predicted travel time for ray i in model $\mathbf{m}$
- d_i^{obs} = observed travel time for ray i
- σ_i = measurement uncertainty for ray i
- $R(\mathbf{m})$ = regularization functional (smoothness constraint)
- λ = regularization parameter (trade-off between data fit and model smoothness)

The velocity model is parameterized on a finite element mesh with velocity values at cell centers or vertices:

$$\mathbf{m} = [v_1, v_2, \ldots, v_{N_{\text{cells}}}]^T \tag{12}$$

For inversion purposes, velocities are typically transformed to **logarithmic slowness**:

$$\mathbf{m}_{\text{slow}} = [\log(s_1), \log(s_2), \ldots, \log(s_{N_{\text{cells}}})]^T \tag{13}$$

Regularization Strategy

PyGIMLi automatically sets up **Tikhonov regularization**. The regularization term enforces smoothness and helps stabilize the inverse solution:

$$R(\mathbf{m}) = \int_\Omega \left(w_z^2 |\partial_z \mathbf{m}|^2 + |\partial_x \mathbf{m}|^2 \right) d\Omega + \alpha |\mathbf{m} - \mathbf{m}_{\text{ref}}|^2 \tag{14}$$

where $\mathbf{m}_{\text{ref}}$ is a reference model and α controls the coupling to the reference model.

Iterative Inversion Steps

The inversion is typically performed using an iterative Gauss-Newton or damped least-squares approach:

$$\mathbf{m}^{(k+1)} = \mathbf{m}^{(k)} + \left(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{R}\right)^{-1}\mathbf{J}^T\left(\mathbf{d}_{\text{obs}} - \mathbf{T}(\mathbf{m}^{(k)})\right) \tag{15}$$

where $\mathbf{J}$ is the Jacobian matrix of partial derivatives of travel times with respect to model parameters.

Convergence Criteria

Inversion iterations continue until convergence based on criteria such as:

- Relative data misfit reduction: $\frac{\chi^2(k)-\chi^2(k+1)}{\chi^2(k)} < \varepsilon_{\text{rel}}$
- Absolute misfit threshold: $\chi^2(k) < \chi^2_{\text{target}}$
- Maximum iteration count: $k < k_{\max}$

Inversion settings

- Manager: `mgr = tt.TravelTimeManager(data)`.
- Inversion call used: `mgr.invert(secNodes=2, paraMaxCellSize=15.0, maxIter=10, verbose=True)`.
- Transformation: data transform = Identity; model transform = Logarithmic transform.
- Regularization: example printed `lam: 20.0` and used chi-square stopping criterion (tutorial stops when $\chi^2 \leq 1$).

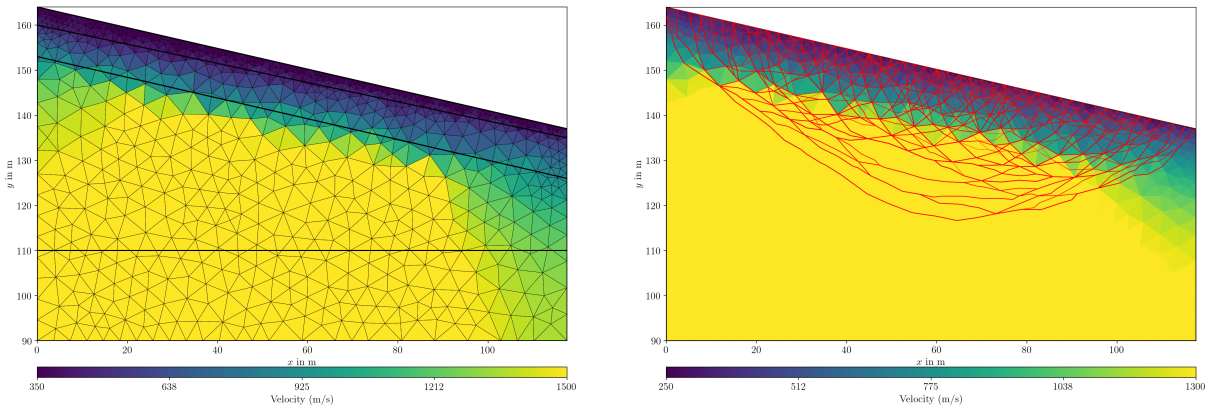


Figure 3: (a.) Final inverted velocity model after convergence. Color indicates velocity magnitude (b.) Ray paths computed through inversion. Different ray colors represent different shot-receiver pairs or ray-type categories. Refracted rays dominate the pattern, bending downward into faster velocity layers and returning to the surface. The ray coverage reveals illumination of all three layers across the model.

