

¹ [COR: Number]
² Cover Letter

³ **formikoj: A flexible library for data management and processing in geophysics - Application
4 for seismic refraction data**

⁵ Matthias Steiner, Adrián Flores Orozco

⁶ Dear Editors-in-Chief,

⁷
⁸ we are submitting our manuscript "formikoj: A flexible library for data management and processing in geophysics -
⁹ Application for seismic refraction data", which we consider is a suitable contribution for Computers & Geosciences.
¹⁰ We confirm that the submission follows all the requirements and includes all the items of the submission checklist.

¹¹
¹² The manuscript presents the open-source python library formikoj for managing and processing geophysical data col-
¹³ lected in environmental and engineering investigations. formikoj was specifically implemented for multi-platform
¹⁴ usage to allow the efficient collaboration and exchange of data between different partners in research projects and
¹⁵ academia. In this regard, we believe that this library aids in providing reproducible data and results as well as estab-
¹⁶ lishing and maintaining good research practices. Accordingly, we consider this manuscript relevant to the audience of
¹⁷ Computers & Geosciences, and in general for geoscientists and practitioners working with geophysical methods.

¹⁸
¹⁹ We provide the source codes in a public repository with details listed in the section "Code availability".

²⁰
²¹ We look forward to your decision.

²²
²³ Yours sincerely,

²⁴
²⁵ Matthias Steiner and Adrián Flores Orozco
²⁶ Research Unit Geophysics, Department of Geodesy and Geoinformation, TU Wien; matthias.steiner@geo.tuwien.ac.at
²⁷

²⁸ **Highlights**

²⁹ **formikoj: A flexible library for data management and processing in geophysics - Application**
³⁰ **for seismic refraction data**

³¹ Matthias Steiner, Adrián Flores Orozco

- ³² • flexible open-source and cross-platform library for managing and processing of geophysical data
³³ • possibility to be deployed for different geophysical methods and/or instruments
³⁴ • application for the modeling and processing of seismic refraction datasets
³⁵ • applicable for seismic refraction data collected in 2D and 3D survey geometries
³⁶ • easily scalable for custom requirements

37 **formikoj: A flexible library for data management and processing in**
38 **geophysics - Application for seismic refraction data**

39 Matthias Steiner^{a,*}, Adrián Flores Orozco^a

40 ^aResearch Unit Geophysics, Department of Geodesy and Geoinformation, TU Wien

41

42 **ARTICLE INFO**

43

44 **Keywords:**
45 geophysical data processing
46 seismic refraction
47 first break picking
48 seismic waveform modeling
49 cross-platform application
50 geophysical python library
51 flexible open-source libraries
52 wave based methods

53

54

55

56

57

58

59 **CRediT authorship contribution statement**

60 **Matthias Steiner:** Conceptualization and implementation of the library, creating the figures, preparation of the
61 manuscript. **Adrián Flores Orozco:** Conceptualization of the library, preparation of the manuscript.

62 **1. Introduction**

63 The acquisition of spatially quasi-continuous data in a non-invasive manner renders geophysical methods suitable
64 for engineering and environmental investigations (e.g., Parsekian et al., 2015; Nguyen et al., 2018; Romero-Ruiz et al.,
65 2018). However, the processing of geophysical data often relies on commercial software solutions and the associ-
66 ated licensing costs might render their use prohibitively expensive, which might be the case for academic projects
67 or institutions. The most popular packages are Res2DInv¹ for electrical methods, Halliburton Landmark SeisSpace
68 ProMAX² or ParkSeis³ for seismic methods, or ReflexW⁴ for ground-penetrating radar and seismic methods. A com-
69 mon limitation of the aforementioned software solutions refers to their specific platform requirements mainly related
70 to the type and version of the operating system; moreover, the possibility to adapt the code are limited if possible at
71 all. Considering the substantial changes regarding the market shares of operating systems within the last two decades,
72 platform-specific software packages are becoming particularly obstructive for academic research and teaching. The

*Corresponding author

ORCID(s): 0000-0002-3595-3616 (M. Steiner); 0000-0003-0905-3718 (A. Flores Orozco)

¹<https://www.geometrics.com/software/res2dinv/>, last accessed on March 3, 2023

²<https://www.landmark.solutions/SeisSpace-ProMAX>, last accessed on March 3, 2023

³<https://www.parkseismic.com/parkseis/>, last accessed on March 3, 2023

⁴<https://www.sandmeier-geo.de/reflexw.html>, last accessed on March 3, 2023

73 increasing popularity of the Python programming language led to the development of various cross-platform open-
 74 source software packages for processing, modeling and inverting geophysical data. Available packages can focus on
 75 specific geophysical methods, for instance, ResIPy (Blanchy et al., 2020) for electrical data, GPRPy (Plattner, 2020)
 76 for ground-penetrating radar data, or ObsPy (Beyreuther et al., 2010) and Pyrocko (Heumann et al., 2017) for seis-
 77 mological data. Other packages provide frameworks for the inversion and permit the inclusion of forward models for
 78 different geophysical methods, e.g., SimPEG (Cockett et al., 2015), Fatiando a Terra (Uieda et al., 2013) or pyGIMLi
 79 (Rücker et al., 2017).

80 The seismic refraction tomography (SRT) is a standard technique in environmental and engineering applications.
 81 Often applied together with other geophysical methods, the SRT is routinely used, e.g., in permafrost studies (e.g.,
 82 Draebing, 2016; Steiner et al., 2021), for the investigation of landfills (e.g., Nguyen et al., 2018; Steiner et al., 2022),
 83 or for hydrogeological characterizations (e.g., Bücker et al., 2021). The market for seismic processing software has
 84 long been dominated by software packages designed for the processing of large datasets, e.g., associated to oil or gas ex-
 85 ploration. Accordingly, these seismic processing solutions may not be suited for small-scale projects in environmental
 86 and engineering studies, or for teaching activities. ReflexW overcomes such limitations by providing processing tools
 87 specifically designed for near-surface investigations at substantially lower costs. In terms of licensing costs, Stockwell
 88 (1999) went a step further by making the Seismic Unix framework available entirely free of charge; whereas Guedes
 89 et al. (2022) recently presented RefraPy, a python processing tool for seismic refraction data. Implemented in python,
 90 RefraPy is potentially suitable for cross-platform usage, yet Guedes et al. (2022) developed and tested solely for Win-
 91 dows operating systems. Moreover, RefraPy does not offer the possibility to generate synthetic seismic waveform data,
 92 as required for survey design, as well as teaching and interpretation purposes.

93 The formikoj library presented here is an open-source framework for creating synthetic datasets, as well as for man-
 94 aging and processing numerical and field data independently from the operating system and without licensing costs;
 95 thus, overcoming limitations associated to existing solutions. The design of the library follows the multi-method
 96 concept of pyGIMLi and SimPEG, which allows for the implementation of custom designed tools for different geo-
 97 physical methods. The usage of transparent file formats, e.g., the unified data format (udf⁵), and data management
 98 concepts (SQLite database) within the formikoj framework facilitates a simple data exchange between partners in re-
 99 search projects and academia, which is required to guarantee the repeatability of results and good research practices.
 100 Considering the diverse applications of the SR method we demonstrate the applicability of the proposed library based
 101 on tools implemented within the formikoj framework for the modeling and processing seismic waveform data. In par-
 102 ticular, we present here a series of illustrative use cases based on the formikoj library referring to (i) the modeling of
 103 synthetic seismic refraction (SR) waveform data, (ii) the processing of a 2D SR field dataset collected with a roll-along

⁵http://resistivity.net/bert/data_format.html, last accessed on March 3, 2023

104 survey geometry, and (iii) the processing of a 3D SR field data set.

105 2. Design and structure of the formikoj library

106 As illustrated in Figure 1, the formikoj library comprises a modeling and a processing module, which both rely
 107 on a common utilities module. The `DataModeler` and the `MethodManager` class provide the basis to add modeling
 108 or processing functionalities for any kind of geophysical methods. In this manuscript, we present the application of
 109 the formikoj library for the modeling and processing of seismic refraction data based on the fundamental use cases
 110 presented in Figure 2 and 3, respectively. These diagrams aim at illustrating the corresponding workflows as well as the
 111 required interactions between the user, the formikoj library and third-party packages. The `SeismicWaveformModeler`
 112 and `SeismicRefractionManager` classes implement these fundamental use cases, yet their actual capabilities are
 113 continuously expanded, e.g., to address specific modeling or processing requirements as well as to enhance the user
 114 experience. Similar to RefraPy, these classes are built upon the functionalities of existing packages such as ObsPy for
 115 the processing of seismological data (Beyreuther et al., 2010) and pyGIMLi for the modeling and inversion of different
 116 geophysical data (Rücker et al., 2017). Other important third party dependencies refer to NumPy (Harris et al., 2020)
 117 and Pandas (McKinney, 2010) for general data handling, as well as matplotlib (Hunter, 2007) and PyVista (Sullivan
 118 and Kaszynski, 2019) for data visualization. In the current version, we implemented and tested formikoj primarily
 119 on Linux machines, yet the library has been successfully tested and used on all major operating systems, i.e., Linux,
 120 MacOS and Windows.

121 2.1. Generation of seismic waveform data for synthetic subsurface models

122 The SR method exploits the ground motion recorded by sensors installed in the surface (e.g., geophones) to char-
 123 acterize the propagation of seismic waves generated at well known locations (i.e., shot stations). The visualization
 124 of the ground motion as function of time yields a so-called seismogram for each geophone position. Accordingly,
 125 the `SeismicWaveformModeler` class provides a flexible way to generate synthetic seismic waveform data either in
 126 a python script, interactively in a jupyter notebook or an ipython shell. To create an instance of the class the user
 127 provides the absolute or relative path to the working directory as parameter to the constructor:

```
128 1 # Import the SeismicWaveformModeler from the formikoj library
129 2 from formikoj import SeismicWaveformModeler
130 3
131 4 # Create an instance of the SeismicWaveformModeler
132 5 swm = SeismicWaveformModeler('..')
133 INFO    : Created instance of SeismicWaveformModeler
```

Table 1

Description of the information to be provided in the columns of a measurement scheme in csv format.

