

XTomo-LM

2D Seismic Tomography System
with Tools for Layered Model Study

Version 3.4.1

User Guide

XGeo Ltd.
St Petersburg
September 2020

XTomo-LM: 2D Seismic Tomography System with Tools for Layered Model Study. Version 3.4.1 (September 2020). User Guide.

Publisher: *XGeo Ltd.*

Author: *Alexander Vinnik*

Table of Contents

What's new in Version 3.4?	1
What's new in Version 3.3?	2
Wat's new in Release 3.2.2	3
Wat's new in Version 3.2?	4
Wat's new in Version 3.1?	8
What's New in Version 3?	10
Introduction	14
1 Kinematic Interpretation	14
2 XTomo-LM 3: Overview	17
3 Waves	18
4 Model	20
5 Data	22
6 Numbers	25
7 Programs	26
Environment	28
1 Workflow	28
2 Project Manager	28
3 Creating Projects	30
4 Project support	34
5 Processing Tree	35
6 Wave Manager	41

7	Velocity Color Spectrum	43
8	Packages. Archives. XTomo-LM 2	47
	Model	49
1	Imaging a model	49
2	Zoom and Selector	52
3	Export	54
4	Editing Model	57
5	Changing Grid Thickness	58
6	Changing Velocity	59
7	Changing Geometry	63
	Observations	68
1	Overview	68
2	SRT Port: Manager	69
3	SRT Port: Import	71
4	SRT Port: TX-curves	73
5	SRT Port: Arhives. Export	77
6	Ray Catalog: Populating	78
7	Ray Catalog: Viewing	81
8	Ray Catalog: M-projects	85
	Forward Problem	91
1	Overview	91
2	The Forward Problem Solver Module	91
3	Ray Picture	95

4	Ray Samples. Creation and Saving	98
5	Studying Time Residuals in I-project	101
6	Ray Coverage Density	103
7	Preparing to Tomography	104
8	Export	105
	Tomography	107
1	Theory	107
2	Inverse Problem Solver	110
3	Viewing solution	112
4	Termination of interpretation cycle	114
	TX-Curve Inversion	115
1	Overview	115
2	Building Initial Velocity	115
3	Building horizons	120
4	Selecting TX-curves	122
5	Building Reflectors	125
6	Building Refractors	129
7	Examples	131
	Utilities	136
1	Comparing velocities	136
2	Model Geometry	139
3	Static Corrections	145
	Appendix	147

1	Export from XTomo LM: Summary	147
2	Basic ASCII File Formats	148
3	Other formats	150

What's new in Version 3.4?

Version 3.4 contains only one new feature, which is of prime importance for entire XGeo product line. This is a bidirectional interface between XTomo-LM and XMF – a platform for manual fitting of layered models. Simultaneously published version 1.1 of XMF contains the second channel of the interface. It turns the entire line DPU—XTomo-LM—XMF into a powerful means of two-dimensional kinematic interpretation capable of processing data with different level of completeness and quality (rather high level is implicitly suggested by XTomo-LM inversion algorithms). From now on, a project initiated in XTomo-LM can be smoothly transformed into an XMF project for further development, and vice a versa. That makes it possible to combine exact XTomo-LM methods with manual fitting of those model components that are poorly lighted with data.

In the updated documentation, the user should read section [Kinematic interpretation/Extensions](#), containing necessary links, and section [Creating Projects/Starting model](#).

What's new in Version 3.3?

Version 3.3 contains new options for interpretation of non-line observations. This type of observations is called *Other 2D* in project properties. In previous versions, they were required to be generated by diving wave propagating in gradient velocity model only. Now these restrictions are lifted, so that the user can solve forward problem and run tomographic inversion for any source-receiver configuration with reflectors or/and refractors in model or, shortly, for layered model. Only one condition must be satisfied: sources and receivers are located above the horizon in study. For example, part of them (or all) may sit in one or more boreholes, if the problem stay two-dimensional. It is just the case of borehole observations that motivated the last update. Clear, that TX-curve inversion tools worked out for line observation are not applicable in that case and stay inaccessible in the user interface.

Wat's new in Release 3.2.2

The release is aimed, mainly, at eliminating minor errors and bugs having been discovered since the first release of version 2.2 in September 2017. Additionally, a few small amendments have been introduced. Some of them are listed below.

1. When comparing two velocity distributions, the user can examine both difference maps and vertical profiles. The latter feature is a novel. Now it is used by the Inverse Problem Solution Viewer module and Velocity Comparator utility (UVC).
2. UVC. The user interface is simplified. Now the utility can be applied to any velocities, not only close to each other and non-constant as in the previous release. Maps of absolute and relative velocity differences can be viewed.
3. Velocity Spectrum Manager. The module is provided with a new tool for rapid building of velocity color spectrum. It allows creating much broader set of spectra.
4. Program Manager. From now on, the user can run a persistent comment for each project. The comment dialog is activated by the main menu command *Project/User Notes*. The beginning of the comment can be shown as a hint when the cursor is over the project in the project list. To activate the option, use the *Working Folder/Show Notes as Hint* command.

Wat's new in Version 3.2?

Essential changes and novelties in version 3.2. pertain to (1) viewing and editing a model; (2) system's framework; and (3) traveltimes curve inversion. In those three areas, functionality has been expanded or efficiency increased and the user interface improved. All bugs and errors discovered have been eliminated or corrected.

1. Comparing velocity distributions

The new utility Velocity Comparator allows comparing two-dimensional velocity distributions stored in any two m-nodes of one or different XTomo-LM 3 projects. Velocity functions are compared at nodes of an orthogonal grid on a rectangle domain common for both models. Result of comparison is represented as a color map and in numeric form. The utility lets the user track velocity changes in the course of interpretation.

2. Editing velocity

2.1. In-place Velocity Editor. To change velocity in a selected domain, one invokes the *Editing Velocity* dialog with a number of editing options. In version 3.2, a new option is added to the list. It enables the user to define velocity in a vertical strip as the power function of depth. Such approximation is often used in modeling. For editing intricate velocity distributions, the dialog is sometimes quite inconvenient. For example, if one needs to modify velocity in a set of isolated blocks of cells, one has to invoke the dialog for each block using the only option of all its machinery. To make the work easier, the in-place velocity editor has been introduced. It looks like a small window floating over the model image. With its controls, one can perform basic editing operations and switch freely to the main window for magnifying the image, making a new selection or viewing velocity range and its average in a selected area (this is a new feature) and so on. Another new option is selecting a subgrid in a dialog by specifying the bounding columns and rows.

2.2. Replacing velocity. After grid deformation, velocity stays attached to grid cells, not to depth. To restore velocity-depth dependence (if necessary), one has to apply the *Replace Velocity* command. It replaces the current velocity function with the one from another m-node in which the grid has the same verticals. In version 3.2, the operation has been extended. First, restrictions on the source grid are lifted. Second, the source velocity function can be taken from a VC file. Such file stores a velocity column set. In both cases the source velocity is redefined at nodes of the current grid. This feature can be applied, in particular, when initial velocity is computed by inversion of diving wave traveltimes curves. The inversion result is stored just in a VC file. In the previous version, this can be used only for creation the starting model of a *new* project. Now it can be used as a source for replacing velocity in any m-node of the current project.

2.3. Copying velocity. In version 3.2, one can copy velocity distribution from a selected model subset to another subset of the same form. One points to the destination by double-clicking a cell that unambiguously defines the target subgrid. The details can be found in [Changing velocity](#).

2.4. Eliminating "black cells". A grid cell is painted black when its velocity is beyond the range stored as a project property in Project Manager. When velocity is edited in Model Editor, its values going beyond the range are automatically replaced by the nearest range limit. But if velocity is borrowed from an external source (as in 2.2), or after tomographic refinement, black cells may appear. Black cells signal data error to the user, so they must be removed. Model editor offers a new command to do that.

2.5. Velocity profiles. When vertical or horizontal velocity profile was plotted, the piecewise-linear velocity representation of $V(*, z)$ or $V(x, *)$ was used. Now one can choose between piecewise-linear representation and step function representation. The context menu of the profile drawing pad is used as a switch. Along with vertical velocity profile, any velocity column permits two ways of interpretation when it is input into the system. In all tasks where model velocity is formed by import of velocity columns, the user can select how to treat the columns: as step functions or as piecewise linear functions. (Step function models a layer-cake on a half-space, while piecewise linear function defines continuous velocity distribution in a layer).

3. Editing model geometry

XTomo-LM works with curvilinear grids. Changing grid geometry is carried out by modifying shape of one or several h-lines, typically those representing seismic horizons. A set of curves defining model geometry is called *model wireframe*. Model wireframe is stored in a text file of the MG format. When a project is created, its starting model can be created by import of a model wireframe, but there were no adequate tool for creating and editing wireframes in versions 2 and 3, though such tool would be an important means of modeling. The new utility MG File Editor or UMG, for short, has been added to XTomo-LM 3.2 to deal with all issues of creating, editing and storing wireframes, in particular, grid h-lines. The utility is launched from the Project Manager's *Tools* menu.

4. Model Editor's user interface

Despite expanded functionality, Model Editor's user interface has been significantly simplified. Commands related to model top line and seismic horizons are excluded from module's menus. Now, an object of an operation is determined automatically. Model Editor runs UMG in a specific mode, as H-Line Editor. Thus, the unified user interface for dealing with model geometry is provided.

5. Environment

5.1. Creating projects. Now this important operation is in competence of the New Project Creator module (NPC) which is launched by Project manager. The main NPC window is designed in the way to minimize difference from the old dialog. However, the second tab differs considerably to reflect the new conception of numeric representation explained in the next paragraph.

5.2. Spatial resolution and numeric formats. In the new version, approach to computational accuracy is essentially simplified. Spatial resolution of objects is defined by size of starting model and bit width of real number representation in a computer. Resolution defines accuracy of decimal approximation of real numbers and, hence, numeric formats which are offered as the defaults. Notions of computational and physical resolutions are no longer in use. Old XTomo-LM 3 projects are smoothly included in the new conception.

5.3. Viewing models of other projects. In previous versions, one could simultaneously view models in any m-nodes of the currently opened project. Starting from version 3.2, the same is true for m-nodes of any project in the current working folder. To view a model from a foreign project, one applies the new *View Model* command of the project list menu. Comparison of models and velocities (see section 1) is especially important for projects created for the same field data to implement alternative approaches to interpretation.

5.4. Copying of an o-node of Processing Tree. In new version, the two-step copy-paste operation of copying the content of an o-node can be performed much easier – by dragging the o-node and dropping it on a target m-node.

A very special and rare error has been discovered in implementation of the operation. We have to go into some detail. Suppose that model has a curved top line T . Let an o-node O_1 created for model M_1 is to be copied to m-node M_2 . Suppose that receiver R is located at the cross point of T with vertical V . Let in model M_2 vertical V is deleted before copying O_1 to M_2 . Then, after copying, R may prove to be hanging well above T . Indeed, if V' and V'' are the neighboring verticals, then the new T segment between V' and V'' can lie either below or above the old segment $[V', V, V'']$ depending on the T shape. If R is above T , then there will be no rays coming to T in the forward problem solution. In the new version, this error is corrected: all "hanging" sources or receivers are landed on T , if the user does not mind. If he or she does, copying is canceled.

5.5. Active module list. This list is withdrawn from the Project Manager main window. Now it is invoked as a separate window by a command of the *Tools* menu. It contains all currently running modules that are launched directly by Project Manager (not those started by other modules). Description of modules has been improved, functionality kept. In particular, one can close any active module or abort it, if it is hung. The change is caused by technical necessity.

5.6. New use of SRT Port. Now SRT port can be used for observation data exchange between different project. The new Processing Tree menu command copies Ray Catalog content from an o- or f-node to a new SRT Port database. Then it can be used as a new data source. In the new version, SRT port is two-level data warehouse: it contains a set of data stores in which databases lies. The existing databases are placed in special data store named *.Main*. New data stores are created by the user when necessary. In previous version, the user could select a port database of any type as observation source for an M-project. That was not correct: in an M-project, the user is allowed to select only from databases of SR type.

6. Reflection and refraction TX-curve inversion

The improved inversion algorithms are used in version 3.2. They are more robust and efficient. A new module for crude estimation of reflector position at source points is added. Examples of building horizons are now supplied as a set of M- and I-projects (see next section).

7. Sample projects and documentation

The product is supplied with the sample projects. Their aim is to demonstrate work with XTomo-LM in general and in getting solution to some problems of layer model interpretation, in particular. The sample projects use systematically some of the new tools added in version 3.2. Processing Tree nodes of each sample project contain explaining comments. Information on installation and use of the sample projects and also tasks for which they are created can be found in the documentation (section Examples of the chapter *TX-curve inversion*).

The documentation describes the software of version 3.2 with new modules and last changes included. In the course of development, some of the base XTomo-LM conceptions have been refined, so the user, even advanced, is strongly recommended to look through the chapters: *Introduction*, *Model*, *TX-curve Inversion*, *Utilities* and also topics *Creating projects*, *Processing tree*, *Ray Catalog: Populating*. The chapter *TX-curve Inversion* is written anew to match last versions of the inversion algorithms.

Wat's new in Version 3.1?

New XTomo-LM version includes two new important features and many minor changes optimizing the user interface.

SRT-Port

The first feature relates to the system architecture. The new component *SRT Port* is now the only point of input and verification of ASCII files with descriptions of observation systems – *observation files*. The following file formats are permitted: SRT, #DT, SR and S+R. The so called "SRT Utilities" are excluded from XTomo-LM as well as the temporary SRT database which was overwritten at every other use. Instead, SRT Port supports the persistent data warehouse in which the verified content of each observation file once imported into the system is stored in internal format ready for repeated use. A file content forms a warehouse storage unit called *SRT database* and identified with a unique name. The utility functions, often overlapping, are included in the SRT port maintenance software presented to the user by the SRT Port Manager module. It performs observation file import and verification and implements necessary maintenance tasks.

SRT Port is an autonomous component in the sense that it does not refers or relates to common folders or projects. It is targeted at isolating technicalities of external file import from the main workflow. Now the workflow does not include import of observation files. Instead, the new operation of extracting data from SRT Port is to be executed on an o-node of Processing Tree. Extracting is not plain copying an SRT database. It means *sampling* data from a database using a reasonable criterion. It is important, because different inverse and modeling tasks often use different subsets of an observation database. Earlier, the user was compelled to manually edit observation files to adjust them to specific problems. Additionally, an annoying problem of adjusting observation data to the project's requirements is, practically, withdrawn. The requirements mentioned concern spatial resolution for sources and receivers and falling strictly inside the model domain.

Description of the new component can be found in first five sections of the chapter [Observations](#).

Inverse problem for diving wave TX-curves

The second novelty concerns the problem of inversion of diving wave traveltime curves to get the initial velocity section for seismic tomography. The Primary Model Builder utility is excluded from XTomo-LM. It was based on the "direct inversion" technique of Herglotz-Wiechert-Chibisov, which imposes too severe condition on TX-curves – strict convexity. Three facts make the condition impracticable: (1) TX-curve curvature always tends to zero with growing offset; (2) possible inverse layers in the medium; and (3) measurement noise. Now XTomo-LM uses well-posed inverse problem and a robust numeric algorithm for getting the solution. Applied to a diving wave

TX-curve set, it allows building a good initial approximation to velocity section by combining the solutions of one-dimensional problems. See [TX-curve Inversion/Building initial velocity](#) section.

Other changes

Now both in the user interface and in documentation, two types of inverse problems are distinguished: tomography and TX-curve inversion. In particular, in the main operating menu of Processing Tree, all commands related to TX-curve inversion are collected in one submenu. A new term [starting model](#) is introduced for a model which is defined at project creation time. Another minor change to be mentioned is the new [common folder](#) called: *common import-export folder*. It is a container for projects' import-export folders and must be defined once before creating a new project. At the same time that does not affect the accessibility of existing data.

What's New in Version 3?

Since the first XTomo-LM release, the software have been widely used as one of or the only means of interpretation in various seismic surveys from WARP in the Arctic Seas to CMP profiling (near-surface layer study), to crosshole tomography, to different kinds of geological engineering investigations. Original way of model representation and ray tracing algorithm make XTomo-LM a convenient and scalable tool for different problems. Version 3 prolongs the software life cycle adjusting it to the present hard- and software environment (multi-core processors and latest Windows versions) with full use of its advantages.

Releases 3.0.X with $X > 1$ present XTomo-LM 3 in full (the preliminary Release 3.0.1 was issued for special purpose). The Version 3 GUI design is, in general, kept as well as basic functionality: project support, model editing, forward problem, tomography inversion, building seismic horizons. However, the implementation of most base features and auxiliary services is absolutely new. It is targeted at:

- full compatibility with the latest Windows versions;
- using new data storage architecture;
- boost of productivity;
- introducing new effective algorithms of solving basic problems;
- more comfortable user interface.

Below only main new features are listed.

1. New data storage architecture

Data storage concept was a weak spot of Version 2, and it was responsible (due to using an obsolete 3d-party software) for low productivity of ray-tracing and problems of use with late Windows versions. The concept was thoroughly revised, and new data storage architecture not only eliminated the named shortcomings but provided general boost of productivity of "heavy" computations and visualizing of large amounts of stored data.

2. Efficiency of Ray-tracing

In version 3, solving kinematic forward problem is organized in a quite a new way and is partly parallelized. The new algorithm takes into account the multi-core architecture of Intel processors and is scalable, which means that the productivity (computation time) depends on the number of logical processor in the system. In the following example rays of diving, reflected and head waves were traced on the grid of $896 \times 160 = 143360$ cells. The observation scheme consisted of 101 sources and 201 receivers with the total number of rays 60903. The below table shows computation time T (min:sec) for N logical processors.

N	1	2	4	8
T	38:27	32:30	17:11	9:01

(The work station had Intel Core i7 3.4 GHz with 8 logical processors). CPU was 95–100% loaded for almost all the time. T value at N = 1 corresponds approximately to the computation time required by version 2 of XTomo-LM.