Koenigsee (field-data inversion)

Objective

Invert a field dataset (`Koenigsee`) to demonstrate ingestion of picks, coverage checks, and inversion with topography/irregular spacing.

Dataset Description and Characteristics

Located near Königsee Lake in the Alpine region, this field dataset captures typical characteristics of near-surface geophysical surveys with moderately heterogeneous geological structures. The Koenigsee data is stored in the **SGT (Shot-Geophone-Time) format** containing the following essential data columns:

$$\text{SGT Data Record: } \{s, g, t, \text{valid}\} \tag{16}$$

where:

- **s**: Shot (source) station number
- **g**: Geophone (receiver) station number
- **t**: Travel time (in seconds)
- **valid**: Validity flag (0 = invalid/excluded, 1 = valid/included)

Acquisition Configuration

Source (shots) and receivers (geophones) are deployed **along a line on the surface**. Both shots and receivers occupy the nearly same horizontal level with slight undulations along the y-axis.

Table 1: Koenigsee dataset acquisition parameters

Parameter	Value
Total number of sensors (shots + geophones)	63
Total data points (shot-receiver pairs)	714
Measurement columns	s, g, t, valid
Travel-time range	0.35-30 ms
Travel-time range (slowness)	3.5×10^{-4}-0.03 s

Geological Setting

The Koenigsee survey area exhibits:

- **Bedrock**: High-velocity consolidated basement rock with distinct velocity contrast
- **Target depth**: Approximately 50-150 m below surface.

Travel-Time Curves (First-Pick Display)

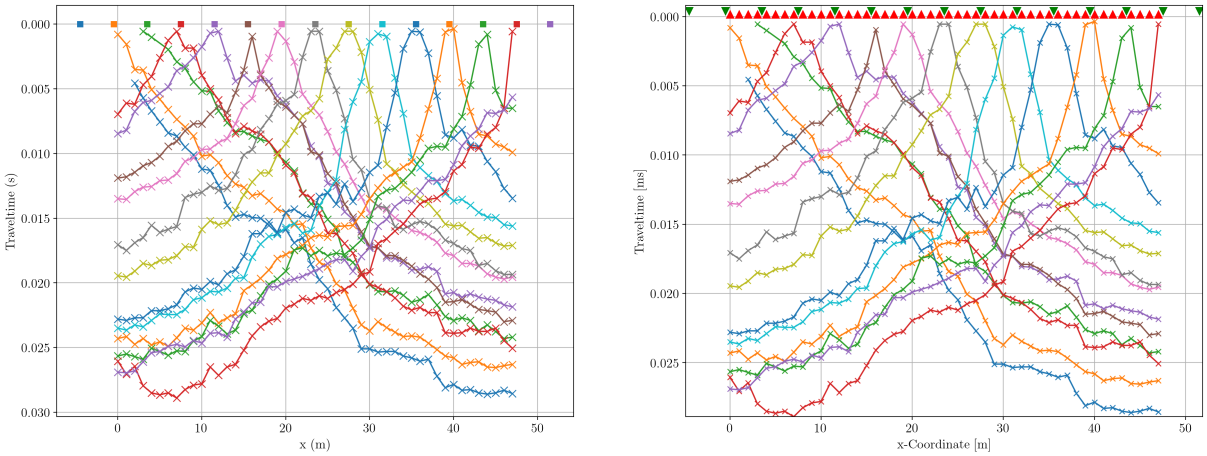


Figure 4: (a.) First-arrival travel-time curves for the Koenigsee dataset. Each curve represents a different shot position. (b.) The same travel-time curves with **green** triangles representing the **shot** indices and **red** pyramids representing the **geophone** indices.