Column	Content	Data type	Description
1	x coordinate	float	Station x coordinate, e.g., given in (m)
2	y coordinate	float	Station y coordinate, e.g., given in (m)
3	z coordinate	float	Station z coordinate, e.g., given in (m)
4	Geophone	bool	1 in case of a receiver station, 0 otherwise
5	Shot	bool	1 in case of a shot station, 0 otherwise

134 In the working directory, the required input files are provided via the subdirectory *in*, whereas the modeling out-
 135 put will be stored in the automatically created subdirectory *out*. The key input file is the measurement scheme as it
 136 contains information regarding the distribution of the shot and geophone stations. If provided in the unified data for-
 137 mat, the measurement scheme is imported directly with pyGIMLi into a DataContainer. In case the measurement
 138 scheme is provided as a csv file, the SeismicWaveformModeler reads the information and writes it to a pyGIMLi
 139 DataContainer object. If provided as csv file, the measurement scheme contains a single line for each station in the
 140 survey layout, where a station either hosts a geophone or a shot, or both (see Table 1). The values provided in each
 141 line need to be separated by a unique delimiter, and the file must not contain a header.

142 For the modeling of the seismic waveform data, the parameters characterizing the base wavelet, the synthetic
 143 subsurface model and the resulting waveform datasets are provided (see Table 2) in a configuration file following the
 144 yaml format:

```

145   wavelet:
146     length: 1.024
147     frequency: 100
148     sampling_rate: 2000
149     pretrigger: 0.02
150
151   model:
152     velmap: [[1, 750], [2, 2500], [3, 4000]]
153     layers: [[1, 3], [2, 5], [3, 15]]
154     quality: 32
155     area: 10
156     smooth: [1, 10]
157     sec_nodes: 3
158
159   dataset:
160     number: 1
161     names: [syn_data]
162     noise: 1

```

```

163     noise_level: 1e-4
164     missing_shots: 1
165     broken_geophones: 1
166     wrong_polarity: 1
167
168 traveltimes:
169     noise_relative: 0.
170     noise_absolute: 0.

```

171 In this exemplary configuration, the first block (`wavelet`) contains the parameterization of the base wavelet, which
 172 controls the modeling of the seismic waveforms (see Table 2). The second block (`model`) contains information regard-
 173 ing the synthetic subsurface model. In the configuration file, the user can define simple models where all layers are
 174 considered to be parallel to the surface topography. The topography is inferred from the station geometry provided
 175 in the measurement scheme, while the seismic velocity (`velmap`) and thickness (`layers`) of the different layers are
 176 defined in the configuration file. The remaining parameters (`quality`, `area`, `smooth` and `sec_nodes`) refer to the
 177 properties of the mesh that is generated for the forward modeling (we refer to the respective pyGIMLi resources⁶ for
 178 further information). In the third block of the exemplary configuration file (`dataset`), the user can set specific names
 179 for the datasets to be created (the number of datasets is automatically determined), or set the number of datasets to be
 180 created and the dataset names are automatically generated with the prefix *dataset_*. The remaining parameters in this
 181 block control the random error (`noise`) and systematic errors in the modeled seismic waveform data (see Table 2).
 182 The number and position of the shot and geophone stations affected by the systematic errors are randomly chosen with
 183 a maximum of 5 % of the total number of stations in order to avoid a high number of invalid trace data. The parameters
 184 in the final block (`traveltimes`) control the forward modeling of the corresponding seismic traveltimes (see Table 2).
 185 In particular, the user can set the relative and absolute noise that is added to the synthetic seismic traveltimes computed
 186 for the given subsurface model.

187 If stored in the input directory such a configuration file can be imported through the `load` method of the `SeismicWaveformMo`

```

188 6 # Load and parse the configuration file
189 7 swm.load('config')

```

190 INFO : Configuration loaded

191 Instead of defining a simple subsurface model in the configuration file, we provide a more complex model in the
 192 binary mesh format (e.g., a `bms` file). In particular, we prepare the model and the corresponding forward modeling
 193 mesh based on the mesh tools provided by pyGIMLi and save the mesh in the `bms` format (a commented version of the

⁶https://www.pygimli.org/pygimliapi/_generated/pygimli.meshTools.html#pygimli.meshTools.createMesh, last accessed March 3, 2023

Table 2

Description of the parameters, which can be defined in a configuration file used for the modeling of the synthetic seismic data.

Parameter	Unit/ Data type	Description
wavelet		
length	s	Length of the base wavelet, which also defines the length of the synthetic seismic waveform data
frequency	Hz	Frequency of the base wavelet
sampling_rate	Hz	Defines temporal resolution of the seismic waveforms
pretrigger	s	Add buffer to the seismic waveforms before the onset of the actual data
dataset		
number	int	Number of datasets to be created
names	list (string)	Names of the datasets
noise	bool	1 in case noise should be added to the synthetic waveforms, 0 otherwise
noise_level	-	Level of the seismic background noise
missing_shots	bool	1 in case the datasets should be affected by missing shot files, 0 otherwise
broken_geophones	bool	1 in case the datasets should comprise broken geophones (i.e., no data in the corresponding seismograms), 0 otherwise
wrong_polarity	bool	1 in case the datasets should contain traces with inverse polarity, 0 otherwise
model		
velmap	list (float/int)	For each layer the first value defines the marker and the second one the seismic velocity within the layer
layers	list (float/int)	For each layer the first value defines the marker and the second one the thickness
traveltimes		
noise_relative	%/100	Relative noise to be added to the forward modeled seismic traveltimes
noise_absolute	s	Absolute noise to be added to the forward modeled seismic traveltimes

194 python script is presented in the Appendix). Similar to the configuration file, a bms file stored in the input directory
 195 can be imported into the workflow through the load method as follows:

```
196 8 # Load the mesh into the workflow
197 9 swm.load('mesh')
```

198 INFO : Mesh loaded

199 The forward modeling process generating the seismic waveform data is then invoked by passing the string 'waveforms'
 200 as parameter to the create method:

```

2010 # Start the modeling of the seismic waveform data
2011 swm.create('waveforms')

203 INFO    : Measurement scheme loaded
204 INFO    : Velocity model created
205 INFO    : Wavelet created
206 [+++++ 100% +++++] 2048 of 2048 complete
207 ...
208 [+++++ 100% +++++] 2048 of 2048 complete
209 INFO    : Dataset 'syn_data' created

```

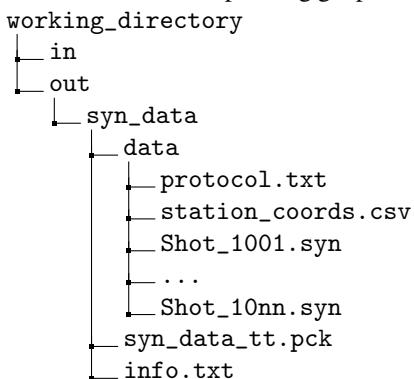
210 As can be seen from the log messages, the `SeismicWaveformModeler` first loads the measurement scheme into the
 211 workflow. In the second step, the provided mesh and the seismic velocity values defined in the configuration file are
 212 combined to create the seismic velocity model to be used for the waveform modeling. Figure 4a presents the subsurface
 213 model considered in this study, which consists of a top and a bottom layer with varying thickness characterized by
 214 seismic velocity values of 750 ms^{-1} and 4000 ms^{-1} , respectively. In the center, the model contains an irregularly
 215 shaped anomaly associated with a seismic velocity of 2500 ms^{-1} .

The third step refers to the generation of a Ricker wavelet based on the wavelet properties provided through the configuration file. In particular, the `SeismicWaveformModeler` uses the pyGIMLi function `ricker`, which creates the Ricker wavelet given as

$$u = (1 - 2\pi^2 f^2 t^2) e^{-\pi^2 f^2 (t-t_0)^2}. \quad (1)$$

216 In Equation 1, f is the frequency of the wavelet given in Hz, t refers to the time base definition, i.e., the length and
 217 resolution in the time domain, and t_0 is the time offset of the wavelet.

218 Subsequently, mesh, velocity model and Ricker wavelet are used to solve the pressure wave equation for each
 219 shot station defined in the measurement scheme with the pyGIMLi function `solvePressureWave`. The resultant
 220 waveforms at the corresponding geophone stations are extracted and saved in the output directory (`out`) as shown here:



221 The subdirectory *data* contains a separate file in the miniseed format (Ahern et al., 2012; Ringler and Evans,
 222 2015) for each shot position, where the file extension *syn* to indicate that the shot files contain forward modeled
 223 seismic waveform data. The measurement protocol (*protocol.txt*) and the station coordinates provided as a csv file
 224 (*station_coords.csv*) are also stored in this directory. The header of the measurement protocol contains the survey
 225 parameters, e.g., sampling rate, recording length, number of geophones and geophone spacing. Moreover, the protocol
 226 associates each shot file of the dataset with a specific location in the survey geometry with respect to the geophone
 227 positions, e.g.:
 228 #####
 229 Line: SYN_syn_data
 230 Sampling rate: 2000 Hz
 231 Recording length: 0.512 s
 232 Number of geophones: 48
 233 Geophone spacing: 2 m
 234 #####
 235
 236 File number | Station
 237 1001 | G001
 238 : | :
 239 1048 | G048

240 The auxiliary file *info.txt* provided in the dataset directory summarizes the parameters from the configuration file and
 241 information regarding the simulated systematic errors in the synthetic seismic waveform data:

242 Number of geophones: 48
 243 Number of shots: 48
 244 Recording length (s): 0.512
 245 Sampling frequency (Hz): 2000
 246 Wavelet type: Ricker
 247 Frequency of the wavelet (Hz): 100
 248
 249 Missing shot(s): 11
 250 Broken geophone(s): 13
 251 Wrong polarity geophone(s): 26

252 Figure 4b presents the seismic waveform data forward modeled for a shot point located in the center of the profile.
 253 The seismograms are shown as curves, whereas the strength of the amplitudes is added as color-coded information.
 254 In the seismograms we see clear first onsets along the entire profile, yet receivers 3 and 45 were modeled as broken
 255 geophones, i.e., the corresponding seismograms contain solely noise. Crosses overlaid on the valid seismograms at
 256 the respective first onset indicate the first break traveltimes *tt* between the shot and the receiver. In addition to the
 257 missing first break traveltimes for the broken receivers, we manually added a systematic error, i.e., we set an erroneous

258 traveltimes for receiver 36.