3. Editing model

The set of tools for editing model was not optimal in previous versions, while their use was sometimes quite convenient. In the new version functionality of Model Editor is directly stipulated by the concept of [model representation](#). Model Editor (MED) allows changing (1) grid thickness; (2) velocity values; and (3) model geometry. The tasks (1) and (2) are implemented basically in the same way as in version 2. The new feature of explicit velocity smoothing is added. Model geometry is defined by shape of h-lines. A powerful curve editing tool is added to MED for in-place h-line modification. It is fast and handy. Remeshing caused by h-line editing is performed momentarily without participation of the user. The same relates to importing an h-line from an ASCII file. Editing model geometry is now simple and fast operation. MED supports "undo/redo" stack to cancel/return a sequence of last changes.

4. New algorithms for reflection and refraction traveltimes inversion

Though building of seismic horizons was present in version 2, the inversion tools were no more than tentative. The user was responsible for making up a horizon curve from so called feasible point set. Version 3 offers new high-performance algorithms requiring from the user minimal a priori information. The result is a horizon curve whose points are grid nodes; it can be immediately viewed on the model image for estimation of relevancy. The accuracy, naturally, depends on grid thickness and regularity properties of time-distance curves and velocity distribution. Similar to the new forward problem implementation, the algorithms are performed in parallel by a set of logical processes available in the system. Building seismic boundaries, together with forward problem and tomography inversion, has turned into a powerful inversion tool. The documentation includes examples of applying the new technique which also emphasizes advantages of using curvilinear grids for representing layered models.

5. Tomography on Ray Samples

In many problems (deep seismic investigations, in the first place) the interpreter deals with observations of different waves in wide offset range for layered model study. Forward problem solution for such model contains rays with different computational errors depending on ray length. To use such mix as input data for tomography inversion is not a good idea. The more so, that different ray sets are used for different layers. For example, one can pick reflections with small offsets for the first layer; refractions or overcritical reflections – for the deep layer. To help the user, XTomo-LM 3 allows sampling rays from forward problem solution for use as input data

for inversion. Sampling criterion includes restrictions on sources, receivers, waves and offsets, time residuals. This is a technical, but a very important moment.

6. Converted waves modeling

In version 3 modeling of converted waves obtained additional support. First, a wide range of numeric codes is reserved for converted waves to extend the old wave encoding system. Second, both incident and reflected wave is allowed to undergo conversion on each interface. At that, each interface is characterized with its conversion coefficient, while each converted wave is described with its set of conversion flags (a flag is set for interfaces on which conversion exists).

7. Forward Problem Solution Analysis

Important modifications are introduced in Forward Problem Solution Viewer (FPV). TX-curve plot can be displayed either docked to model-rays plot window or in a separate window. Special attention was paid to visualizing of large amounts of rays (tenths of thousand). In particular, process of rendering can be interrupted. It is possible to output ray coverage colored maps and export them to graphic files. Operation of freezing velocity in a subgrid (needed for layer-by-layer study) is made more convenient. One can fix velocity in cells with given ray coverage value.

8. Verification of SRT Files

SRT file contain input data for an inversion project. If SRT file is prepared manually or by irrelevant software, it can be erroneous in many ways, especially with respect to space resolution defined in a project. In previous versions, SRT file content was checked at import time up to the first error occurred. In the new version, the SRT file to be imported is undergone a thorough check. Information of all errors is logged and can be saved to a user file for later study. Similarly, SRT file preprocessing begins with the new SRT Checker utility producing detailed log and capable of saving SRT data to temporary database for other utilities. For profiling data, there is a new utility representing SRT data as a set of time-distance curves for viewing and editing. In particular, an external SRT file can be divided into several such files for specific problems by operating on the TX-curve image from the initial file.

9. Packages and Archives

Consider the situation when two users work on the same data at different remote sites and wish to exchange intermediate or final results. The XTomo-LM 3 users can use *packages* or *archives* for that aim. User 1 selects a node of Processing Tree in Project Manager (PM). On a command from the popup menu, PM packs the data from the part of tree branch (Model to Selected node) into a single compressed file called a package. Besides the data, the package contains the entire project infrastructure. PM puts the package into the predefined folder. The file can be delivered to User 2 in any way. On receipt, User 2 puts the package into the known folder, runs PM and selects a command in its main menu. PM creates a new project with Processing Tree containing part of the branch from the package. User 2 gets exactly the same data views as User 1. Unlike package, an archive is used for packing the entire project. Creating and unpacking archives is also Project Manager's task.

10. Version 2 and Version 3

Because data structures and formats have been thoroughly changed, XTomo-LM 3 cannot open a project created in version 2. However, the input data from any step of processing (i.e. data of Model and Observations nodes) can be converted into a new XTomo-LM 3 project. This operation is carried out by Project Manager.

XTomo-LM 3 can be installed not instead of but side by side with the version 2. XTomo-LM 3 can work with Data Preparation Unit 2 (DPU 2), but displaying travel times computed by XTomo-LM 3 on seismograms can be carried out only with DPU 3.

Introduction

1 Kinematic Interpretation

[Terminology and Problems](#) – [Formal Description](#) – [Tomography Inverse Problem](#) – [Successive Iterations](#) – [Traveltime Curve Inversion](#) – [Extension](#).

Terminology and Problems

Kinematic interpretation belongs to a class of problems rising in planning of physical experiment and interpretation of experimental data. In our case, experiment means field seismic observations; data include traveltimes of observed seismic waves and positional information; interpretation means estimation of 2D model's parameters; classes of permissible models are gradient velocity model and layered model with nonintersecting interfaces. Both models are supposed to be isotropic. The first model is described with velocity function $V(x, z)$, the second model, additionally, includes equations of interfaces. The universal approach to such problems is modeling, in our case – kinematic modeling of seismic wave propagation in isotropic medium. The modeling problem or *forward kinematic problem* assumes tracing rays of seismic waves for given model and source-receiver configuration and evaluation of traveltimes. Kinematic interpretation consists in finding a model that minimizes residuals between observed and computed traveltimes. The basic instrument of fitting model is software tools for solving forward problem. The other common terms for kinematic interpretation are: solving *inverse kinematic problem*, *arrival time analysis/inversion*. The term *kinematic interpretation* is used mostly in Russian literature.

The general inverse problem for the layered model in its strict statement is not considered. However, XTomo-LM means of solving inverse problems for gradient velocity model and building seismic horizons, together with the principle of layer-by-layer interpretation, make it possible to study layered models, at least, using informal, empirical approach.

Formal Description

For brevity, consider the case of gradient velocity model. Let \mathbf{T}^* be an n -dimensional vector of observed times, $\mathbf{F}: \mathbf{V} \rightarrow \mathbf{T}$ is forward problem operator. Then the problem of kinematic interpretation can be expressed as

$$(1) \quad \|\mathbf{T}^* - \mathbf{F}(\mathbf{V})\|_n \rightarrow \min.$$

Here $\|\mathbf{X}\|_k$ denotes norm (length) of a k -dimensional vector. It is clear that such a problem hardly make sense: observed traveltimes bear too little information for evaluating \mathbf{V} . However, usually the interpreter has at his/her disposal a great amount of geological and geophysical data and information, let not formalized, which make it possible to form the hypothesis of *initial*

approximation to the model. If V_0 is velocity function of initial approximation, then (1) can be reduced to more realistic problem of finding *corrections* ΔV satisfying

$$(2) \quad \| \mathbf{T}^* - \mathbf{F}(V_0 + \Delta V) \|_n \rightarrow \min.$$

Problems (1) and (2) are basically different. While there is no a priori information on the model in (1), V_0 in (2) contains significant, if not the major, part of it.

Let $\Delta \mathbf{T}$ be a vector of time residuals, i.e. differences between observed times and times obtained as forward problem solution for the model V_0 : $\Delta \mathbf{T} = \mathbf{T}^* - \mathbf{F}(V_0)$. After examination of the residuals $\Delta \mathbf{T}$, one can, applying one's professional knowledge and insight, change model from V_0 to V_1 "manually", solve forward problem for V_1 and study new residuals. If they are found less than the previous in a certain sense, one can continue changing the model in the same direction to fit it still better the observed times. In this way one implements *manual model fitting*. It is manual in the sense that there is no machine to swallow up the data on one end and spit out the result on the other for general enough problem.

Tomography Inverse Problem

The problem (2) can be solved much more efficiently, if one supposes that corrections ΔV are small enough. In computing, V is represented as a vector of its values at the nodes of a mesh. Hence, one can apply standard consideration of differential calculus in Euclidean space and replace the forward problem operator in (2) with its linear approximation:

$$\mathbf{F}(V_0 + \Delta V) \approx \mathbf{F}(V_0) + \mathbf{D} \cdot \Delta V.$$

Here \mathbf{D} is a matrix whose elements are partial derivatives of components of \mathbf{F} computed at $V = V_0$. Then (2) takes the form

$$(3) \quad \|\Delta \mathbf{T} - \mathbf{D} \cdot \Delta V\|_n \rightarrow \min.$$

Problem (3), in which corrections ΔV are to be found, belongs to a known class of mathematical problems: Linear Least Square Problems. And though it is an *ill-posed* problem, it is known how it can be *regularized* (all terms will be explained later). The approach to interpretation based on reduction (2) to (3) is called *traveltimes seismic tomography*; solving (3) is called *tomography inversion*. The term *first arrival tomography* relates to a special case, in which traveltimes are identified with specific events on seismic records. Tomography inversion, as it is implemented in XTom-LM, is in no way related to any specific kind of wave. Observations of all types of waves – diving, transient, reflections, refractions and their mix can be used for finding the true velocity distribution.

Successive Iterations

Regardless of whether one uses manual fitting or applies tomography inversion, interpretation is an iterative process. Manual fitting was described above exactly as such process. As for tomography inversion, its main assumption allows only small change of initial velocity. However,

one can consider the changed velocity as the initial and make the next step and so on, until time residuals become acceptably small. Iterative approach is common for all problems of interpretation. Such approach requires from the software to implement multivariant and iterative processing in its dynamic data structures and user interface.

Traveltime Curve Inversion

XTomo-LM 3 deals with inverse problems arising in *profiling*, where observations forms *traveltime curves* (or time-distance curves, or TX-curves). A problems of extracting specific information on the medium from TX-curves will be called *traveltime curve inversion*.

In deep seismic surveys the *diving wave* is often observed, whose rays return to the surface due to continuous refraction in medium with velocity basically growing with depth. The inverse problem for diving wave TX-curve set allows building a model $V(x, z)$ in a class of models with function V decreasing by z for each x (z -axis is directed vertically upward). The obtained velocity distribution can be used as a good initial approximation for tomography inversion.

Two more inverse problems relate to reconstruction of seismic horizons using reflection and refraction time-distance curves. Velocity distribution in covering medium is supposed to be known. Solving such problems is based on the evolutionary equation of seismic wave propagation. We use only kinematic part of the theory: the eikonal equation for wave travel time $T(x, z)$ from a fixed source. According to the theory, if the trace of $T(x, z)$ on a curve L is known at the moment t_0 , it is possible to restore $T(x, z)$ at any point outside L at any time $t > t_0$ or $t < t_0$. Backward continuation of time field $T(x, z)$ from its trace on the surface line (which is TX-curve) is called *migration*. Comparing migrated time field T with the incident wave traveltime, it is possible to determine location and geometry of a reflector or a refractor.

Though the first of these problems relates to the gradient velocity model and the rest two – to the layered model, they are placed in one class. The reason for that is partly technical: all of them use common data structures.

Extensions

The main purpose of XTomo-LM is implementation of inversion algorithms listed above. However, to productively apply the software, the following two questions are to be answered: (1) how input data are supplied? and (2) is there a convenient tool for manual model fitting? Both problems can be treated by appropriate XGeo products.

1. Input data. XTomo-LM input data are delivered in text files of the SRT format which contain coordinates and wave travel times (see. [Introduction/Data](#)). However, those data are secondary. Real observed data are represented by seismic records acquired on a seismic line. Normally, they are arranged as a set of common source point (CDP) seismograms. To get from a set of seismograms to an SRT file, one need a special software. Such software is provided by [Data Preparation Unit](#) (DPU), a stand-alone application which implements picking wave arrival times from line seismograms in a half-automatic manner.

2. Layered model manual fitting. However smart XTomo-LM inversion algorithms are, the result will depend essentially on data completeness and quality. It is common situation that only some

layered model components are lighted well enough with data to be reliably built. Hence, in some measure, manual model fitting is unavoidable. XTomo-LM is not adjusted for that. But there is an XGeo product implementing sort of a software platform for the task – [Kinematic Seismic Model Fitter](#) (shortly, XMF). This is a self-contained application working directly with line CDP seismograms. Despite difference in model representation, XMF is connected to XTomo-LM with a bidirectional program interface which allows joint use of both applications for kinematic interpretation in the way that a project initiated in XTomo-LM can be continued in XMF and vice versa (see [Environment/Creating projects](#) and, with more details, in [XMF documentation](#), in section XMF and XTomo-LM).

2 XTomo-LM 3: Overview

Basic tools

XTomo-LM 3 includes the following means of [kinematic interpretation](#):

1. Tools for creating and editing a model.
2. Means of creation, import and editing an observation system (source/receiver configuration).
3. Means of kinematic modeling or solving forward problem. For given layered model (or gradient velocity model), an observation system and a wave, the wave rays are traced, traveltimes along the rays are computed and the ray picture is presented.
4. Means of tomography inversion.
5. Means of TX-curve inversion for diving, reflected and head waves.

Framework

Because of the iterative and multivariant nature of interpretation process, a large part of the software is used for support of environment or framework within which this process is evolving. The environment includes the means of data organization, support of the workflow and the appropriate user interface. Each task of modeling or field data interpretation is treated as a *project*, a *modeling project* (M-project) or an *inversion project* (I-project). A project is a container for data including the entire history of processing. Managing projects and workflow support is in competence of *Project Manager* – the XTomo-LM head application. It enable the user to run other applications for solving interpretation problems and servicing tasks. XTomo-LM provides rich capabilities of exporting internal data to ASCII files for interfacing with external applications, in particular, with brand tools of presentation graphic, Golden Software Surfer™ being a primary target.

Documentation

In the rest of the document, "XTomo-LM" means "XTomo-LM 3.x". Version number is mentioned if only it is required by the context. The commonly used terminology is certainly preferred, but the user interface shifts its choice toward brevity and expressivity. Thus, *TX-curve* is the only term for time-distance curves; *spread* is often used side by side with *observation system* and so on. The terms used are necessarily explained either when they appear for the first time or when they fall in the focus of study. The documentation is supplied in PDF and EWriter files (HTML packed in exe

file). Both manuals can be viewed from the Windows Start menu (*Start/Programs/XTomo-LM*). Context help topics can be invoked by the F1 key, or help buttons, or commands of help menus.

3 Waves

Interpretation models

It is assumed in XTomo-LM that kinematic interpretation can be based on two media models. The first is the gradient velocity model. It is used, for example, in seismology, where the classic first arrival tomography first appeared. First arrival is an event on seismic record: the first sharp break of seismic energy on a trace. The set of first arrivals is regarded as a set of arrivals of a wave propagating in a media with gradient velocity. The goal of first arrivals tomography is determination of velocity section.

The second model is the layered model, using recording of reflected and refracted waves, generated by seismic boundaries. Layered model study includes mapping seismic horizons and determination of velocity in layers. In the latter problem the tomography approach again can be applied, with using arrivals of different waves. Thus, both models complement each other in interpretation problems. The above considerations motivate XTomo-LM classification of waves.

Wave types

The following kinds of waves are used:

- diving or transient wave;
- reflection;
- head wave;
- first wave.

An observed event on seismic record is supposed to fall into one of these categories. Diving or transient wave is formed due to continuous refraction. Which of two terms is better, depends on source-receiver configuration and the context. The term *diving* is relevant when we speak of line observations with sources and receivers set near the surface; the diving wave rays turn back at certain depth due to refraction caused by velocity growth and return to the surface. The standard term in the user interface is *diving wave*.

Head wave is a wave propagating along an interface of small curvature which divides layers with velocity limit values V_1 (from above) and V_2 (from below) with $V_1 < V_2$. The implied physical model is a waveguide: a wave propagates within a thin layer under the interface generating upward rays attaining the surface. In the documentation and user interface the term "head wave" is used rather than "refraction" to avoid ambiguity.

First wave is placed in the same list, despite its formal definition, because its arrivals are interpreted as those of continuously refracted wave. First wave can include arrivals of different waves. For example, head waves from different horizons successively come out in first arrivals at large offsets; thus first wave may consists of diving wave and head wave arrivals. When data are being prepared for input into XTomo-LM, some arrivals can be used more than once, say, as part of

first wave and as arrivals of head wave. At processing time, they may be used for different purposes, say, to determine velocity section (as first arrivals) and to build a refractor (as head wave arrivals).

Converted waves are allowed to come into play only in modeling projects. The fact of conversion of a refraction or reflection on a seismic boundary H is treated as changing velocity of propagation from $V(x,z)$ to $C_H \cdot V(x, z)$, where V is defined by the model, C_H is *conversion coefficient (CC)*, depending only on a horizon H .

Numeric Codes

A wave is identified with its numeric code or ID in the following way:

ID = 0 for diving wave;

ID = <Horizon ID><Wave type> for reflection or head wave.

Horizon ID is an integer number in the range 1 to 99, for example, the ordinal number in the model (details are [here](#)). Wave type is encoded with one- or three-digit number:

0 – monotype reflection;

1, 100 - 199 – converted reflection;

2 – monotype head wave;

3, 300 - 399 – converted head wave.

Here are examples of wave IDs:

(1) 0, 10, 1101, 3302, 102.

They encode, respectively, diving wave, reflection from horizon 1, converted reflection from horizon 1, head converted wave from horizon 3, head wave from horizon 10. Encoding cannot be called lucid, but the need for backward compatibility prevails. In the user interface, wave ID is displayed with hyphen between horizon ID and wave type:

(2) 0, 1-0, 1-101, 3-302, 10-2.