Apparent Velocity Display

An alternative visualization uses apparent velocities v_{app} arranged in a matrix format with shot positions on one axis and geophone positions on the other axis, with color indicating apparent velocity magnitude:

$$v_{\text{app}} = \frac{\Delta x}{\Delta t} = \frac{|x_g - x_s|}{t(x_s, x_g)} \tag{17}$$

This matrix representation is particularly useful for identifying data gaps and acquisition coverage, anomalous velocities indicating structural features, quality of data (noisy vs. clear picks)

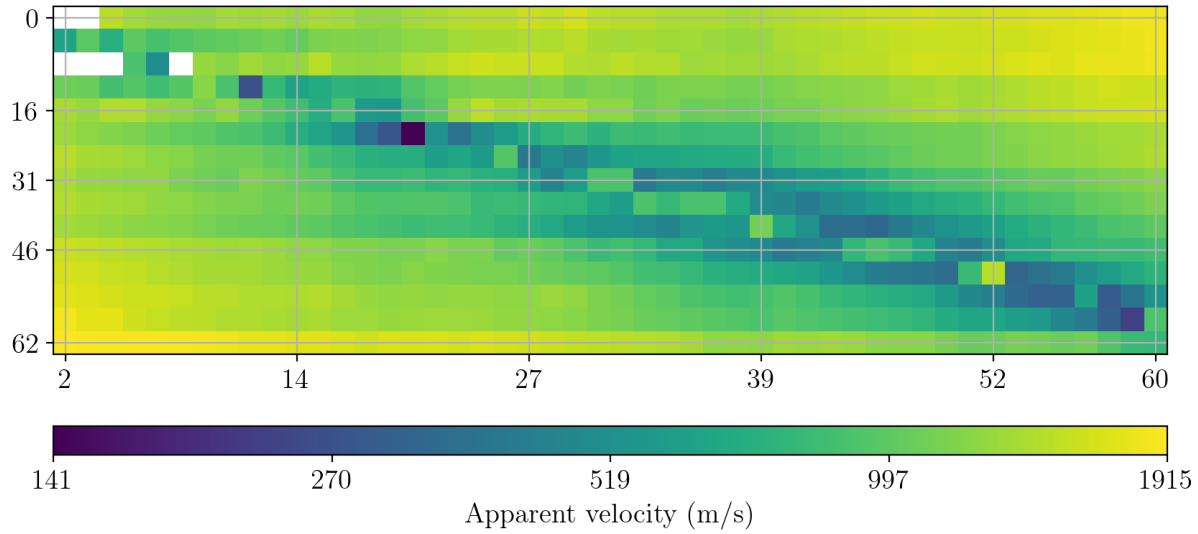


Figure 5: Apparent velocity matrix plot of the Koenigsee dataset. Shot positions are plotted on the horizontal axis, geophone positions on the vertical axis. Color scale represents apparent velocity (m/s). The visualization reveals systematic velocity variations reflecting subsurface structure and heterogeneity.

Inversion Parameters for Koenigsee

The Koenigsee dataset is inverted using the following parameters:

Table 2: Inversion parameters for Koenigsee dataset

Parameter	Value	Description
`secNodes`	3	Secondary mesh nodes (refinement level)
`paraMaxCellSize`	5.0 m	Maximum parametrization cell size
`zWeight`	0.2	Vertical weighting (anisotropic smoothness)
`vTop`	500 m/s	Starting velocity at surface
`vBottom`	5000 m/s	Starting velocity at depth
λ	20	Damping Factor for regularization $\chi^2_{\text{total}} = \chi^2_{\text{data}} + \lambda \cdot R(\mathbf{m})$

Parameter Interpretation

vTop and vBottom

PyGIMLi internally creates a **linear model** when calling `mgr.invert(.., vTop=500, vBottom=5000, ..)`. The linear gradient starting model is hence:

$$v_{\text{ref}}(z) = v_{\text{top}} + (v_{\text{bottom}} - v_{\text{top}}) \frac{z}{z_{\text{max}}} \tag{18}$$

Final Inversion Map

Final results are available at `travel-time-tomography/data/TravelTimeManager`.