A pseudosection as presented in Figure 4c, illustrates apparent velocity (v_{app}) values computed as

$$v_{app} = \frac{aoffset}{tt}, \quad (2)$$

259 with $aoffset$ referring to the absolute offset between shot and receiver. The v_{app} values are plotted at pseudolocations,
 260 where the location along profile direction is defined by the midpoint of the corresponding shot-receiver pair and the
 261 pseudodepth is computed as 1/3 of $aoffset$. As demonstrated in Figure 4c, a pseudosection allows for the identi-
 262 fication of missing data (e.g., receiver 3 and 45) as well as systematic errors or outliers, i.e., velocities erroneously
 263 influenced by a single shot or receiver (see receiver 36). The main assumption here is that the pseudosection should
 264 reveal smooth transitions between lateral and vertical neighbors, considering that the data were collected with gradual
 265 changes in the position of the source (i.e., hammer blow) and the receiver (i.e., geophone). The pseudosection will
 266 reveal large variations in case of abrupt changes in the topography or the geometry of the array, yet this can be taken
 267 into account by the user during the identification of outliers and possible systematic errors.

Based on pseudosections erroneous traveltimes can be identified and subsequently removed or corrected, which is a critical processing step prior to the inversion of the data. In particular, the inversion of first break traveltimes aims at resolving a model of the P-wave velocity of the subsurface materials. Since the inversion of geophysical data is not within the scope of the formikoj library we rely for this purpose on the modeling and inversion capabilities of pyGIMLi (Rücker et al., 2017). The default inversion framework of pyGIMLi uses a generalized Gauss-Newton method to solve the inversion problem through the minimization of an objective function given as:

$$\Psi(\mathbf{m}) = \Psi_d(\mathbf{m}) + \lambda \Psi_m(\mathbf{m}). \quad (3)$$

268 The first term on the right-hand side of Equation 3 refers to the data misfit, whereas the second term denotes represents
 269 the model constraints, i.e., the regularization. The regularization parameter λ controls the influence of the regulariza-
 270 tion term on the inversion process.

271 The inversion of the synthetic first break traveltimes considered here resolves the subsurface model presented in
 272 Figure 4d. To aid in the evaluation of this inversion result we superimposed the known interfaces between the different
 273 subsurface unit of the synthetic model. As can be seen from this plot, the imaging result resolves the fundamental
 274 structural features and reflects the P-wave velocity distribution of the synthetic subsurface model. Deviations from the
 275 true velocity model are due to the smoothness-constraint inversion scheme applied by pyGIMLi. To obtain sharper
 276 contrasts between the different subsurface units in the imaging result structural constraints could be incorporated in
 277 the inversion, e.g., as demonstrated by (Steiner et al., 2021). Nonetheless, we consider Figure 4 to demonstrate the

278 applicability of the `SeismicWaveformModeler` class for generating synthetic seismic waveform data to be used in
 279 numerical P-wave refraction seismic investigations.

280 **2.2. Processing of seismic refraction datasets**

281 The SR method is based on the measurement of the traveltimes of seismic waves determined from the the first onset
 282 of the seismic energy recorded by geophones. In the SRT, the inversion of traveltimes gathered from tens to hundreds
 283 of seismograms permits the computation of variations in the seismic velocities in an imaging framework. Measuring
 284 the traveltimes in seismograms (i.e., first break picking) is commonly done manually (or semi-automatically) in an
 285 iterative process. The signal-to-noise ratio in the recorded seismograms substantially influences the quality of the
 286 traveltimes picked in the seismograms, and thus a proper enhancement of the perceptibility of the first onsets is crucial.
 287 Accordingly, the `SeismicRefractionManager` class provides functionalities that permit the processing of seismic
 288 waveform data for a refraction seismic analysis. These functionalities involve the reading of seismic waveform data,
 289 combining the data with information about the survey geometry, processing of the waveforms as well as the picking of
 290 first break traveltimes. Since the first break picking is a highly interactive process, the `SeismicRefractionManager`
 291 is designed primarily for usage from within an ipython shell.

292 For the demonstration of the fundamental `SeismicRefractionManager` capabilities we consider the synthetic
 293 seismic data created with the `SeismicWaveformModeler` above. This allows us to show that both classes can be
 294 combined in subsequent workflows, and to present the forward modeled seismic waveform data, the synthetic first
 295 break traveltimes as well as the seismic P-wave velocity model obtained through the inversion of these traveltimes.

296 **2.2.1. Compiling the survey information and creating a project**

297 To create a new or load an existing `SeismicRefractionManager` project, the working directory needs to contain
 298 specific subdirectories:

```
working_directory
  └── 01_data
    └── raw
  └── 02_geom
  └── 03_proc
```

299 In this directory structure, the seismic shot files are stored in `01_data/raw` and the geometry file (`geometry.csv`) is
 300 provided in `02_geom`. The geometry file is a csv file that provides an abstract representation of the survey layout and
 301 must not contain a header. The fundamental element for the description of the survey layout is the station, which refers
 302 either to a geophone position, a shot position or a position with co-located shot and geophone. For each station the
 303 geometric and semantic information described in Table 3 are provided column-wise, i.e., each line in the geometry file
 304 corresponds to a single station with a unique position within the survey layout.

305 An instance of the `SeismicRefractionManager` can be created by providing the path to the working directory

Table 3

Description of the information to be provided in the columns of the geometry file.

Col	Content	Data type	Description
1	x coordinate	float	Station x coordinate, e.g., given in (m)
2	y coordinate	float	Station y coordinate, e.g., given in (m)
3	z coordinate	float	Station z coordinate, e.g., given in (m)
4	Geophone	bool	1 if a geophone was deployed at station, 0 otherwise
5	Shot	int	The numerical part of the file name, e.g., 1001, if a shot was conducted at this station, -1 otherwise
6	First geophone	int	First active geophone (-1 if no shot was conducted at the station)
7	Number of geophones	int	Number of active geophones (-1 if no shot was conducted at the station)

Table 4

Excerpt from the geometry file describing the survey geometry of the synthetic dataset generated in this study.

x (m)	y (m)	z (m)	Geo	Shot	1st Geo	# Geo
0.0	0.0	0.0	1	1001	1	48
2.0	0.0	0.0	1	1002	1	48
:	:	:	:	:	:	:
20.0	0.0	0.0	1	-1	1	48
:	:	:	:	:	:	:
74.0	0.0	0.0	0	1038	1	48
:	:	:	:	:	:	:
92.0	0.0	0.0	1	1047	1	48
94.0	0.0	0.0	1	1048	1	48

306 as parameter to the constructor. Based on the content of the working directory, the `SeismicRefractionManager`
 307 automatically decides whether (i) to start in the data preview mode (first break picking not possible), (ii) create a new
 308 project, or (iii) load an existing project from disk. In case the shot files as well as the geometry file are provided and a
 309 basic sanity check of the geometry file was successful, the `SeismicRefractionManager` creates a new project:

```
310 1 # Import the SeismicRefractionManager from the formikoj library
311 2 from formikoj import SeismicRefractionManager
312 3
313 4 # Create an instance of the SeismicRefractionManager
314 5 srm = SeismicRefractionManager('.')

315 INFO    : Read geometry information from file
316 INFO    : Extracted shot geometry
317 INFO    : Extracted receiver geometry
318 INFO    : Applied geometry
319 INFO    : Standard pickset 'picks' created
320 INFO    : Pickset 'picks' loaded
321 INFO    : 'picks' set as active pickset
322 Progress <=====> 100.0% completed
323 INFO    : Read 48 files
```

324 In a first step, the `SeismicRefractionManager` creates an SQLite database `prj.db` in the working directory based on
 325 the entity-relationship diagram shown in Figure 5. The geometry information is then read from the geometry file and
 326 stored in the database table `geometry` with consecutively numbered stations (see Figure 6). To allow for an efficient data
 327 selection for the user the `SeismicRefractionManager` creates database tables `shots` and `receivers`, which link the
 328 station numbers to shot index numbers (SIN) and receiver index numbers (RIN), respectively. For each shot-receiver
 329 pair the corresponding SIN and RIN are stored in the table `applied_geometry` together with the absolute offset and
 330 midpoint between these stations, i.e., the geometry is applied. In the last step, the database table `fbpicks` is created,
 331 which stores the first break traveltimes for each SIN-RIN pair together with the name of the corresponding pickset, i.e.,
 332 a common label for an entire set of first break traveltimes. By default, each project contains the default pickset 'picks',
 333 which is loaded and activated on startup. Once the database is initialized, the waveform data are read from disk and
 334 the project is ready for processing.