Internally, wave ID are stored as 4- or 5-digit numbers:

(3) 0, 1000, 1101, 3302, 10200.

In SRT files one can use representations (1) or (3).

Additionally to numeric codes, each wave is assigned a set of *drawing attributes*, in particular, – a unique color. Drawing attributes are used when wave rays or traveltimes curves are drawn on screen. See [Wave Manager](#) for more details.

4 Model

Traditionally, mathematical model of layered medium is taken in the form of velocity vector function

$$\mathbf{V}(x, z) = (V_1(x, z), V_2(x, z), \dots, V_m(x, z)),$$

in which components $V_k(x, z)$ are smooth functions defined on domains bounded by smooth curves representing seismic horizons. To digitize such model, rectangular grids are used for layers and other grids for interfaces. Ray tracing is implemented by getting numeric solution to the ray equations.

XTomo-LM model concept

In XTomo-LM, model is described with velocity function $V(x, z)$, defined at nodes of a curvilinear grid, formed by a set of verticals and a family of non-intersecting simple curves. Thus, a grid cell is a trapezium whose parallel sides are vertical. The left and top cell edges are regarded belonging to it; their intersection point is called *cell vertex*. Velocity function is continued from grid nodes to the entire model area, so that its value at any point of a cell is equal to its value at the cell vertex. After continuation, velocity becomes piecewise constant function on the grid domain and a step function of z on any grid column. XTomo-LM ray-tracing algorithm does not require from V to be smooth and even continuous because it uses the integral forms of Fermat and Huygens principles. The model described can be obtained by discretization of a smooth model, but that is only the special case.

In geology and geophysics, velocity distribution is usually described in geometrical terms (layer, horizon, structure). Curvilinear mesh suites better for representing model geometry than the rectangular. For modeling a reflection from a curved horizon, it is better to use the grid in which the horizon coincides with one of coordinate lines (fig. 1). In this case no special description of the horizon curve is required for tracing rays.

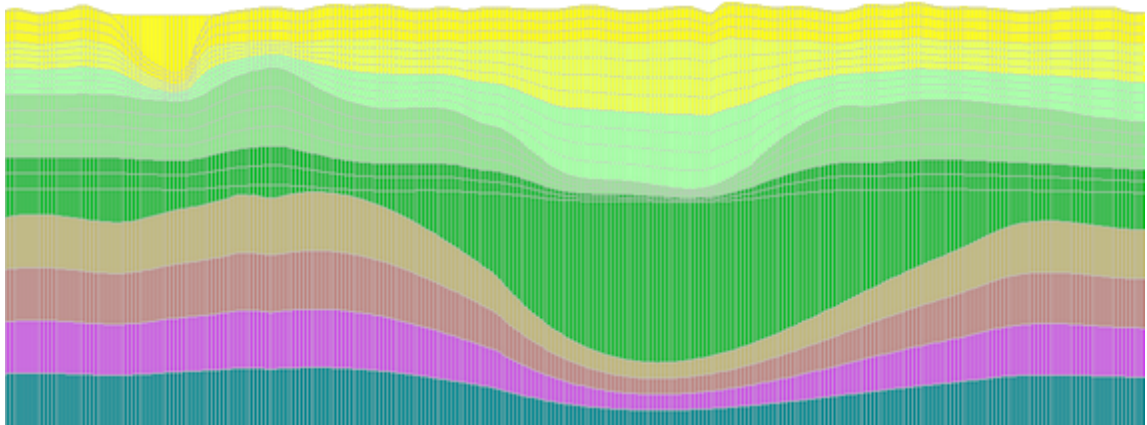


Fig. 1. This layered model is represented accurately by a curvilinear grid of the class under consideration.

Details

XTomo-LM uses plain right-hand Cartesian coordinate system with horizontal X axis directed left-to-right and Z axis directed upward. Model area D is a quadrangle bounded on the left and right with verticals $x = L$, $x = R$. Model bottom is horizontal line $z = B$; model top is a simple curve, which is referred to as *model top (line)*. A curve is *simple* if any vertical cuts it at one point at the most (it implies that a simple curve can be defined by equation $z = f(x)$, $x \in [L, R]$). Domain D can be positioned arbitrarily with respect to the coordinate system origin.

The least rectangle containing a model is called *model rectangle*. It coincide with the model domain if the model top is horizontal. Model rectangle defined at the first definition of a model cannot be changed in the course of editing within one project, unlike other model elements.

The grid or mesh is formed by a set of verticals (or *v-lines*) and a set of simple nonintersecting curves (or *h-lines*). Each h-line necessarily crosses the left and right model borders and, hence, cannot get in or out through the model top or bottom. Points of intersection of v-lines and h-lines make up the set of *grid nodes*. Neighboring v-lines (h-lines) form grid columns (rows). V-lines and columns are numbered left-to-right, h-lines and rows – top-to-bottom. Cells are defined by double-index: (column number, row number); the same relates to nodes. Nodes on the right and bottom model borders make no sense for the user. Other grid subsets are named in the explanatory text to fig. 2.

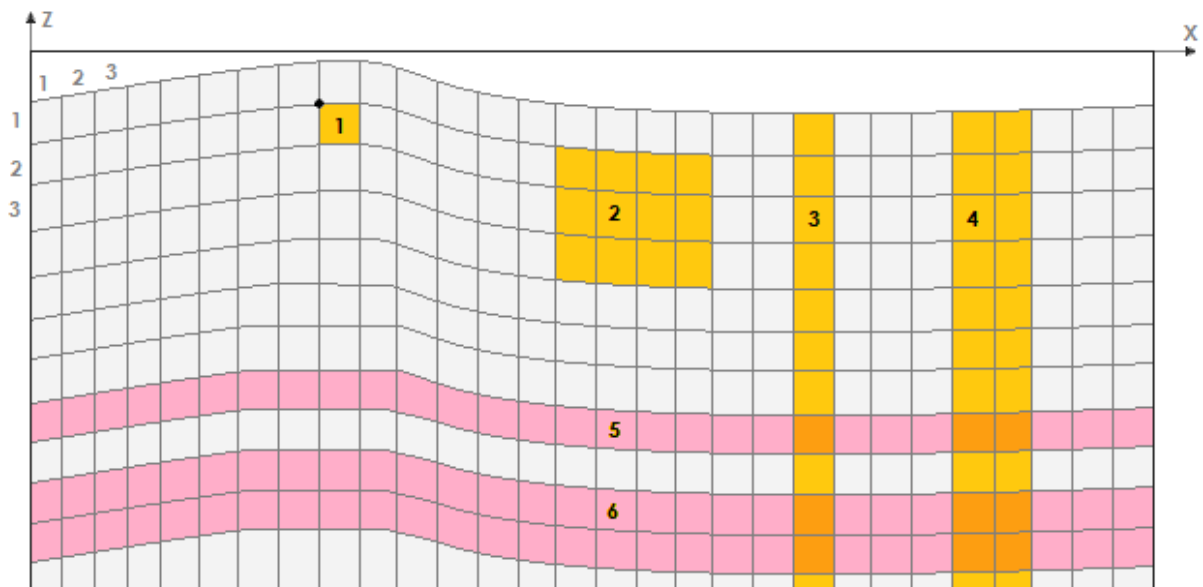


Fig. 2. Standard grid subsets.

1 – cell and its vertex; 2 – subgrid; 3 – column; 4 – vertical strip; 5 – row; 6 – horizontal strip.
Digits 1,2,3 at the top-left are column and row numbers.

Restrictions imposed on h-lines are strong enough. They suggest that h-lines are, in a way, *subhorizontal*. Fig.1 shows a typical model of the class under consideration.

Frequency of grid lines is chosen following physical considerations. At that, the user should keep in mind, that XTomo-LM is based on ray approximation to the true description of wave propagation. Therefore, model changes at the distance less than two wavelength do not make physical sense. Thus, speaking strictly, thickness depends on wave spectrum, and so in upper layers the grid should be thicker than at the model bottom if we speak of deep seismic surveys. On the computational reasons, ratio of cell dimensions must be within the range 0.1 to 10. But the closer this ratio is to 1, the better. A model without seismic boundaries is described numerically as the ordered set of triples $\{(x_k, z_k, V_k)\}$. Each triple defines a grid node coordinates and velocity value at the node.

Seismic Horizons

The main idea of using a curvilinear grid is that all seismic horizons coincide with some of grid lines. Once equation of a horizon is available, it is implanted into the current grid, changing it, of course, and becomes its h-line. In particular, that can be done at project create time, if one or several seismic horizons are known and properly described.

The detailed explanation of the above procedure is given in the section [Changing model geometry](#). Now it is enough to note, that in XTomo-LM, to define a horizon means to fix h-line number, horizon ID and, at least, one wave W generated by the horizon. Horizon IDs are integers increasing with horizons' depth. The horizon ordinal number in the model (counting from its top) can be used as horizon ID, but to avoid problems with inserting new horizons, it is better to use a sequence of integers with some step. Triplet (ID, H, W) defines a horizon unambiguously (and even with some redundancy: ID is also a part of W). Layered model **M** is described, first, with velocity distribution $V(x, z)$ and, second, with the *horizon list* **H**, i.e. a set of triplets (ID, H, W). Briefly: **M** = (V, **H**).

5 Data

Data sets, with which the user deals in XTomo-LM, are divided into two classes. The first class include observation data (or, simply, observations), i.e. positioning data and wave traveltimes. The second class includes information on the model. It can be obtained either from external sources or by XTomo-LM tools. For example, in deep seismic investigations, information about interfaces of upper layers can be obtained either by inversion of reflection TX-curves or from results of a CDP survey. The same relates to information on model velocity. In this topic, we describe how the data mentioned above should be made eligible for XTomo-LM or, more precisely, what are formats of ASCII files in which the data are stored.

Observation files

Observations are stored in ASCII files of the [SRT](#) format. A line of such file consists of three blocks: **S**, **R**, **T**. The **S** and **R** blocks are triplets ID, X, Z specifying a source and receiver. ID is a device identifier or number, X and Z are coordinates. The **T** block is a pair T, W: T is **S**-to-**R** traveltime of wave with ID = W. Actually, each file line bears information about a wave ray. The #DT format used

in the first arrival tomography system Firstomo® is also allowed but only for tomography inversion.

For positioning data in modeling projects, the reduced formats [SR](#) and [S+R](#) are applied. The former is composed from lines with **S** and **R** blocks, the latter requires two files, one for sources, the other – for receivers. No ordering of lines in an observation files is required.

SRT Port

Importing observation files is not a simple task. First, file content must be thoroughly verified to exclude any ambiguity in input data. Second, the user needs tools for viewing data in numeric and graphical mode. To deal with those tasks, XTomo-LM uses a special component – *SRT Port*. It consists of a data warehouse and processing software. The data warehouse contains verified data from input files of SRT, #DT, SR and S+R formats stored as *SRT databases* and ready for repeated use. SRT Port is a standalone component designed to isolate main processing workflow from technicalities of data import. It serves as an observation data source for all XTomo-LM projects. Work with SRT Port is described in the chapter [Observations](#).

Composing observation files

Formally, the user can make up an observation file in any text processor or spreadsheet. But for industrial-scale observations, the way from field records to observation files is long enough. First, field coordinates must be mapped onto XTomo-LM coordinate system (X, Z). Further, wave identification on seismic records must be carried out and traveltimes picked. Finally, the results must be stored in a (possibly, very large) SRT file. Preparation of data can hardly be carried out without special software. XGeo delivers such software as a separate product – [Data Preparation Unit](#) (DPU).

Observation geometry. TX-curves

An inversion tool kit depends on observation geometry. XTomo-LM divides observations into two classes: *2D profiling* and *Other 2D surveys*. In 2D profiling the measuring system is moved along a line. Additionally, the following formal condition is supposed to be fulfilled:

- (1) *source or receiver position is unambiguously defined by its x-coordinate* (in other words, devices of the same type are not set one under another).

It follows from (1) that a set of observations $\{ (S, R, T, W) \}$, where wave W and source S are fixed and receivers (R) are ordered by X_R , can be described with function $T = T(X) = T_S(X_R)$, which is called *TX-curve* of wave W . Often TX-curve is regarded as a union of two branches: *direct* TX-curve defined for $X_R > X_S$ and *reversed* TX-curve with domain $X_R < X_S$. The receiver with $X_R = X_S$ can be considered as a member of either the direct or the reversed curve. *TX-curve set* of wave W is a set of curves, generated by all sources.

Thus, 2D profiling generates TX-curve sets. One can apply to such observations all XTomo-LM inversion tools, in particular:

- building initial velocity section for tomography by inversion of diving wave TX-curve;
- using observations of different waves for tomography inversion;
- building seismic horizons by inversion of reflection and head wave TX-curves.

For observations with other geometry, XTomo-LM provides solving forward problem and tomographic inversion for diving wave, reflections and head waves. At that, source-receiver configuration may be arbitrary, with devices located above seismic horizons. In particular, part of devices may be located inside a borehole.

Observation geometry is declared by the user, when he or she performs import of an observation file to SRT Port or creates an inversion project.

Information on model velocity

XTomo-LM accepts velocity information in the form of *velocity column set*. The term *velocity column* is used for sampled one-dimensional distribution $V = V(z)$. Thus, velocity column is a sequence of couples $VC = \{ (z_k, V_k), k = 1, 2, \dots, n \}$, in which z_k follow in descending order (i.e. in the order of growing depth).

When velocity column is being imported, XTomo-LM can treat it in two ways: as *step function* or as *piecewise linear continuous* function. Step function models a layer-cake on a half-space in which V_k is velocity in layer with the roof $z = z_k$, while piecewise linear function describes continuous velocity distribution in interval $[z_n, z_1]$. The final decision is made by the user.

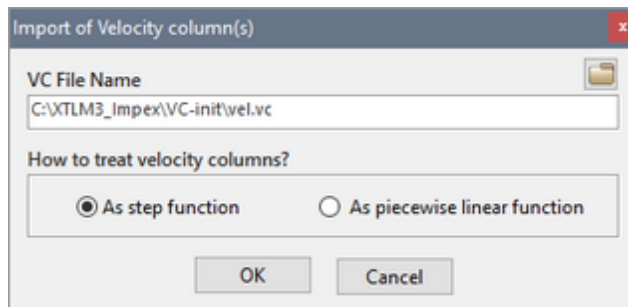


Fig. 1. Import of velocity columns.

When importing a VC or a set of VCs, XTomo-LM displays the dialog shown on fig. 1. In the dialog, the user defines file name and the way of treating velocity columns. All VC in the file are treated similarly.

In a velocity column set, each column is attached to a point of the day surface, has its own sequence z_k and its own length n . Points of attachment are supposed to be defined by their x -coordinates. Velocity column set is a natural way to define two-dimensional velocity distribution. After all, velocity $V(x, z)$ on a grid is a large set of velocity columns. To store velocity column set, the **VC** format is used. VC file can be composed using well-logs or velocity columns obtained by inversion of diving wave TX-curve set. VC file can be imported to define the starting model at project create time or to correct a model in the course of interpretation.

Information on model geometry

Model geometry is conditioned by the shape of one or several h-lines, in particular, those representing seismic horizons. In modeling, the user defines several horizons and then inserts v- and h-lines to build a grid for velocity function. These standard actions justifies the notion of *model wireframe*. This is a term for a set of simple nonintersecting curves defined at nodes of a common x -net. Wireframes are stored in ASCII files of the **MG** format. In an inversion project, such file contains one or more curves from external sources or obtained by inversion of reflection or head wave TX-curve set. A wireframe can be imported to form the starting model at project create

time; or a curve from a wireframe can be implanted in a model in the course of processing. Work with model wireframes – creation, editing, import-export – is implemented by the UMG utility (full name is MG File Editor), which, as SRT Port, is a standalone XTomo-LM component.

6 Numbers

Main variable and measurement units

In kinematic study, the main physical variables are distance (D), time (T) and velocity (V). Units of measurement are not used in the user interface, but the following conventions are applied: time unit is second; velocity unit is <distance unit>/second. Object position is defined by coordinates X and Z ($|X|$ and $|Z|$ are also distances). About distance unit see below.

Computer calculations

Real numbers are represented in computer approximately. XTomo-LM uses 4-byte floating-point representation, which is (approximately) equivalent to using real numbers with no more than 7 significant digits. The main problem of computations with finite accuracy is rounding error accumulation, which can lead to unpredicted results for some algorithms. Finite accuracy implies restriction on object nearness. Indeed, distance is calculated by subtracting coordinates, and this operation, applied to close objects, causes loss of significant digits. The remaining low decimal positions can be spoiled by rounding errors. As a result, objects may lose their initial relative position, and computations become pointless. Objects under consideration are elements of model (grid lines, cells etc.) or devices of observation system.

Spatial resolution

In order to cope with the finite accuracy problem, XTomo-LM uses the following restriction:

distance D between any two objects of same type is bounded below:

- (1) $D \geq R = 0.5 \cdot 10^{-5} \cdot \max(W, H)$, where W is width, H is height of model rectangle; R is called (spatial) resolution.

In most practical cases restriction (1) really secures correctness of all computations in the course of processing. The second requirement provides convenient way of numeric data representation in the user interface and text files:

- (2) *values of the main variables are represented in the form of fixed point numbers with no more than 7 significant digits.*

Requirement (2) is related to (1) in the following way: if (1) holds, (2) can be attained by appropriate choice of distance unit and, if necessary, shift of the coordinate origin. In typical tasks, meter or kilometer can be used as length unit; time and velocity ranges also permit representation (2).

Numeric data formats

At project create time, XTomo-LM computes resolution R and displays a *format string* for each of the main variable in the form

(2) **9999.999** or **99999.99** or **99999999.** etc.