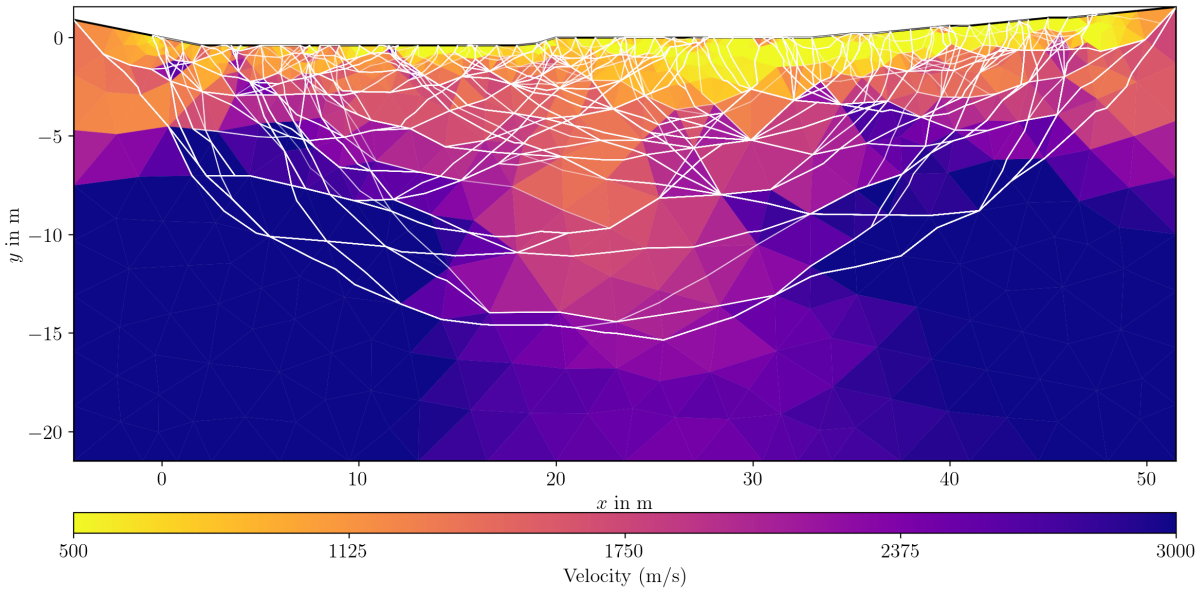


Figure 6: Final inverted velocity model for Koenigsee dataset. Color scale represents seismic P-wave velocity in m/s. The model reveals a three-layered structure: upper low-velocity layer (500–1200 m/s) representing weathered zone, intermediate moderate-velocity layer (1200–2000 m/s), and high-velocity bedrock (2000–2693 m/s). Superimposed ray coverage (or ray paths) indicates illumination of the model by the seismic ray paths