335 2.2.2. Selecting and visualizing seismic waveform data

336 Once the geometry is applied the `select` method of the `SeismicRefractionManager` allows to gather the seis-
 337 mic waveform data based on a common absolute offset (`aoffset`), a common RIN (`rin`), or a common SIN (`sin`)

```
338 6 # Select traces with common absolute offset
339 7 srm.select(by='aoffset', num=6)

340 INFO : 90 traces selected
```

```
341 8 # Select traces with receiver
342 9 srm.select(by='rin', num=10)
```

```
343 INFO : 24 traces selected
```

```
344 10 # Select traces with receiver
345 11 srm.select(by='sin', num=23)
```

```
346 INFO : 24 traces selected
```

347 For the currently selected trace data the frequency spectrum can be computed and visualized through the `plot`
 348 method:

```
349 12 # Plot the frequency spectrum
350 13 srm.plot(type='spectrum')
```

351 Examining a frequency spectrum as shown in Figure 7 provides information regarding the dominating signal frequen-
 352 cies. Discriminating signal and noise frequency ranges is critical for the selection of adequate filter types and settings

353 in order to enhance the signal-to-noise ratio of the seismograms. Filters can be applied to the currently selected traces
 354 through the `filter` method, which utilizes the frequency filters implemented in the ObsPy package (lowpass, high-
 355 pass, bandpass and bandstop; Beyreuther et al., 2010), e.g.:

```
356 14 # Apply bandpass filter
357 15 srm.filter(type='bandpass', freqmin=10, freqmax=120)
```

358 INFO : Applied bandpass filter (10.0 to 120.0 Hz)

359 By default, filters are solely applied to the currently selected traces, yet setting the filter to hold on automatically filters
 360 all subsequently selected traces with the same settings:

```
361 16 # Apply lowpass filter
362 17 srm.filter(type='lowpass', freq=120, onhold=True)
```

363 INFO : Applied 120.0 Hz lowpass filter

364 INFO : Set filter hold on

365 The currently selected traces can be visualized by calling the `plot` method without passing any parameter:

```
366 18 # Plot selected traces
367 19 srm.plot()
```

368 By default, the seismogram plot visualizes the seismic waveforms in a combination of wiggle trace and variable area
 369 mode, i.e., the trace data are shown as curves. The area of the curves is colored red for negative and blue for positive
 370 amplitudes, respectively (not shown for brevity). Pressing the up or down arrow key on the keyboard toggles between
 371 variable area and variable density mode, with the latter reflecting the strength of the amplitudes by the color saturation,
 372 i.e., high amplitudes refer to a stronger shade than low amplitudes (see Figure 8).

373 The active processing mode and data scaling mode are reported together with the traveltimes at the current cursor
 374 position in the status bar of the seismogram plot window (see Figure 9). The initial processing mode is 'Fb pick',
 375 i.e., first break picking is possible. The user can switch between the different modes by pressing specific keys on the
 376 keyboard. The 'm' key activates the trace mute mode ('Trc mute'), which allows to set the amplitude of a trace to
 377 zero by clicking with the left mouse button; clicking again on the same trace restores the amplitude information. The
 378 trace reverse mode ('Trc rev') is activated by pressing the 'r' key and enables the user to toggle the polarity of a trace
 379 by clicking on it with the left mouse button. The default data scaling mode is 'Zoom', which allows the scaling of
 380 the y-axis by turning the mouse wheel. By pressing the 'a' key the amplitude scaling mode ('Amp scal') is activated.
 381 Turning the mouse wheel increases or decreases the amplitudes of the traces currently shown in the seismogram plot,
 382 and thus might help to enhance the perceptibility of the first onsets. By pressing the key of the currently active mode
 383 again, the `SeismicRefractionManager` returns to the default mode; yet, the different modes can be activated in any
 384 arbitrary order (as illustrated in Figure 9).

385 **2.2.3. Analysis of the seismic waveforms and first break traveltimes picking**

386 In the seismogram plot, the waveforms can be analyzed and processed to obtain information about the subsurface
 387 conditions. Activating the velocity estimation mode ('Vel est') by pressing the 'v' key on the keyboard, enables the
 388 user to estimate velocities for different wave phases (e.g., originating from a refractor) by pressing the left mouse button
 389 and moving the cursor. Once the left mouse button is released, a line between the start and end point is drawn and
 390 labeled with the corresponding velocity (estimates can be deleted by clicking with the right mouse button).

391 If the processing mode 'Fb pick' is activated, picking of first break traveltimes is done individually by clicking
 392 with the left mouse button on the respective trace. Clicking again on the same trace will set the first break pick to the
 393 new location as there can only be one traveltimes for each SIN-RIN pair; whereas clicking with the right mouse button
 394 deletes the pick. Alternatively, first break picks can be set for multiple traces by pressing the left mouse button and
 395 moving the cursor. Once the left mouse button is released, first break picks are defined at the intersections between
 396 the line and the seismograms. In the same way, multiple picks can be deleted if a line is drawn with the right mouse
 397 button pressed. The first break traveltimes determined in the seismogram plot are automatically written to the project
 398 database when the window is closed or another set of traces is loaded by pressing the 'left' or 'right' arrow keys on the
 399 keyboard.

400 The traveltimes diagram for the currently active pickset can be created through the `plot` method:

```
40120 # Plot traveltimes diagram
40221 srm.plot(type='traveltimes')
```

403 Figure 10 presents an exemplary traveltimes diagram, which is a common way to examine the quality of the first break
 404 picking. Such illustration of the traveltimes can be used to identify outliers or erroneous measurements, which are
 405 commonly associated to traveltimes with substantial deviations from those observed at adjacent stations such as the
 406 first break pick for the SIN-RIN pair (23, 11). Outliers can be removed by clicking on the respective data point in
 407 the traveltimes diagram, which is instantly synchronized with the project database. If the seismogram plot and the
 408 traveltimes diagram are used side-by-side, changes made to the first break picks in one window will interactively trigger
 409 an update of the other one and vice versa. Once the user is satisfied with the quality of the first break traveltimes, the
 410 data points of the pickset can be exported in the unified data format, which is compatible with the pyGIMLi framework.
 411 In particular, the udf file is stored in the subdirectory *03_proc/picks* with the current timestamp as suffix:

```
41222 # Export first break traveltimes
41323 srm.picksets(do='export', name='pick')
```

414 INFO : Pickset 'pick' saved to `pick_20230303T140447'`

Table 5

Excerpt from the 3D survey geometry file.

x (m)	y (m)	z (m)	Geo	Shot	1st Geo	# Geo
40336.056	292470.692	120.0	1	1001	1	96
40334.751	292469.177	120.0	1	1002	1	96
:	:	:	:	:	:	:
40284.472	292455.431	120.0	1	1058	1	96
40285.896	292454.027	120.0	1	1059	1	96
:	:	:	:	:	:	:
40339.421	292402.681	120.0	1	1096	1	96
40305.228	292445.050	120.0	0	1097	1	96
:	:	:	:	:	:	:
40265.415	292431.957	0	0	1115	1	96
40255.453	292431.395	0	0	1116	1	96

415 3. Application to field data: Processing a 3D seismic refraction dataset

416 The SeismicRefractionManager allows also the visualization and processing of data collected with a 3D survey
 417 layout. To illustrate these capabilities, we present in Figure 11 the geometry of a 3D survey conducted in a soda lake
 418 located close to Vienna. The soda lake corresponds to quaternary sediments where capillary forces have developed a
 419 low permeable layer close to the surface (between 50 and 100 cm) with a high clay and salt content. The seismic survey
 420 aims to support the interpretation of the electrical and electromagnetic models obtained in a monitoring framework.
 421 Accordingly, the survey geometry shown in Figure 11 was specified by previously conducted electrical measurements
 422 with electrodes arranged along two perpendicular lines. The seismic data were collected with 48 geophones deployed
 423 along the North-East to South-West oriented line, and 48 geophones along the North-West to South-East oriented line,
 424 with a spacing of 2 m between the geophones. Shots were generated with an 8 kg sledgehammer at the geophone
 425 positions as well as at positions along the diagonals to obtain a sufficient coverage, as indicated in Figure 11.

426 In the field, each segment was measured separately, yet for the processing all measurements are combined in a
 427 single profile. We can implement such measurement layout in a geometry file as illustrated in Table ??.

428 Based on the shot files and the geometry file stored in the required directory structure the SeismicRefractionManager
 429 creates the project database.

430 To review the data quality along the entire profile it is possible to visualize the picking percentage, i.e., the ratio of
 431 actually picked traveltimes and total number of SIN-RIN pairs:

```
432     srm.plot('pickperc')
```

433 Figure 12 shows that the picking percentage is visualized separately for each SIN. In this way, a low picking percentage
 434 indicates shots affected by a low signal-to-noise ratio. For the Danube island dataset, the picking percentage is low
 435 in the first segment, which confirms the lack of datapoints observed in the pseudosection. Moreover, the picking
 436 percentage plot can be used to track the picking progress, for instance if the traveltimes cannot be determined in one

437 session or to identify single shots that might have been forgotten during the first break picking. Accordingly, it is
 438 advisable to check the picking percentage prior to exporting the traveltimes for the inversion.

439 The geometry file contains the 3D coordinates for each shot or receiver station, and thus the
 440 `SeismicRefractionManager` automatically configures the project for 3D processing. Figure 13 presents the seismic
 441 waveform data recorded for SIN 1, i.e., the shot position co-located with the first geophone (Station 01 in Figure 11).
 442 The data for RIN 1 to 48 appear familiar as the corresponding SIN-RIN layout refers to the one of conventional
 443 2D profiles; whereas the seismic waveforms for RIN 49 to 96 show an entirely different pattern. To understand such
 444 visualization, we need to take into account that RIN 49 to 96 are deployed perpendicular to the direction of propagation
 445 of the waveform originating from SIN 1. Accordingly, the observed curvature in the first onsets is due to the varying
 446 offset of the different SIN-RIN pairs.