Here '9' stands for a decimal digit, number of digits is 7, decimal separator is the dot regardless of the Windows settings. A format string defines decimal approximation of a real number, while the dot position points to its accuracy, which is equal to the value of half of unit in the last position. Format strings in (2) define accuracies as 0.0005, 0.005, 0.5. Format strings defined at project create time are used to display the main variables by all programs. *Format strings for coordinates are consistent with resolution.* That means that accuracy of coordinate decimal approximations is bounded below by resolution R. The user is allowed to edit the coordinate format strings, but only to lower their accuracy. The user chooses format strings for time and velocity following measurement precision typical for the seismic survey.

Resolution error

The *resolution error* occurs each time when the user forces two objects to come closer than resolution value permits. When that happens, the current operation is canceled and the resolution error message is displayed. If the user, when typing in an array of numbers or preparing a text file, uses the project format strings for coordinates, the resolution error does not occur. However, it can happen during graphic editing or import of external data.

About projects of earlier versions

In projects created by XTomo-LM 3.1 and earlier, the problem of providing correct computations was solved differently. Full back compatibility is secured for such projects.

7 Programs

The user needs to have an idea of how the XTomo-LM software is structured. That might be helpful to correctly respond to abnormal situations. The software include more than 40 Windows applications. One of them, Project Manager (PM), is the main application in the sense that it is launched after the user double-clicks the XTomo-LM icon on the desktop, and work session carries on until PM is shut down. The rest of applications are called *XTomo-LM modules*. Some modules are launched by PM (they are called *primary*), some are started by primary modules. If a module is a window application, its icon appears on the task bar (all modules have identical icons), but, additionally, PM supports *active module list*, which offers the user some functionality of module control. Two commands are of special importance. They enables the user to *close* an active module or to *abort* it. When a module is closed, it frees correctly all the resources, closes files and databases, correctly terminates secondary modules with no data loss. When a module is aborted, Windows destroys its process, file and database buffers are freed (data loss is possible), running secondary modules are left to run. On this reason, abort is the last resort (see also [here](#)).

Some XTomo-LM modules are *graphic applications*. They implement data viewing and editing in graphic form. Graphic modules can be divided into two classes depending on the base image type. Some, as Model Editor, work with a model image, others, as the UMG utility, display a curve family. Modules of both classes display the base image on the drawing surface called *plotter*,

which allows image magnification, scrolling and selection of objects for editing. Some graphic modules are resource-consuming programs because of necessity to frequently redraw large volumes of data (ray picture, for example).

Because XTomo-LM provides iterative and multivariant processing, intermediate data volume grows fast enough. They are stored in flat files and databases controlled by local DBMS. On that reason, the user has to look after the data, remove unnecessary files and databases, archive projects regularly. All these tasks can be carried out in Project Manager.

XTomo-LM is not a network application in the sense that files and databases are created and processed on the work station where the software is installed. Of course, files to import or exported files can be located in network folders and projects can be copied from another work station, but processing is performed locally.

Environment

1 Workflow

Cycles of Processing

Let $\mathbf{M}_0 = (V_0, \mathbf{H}_0)$ is initial model: $V_0 = V_0(x, z)$ is velocity function, \mathbf{H}_0 – horizon list (possibly, empty). Processing within an I-project can be described as succession of cycles, each cycle being a combination of the following base steps:

1. Editing model: $\mathbf{M}_0 \rightarrow \mathbf{M}_1$.
2. Selecting observation data from SRT port or copying them from one of the previous cycles.
3. Diving wave TX-curve inversion for getting more precise initial velocity model V_1 .
4. Building a seismic horizon (reflector or refractor).
5. Solving forward problem for \mathbf{M}_1 and current observation system; analysis of the solution.
6. Solving inverse tomography problem to get the refined velocity distribution V_{1r} and, hence, refined model \mathbf{M}_{1r} .

Reasonable combinations of steps in one cycle are: 2-3; 1-2-4; 1-2-5; 1-2-5-6. One cycle makes up an iteration in the process of getting the interpretation result. Each iteration may have several variants because of necessity to find the optimal set of parameters for each processing program. Interpretation process (excluding the combination 2-3) terminates only after step 5 (solving forward problem). Only at this point the statistics of time residuals between forward problem solution for current model and observed traveltimes can be estimated and found either acceptable or requiring further refinement. Combination 1-2-5 is used in the case of manual model fitting.

In M-projects, a number of cycles depends on the aims of modeling. Usually, one M-project includes modeling for a set of similar models and a set of observation systems.

Each cycle step uses input data from previous steps and generates its output data, often very large in volume. Thus, the workflow produces data flow or, more exactly, data hierarchy, which is called *Processing Tree*. It will be explained in due time.

2 Project Manager

Project Manager (PM) is the head XTomo-LM program. It is launched when user double-clicks the XTomo-LM icon on the desktop. PM represents the framework of processing to the user and enables him/her to control the workflow. This section gives general overview of the application.

Main window

PM main window is shown on fig. 1.

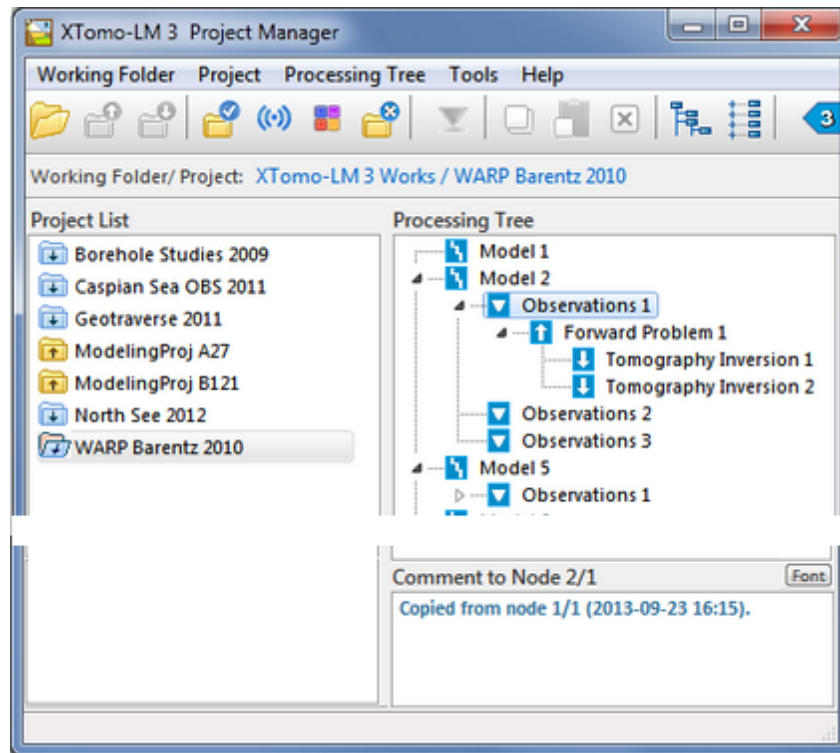


Fig. 1. Project Manager main window.

The window displays three panels: *Project List* (on the left); *Processing Tree* (on the right); *Comment* to selected tree node (bottom-right). Additionally to main menu and tool bar, there are three popup menus owned by all panels but the *Comment* panel. The most important is the *Processing Tree* menu. It contains commands launching executing modules for solving base and service tasks. The rest of the current chapter is fully devoted to learning Project Manager.

Creating Common Folders

To start work, three folders must be created for XTomo-LM data, which are shared by all projects:

- *Working Folder* serves as a container for project folders;
- *Common Import-Export Folder* is a container for projects' import-export folders;
- *Archive Folder* is a storage for projects' package and archive files.

Several common folders of each type can be used if one finds it convenient. For instance, in order to separate different groups of projects. To create or change common folder, select PM main menu command *Folders/Common Folders* or click on the first tool bar button. The command invokes the *XTomo-LM Common Folders* dialog (fig. 2).

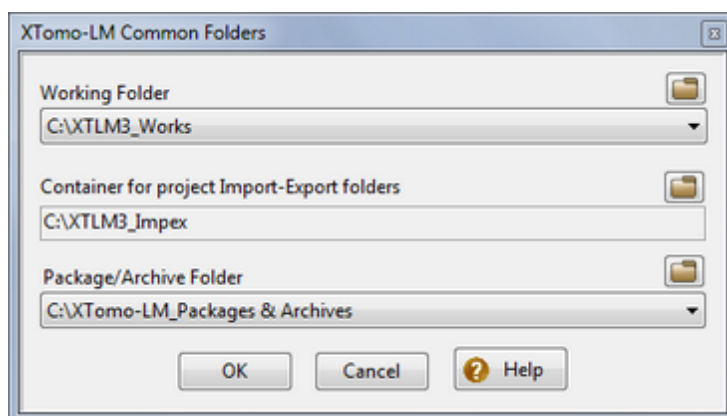


Рис. 2. The dialog to manage common folders.

Three dialog fields either are empty or display common folder of each kind currently in use. There is a button to call the local folder browser above the right end of each field. Use it to select a new folder. The *Working Folder* and *Package/Archive* fields are, actually, the drop-down lists allowing to promptly switch between different folders defined earlier. The common folders must be different, i.e. one of them must not contain the other(s). No project can be created until all common folders are defined.

3 Creating Projects

Information required for creation of an XTomo-LM project sometimes must be prepared beforehand. Formally, the user has to:

- (1) type in project name;
- (2) define observation geometry – for an I-project;
- (3) create starting model;
- (4) change numeric data formats, if necessary.

Observation type can be *2D Profiling* or *Other 2D Surveys*. Choosing the type is important, because it affects processing workflow, as was explained in [Introduction](#).

Starting Model

A project is created together with *the starting model*. It is the first member in sequence of successive approximations to the resulting model. One of starting model properties *cannot* be changed in future refinements: it is [model rectangle](#). In other respects, the starting model can be changed drastically. It is reasonable to define the starting model using available a priori information. XTomo-LM offers several ways to do that.

Table 1. Methods of creating the starting model.

?	A priori data	How starting model is built?
1	None	<i>Simple model</i> : orthogonal grid with velocity defined at four corner cells. The velocity values, grid size and line frequency are defined by

		the user. Velocity is calculated by the program using interpolation of its values at the corner cells.
2	Velocity column set	The user defines an orthogonal grid and provides a VC file. The program defines velocity in grid cells using velocity columns imported from the file.
3	Model wireframe	The user defines model rectangular, a number of grid verticals, values of velocity at corner cells and provides an MG file. The program imports the wireframe and builds layered model with constant layer velocities. If the first line number is 0, it is interpreted as model top. If model top is the only curve imported, the first thing to do after the project is created is to add additional h-lines by Model Editor's command <i>Double Number of Rows</i> .
4	Model defined by VFT file	The VFT format was inherited from Firstomo®. The program forms the model corresponding to the file content.
5	Model defined in an XTomo-LM 3 project	The user points to a Processing Tree m-node of an XTomo-LM 3 project in any working folder. The program copies the model to the new project's folder.
6	Model defined in an XMF project	Starting model is imported from an XMF project in the course of joint interpretation .

Project Constructor

I- and M-projects are created by different commands of the *Working Folders* menu, but both launch module New Project Constructor (NPC). The user's actions in both cases are almost the same. Consider creation of an I-project. If there is an open project in Project Manager, close it and click on the *Working Folders|New I-Project* command. The NPC main window with two tabs is shown on fig. 1.

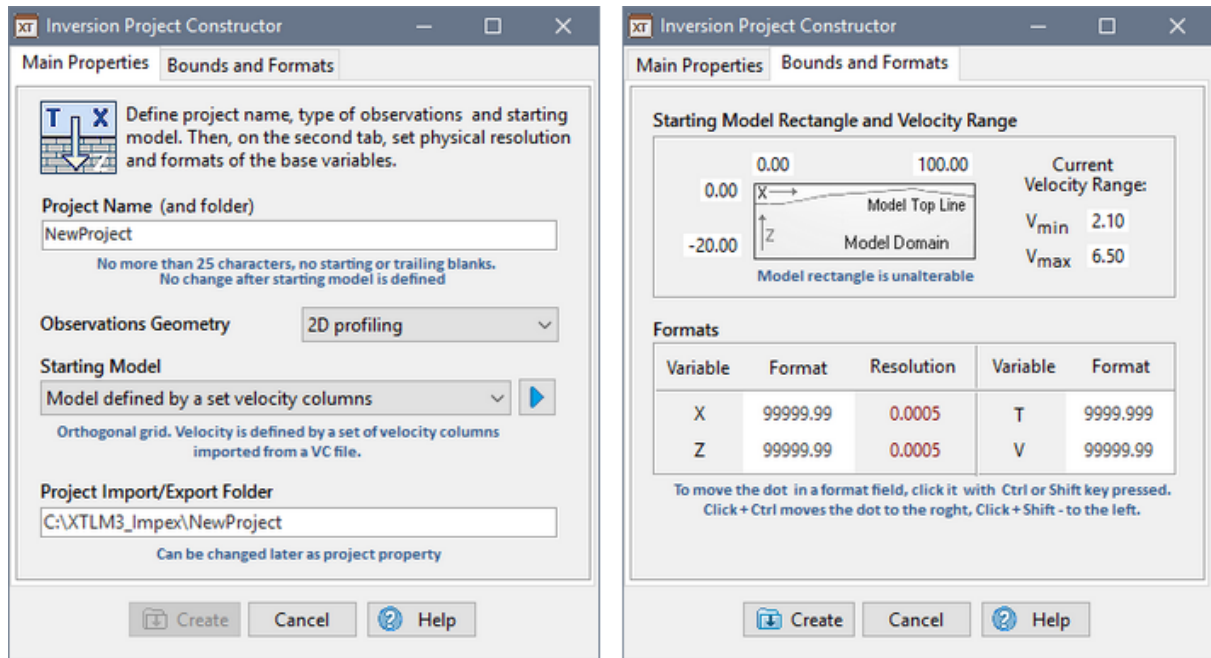


Fig. 1. New Project Constructor's main window.

The first field to fill out is *Project Name*. Use no more than 25 characters allowed for file names with no starting or trailing blanks. Parallel to project name, the project import-export folder name is being automatically formed in the bottom field. It cannot be changed at this point but, if necessary, that can be done after the project is created. Now select one of two options for observation geometry. Then select one of options for the starting model. The drop-down list of starting model options corresponds to table 1.

Creating Starting Model

Click on the button to the right of the *Starting Model* field. It invokes the dialog matching the model type. In the below description, the dialogs are numbered as in the first column of table 1.

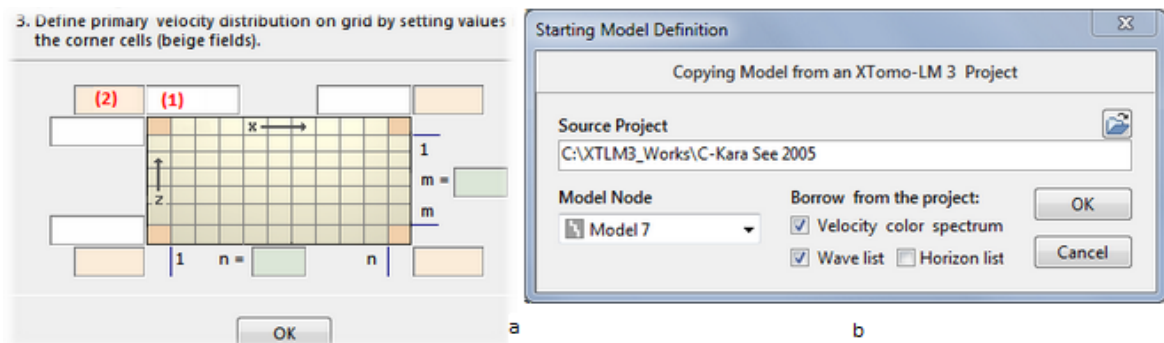


Fig. 2. Creation of starting model. a – Dialog 1; b - Dialog 5.

Dialog 1 (simple model) is shown on fig. 2a. In the beginning, field (1) is focused. Type in it the value X_{min} . Press the *Tab* key to enter successively X_{max} , Z_{max} , Z_{min} . Hit the *Tab* again to set

focus to the field (2). Enter velocity value for the top-left grid cell. Then, using Tab, enter velocity values for the rest three corner cells. The next *Tab* shifts focus to the *m* field. *m* is a number of inner horizontals. The next and the last tab sets focus to the *n* field – a number of inner verticals. Click on *OK* to confirm the input and close the dialog.

Dialog 2, unlike dialog 1, does not contain velocity fields. Define only the grid parameters. The *OK* buttons invokes the *Import Velocity Columns* dialog, in which the user selects a VC file to import, and specifies how to interpret velocity columns (see [Data](#)). Clicking on *OK* in the dialog starts import. To get an idea of how velocity function on grid is formed, see [here](#).

Dialog 3 differs from dialog 1 in that the *m* field is absent because h-lines are imported. After the model rectangle and velocity values at corner cells are defined, click on *OK* to invoke the Open file dialog to select an MG file.

Dialog 4 is the Open File dialog for selecting a VFT file.

Dialog 5 and 6. The former is shown on fig. 2b. The folder of a source project is selected in the local folder browser (the button at the end of the *Source Project* field). After that, the *Model Node* drop-down list is ready for selecting a model to copy. along with the model, some or all of the project infrastructure objects (spectrum, wave list, horizon list) can be copied. Check the necessary ones and click on *OK*.

Dialog 6 differs only in that an XMF project iteration should be selected in the similar way and only one additional option is offered: using spectrum from the source project. Import is allowed only when in XMF project, each horizon is bound to, at least, one wave.

Bounds and Formats

After the starting model is created, the second tab *Bounds and formats* becomes active (fig. 1b). It displays the model rectangle and velocity range. The latter is computed after the starting model and later can be changed (see the next paragraph). The model rectangle is a project constant. The *Formats* table contains the [resolution](#) value and the default [format strings](#) for the main variables. The user can move the decimal point in each string by clicking on it with Ctrl or Shift pressed. Ctrl moves the point to the right, Shift – to the left. Moving the point to the left in the coordinate format strings may bring about the resolution error.