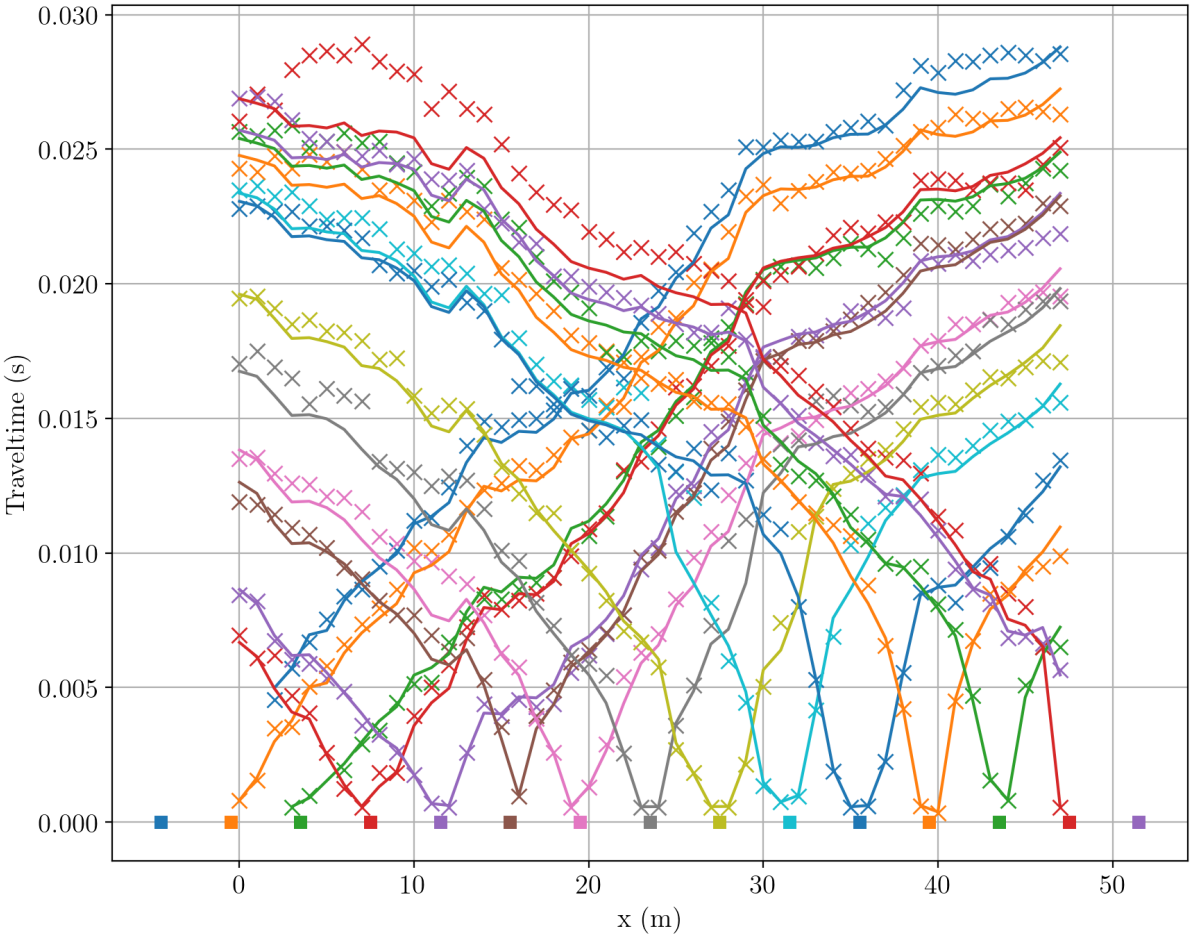


Figure 7: The corresponding predicted travel-time arrival data calculated using the inverted velocity model through forward ray tracing indicating residual misfit.

Conclusions

Travel-time tomography via Dijkstra ray tracing and Gauss-Newton inversion is a mature, effective method for near-surface velocity reconstruction. The combination of physical simplicity (ray paths on a mesh), mathematical robustness (well-posed inverse problem with regularization), and computational efficiency ($O(n \log n)$ forward solves) makes it ideal for practical applications. PyGIMLi provides a high-level, user-friendly implementation enabling reproducible workflows from raw data to interpreted models in a few lines of code.

Bibliography

[1] Bensen, G. D., Ritzwoller, M. H., Barmin, M. P., Levshin, A. L., Lin, F., Moschetti, M. P., Shapiro, N. M., and Yang, Y. Processing seismic ambient noise data to obtain reliable broad-band surface wave dispersion measurements. *Geophysical Journal International 169*, 3 (06 2007), 1239–1260.

[2] Kennett, B. L. N., and Engdahl, E. R. Traveltimes for global earthquake location and phase identification. *Geophysical Journal International 105*, 2 (05 1991), 429–465.

[3] Kennett, B. L. N., Engdahl, E. R., and Buland, R. Constraints on seismic velocities in the earth from traveltimes. *Geophysical Journal International 122*, 1 (07 1995), 108–124.

[4] Park, C. Imaging dispersion curves of surface waves on multi-channel record. *Seg Technical Program Expanded Abstracts 17* (01 1999).

[5] Rücker, C., Günther, T., and Wagner, F. M. pyGIMLi: An open-source library for modelling and inversion in geophysics. *Computers and Geosciences 109* (2017), 106–123.