447 Due to the 3D survey geometry, a 2D pseudosection is not suitable to illustrate the apparent velocity values obtained
 448 from the picked first break traveltimes. Hence, the `SeismicRefractionManager` automatically switches to a 3D
 449 representation where the apparent velocity values are illustrated in an interactive 3D pseudosection. This plot can be
 450 rotated and the image section can be zoomed and panned allowing the user to easily investigate the data quality for
 451 3D geometries. Figure 14 shows a screenshot of the 3D pseudosection for the salt lake dataset; yet, such screenshot
 452 cannot reveal the full capabilities implemented in the `SeismicRefractionManger` for the interactive analysis and
 453 visualization of 3D pseudosections.

454 Once the first break picking is finished, the corresponding pickset can be exported for the inversion. The inversion
 455 results and their interpretation are not the scope of this manuscript, yet Figures 13 and 14 reveal the capabilities
 456 provided by the proposed framework for the visualization and processing of data collected in 3D survey geometries.

457 4. Conclusions and Outlook

458 We have presented formikoj, a flexible open-source library enabling the development of modeling and processing
 459 tools for geophysical data. Implemented in python and tested on all major operating systems (Linux/Unix, MacOS,
 460 Windows), formikoj is suitable for multi- and cross-platform applications; thus, allowing collaboration between users
 461 free from licensing costs and platform requirements.

462 We demonstrated the capabilities of the formikoj framework to develop versatile and easily scalable classes for
 463 the modeling and processing of waveforms in seismic refraction surveys. The required interaction with the user
 464 is reduced to a minimum as crucial processing steps are automatized within the `SeismicWaveformModeler` and
 465 `SeismicRefractionManager` based on efficient data input strategies, for instance the preparation and import of the
 466 geometry file or the keyboard-based interaction related to the first break picking. In this regard, the user controls the
 467 formikoj library by providing text-based commands preferably through an ipython shell to exploit the full interactive

468 potential modeling and processing tools. However, applications of the formikoj library can also be automatized by
469 implementing workflows in python scripts or jupyter notebooks.

470 Based on three exemplary use cases, we illustrated the applicability of both the `SeismicWaveformModeler` and
471 the `SeismicRefractionManager`. In the first use case, we showed the possibility to forward model seismic wave-
472 form data based on custom subsurface models and survey geometries with the `SeismicWaveformModeler`. Addi-
473 tionally, the resulting waveforms can be subjected to systematic and random noise sources. The capabilities of the
474 `SeismicRefractionManager` were demonstrated through the processing of field datasets collected in complex sur-
475 vey layout, namely a roll-along and a 3D geometry. Moreover, we showed how the different data visualization options
476 can assist during the data processing to ensure consistency in the first break traveltimes. In particular, we developed a
477 visualization of the traveltimes by means of pseuodsections illustrating the corresponding apparent seismic velocities.
478 Such plots allow for a quick identification of systematic errors and outliers in both 2D and 3D datasets.

479 By making the source code of the formikoj library available under the MIT license we intend to spark the develop-
480 ment of further modeling and processing tools for various geophysical models based on this framework. Our further
481 efforts will focus on implementing tools for other wave-based geophysical methods used in frame of our research
482 activities, such as multi-channel analysis of (seismic) surface waves or magnetotelluric surveys.

483 5. Acknowledgments

484 The authors are grateful to Nathalie Roser and Lukas Aigner for benchmarking the formikoj library against estab-
485 lished software packages in frame of their research activities. Furthermore, we would like to thank Clemens Moser,
486 Martin Mayr, Vinzenz Schichl and Harald Pammer for their constructive comments during first tests of the formikoj
487 framework as well as for their help during the seismic surveys.

488 Code and data availability section**489** Name of the code/library: formikoj**490** Contact: matthias.steiner@geo.tuwien.ac.at**491** Hardware requirements: No specific requirements**492** Program language: Python**493** Software required: Anaconda/Miniconda recommended**494** Total program and dataset size: 250 MB**495** The source codes and exemplary data sets are available for downloading at the link:**496** <https://git.geo.tuwien.ac.at/msteine1/formikoj.git>**497** The orthophotos used in Figures ?? and 11 were published by geoland.at under a CC BY 4.0 license.**498 A. Source code for generating the subsurface model considered in this study**

```

499 1 # Import required packages
500 2 import numpy as np
501 3 import pygimli as pg
502 4 import pygimli.meshTools as mt
503 5
504 6 # Create the model geometry
505 7 # - top layer
506 8 top = mt.createPolygon([[0, 0], [94, 0],
507 9                 [94, -3.5], [72, -3.5],
508 10                [20, -2], [0, -2]],
509 11                isClosed=True, marker=1, area=0.1)
510 12
511 13 # - bottom layer
512 14 bottom = mt.createPolygon([[0, -2], [20, -2],
513 15                [22, -6], [70, -6],
514 16                [72, -3.5], [94, -3.5],
515 17                [94, -10], [0, -10]],
516 18                isClosed=True, marker=3, area=0.1)
517 19
518 20 # - anomaly/infill
519 21 infill = mt.createPolygon([[20, -2], [72, -3.5],
520 22                [70, -6], [22, -6]],
521 23                isClosed=True, marker=2, area=0.1)
522 24
523 25 geom = top + infill + bottom

```

```

52426
52527 # Define shot and receiver stations and create corresponding nodes
52628 nstats = 48
52729 spc = 2
52830
52931 stations = np.vstack((np.linspace(0, (nstats-1)*spc, nstats),
53032                 np.zeros(nstats))).T
53133
53234 for p in stations:
53335     geom.createNode(p)
53436
53537 # Create mesh for the finite element modeling
53638 mesh = mt.createMesh(geom, quality=34)
53739
53840 # Save the mesh in the binary mesh format for later use
53941 # with the SeismicWaveformModeler
54042 mesh.save('mesh.bms')

```

541 References

- 542 Ahern, T., Casey, R., Barnes, D., Benson, R., Knight, T., Trabant, C., 2012. SEED Reference Manual, Version 2.4. URL: http://www.fdsn.org/pdf/SEEDManual_V2.4.pdf.
- 543 Beyreuther, M., Barsch, R., Krischer, L., Megies, T., Behr, Y., Wassermann, J., 2010. Obspy: A python toolbox for seismology. *Seismological Research Letters* 81, 530–533. doi:10.1785/gssrl.81.3.530.
- 544 Blanchy, G., Saneiyan, S., Boyd, J., McLachlan, P., Binley, A., 2020. Resipy, an intuitive open source software for complex geoelectrical inversion/modeling. *Computers & Geosciences* 137, 104423. URL: <https://www.sciencedirect.com/science/article/pii/S0098300419308192>, doi:<https://doi.org/10.1016/j.cageo.2020.104423>.
- 545 Bücker, M., Flores Orozco, A., Gallistl, J., Steiner, M., Aigner, L., Hoppenbrock, J., Glebe, R., Morales Barrera, W., Pita de la Paz, C., García García, C.E., et al., 2021. Integrated land and water-borne geophysical surveys shed light on the sudden drying of large karst lakes in southern mexico. *Solid Earth* 12, 439–461. doi:10.5194/se-12-439-2021.
- 546 Cockett, R., Kang, S., Heagy, L.J., Pidlisecky, A., Oldenburg, D.W., 2015. Simpeg: An open source framework for simulation and gradient based parameter estimation in geophysical applications. *Computers & Geosciences* 85, 142–154. URL: <https://www.sciencedirect.com/science/article/pii/S009830041530056X>, doi:<https://doi.org/10.1016/j.cageo.2015.09.015>.
- 547 Draebing, D., 2016. Application of refraction seismics in alpine permafrost studies: A review. *Earth-Science Reviews* 155, 136–152. doi:<https://doi.org/10.1016/j.earscirev.2016.02.006>.
- 548 Guedes, V.J.C.B., Maciel, S.T.R., Rocha, M.P., 2022. Refrapy: A python program for seismic refraction data analysis. *Computers & Geosciences* 159, 105020. URL: <https://www.sciencedirect.com/science/article/pii/S0098300421003022>, doi:<https://doi.org/10.1016/j.cageo.2021.105020>.
- 549 Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern,

- 561 R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Rio, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard,
562 K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E., 2020. Array programming with NumPy. *Nature* 585, 357–362. URL:
563 <https://doi.org/10.1038/s41586-020-2649-2>, doi:10.1038/s41586-020-2649-2.
- 564 Heimann, S., Kriegerowski, M., Isken, M., Cesca, S., Daout, S., Grigoli, F., Juretzek, C., Megies, T., Nooshiri, N., Steinberg, A., Sudhaus, H.,
565 Vasyura-Bathke, H., Willey, T., Dahm, T., 2017. Pyrocko - an open-source seismology toolbox and library doi:10.5880/GFZ.2.1.2017.001.
- 566 Hunter, J.D., 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering* 9, 90–95. doi:10.1109/MCSE.2007.55.
- 567 McKinney, W., 2010. Data Structures for Statistical Computing in Python, in: Stéfan van der Walt, Jarrod Millman (Eds.), *Proceedings of the 9th*
568 *Python in Science Conference*, pp. 56 – 61. doi:10.25080/Majora-92bf1922-00a.
- 569 Nguyen, F., Ghose, R., Isunza Manrique, I., Robert, T., Dumont, G., 2018. Managing past landfills for future site development: A review of the
570 contribution of geophysical methods, in: *Proceedings of the 4th International Symposium on Enhanced Landfill Mining*, pp. 27–36.
- 571 Parsekian, A., Singha, K., Minsley, B.J., Holbrook, W.S., Slater, L., 2015. Multiscale geophysical imaging of the critical zone. *Reviews of*
572 *Geophysics* 53, 1–26. doi:10.1002/2014RG000465.
- 573 Plattner, A.M., 2020. Gprpy: Open-source ground-penetrating radar processing and visualization software. *The Leading Edge* 39, 332–337.
574 doi:10.1190/tle39050332.1.
- 575 Ringler, A.T., Evans, J.R., 2015. A quick seed tutorial. *Seismological Research Letters* 86, 1717–1725. doi:10.1785/0220150043.
- 576 Romero-Ruiz, A., Linde, N., Keller, T., Or, D., 2018. A review of geophysical methods for soil structure characterization. *Reviews of Geophysics*
577 56, 672–697. doi:10.1029/2018RG000611.
- 578 Rücker, C., Günther, T., Wagner, F.M., 2017. pygimli: An open-source library for modelling and inversion in geophysics. *Computers & Geosciences*
579 109, 106–123. doi:10.1016/j.cageo.2017.07.011.
- 580 Steiner, M., Katona, T., Fellner, J., Flores Orozco, A., 2022. Quantitative water content estimation in landfills through joint inversion of seismic re-
581 fraction and electrical resistivity data considering surface conduction. *Waste Management* 149, 21–32. URL: <https://www.sciencedirect.com/science/article/pii/S0956053X22002793>, doi:<https://doi.org/10.1016/j.wasman.2022.05.020>.
- 582 Steiner, M., Wagner, F.M., Maierhofer, T., Schöner, W., Flores Orozco, A., 2021. Improved estimation of ice and water contents in alpine permafrost
583 through constrained petrophysical joint inversion: The hoher sonnblick case study. *Geophysics* 86, 1–84. doi:10.1190/geo2020-0592.1.
- 584 Stockwell, J.W., 1999. The cwp/su: Seismic unix package. *Computers & Geosciences* 25, 415–419. URL: <https://www.sciencedirect.com/science/article/pii/S0098300498001459>, doi:10.1016/S0098-3004(98)00145-9.
- 585 Sullivan, C.B., Kaszynski, A., 2019. PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK).
586 *Journal of Open Source Software* 4, 1450. URL: <https://doi.org/10.21105/joss.01450>, doi:10.21105/joss.01450.
- 587 Uieda, L., Oliveira Jr, V.C., Barbosa, V.C., 2013. Modeling the earth with fatiando a terra, in: *Proceedings of the 12th Python in Science Conference*,
588 pp. 96–103.