Completion. Viewing Project Properties

After clicking on the *OK* button, NPC creates project folder, Processing Tree with the "Model 1" node and the project infrastructure. Wave list and velocity color spectrum are created with the default properties, if they have not been copied from another project. The "Model 1" node contains the starting model. NPC closes, and new project name appears in the project list.

For an opened project, the *Project/Properties* command of the PM launches NPC in special mode. In this mode, the user can change the following properties:

- import/export folder;
- X, Z and T format strings.

Velocity range and format string can be edited in [Velocity Spectrum Manager](#).

4 Project support

Commands for work with projects can be found in (1) *Working Folder* menu; (2) *Project* menu; and (3) the project list popup menu. The *Working Folder* menu contains commands for adding new projects to the current working folder from other sources. The *Project* menu deals with the currently open project. All commands are listed explained in tables 1–3.

Table 1. The *Working Folder* menu commands: Adding projects to current working folder

Command	Operation
<i>Common folders</i>	Invokes the dialog for setting/changing XTomo-LM common folders. Details.
<i>New M-Project</i> <i>New I-Project</i>	Runs the module New Project Constructor for creation of M- and I-projects. Details.
<i>Add Project from Network Location</i>	Invokes the dialog for adding an XTomo-LM 3 project located on a local network to the current working folder. Project name can be modified.
<i>Add Project from Package or Archive</i>	Invokes the dialog for selecting a package or archive from the project's archive folder and creating from it a new project in the current working folder. Details
<i>Add Project from Version 2 Project</i>	Invokes the dialog for creating a new project based on XTomo-LM 2 project data. Details.
<i>Show Notes as Hint</i>	This is a switch. If on, the beginning of the user notes (see table 2) is displayed as a hint, when the cursor is over the project in the project list.

Table 2. Commands of the *Project* menu. Operations on the currently opened project

Command	Operation
<i>Properties</i>	Displays <i>Project Properties</i> . Details.
<i>Velocity Color Spectrum</i>	Runs Velocity Spectrum Manager. Details.
<i>Wave Manager</i>	Runs Wave Manager. Details.
<i>Close Project</i>	Closes the opened project.
<i>User Notes</i>	Invokes the dialog that allows running a persistent text notes about the project.

Table 3. The project list popup menu commands. (operate on a project, currently selected in the list).

The menu is accessible only when all projects are closed and no module working with data of a closed project is active (e.g. [Velocity Comparator](#) or Model Viewer launched to view a [closed project](#) data).

Command	Operation
<i>Open</i>	Opens the selected project. Alternatively, double-click the project name).
<i>Rename</i>	Allows changing project name.
<i>Clone</i>	Creates project copy with a new name in the current working folder.
<i>Copy to</i>	Invokes a dialog for copying a project to a specified (possibly, network) folder with optional change of name.
<i>Delete</i>	Removes a project from the list and its folder – from disk.
<i>Archive</i>	Invokes a dialog for creating the archive file of the entire project. Details .
<i>View Model</i>	Allows viewing model of the project selected in the list (not that is currently opened). Details

5 Processing Tree

[View. Terminology](#) – [Comment field](#) – [Workflow implementation](#) – [Tree support](#) – [Copying o-node](#) – [Tree context menu](#) – [Viewing models from other projects](#) – [List of active modules](#).

View. Terminology

Processing Tree is the data structure dynamically created in the project folder in the course of interpretation. Project Manager (PM) represents the data for the user and provides access to them. The same term is used for both the data structure on disc and its visual representation in the PM's main window (fig. 1).

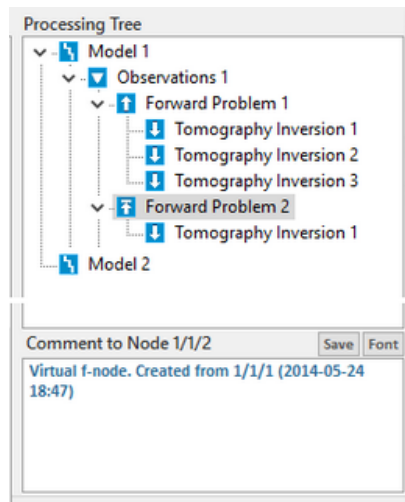


Fig. 1. A fragment of Processing Tree.

The names of the tree nodes are fixed, while numbering is maintained by PM. When nodes are deleted, the numbers of the rest of nodes are kept. In the documentation and user interface, the following shortcuts are used for nodes of different levels:

Model – m-node

Observations – o-node;

Forward Problem – f-node;

Tomography Inversion – i-node.

Processing Tree represents also a subtree of the file system, so that a folder stands behind each node. The phrases like "node folder" or "stored in a node", "copied from node" are widely used in the documentation and the user interface without explanation.

Comment field

A small field *Comment to Node* <code> contains short user notes relating to the selected node. The *node code* is a slash-delimited string of the node's and its ancestors' numbers counting from the root, e.g. 1/1/2 for the f-node on fig. 1. Maximal length of the comment is 300 characters. Sometimes, after node is created, PM itself adds information of the node origin as on fig. 1. Further notes are entered by the user. In the course of multivariant processing comments are necessary. To save changes in the field, click on the *Save* button or press the *Enter* key. To cancel changes, press *Esc* or simply click somewhere on the project list or Processing Tree. The *Font* button allows changing font of the comment field.

Workflow implementation

Processing Tree is PM's interface element with which the user implements the XTomo-LM [workflow](#). Steps of each cycle of the workflow suggest creating Tree nodes or/and processing data stored in them. After a project is created, Processing Tree contains the only node Model 1. It stores the starting model. Further growth of Processing Tree is initiated by the user. The sequence of m-nodes from Tree's top to its bottom represents model evolution from the starting model to the interpretation result. One creates new m-nodes as copies of the existing m-nodes; one creates new empty o-nodes and then fills them with observation data from SRT Port. For further iterations, o-nodes are copied with all data stored in them. On an o-node, one runs modules for getting forward problem solution and for TX-curve inversion. F- and i-nodes are created automatically by the modules of solving forward problem and tomography inversion. The rest of this section is devoted to description of Tree support tools and formal listing of commands containing in the Tree context menu.


Tree support

Base commands of Tree maintenance can be found in the *Processing Tree* menu. Partially, they are duplicated by the tool bar buttons. Accessibility and action of some commands depend on the node currently selected on the tree. The summary is given in tables 1 and 2.

Table 1. Processing Tree support commands.

Commands	Selected node	Action
<i>Copy Node</i>	m	Creates copy of the selected m-node.
	o	Starts two-step copy operation of the "copy-paste" type. Details are below.
	f	Creates new virtual sibling f-node, containing ray sample. Accessible, if the selected node is not virtual (explained here).
	i	Creates the new m-node and copies to it the refined model from the selected i-node.
<i>Create O-node</i>	m	Inserts a new (<i>Create</i>) or the copied (<i>Paste</i>) o-node as the selected m-node's child.
<i>Paste O-node</i>	o, f, i	Inaccessible.
<i>Delete Node</i>		Removes the selected tree node together with all its descendants. The node is deleted from disc. The command is duplicated in the Tree context menu.
<i>Expand Tree</i>		Unfolds all Tree nodes. To expand a selected node, use the <i>Expand Node</i> command of the Tree context menu.
<i>Collapse Tree</i>		Folds up Processing Tree up to the root nodes.
<i>Rebuild Tree</i>		Rebuilds tree using folder structure inside the project folder. The command is used when the user detects discrepancy between project data and the tree view or when the load tree error occurs.

Copying o-node

This is an operation of the "copy – paste" type. Select the o-node and use the *Copy Node* command or the button . Then select the target m-node and issue the *Paste O-node command*. Alternatively, – drag the selected o-node and drop it onto the target m-node. PM executes copying and then checks if there are sources or receivers in the copied o-node that are hanging above the top line of the model stored in the target m-node. Such situation may occur for models with the curvilinear top and is explained [here](#). If hanging devices are found, the program asks if the user agrees to land them on the model top. If the answer is "OK", PM runs the special module to fulfill correction, otherwise, copying is canceled.

Tree context menu

This menu is, actually, the main *processing console*. Most of its command launches XTomo-LM modules for getting solution to basic problems or servicing tasks. The set of commands and their

accessibility depend on the node on which the menu is popped up (= which node is selected) and on the state of processing. By hiding and disabling menu commands, PM controls the workflow and workload on the system. In the below table, the Node/Project column displays node and project types for which the command in column 1 is visible.

Table 3. List of commands of Processing Tree context menu

Command	Node Project	Action
<i>Work with model</i>		
<i>View Model</i>	m Any	Runs graphic module for viewing model (Model Viewer).
<i>Edit Model</i>	m Any	Starts graphic module for editing model (Model Editor). Accessible, if the node has no children.
<i>Work with observation system</i>		
<i>Extract Data from SRT Port</i>	o Any	Extracts data sample from an SRT Port database and creates Ray Catalog. The command is accessible if an o-node has no children.
<i>View *** on Model Image</i>	o Any	In this and following commands *** means <i>Spread</i> for M-project or <i>Observations</i> for I-project. This command runs graphic module for viewing observation system on model image (Graphic Spread Viewer).
<i>View *** As Ray Catalog Database</i>	o Any	Runs module for viewing Ray Catalog as a set of numeric tables (Ray Catalog Viewer).
<i>View Observations As TX-Curve Set</i>	o I-project	Runs graphic module for viewing Ray Catalog in the form of TX-curve set (TX-Curve Viewer). Accessible for 2D-profiling data only.
<i>Edit Spread</i>	o M- project	Runs graphic module for editing observation system (Spread Editor). Accessible, if the node has no children.
<i>Edit Ray Catalog Database</i>	o M- project	Runs module for editing Ray Catalog (Ray Catalog Editor) , if the node has no children.
<i>TX-curve inversion (for 2D profiling data) The submenu contains the following commands:</i>		

<i>Build Initial Velocity Distribution</i>	o l-project	Runs the module for building initial velocity distribution by means of diving wave TX-curve set inversion.
<i>Select TX-curves for Inversion</i>	o l-project	Starts module TX-Curve Selector for composing TX-curve set for building a horizon.
<i>Evaluate Reflector</i>	o l-project	Estimates reflector depth at source points using traveltimes along vertical rays.
<i>Build Reflector</i>	o l-project	Starts the Reflector Builder module.
<i>Build Refractor</i>	o l-project	Starts the Refractor Builder module.
<i>Preview Built Horizons</i>	o l-project	Starts the Horizon Previewer module.
<i>Solving Forward Problem and exploring of the solution</i>		
<i>Solve Forward Problem</i>	o Any	Creates child f-node and launches the Forward Problem Solver (FPS).
<i>View FPS Log</i>	o, f Any	Runs the Text Viewer module for viewing FPS log file. If FPS was a success, it stores its log in newly created f-node; if it failed, the log is placed in the o-node, from which FPS was launched.
<i>View Catalog of Traced Rays</i>	f Any	Runs Ray Catalog Viewer with the option of viewing computed traveltimes, and time residuals (the latter – for l-project).
<i>View Forward Problem Solution</i>	f Any	Launches the Forward Problem Viewer module for viewing ray picture, computed TX-curves and exploring the solution.
<i>Solving tomography inverse problem</i>		
<i>Solve Inverse Problem</i>	f l-project	Creates child i-node and launches Inverse Problem Solver for tomography inversion.
<i>Info on Inversion</i>	i l-project	Runs Text Viewer for reading special information on completed inversion.
<i>View Inverse Problem Solution</i>	i l-project	Starts the Inverse Problem Viewer module for viewing initial and refined velocity distribution and their comparison.

<i>Processing Tree Support</i>		
<i>Expand Node</i>		Unfolds the selected tree node.
<i>Delete Node</i>		Duplicates the namesake command in table 2.
<i>Create Package</i>		Creates the compressed file for tree branch ending with the selected node. Details.

Viewing models from other projects

Processing Tree provides access to the opened project's data. Data from other projects are not accessible. Meanwhile, often two or more projects are based on the same data, and the possibility of comparing models from those projects is important. PM allows to do that using project list menu. Invoke the menu on a project required (not that is currently opened) and select *View Model* command. It displays the list of the project's m-nodes on screen. Double-click a node of interest to display the model in Model Viewer's window.

List of active modules

Several modules can run simultaneously, launched by the commands of the Tree menu on one or several nodes. At the same time, as was explained in the previous paragraph, Model Viewer instances can work with other project's data. To control the entire set of running modules, use the component Active Module List, which can be activated by the *Tools/Active Modules* command (fig. 2).

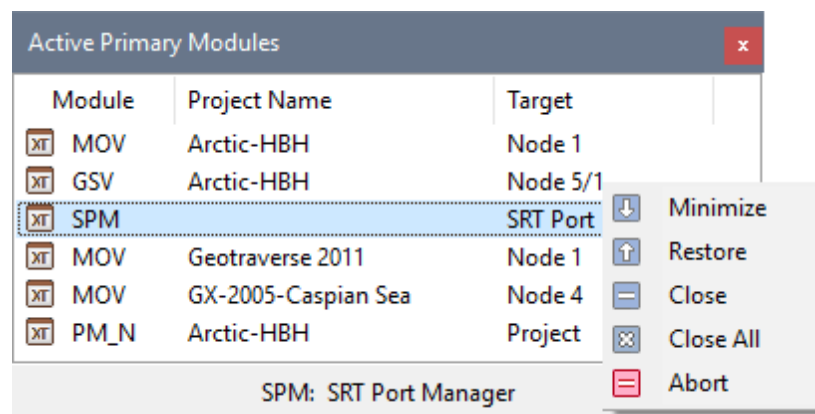


Fig. 2. Active Module List

The screenshot shows six running modules. The first column displays *short module name*. Short names are used in the documentation and (rarely) in the user interface. The full name of the selected module is displayed at the bottom. The list context menu contains control commands. All commands but *Close All* operate with the selected module. The most important commands are *Close*, *Close All* and *Abort*; they were discussed in [Introduction](#).

6 Wave Manager

The wave list is important infrastructure element of projects created for *2D Profiling* data. The module is launched from Project Manager by the *Project/Wave Manager* command or the matching tool button. The module's main window is shown on fig. 1. The left panel contains *the project wave list*, the right one – drawing attributes of the wave currently selected in the list. If the wave list was not imported at the project create time, Wave Manager contains the default list containing the only wave – the diving (transient). One works with the program only in projects in

which interpretation model is the layered model. In this case, one adds to the list reflections and/or head waves at the beginning or in the course of processing.

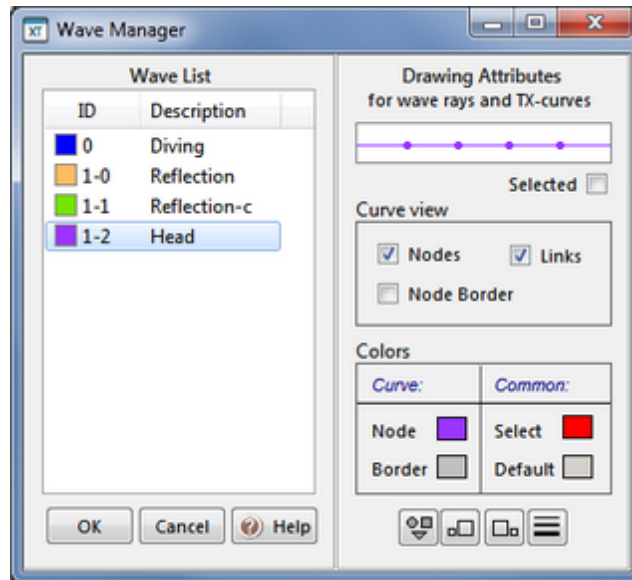


Fig. 1. Wave Manger main window.

Adding waves

The wave list is provided with the context menu with three commands:

- *Add*;
- *Export Waves*;
- *Import Waves*.

The *Add* command invokes the *New Wave dialog*. Fig. 2 demonstrates adding a converted wave reflected from horizon 2.

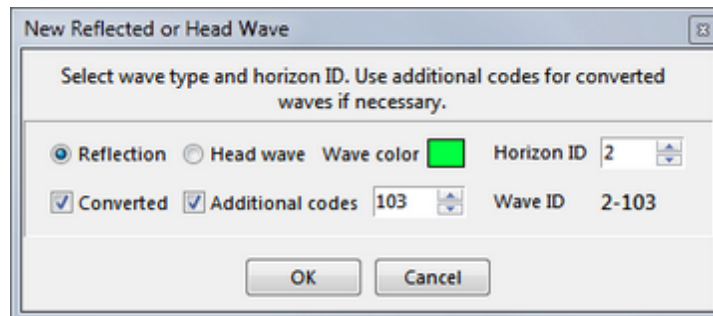


Fig. 2. Adding a wave.

The dialog contains two lines of controls. On the top line, the user defines wave type and color and horizon association (the spin-edit field contains possible values of horizon ID). Wave color must be unique in the list. To change the color, click on the *Wave color* rectangle and select color from the Windows color dialog. The bottom line of controls is used for converted waves only, i.e. when the user checks the *Converted* box. If wave code is three-digit, as on fig. 2, click the

Additional codes box and select the code from the spin-edit field. A click on the *OK* button adds the new wave to the list and assigns the default drawing attributes to it. *Waves cannot be deleted from the list, because they are referred to by various project data.*

The two other menu commands allows saving wave list (together with drawing attributes) to a user file and import the list from such file. The last command can be applied, if only the list currently contains only diving wave.

Drawing attributes

Drawing attributes define view of a curve related to the wave as ray path or TX-curve. A curve consists of points (nodes) and links connecting neighboring points. Both can be made invisible, but not simultaneously. Drawing attributes include:

- point flag (*Points/Node*);
- link flag (*Points/Links*);
- point border flag (*Points/Node Border*);
- point and link color = wave color (*Colors/Curve/Node*);
- point border color (*Colors/Curve/Border*).
- point shape (button);
- point size (button);
- link width (button).

Buttons are located at the right panel's bottom. At the panel top one can view how the curve with currently selected attributes looks. The *Colors/Common* panel contains two colors related to all waves: *Select* is used to mark a selected curve on the image; *Default* is wave color used in the *New Wave* dialog before the user assigns the color to it. These two colors are reserved and cannot be used for other needs.

When Wave Manager is active, access to Project Manager is blocked. Editing wave list is blocked, if at least one other module is active.

7 Velocity Color Spectrum

Color spectrum is used to visually represent velocity values on the model image encoding a value with color. At project create time, the spectrum gets the default color set. To change it, use Project Manager main menu command *Project/Velocity Color Spectrum*. It starts module Spectrum Manager whose main window is shown on fig. 1a. The window contains the current spectrum image (the left panel), velocity range at the moment of the program's start (*Initial velocity range*) and current velocity format.

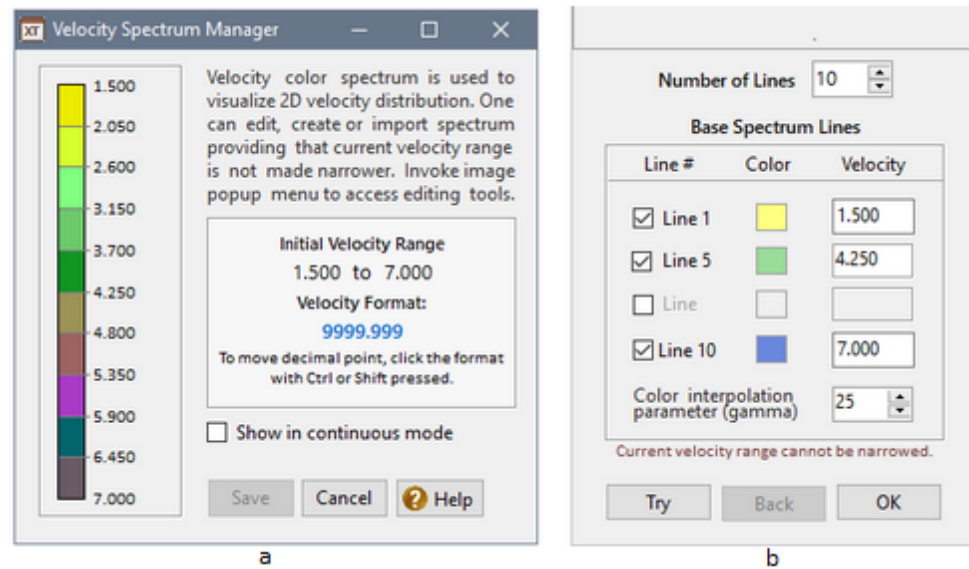


Fig. 1. Velocity Color Spectrum Manager. a) – main window; b) – creating a new spectrum.

Velocity range and format can be changed only in this module. Velocity format string is edited in the same way as in New Project Constructor: decimal point is moved by Click+Ctrl or Click+Shift.

Spectrum is a set of couples "Color – Velocity value". Such couple is imaged as a colored rectangle (spectrum line) with value at its top-right. Fig. 1a displays *line spectrum*. If the *Show in Continuous Mode* flag is set, the image shows spectrum in which color changes continuously from top to bottom.

Editing commands are gathered in the image popup menu (table 1). All commands, but one, invoke dialogs; each dialog has the *Try* button, which allows viewing spectrum after a change. The edited spectrum becomes the project spectrum after the *Save* button is clicked (it closes the module).

Table 1. The editing commands

Commands	Action
<i>Create new</i>	Invokes the <i>Create Spectrum</i> dialog (see below).
<i>Adjust Luminance</i>	Displays the dialog for adjusting spectrum luminosity.
<i>Edit Color Lines</i>	Displays the dialog <i>Line Editor</i> for editing spectrum lines, i.e. colors and values (see below).
<i>Default</i>	Replaces the current spectrum with the default one keeping velocity range.
<i>Export/Import</i>	Invokes dialog for saving the current spectrum to a user file (.usf) and for import spectrum from such file. The USF file contains the entire spectrum: all colors and velocity values. If velocity range of the imported spectrum is narrower than that of the current, import operation is canceled.

Color SetsDisplays the *Color Sets* dialog for work with color sets (see below).**Creating spectrum**

To create a new spectrum, the user has to define 2 to 4 base lines, each one with its own color and velocity value. The rest spectrum lines are computed by interpolation between the base ones. In the *New Velocity Color Spectrum* dialog (fig. 1b), first, set the total number of spectrum lines in the spin edit field *Number of lines*. Then check the lines to be used as the base ones. The first and the last lines are obligatory – they are already checked. To change the line color, click on the color square and adjust color in the *Window Color* dialog. Velocity values are defined only for the first and last lines in the corresponding edit fields. Mind that the current velocity range cannot be narrowed. The last setting is color interpolation parameter gamma. It affects the way of interpolation and is adjusted experimentally. Click on the *Try* button to replace the current spectrum in the main window with the new one. To cancel the change, click on the *Back* button. To fix the final variant, click *OK*.

Editing spectrum lines

The main tool of editing spectrum is the *Line Editor* dialog (fig. 2a) representing the spectrum as a list box. In particular, it allows the user to widen velocity range. The dialog is invoked by the *Edit Color Lines* command of the spectrum menu or by double-clicking the spectrum image. Select a list item and right-click it to invoke the context menu. Its commands are listed in table 2.

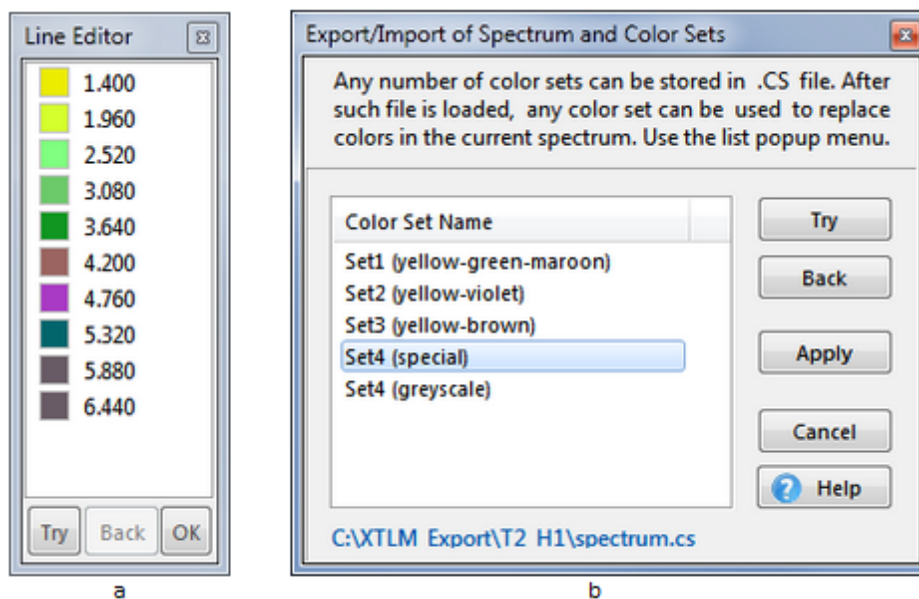


Fig. 2. Velocity Color Spectrum Manager. a – editing spectrum lines; b – list of color sets in the loaded CS file.

Table 2. The commands for editing spectrum lines

Command	Action
---------	--------

<i>Edit Value</i>	Allows changing velocity value associated with the selected line. Edit mode can be entered also by clicking on the value.
<i>Edit Color</i>	Calls the Windows Color dialog to change line color. Double-clicking an item has the same effect.
<i>Insert before</i> <i>Insert after</i>	Inserts a new line painted in the default color before or after the selected line. Change the default color with the <i>Edit Color</i> command.
<i>Delete</i>	Removes the selected line from the spectrum.

The program controls closeness of the neighboring velocity values, using format string accuracy as its limit. Inversely, changing format in the way that neighboring velocity values on the image become identical is blocked.

Color sets

Usually, the user works with several standard color sets in all his or her projects. A color set is not associated with velocity values and is regarded as an independent spectrum component, which is stored in files of the CS format. One such file can contain any number of color sets labeled with unique names. Files are created by the user. The *Color Sets* command of the spectrum menu opens the dialog shown on fig. 2b. It contains the list of color sets found in a loaded CS file. Use the list popup menu and buttons on the right panel to carry out the typical tasks listed below.

1. *Saving the current spectrum color set to a CS file.* To do the job, use the popup menu. First select *Add Current* to add the current color set to the list with the default name. Then choose *Edit Name* to change the default name for meaningful one. Finally, use *Save As*.
2. *Adding current spectrum color set to an existing CS file.* First load the file using the menu command *Load CS File*. The list will show the names of color sets in the file. Then act as in the case 1.
3. *Viewing color sets in a CS file.* Load the file, as in 2 and select a color set name in the list. Click on the *Try* button. Colors of the current spectrum (fig. 1a) will be replaced with colors from the selected set. The *Back* button restores the initial spectrum. If the current spectrum is longer than the selected set, the redundant lines are painted with the default color.
4. *Applying a color set from CS file.* Act as in the case 3, but use the *Apply* button instead of (or after) the *Try* button.

Black cells

If in the course of processing, velocity value of a grid cell comes out of the current spectrum range, the cell becomes black to signal a data error. This situation is a prompt for the user to invoke Spectrum Manager and extend velocity range by editing the end spectrum lines, or edit the erroneous cell in Model Editor, or apply Model Editor's tool for removing black cells.

8 Packages. Archives. XTomo-LM 2

XTomo-LM maintains its own tools for creating project backups and compressed files for data subsets (packages). Archives and packages are stored in the common [archive folder](#).

Packages

Packages are used for swift data exchange in the situation when the same data are being processed simultaneously in different sites, say, onboard and in the on-shore lab. Packages can be used also for getting an advice from an expert or addressing the XGeo Support Service. A package contains data from one branch of Processing Tree. In an open project invoke Tree context menu on a node and select *Create Package*. The command creates the package containing (1) data of the selected node and all its direct ancestors up to the root node; (2) project infrastructure. Package name is composed by the program. It includes creation time and the node numeric code. The name is not to be changed. Move the package created from the archive folder to an external storage or send it to the addressee through LAN or the Internet.

On receiving the package, the addressee puts it to the archive folder, starts Project Manager and applies the *Working Folder |Add Project from|Package or Archive* command. It invokes the dialog *Creating Project from Package or Archive* (fig. 1a). It lists all packages or archives in the archive folder depending on the switch under the list. The addressee finds and selects the package in the list and changes, if necessary, the project name in the *Project Name* field. After that, he or she clicks on the *Create* button. After dialog is closed, the new project can be found in the project list. In the created project the path to import/export folder may be invalid. It should be corrected in the *Project Properties* dialog.

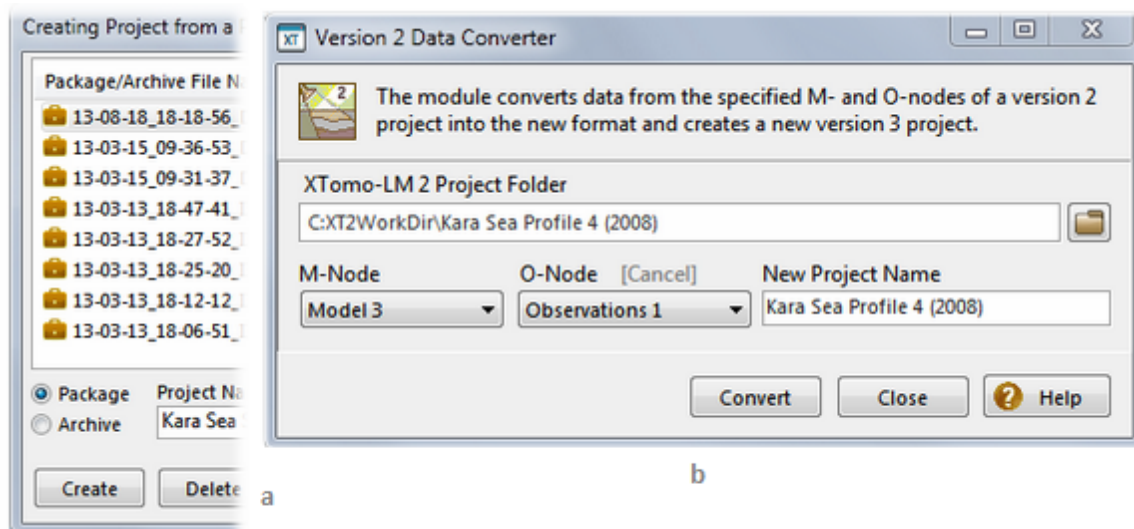


Fig. 1. Creating a project: a – from a package or an archive; b – by conversion data of a version 2 project.

The dialog can be used to clear the archive folder: the *Delete* button removes from the folder all files currently selected in the list (multiple selection is available).

Archives

Archive is a compressed file containing all project data. To create a project archive, close the project, right-click its name in the project list to invoke the context menu and select the *Archive* command. To extract a project from its archive file, act exactly the same way as in the case of a package.

Conversion of version 2 project data

Project Manager is able to create a new project from data of an XTomo-LM 2.x project, if only this software is installed on the computer. Use the *Working Folder|Add Project From|XTomo-LM 2* command to launch the converter. The module blocks access to Project Manager until operation is over. The module's main window is shown on fig. 1b.

A button to the right of the *XTomo-LM 2 Project Folder* field invokes the local folder browser. Select the folder of the source version 2 project. After that, the drop-down list *M-Node* is filled with names of root nodes of the project's Processing Tree. Select the needful m-node. From the *O-Node* list select the *Observations* node. If only m-node is to be converted, clear the *O-Node* with the *[Cancel]* link. Change project name, if necessary, and click on the *Convert* button.

The created project has the same wave list and velocity color spectrum as the source project. The path to import/export folder is, most likely, invalid and needs to be corrected in the project properties dialog.

Model

1 Imaging a model

In this topic, the user interface of the simplest [graphic module](#) based on model image is explained. The module under consideration is Model Viewer or MOV. To launch MOV, invoke Processing Tree context menu on an m-node and apply the *View Model* command.

Model image. MOV main window

Model image depicts model grid with cells painted with colors from [velocity color spectrum](#). The latter encodes velocity values with colors. Both grid lined and cell coloring can be hidden. On fig. 1 showing MOV main window, grid lines are hidden because their frequency is too high for adequate view. Heavy lines represent seismic horizons.

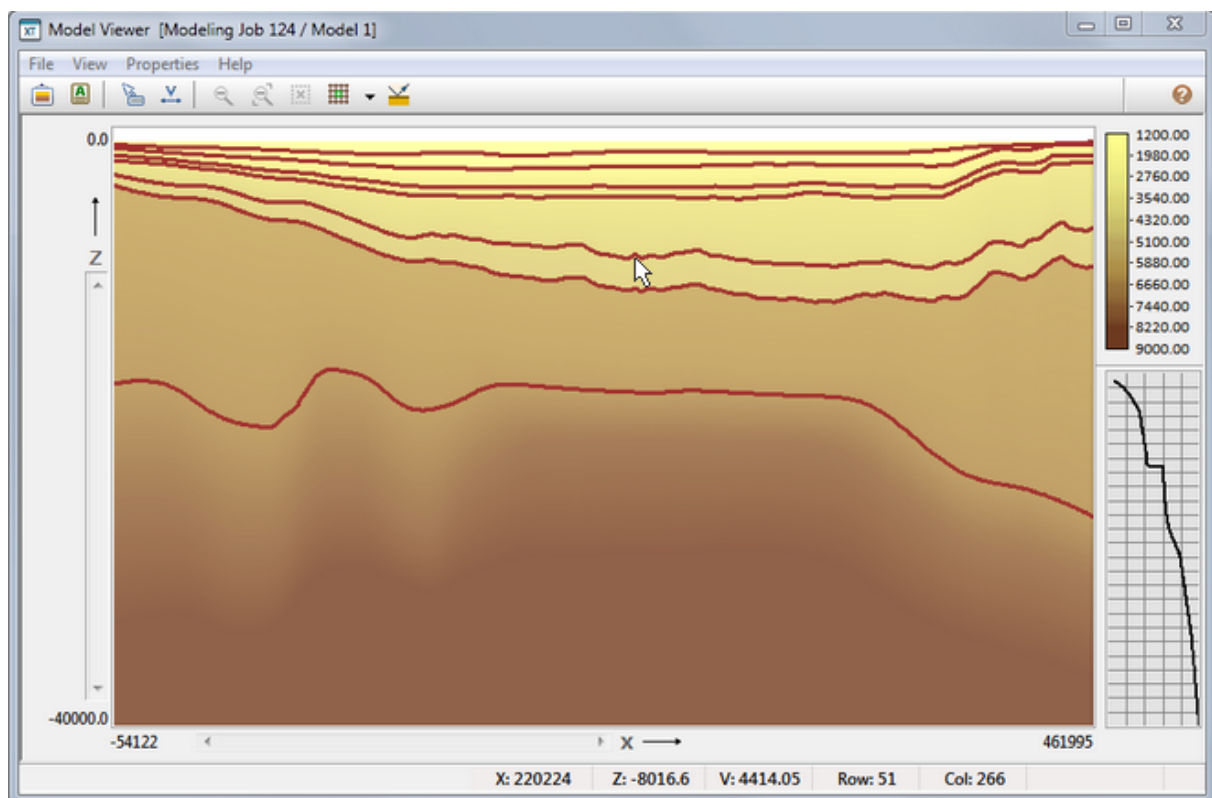


Fig. 1. Model image in MOV.

Velocity color spectrum is switched to continuous mode to avoid artifacts of line spectrum. The central part of the window is taken up by the *plotter* on whose surface the model image is drawn. The left and bottom plotter margins contain labels and scroll bars. Scroll bars come into play when

the image is magnified, and plotter displays its part or *frame*. Under the bottom margin, the *status panel* is fixed for displaying context information associated with the cursor position on the image: point coordinates X and Z, velocity value in grid cell, column and row indexes. The right margin contains two panels that can be resized by dragging the dividing line. The top panel displays velocity color spectrum, the bottom one – velocity profile $V(z)$ along the column that is currently under the cursor.