591 List of Figures

592 1	Fundamental use case illustrating the modeling of seismic refraction data within the formikoj ecosystem.	22
593 2	Fundamental use case illustrating the processing of seismic refraction data within the formikoj ecosystem.	23
594 3	Typical formikoj workflow for the picking of first break traveltimes from seismic waveform data.	24
595 4	Synthetic seismic data generated with the SeismicWaveformModeler:	
596 (a)	Two-layered subsurface model without topography characterized by a distinct anomaly in the center. Triangles along the surface indicate the positions of geophones.	
597 (b)	Synthetic seismic waveform data computed from the subsurface model for a shot point (star-shaped symbol) located in the center of the profile.	
598 (c)	Pseudosection illustrating the apparent velocity computed from the seismic travel times based on the survey geometry.	
600 (d)	Subsurface model of the seismic P-wave velocity distribution obtained through the inversion of the synthetic P-wave travel times.	25
604 5	Entity-relationship diagram illustrating the SQLite database used in a SeismicRefractionManager project to store and manage processing related information.	26
606 6	The SeismicRefractionManager addresses the stations through consecutive station numbers based on the sort order in the geometry file. The shot index numbers (SIN) and receiver index numbers (RIN) are assigned to the shot and receiver stations separately to allow for an intuitive data handling for the user.	27
610 7	The frequency spectrum illustrates the frequency content of the currently selected traces, which allows for the identification of frequency ranges associated to noise, which can be omitted through the application of corresponding frequency filtering.	28
613 8	The seismogram plot presents the currently selected traces along the x axis, where the sort order is determined through the geometry. The corresponding trace data are illustrated as a function of time along the y axis (solid curves), with positive and negative amplitudes depicted in blue and red, respectively. The selection criterion and the applied filter are shown in the upper left corner of the plot. Green crosses refer to the picked traveltimes stored in the currently active pickset.	29
618 9	The status bar in the interactive seismogram plot displays the currently active processing and data scaling modes as well as the time (in seconds) at the current cursor position. By pressing the keys 'a', 'm', 'r', 'v' on the keyboard the different modes can be activated.	30
621 10	The traveltime diagram shows the first break traveltimes stored in the currently active pickset along the y axis (x symbols), where solid lines connect traveltimes assigned to a common shot. The sort order of the stations along the x axis reflects the geometry. Filled circles indicate stations with co-located shot and geophone (receiver), triangles refer to receiver stations (no shot) and stars refer to shot stations (no geophone).	31
626 11	The soda lakes field dataset was collected in a 3D survey layout with stations (co-located geophones and shots indicated by filled dots) deployed in form of a cross. Additional shots (yellow stars) were conducted in front of a cross rotated by 45° to increase the coverage of the dataset.	32
629 12	Picking percentage for each shot in the Danube island roll-along dataset. Low values in the first third of the dataset indicate a low signal-to-noise ratio in the seismograms hindering the picking of first break traveltimes for each shot-receiver pair.	33
632 13	Exemplary seismic waveforms from the soda lakes dataset shown for shot index number (SIN) 1 with a 100 Hz lowpass filter applied to suppress high frequency noise. The recorded seismic waveforms clearly reflect the geometry of the geophones with RIN 1 to 48 deployed along the direction of wave propagation, while RIN 49 to 96 are deployed perpendicular to the propagating wavefront.	34
636 14	3D pseudosection showing the apparent seismic velocities determined from the first break traveltimes obtained from the soda lakes dataset and the corresponding absolute offset between the shot and receiver stations. The apparent velocity for each shot-receiver pair is illustrated at the corresponding 2D midpoint and pseudodepth (1/3 of the absolute offset), thus yielding a 3D representation.	35

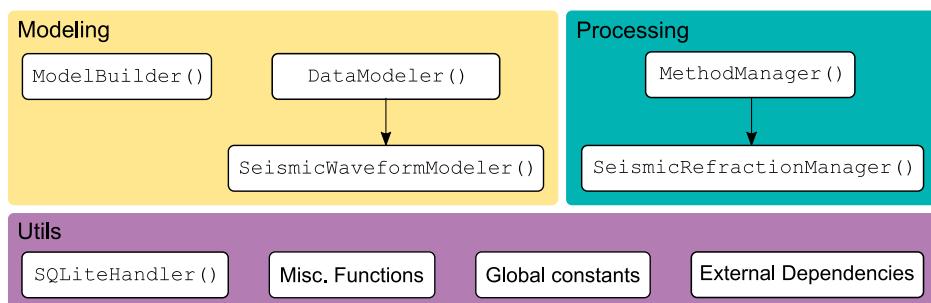


Figure 1: Fundamental use case illustrating the modeling of seismic refraction data within the formikoj ecosystem.

formikoj - Flexible geophysical data processing

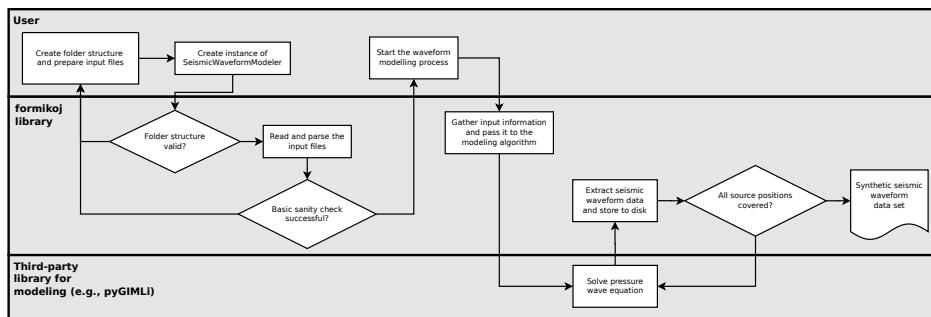


Figure 2: Fundamental use case illustrating the processing of seismic refraction data within the formikoj ecosystem.

formikoj - Flexible geophysical data processing

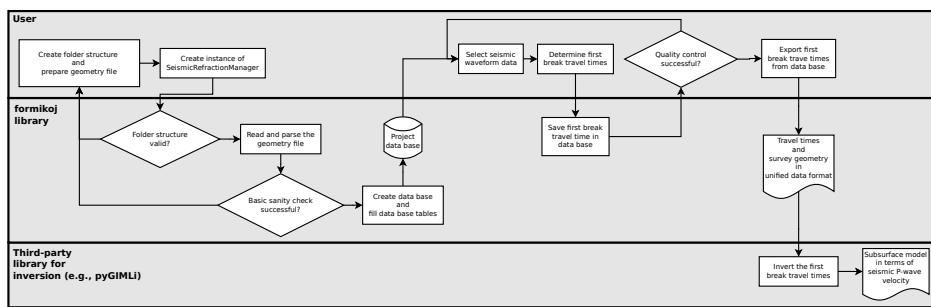


Figure 3: Typical formikoj workflow for the picking of first break traveltimes from seismic waveform data.