Menus

MOV control tools include the main menu, tool bar and popup menus. The popup menu owners are: plotter, spectrum and velocity profile and the rubber-band. The plotter and rubber-band menus are explained in the next topic. The spectrum menu allows switching between line and continuous modes. The velocity profile switches the method of representing $V(z)$: as piecewise linear function or as step function.

Table 1. Main menu commands

Commands	Action
File	
<i>Reload Model</i>	Reloads model file, if only it has been changed (by Model Editor) during the current work session.
<i>Export Model Image</i>	Invokes the dialog for exporting the model image to a graphic file. Details .
<i>Export Model to ASCII Files</i>	Invokes the dialog for exporting the model components to ASCII files. Details .
<i>Exit</i>	Closes down the program.
View	
<i>Zoom out</i>	Zooms out the image. Details .
<i>Decrement Zoom</i>	Decrements zoom. Details .
<i>X, Z, V under Cursor</i>	A switch: shows/hides the hint window under the cursor displaying the context information.
<i>Color Spectrum</i>	A switch: shows/hides spectrum image.
<i>Right Margin</i>	A switch: shows/hides the entire right margin.
<i>Velocity Profile on a Row</i>	A switch: shows/hides the window with horizontal velocity profile along the grid row that is currently under the cursor.

<i>Repaint at Resize Time</i>	A switch: repaints/does not repaint the image at window resize time. The default mode is "not repaint" to avoid chaotic blinking while resizing.
<i>Update View</i>	Repaints the current plotter content.
Properties	
<i>Grid Properties</i>	Displays the dialog with grid view settings (see below).
<i>Horizon List</i>	Displays the model horizon list (see below).
<i>Rubber Band Properties</i>	Displays the dialog for adjusting colors and line width of the rubber-band.
<i>Scroll Properties</i>	Displays the dialog for changing scroll increments.
Help	
<i>Introduction</i>	Displays the first section of this document.
<i>Help TOC</i>	Displays Table of Contents and section with XTomo-LM 3 overview.
<i>Model Viewer</i>	Displays the first section of this chapter.
<i>Hot Keys</i>	Displays information on the hot keys for navigation and selection.
<i>About</i>	Displays information on the product and the module.

Grid properties

The *Properties/Grid Properties* menu display info on the model and grid. Fig. 2a shows the *Grid Properties* dialog.

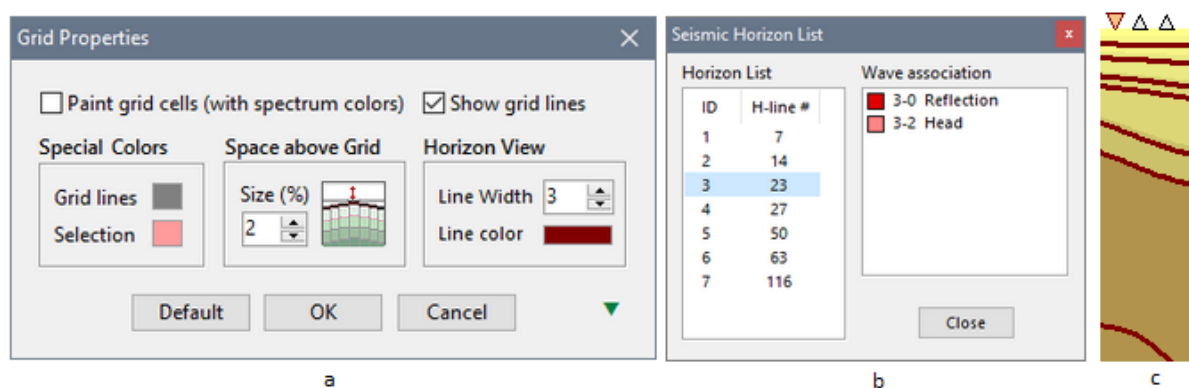


Fig. 2. Dialogs invoked from the Properties menu.
a – grid properties; b – horizon list; c – empty space above grid.

The *Paint Grid Cells* flag controls painting grid cells. If the box is checked, they are painted with the color defined by the velocity color spectrum. If the *Show Grid Lines* box is checked, grid lines become visible. One can switch both flags using drop-down menu of the tool button matching the *Grid properties* command.

On the *Special Colors* panel, one can change the cell select color and grid line color. The *Space above Grid* panel contains a spin-edit field for empty space height over the model, which is useful for drawing sources or receivers when they are located on the model top (fig. 2c). The height is defined as percentage of the model rectangle height. The *Horizon View* panel allows adjusting color and width of the horizon line.

The down arrow button at bottom-right drops down the hidden part of the dialog with additional information: grid size; actual velocity range; minimal distances between h- and v-lines in absolute units and with respect to space resolution.

The *Seismic Horizon List* dialog (fig. 2b) displays model horizons. Every item contains the horizon ID and the corresponding h-line number. For the selected horizon, the *Wave Association* field lists waves generated by this horizon and found in the project wave list.

2 Zoom and Selector

Tools for image magnification and selection of grid subsets are called *Zoom* and *Selector*. Selector allows marking image elements as targets for editing. Both tools use the *rubber-band* and its menu (*rb-menu*).

Rubber-band

To stretch the rubber-band, press left mouse button and drag the point pinned by the cursor in a direction that is neither vertical nor horizontal. A rectangle appears on screen stretching in step with dragging. At the moment the mouse button is released, the rectangle is redrawn in another color, and *rb-menu* is displayed with the *Magnify...* and *Select...* commands. Fig. 1 shows *rb-menu* and plotter menu.

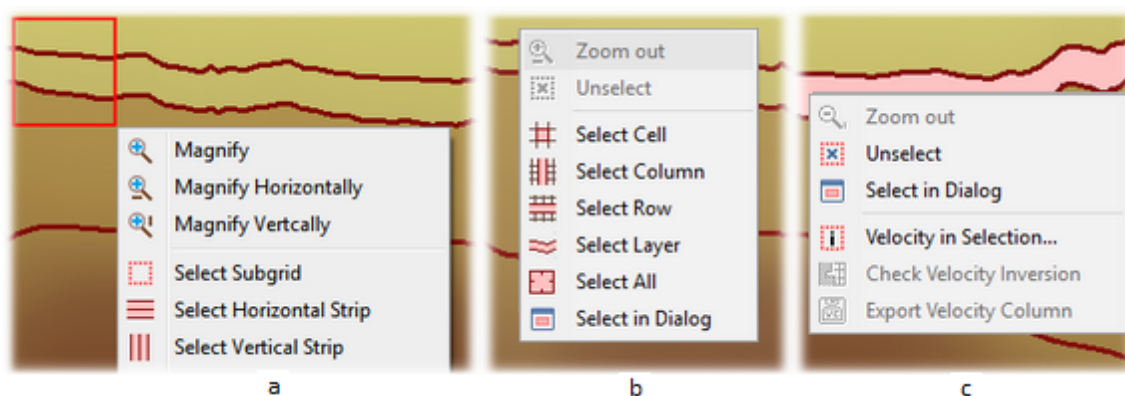


Fig. 1. Selector. a – *rb-menu*; b – plotter menu with no selection; c – plotter menu in the presence of a selection.

Select Options

One can select only standard grid subsets. A subset is unambiguously defined either by a given plotter point (a cell, a row or a column) or by a given rectangle (a subgrid or a strip). Subsets of the

first type are selected by commands of the plotter menu which make use of a point right-clicked to invoke the menu (fig. 1b). Subsets of the second type are selected by rb-menu commands (fig. 1a). Selected subsets are painted in the select color defined in [grid properties](#). Description of the selected subset appears under the plotter; it looks like this: *Row 14 – 54; Col 45 – 67*. The *Velocity in Selection* command of the plotter menu displays velocity range and mean value in the selected subset. As fig. 1b and 1c show, the plotter menu depends on whether there is a selection on the grid or not. Moreover, it depends on the kind of selection.

Before making a new selection the existing selection must be canceled by double-clicking the image or using the Unselect command. The latter can be found on the plotter menu or the tool bar.

Another option is selecting in a dialog (command *Select in Dialog* on fig. 1a and 1c). The dialog is shown on fig. 2.

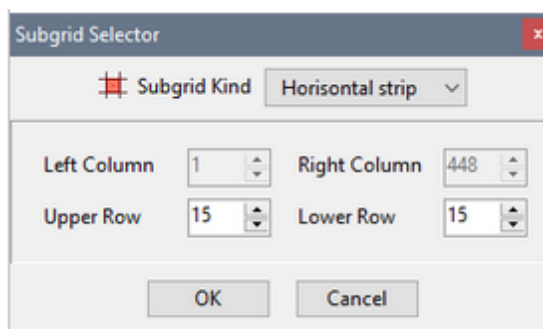


Fig. 2. Dialog *Subgrid Selector*.

Select standard subgrid type in the drop-down list on the top panel; then enter bounding columns and rows numbers in the spin-edit fields (border is included in a subgrid); then click on *OK*. This way of selection is helpful for thick grids or when subgrid location is exactly known beforehand. The dialog can be invoked irrespectively of whether there is a current selection or not.

Zoom and scrolling

The zoom tool stretches the rubber-band content up to size of the entire plotter. The *Magnify...* commands are in the rb-menu (fig. 1a). The vertical and horizontal magnifying works only in one direction. Magnification can be iterated until the domain on the plotter becomes comparable with resolution. When the image is magnified, its scrolling within the plotter frame is provided with:

- scroll bars (vertical and horizontal);
- mouse wheel (scrolls vertically or, with Shift pressed, horizontally);
- dragging the image with the Ctrl key pressed (any direction).

To cancel magnification, use the *Zoom out* command of the *View* menu or the plotter menu. Magnification can be decremented by steps using *View/Decrement Zoom* command. Both commands are duplicated by tool buttons.

Hot keys

Some zoom and select operations can be carried out using mouse and key-modifiers as shown in Table 1.

Table 1. Special ways of work with Zoom and Selector

Operation	Shortcut
Zoom out	Ctrl + Z
Scroll under Zoom	Ctrl + drag the image
Select grid cell	Double-click on the cell
Select grid column	Ctrl + double-click on the column
Select grid row	Shift + double-click on the row
Unselect all	Double-click on the image
Select subgrid	1. double-click on the top-left subgrid cell (the cell gets selected); 2. Ctrl + double-click on the bottom-right subgrid cell. This is the only way to select a large magnified subgrid.

Selected column

If a column is selected, two last commands of the plotter menu become accessible (fig. 1c). The *Velocity Inversion Point* checks whether there is velocity inversion in the column when moving from top to bottom. The *Export Velocity column* adds the selected column to a specified VC file.

When a column is selected, vertical velocity profile on the right margin gets frozen: it does not change along with the moving cursor. The selected column number appears under the plot. Finally, when the cursor moves over the profile plot, the coordinates (Z, V) are displayed in the hint window. The *horizontal velocity profile* window behaves similarly when a grid row is selected.

3 Export

Export to ASCII files

Model components can be exported to ASCII file for data exchange with other applications. The *File|Export Model to ASCII File* command displays the dialog shown on fig. 1.

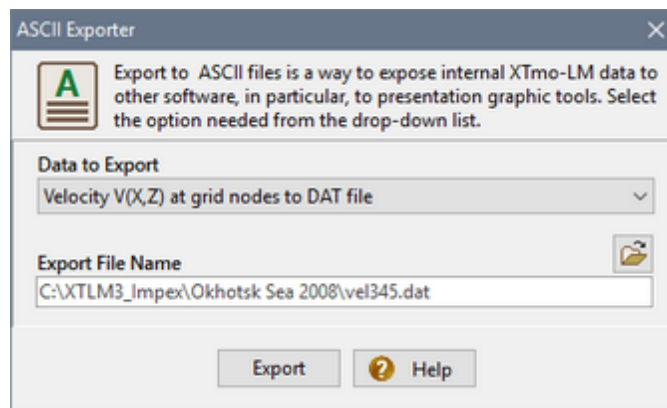


Fig. 1. Dialog for ASCII export.

The *Data to Export* drop-down list includes the following options:

- velocity distribution $V(x, z)$ to grid nodes to VFT file;
- velocity distribution $V(x, z)$ to grid nodes to DAT file;
- model horizons to MG file;
- model horizons to BLN file.

VFT and MG are XTomo-LM formats. The DAT and BLN formats are used for similar aims in other applications including Surfer™. When exporting horizons, check the *Include Model Top* box to export the model top along with horizons. After data to export are defined, select file in the Window file dialog and click on the *Export* button. The file dialog, when called for the first time, displays the project import/export folder. Another export option was described in the previous topic: adding velocity column to a VC file.

Export to graphic files

The plotter image in a graphic module can be exported to a graphic file. That does not mean that the file content is simply a snapshot of the plotter frame. First, the user can export any rectangle model subdomain and specify its pixel size. Then he or she can define axis with ticks and labels and headings. The *Image Export* dialog is invoked by the *File/Export Model Image* command. Both its tabs are shown on fig. 2.

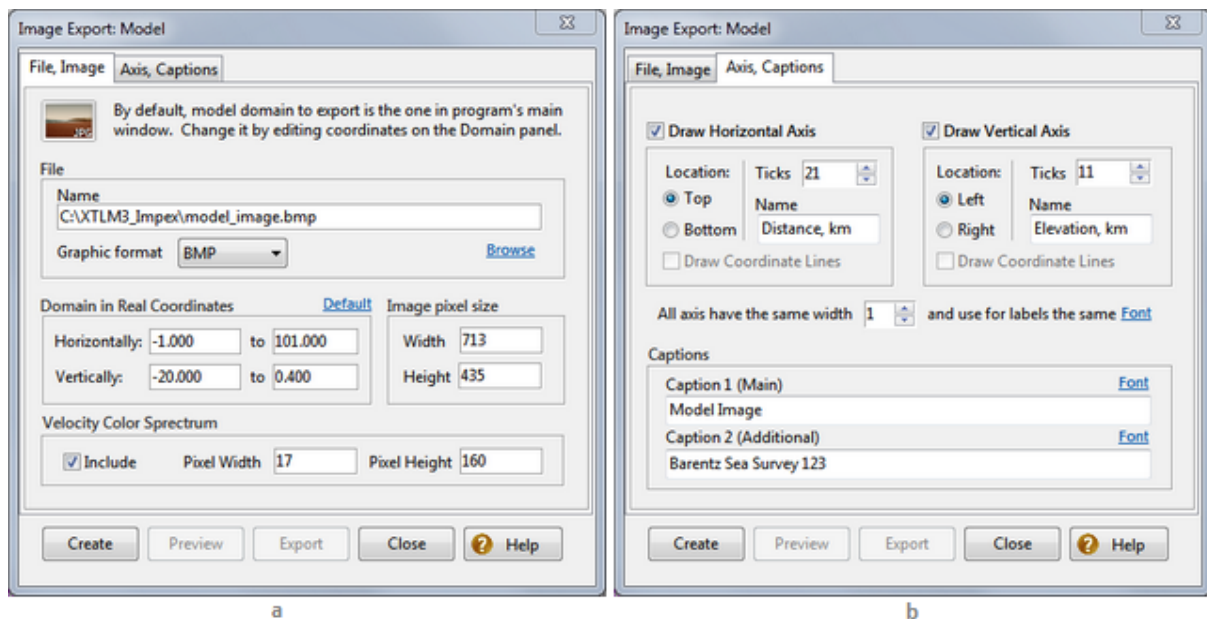


Fig. 2. Graphic export dialog. a – tab 1; b – tab 2.

Export is performed in three steps:

- 1) creating task by way of filling out or editing the dialog fields;
- 2) creating the image (the *Create* button) and studying the preview (the *Preview* button);
- 3) export to a file of the specified format (the *Export* button).

Specifying format and file path name may be postponed till step 3. On the *Domain in Real Coordinates* panel, the user defines a rectangular subset of the model. By default, it coincides with the current content of the screen plotter frame. The *Default* link restores this subset. When the user edits the range fields, the values are automatically corrected if they come out of model dimensions.

On the *Image Pixel Size* panel, the user defines the target image size in pixels. By default, it is plotter frame size. On the *Velocity Color Spectrum* panel, the user checks the box to include spectrum image and adjusts its size.

The second dialog tab (fig. 2b) is for axes and captions. The *Draw Axis* flags mean "draw/not to draw" the matching axis. There is a panel for each axis to specify its location with respect to the image (the *Top/Bottom* or *Left/Right* flags), number of ticks (*Ticks*) and axis name (*Name*). Below, the axis line width and label/name font are specified. To know the currently set font, position the cursor over the *Font* link. The font properties appear in the hint window.

On the *Captions* panel, define the print captions and their fonts. Both caption are printed above the model image. If the caption field is empty, the caption is absent on the picture.

After the task is ready, click on the *Create* button and wait until the picture is created. The *Preview* button outputs the picture with the application which is fixed in Windows for BMP files. If the preview is approved, select export file name (the *Browse* link) and graphic format. There are four

to choose from: BMP, GIF, PNG, JPG (without compression), all with color depth 24 bits. It is possible to output the preview picture successively in each format.

4 Editing Model

Model Editor (MED)

This and the following sections of the chapter are devoted to Model Editor (MED). It is launched by the command of the Processing Tree menu on the childless root node. If an m-node has at least one child, the model can be only viewed. MED is a [graphic module](#) inheriting all the functionality of Model Viewer including export. Common editing features are:

- reloading model from disc at any moment (*File/Reload*);
- saving the current state of model at any moment (*File/Save*);
- cancellation of all changes made in the session before exit;
- support of rollback stack and Back/Forward commands to cancel last changes and restore them.