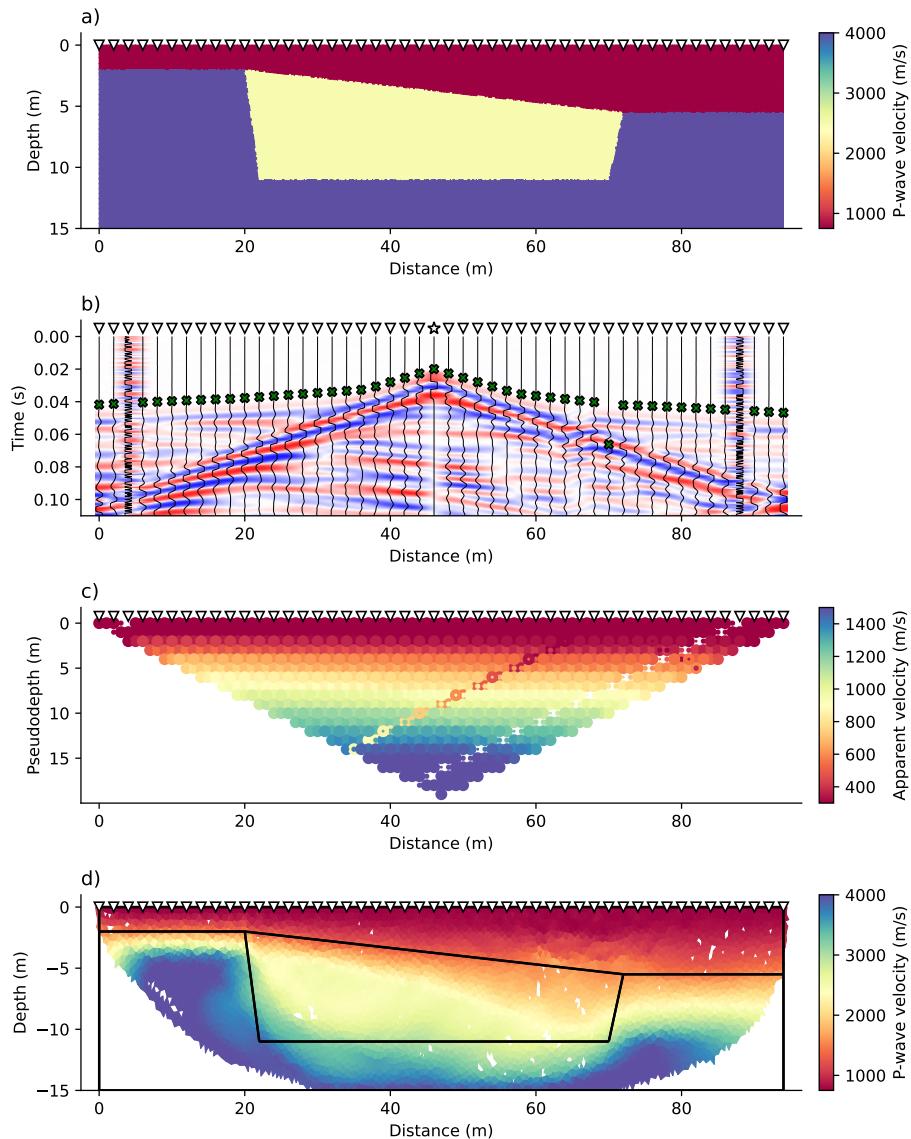


Figure 4: Synthetic seismic data generated with the `SeismicWaveformModeler`:

- (a) Two-layered subsurface model without topography characterized by a distinct anomaly in the center. Triangles along the surface indicate the positions of geophones.
- (b) Synthetic seismic waveform data computed from the subsurface model for a shot point (star-shaped symbol) located in the center of the profile.
- (c) Pseudosection illustrating the apparent velocity computed from the seismic travel times based on the survey geometry.
- (d) Subsurface model of the seismic P-wave velocity distribution obtained through the inversion of the synthetic P-wave travel times.

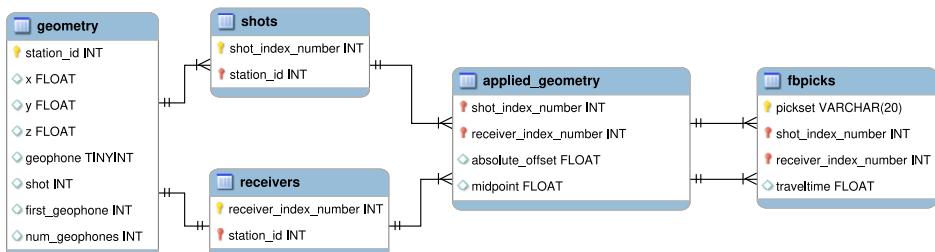


Figure 5: Entity-relationship diagram illustrating the SQLite database used in a SeismicRefractionManager project to store and manage processing related information.

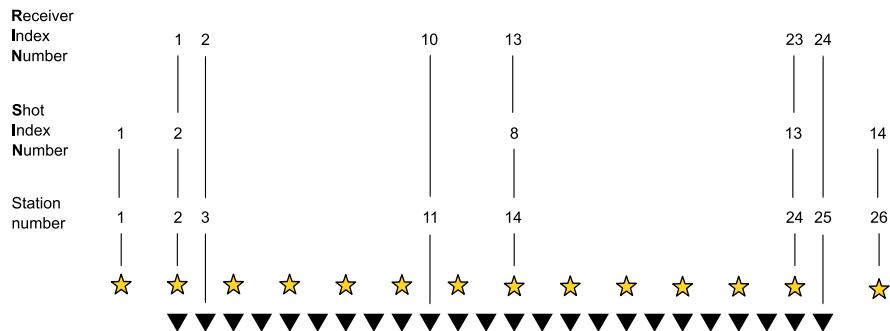


Figure 6: The SeismicRefractionManager addresses the stations through consecutive station numbers based on the sort order in the geometry file. The shot index numbers (SIN) and receiver index numbers (RIN) are assigned to the shot and receiver stations separately to allow for an intuitive data handling for the user.

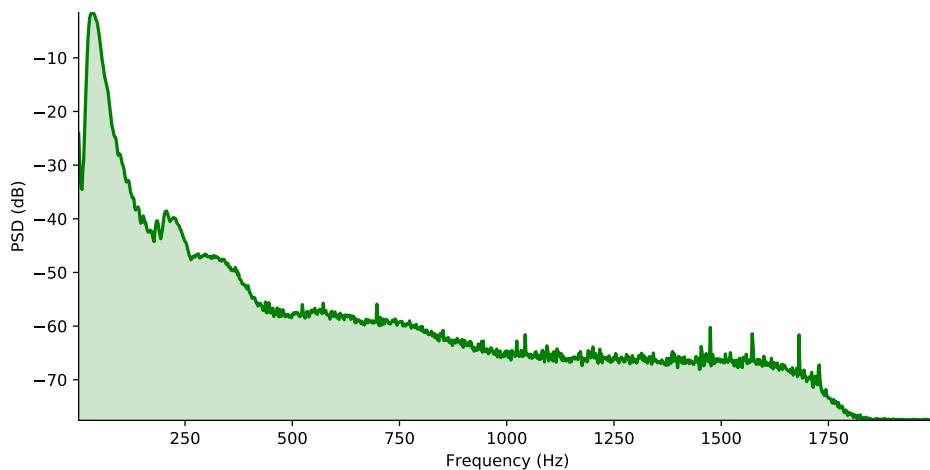


Figure 7: The frequency spectrum illustrates the frequency content of the currently selected traces, which allows for the identification of frequency ranges associated to noise, which can be omitted through the application of corresponding frequency filtering.

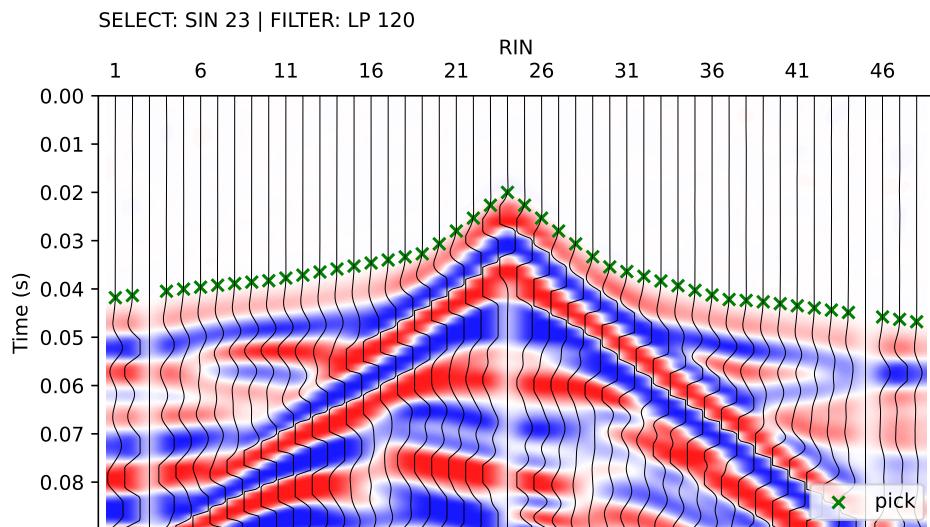


Figure 8: The seismogram plot presents the currently selected traces along the x axis, where the sort order is determined through the geometry. The corresponding trace data are illustrated as a function of time along the y axis (solid curves), with positive and negative amplitudes depicted in blue and red, respectively. The selection criterion and the applied filter are shown in the upper left corner of the plot. Green crosses refer to the picked traveltimes stored in the currently active pickset.

Proc: Fb pick | Scroll: Zoom | Time (s) = 0.000



Figure 9: The status bar in the interactive seismogram plot displays the currently active processing and data scaling modes as well as the time (in seconds) at the current cursor position. By pressing the keys 'a', 'm', 'r', 'v' on the keyboard the different modes can be activated.

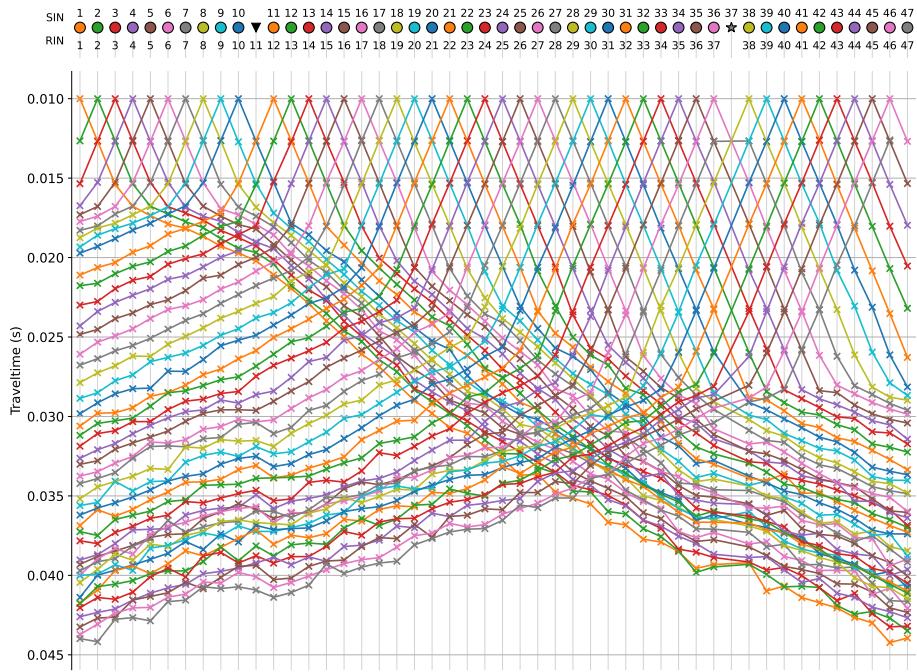


Figure 10: The travelttime diagram shows the first break traveltimes stored in the currently active pickset along the y axis (x symbols), where solid lines connect traveltimes assigned to a common shot. The sort order of the stations along the x axis reflects the geometry. Filled circles indicate stations with co-located shot and geophone (receiver), triangles refer to receiver stations (no shot) and stars refer to shot stations (no geophone).

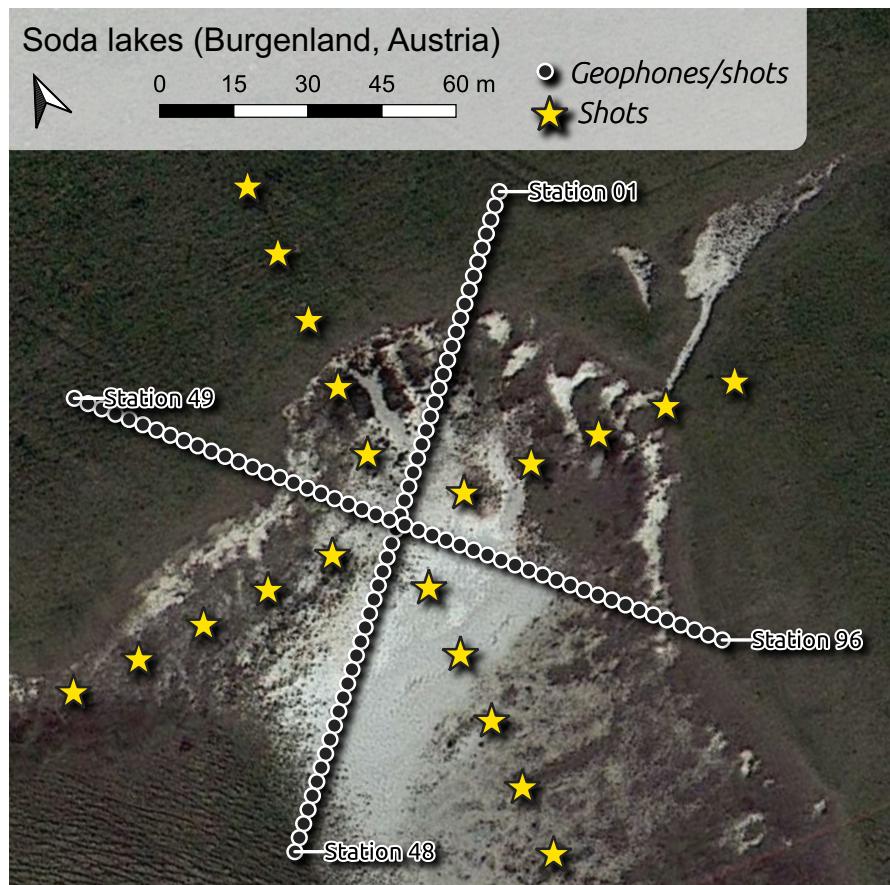


Figure 11: The soda lakes field dataset was collected in a 3D survey layout with stations (co-located geophones and shots indicated by filled dots) deployed in form of a cross. Additional shots (yellow stars) were conducted in front of a cross rotated by 45° to increase the coverage of the dataset.

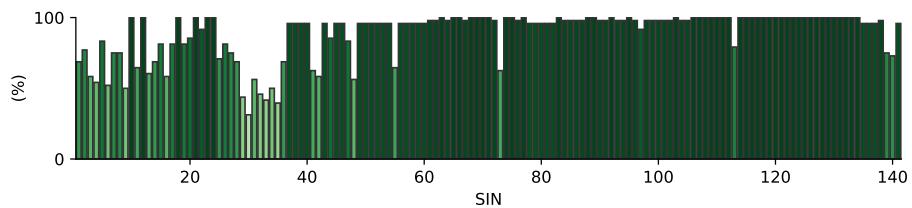


Figure 12: Picking percentage for each shot in the Danube island roll-along dataset. Low values in the first third of the dataset indicate a low signal-to-noise ratio in the seismograms hindering the picking of first break traveltimes for each shot-receiver pair.

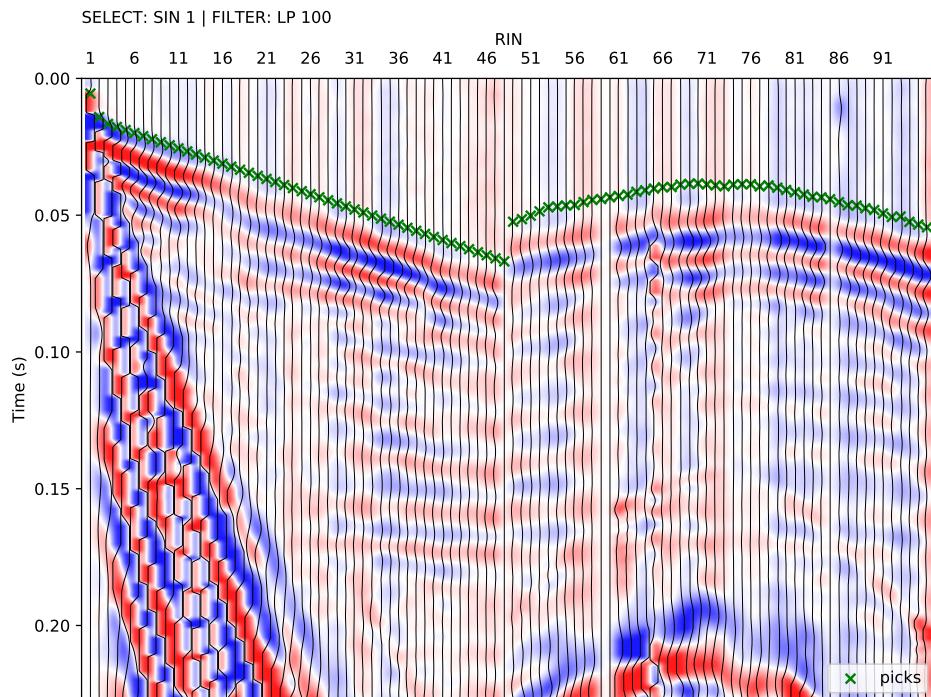


Figure 13: Exemplary seismic waveforms from the soda lakes dataset shown for shot index number (SIN) 1 with a 100 Hz lowpass filter applied to suppress high frequency noise. The recorded seismic waveforms clearly reflect the geometry of the geophones with RIN 1 to 48 deployed along the direction of wave propagation, while RIN 49 to 96 are deployed perpendicular to the propagating wavefront.

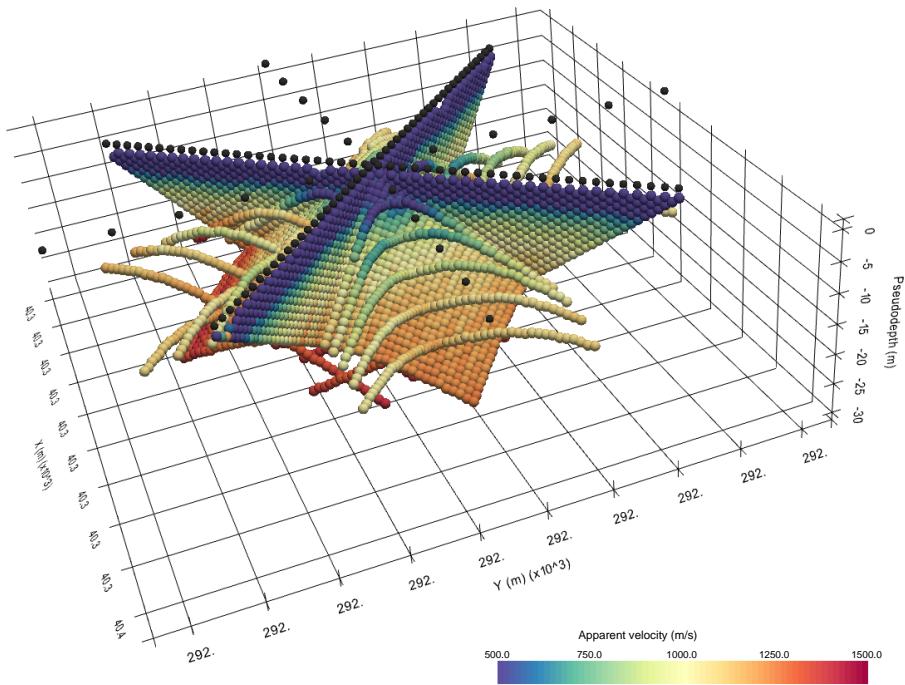


Figure 14: 3D pseudosection showing the apparent seismic velocities determined from the first break traveltimes obtained from the soda lakes dataset and the corresponding absolute offset between the shot and receiver stations. The apparent velocity for each shot-receiver pair is illustrated at the corresponding 2D midpoint and pseudodepth (1/3 of the absolute offset), thus yielding a 3D representation.