The edit commands can be found in the *Edit* menu, *rb*-menu and *plotter* menu. Edit operations are explained in following order:

- changing grid thickness (frequency);
- changing velocity;
- changing grid geometry;

Modifying model

In the previous section, changing velocity and grid geometry are listed as independent model component. Clearly, such discrimination is made only for convenience: after all, velocity is defined at grid nodes. Often grid is a means of digitization of continuous functions. At that, due to continuity, replacing one grid with another is not essential, if only both grids are thick enough. In XTomo-LM, grid can serve both as a means of digitization and as the way to depict structural nonuniformity. That's why the user should have an idea of what is going on "under the hood" at various transformations of the model.

For example, what is actually going on when a velocity column $C = \{ (z_k, v_k), k = 1, 2, \dots, n \}$ from a VC file is implanted into the grid? The answer is this: the program finds the grid vertical U with the x -coordinate nearest to x of C in the file; then C is redefined on the z -net formed by grid nodes on U ; the way velocity from C is calculated at nodes on U depends on how C is treated: as step function or as piecewise linear function (see [here](#)). If velocity on the grid is formed by a velocity column set, each column replaces velocity in a grid column, as explained above, and then velocity is reconstructed on each grid row using linear interpolation. If some structural features are lost after the operation, the user has to restore them using available editing tools. Description of other model transformation will contain the detailed explanation of internal machinery.

Editing the model top line

This operation (if required) can be performed only in the very beginning of processing, when there is only one node Model 1 with the starting model. Changing the model top in other m-nodes is forbidden. Only at this moment, one can enter a leveling section which becomes first grid h-line. At that, the model rectangle can change: its top may go down, so what Model Viewer or Model Editor displays as a top, may be less than in project properties. The operation does not differ from common operation of editing/importing of an h- line described in "Changing geometry", in particular, [here](#).

5 Changing Grid Thickness

Commands

Grid line frequency defines details of variability of velocity and h-line shape. First, select an appropriate grid subset using [selector](#). Then invoke the plotter menu and choose one of the command collected in Table 1. The *Selection* column shows to which subsets the command can be applied.

Table 1. Commands of changing grid thickness

Command	Selection	Action
<i>Insert Rows</i>	row	Inserts the specified number of h-lines inside the row, possibly, with variable step (see below).
<i>Insert Columns</i>	column	Inserts the specified number of v-lines inside the column, possibly, with variable step (see below).
<i>Delete Selected</i>	row, column, strip	Deletes all selected rows or columns.
<i>Double Number of Rows</i>	row, h-strip	Doubles number of rows in the selected subset (h means "horizontal").
<i>Halve Number of Rows</i>	h-strip	Halves number of rows in the selected strip.
<i>Double Number of Columns</i>	column, v-strip	Doubles number of columns in the selected subset (v means "vertical").
<i>Halve Number of Columns</i>	v-strip	Halves number of selected columns in the selected strip.

Inserting rows and columns

The *Insert Rows* command displays the dialog shown on fig. 1. The user defines the number N of h-lines to insert, and step modifier M . M is a real number from the interval $[0.5, 2.0]$. If $M = 1$, new h-lines are equidistant. Otherwise, distance between them will grow top to bottom, if $M > 1$ or drop, if $M < 1$ exponentially as power of M .

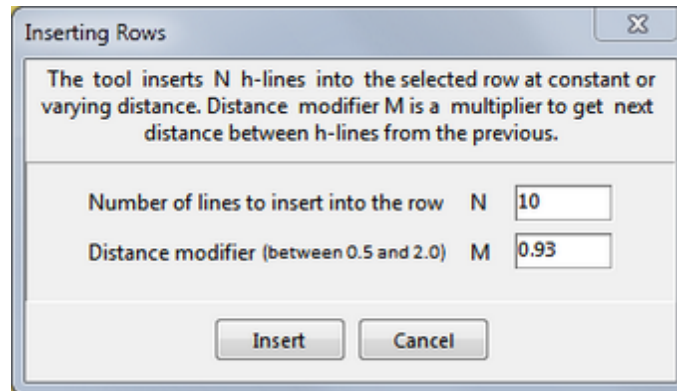


Fig. 1. Dialog for inserting rows.

Inserting columns is carried out similarly. The *Insert* and *Double* operations can cause the [resolution error](#). If it happens, the operation is canceled and the grid stays intact.

6 Changing Velocity

[The Editing Velocity dialog](#) – [Operations](#) – [In-place editing](#) – [Copying velocity](#) – [Replacing velocity in a grid column](#) – [Replacing velocity function on entire grid](#) – [Removing black cells](#)

The *Editing Velocity* dialog

Operations of changing velocity normally affect a selected grid subset. The *Edit Velocity* command of the *plotter* menu invokes the *Editing Velocity* tabbed dialog. When invoked, it displays the only tab *Options* with the list of operations (fig. 1a). Which of them can be performed depends on the kind of selection. After checking the option, the *Operation* tab becomes visible. It contains the operation details (fig. 1b). After clicking *OK*, the operation is carried out. The modified velocity values must stay inside the current velocity range. If they do not, they are replaced silently with the nearest range bound. If the *Unselect* flag is checked, selected subset gets unselected just after the operation.

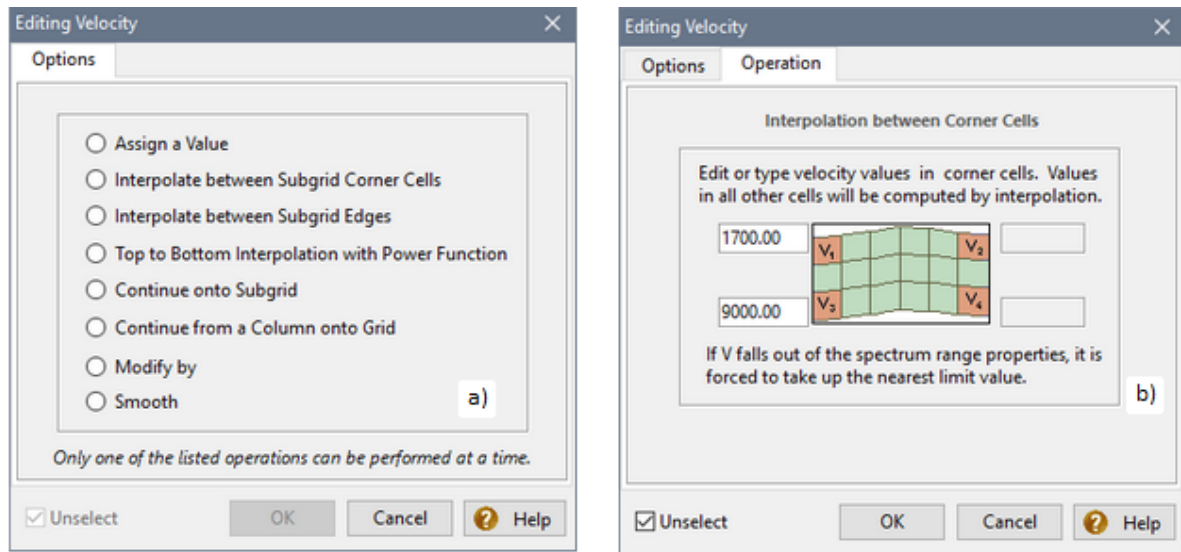


Fig. 1. Dialog for editing velocity. a – tab 1: operation list; b – task for an operation.

Operations

The commented operation list is presented in Table 1.

Table 1. Operations.

#	Operation	Description
1	<i>Assign a Value</i>	In all selected cells, velocity is set to the value typed into the edit field. The assignment can be implemented without calling the dialog. Move the cursor over the spectrum image until the hint window under the cursor shows the needful value. Double-click the spectrum at this spot, – and the selected subset gets the pointed velocity value. Instead of double-clicking, one can right-click the spot and choose in the popup menu the <i>Assign to Selected</i> command.
2	<i>Interpolate between Subgrid Corner Cells</i>	Enter velocity values in the edit fields corresponding to the corner cells of the subset (fig. 1b). After clicking OK, velocity in the rest subset cells is computed using linear interpolation.
3	<i>Interpolate between Subgrid Edges</i>	There are two options: (1) interpolation between side columns and (2) between top and bottom rows. An option is selected from the drop-down list. If (1) is selected, velocity V in an arbitrary cell is calculated by interpolation between values at the leftmost and rightmost cells of each row. If (2) is selected, V is computed by interpolation between values at the top and bottom cells of each column.
4	<i>Top to Bottom Interpolation with Power Function</i>	The operation applies to a column or vertical strip and forces velocity in each selected column to increase as the specified

		power of depth from value V_{\min} at the top cell down to value V_{\max} at the bottom cell. The power index is selected from interval $[0.1, 1.1]$. If $V_{\min} \geq V_{\max}$, the column stays intact.
5	<i>Continue onto Subgrid</i>	The user selects one of four options in the drop-down list. An option defines one of the border rows/columns as a source. For example, let the source be the leftmost column. Then velocity value in a subset cell will be the same as in the leftmost cell of the row. If the source is the top row, velocity value in a subset cell will be equal to its value at the top cell of the same column.
6	<i>Continue from a Column onto the Grid</i>	The option is accessible if the selection consists of one column. The three options of continuation (to the left, to right and in both directions) are the items of the drop-down list. The user selects the item and clicks on <i>OK</i> . All columns in the chosen part of the grid become clones of the selected one.
7	<i>Modify by</i>	The user chooses between (1) <i>absolute</i> or (2) <i>relative</i> change using radio-buttons. In the case of (1), the correction entered into the edit field is added to velocity values of every selected cell. In the case of (2), the correction is interpreted as percentage and so the changes are relative.
8	<i>Smooth</i>	Moving-average smoothing assigns to a cell (Col, Row) the value equal to the mean value of velocity values in rectangle $[\text{Col} - R_x, \text{Col} + R_x] \times [\text{Row} - R_z, \text{Row} + R_z]$. The user must specify the R_x and R_z parameters defining the degree of smoothing. If the grid selection is a row, then, additionally, moving-median smoothing is available. It is especially helpful for eliminating velocity oscillations having no physical or geological sense.

In-place editing

In many cases it is not convenient to work with the dialog, because it blocks access to the image. As an alternative, the tool Velocity In-place Editor (VIP) can be used. It is a small window (fig.2) floating over the plotter.

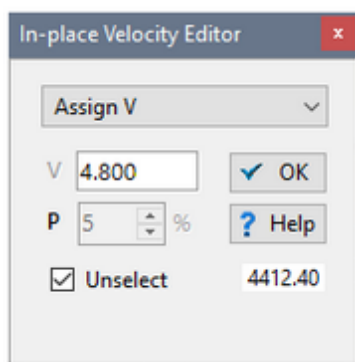


Fig. 2. Velocity In-place Editor.

The tool is invoked by the *Invoke Velocity In-place Editor* command of the plotter menu. It doesn't matter if there is a selection or not yet: at any moment the user can switch to the main window to zoom in the image or change selection. However, it cannot hang all the time because it blocks some editing operations. The drop-down list of VIP operations is at the top. VIP implements only part of operations from table 1. Along with the list, VIP displays two edit fields *V* and *P* for velocity value to assign and for percentage by which velocity should be changed. The *OK* button carries out the currently selected operation. The *Unselect* flag has the same purpose as in the dialog.

When the *Increase* or *Decrease* operation is chosen, the flag is automatically reset to allow multiple use. Under the *Help* button, velocity in a cell pointed by the cursor is shown.

Copying velocity

The user can copy velocity distribution from a selected subset *S* to another subset *T* of the same type. *S* and *T* must not intersect. It is clear that *T*, being of the same type, may have quite different shape, so the cell-to-cell mapping is implemented by a cell position relative to the subset top-left. Copying is performed this way. After *S* is selected, apply the *Copy Velocity* command in the plotter menu. A message appears on screen with information what should be double-clicked to identify *T*. The place depends on the subset type. Say, for column *S*, the target is any cell of *T*, for a subgrid – the top-left cell of *T* and so on. Leaving the message as it is, perform the double-click to start copying. Response to the message must be quick enough, no more than 10 seconds, otherwise the operation is canceled. The time elapsed is shown by the progressor attached to the message bottom.

Replacing velocity in a grid column

Model Viewer allows replacing velocity in a selected grid column by importing the **first** velocity column from a VC file. To do that, select a grid column and apply the *Import Velocity Column* command of the plotter menu. It displays the *Import Velocity Column* dialog to choose a VC file and set how to treat the velocity column: as step function or piecewise linear function. A click on the dialog's *OK* button starts the operation. The value of velocity column's *X* in file is not used. About how replacing is performed, read [here](#).

Replacing velocity function on the entire grid

Replacing velocity on the entire grid is implemented for two velocity sources: (1) another m-node of the project and (2) a velocity column set stored in a VC file. For each case, there is a command in submenu *Edit/Replace Velocity*, which opens a dialog to let the user select m-node or VC file. In case of VC file, the user, additionally, defines how to treat velocity columns. Redefinition of the velocity column set on the current grid is performed as explained [here](#). In case of (1), the source velocity defined on the source grid must be redefined on the current grid. This is done similarly to the case of (1), because the source model can be regarded as a set of velocity columns $V(z)$ treated as step function of *z* according to grid [definition](#).

Removing black cells

Black cells signal to the user breaking data integrity. To remove black cells, use the *Edit/Remove Black Cells* command. It checks all grid cells to find those in which velocity value falls beyond the current velocity range. Each such value is replaced with the nearest range bound.

7 Changing Geometry

[What does it mean?](#) – [Grid Perturbation Area \(GPA\)](#) – [Work with GPA](#) – [Moving an h-line](#) – [Editing an h-line](#) – [Importing an h-line](#) – [Changing model top](#) – [Seismic horizons](#) – [Velocity and changing geometry](#)

What does it mean?

We speak of changing grid geometry if one of h-lines changes its position or shape or a new h-line is inserted in the grid. Accordingly, the main operations of changing geometry are vertical shift and editing/importing an h-line. Changing geometry implies rebuilding, at least, a part of the grid to avoid intersection of h-lines. An example of grid perturbation caused by changing the shape of an h-line shown on fig.1.

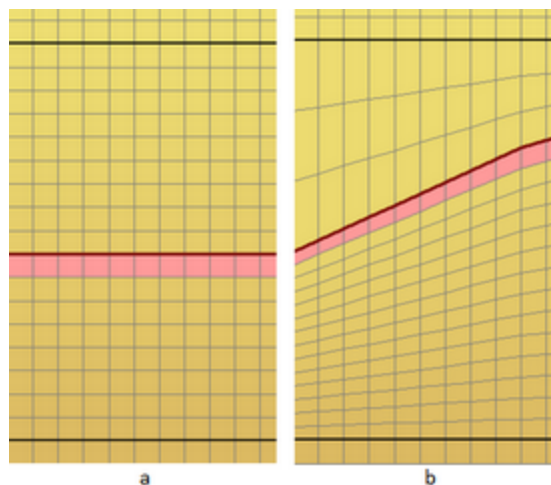


Fig. 1. Remeshing caused by changing the form of horizon. GPA is bounded with heavy black h-lines.
a – initial view; b – after horizon has been changed.

The term *grid perturbation* is used to describe the effect shown on fig. 1. Just after an h-line has been edited or imported, it becomes a *special* object whose shape is fixed, at least, for the moment, while the neighboring h-lines are regarded as *ordinary* and are changed or replaced automatically. Thus, some h-lines are formative for geometry, the other serve to define velocity function with sufficient accuracy. Those special h-line form model [wireframe](#). Their shape can be changed only "manually". Usually, they "underlie seismic horizons. On fig. 1, the horizon being edited is selected. *To select an h-line* means to select the row for which it is a roof.

Grid Perturbation Area (GPA)



Grid perturbation is restricted by a horizontal strip called Grid Perturbation Area (GPA). An h-line to edit or import must lie strictly inside GPA. On fig. 1, GPA is bounded by the heavy black lines. Restriction of perturbation, in particular, allows inserting several horizons without distortion of

their shape. GPA is to be chosen so as not to contain inside it more than one horizon and if it contains one, it is the only possible object of editing.

Grid perturbation is implemented with the built-in algorithm. It is enough for the user to know the following facts:

- 1) number of lines in GPA before and after perturbation is the same;
- 2) when an h-line H is moved, numbers of lines above and below H are kept; the set of existing lines gets denser on one side of H and rarefied on the other; the value of shift is restricted only by occurrence of the [resolution error](#);
- 3) after an h-line H has been edited or imported, numbers of lines above and below H is proportional to minimal distances between H and GPA borders; if one of those distances is too small, the resolution error occurs.

Work with GPA

GPA is defined by its top and bottom h-lines. The easiest way to define GPA is to use the rubber-band, similar to selecting a horizontal strip. The rb-menu has a special command for that: *Define Grid Perturbation Area*. Another way is to invoke a dialog by the *Edit/Set Grid Perturbation Area* command or the matching tool button . The dialog allows viewing the current GPA borders and typing in new ones. If one of edit fields is left empty, GPA is canceled. One can also change drawing attributes of GPA borders. Fast commands for creation/destroying GPA can be found in the drop-down menu of the button .

Moving an h-line

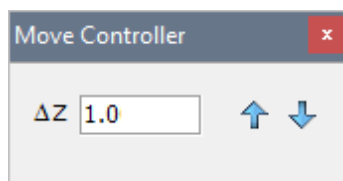


Fig. 2. Move Controller

Select an h-line and issue the *Move H-line* command of the plotter menu. The *Move controller* dialog appears on screen (fig. 2). Type the move step value in the ΔZ field. At each click on the arrow button, the h-line moves by ΔZ in arrow's direction. If the desired shift is known beforehand, type it in the field and click on the button once. When the h-line attains the required position, close the dialog. To cancel shift use the *Undo button*.

Editing an h-line

To edit an h-line, select it and choose the *Edit H-line* command in the plotter menu. It makes Model Editor to start the UMG utility in the special mode of H-line Editor or UMG/HE, for short. In this section, only its difference from UMG is described; for the details of curve editing, address the UMG [topic](#). The main window of UMG/HE is shown on fig. 3.

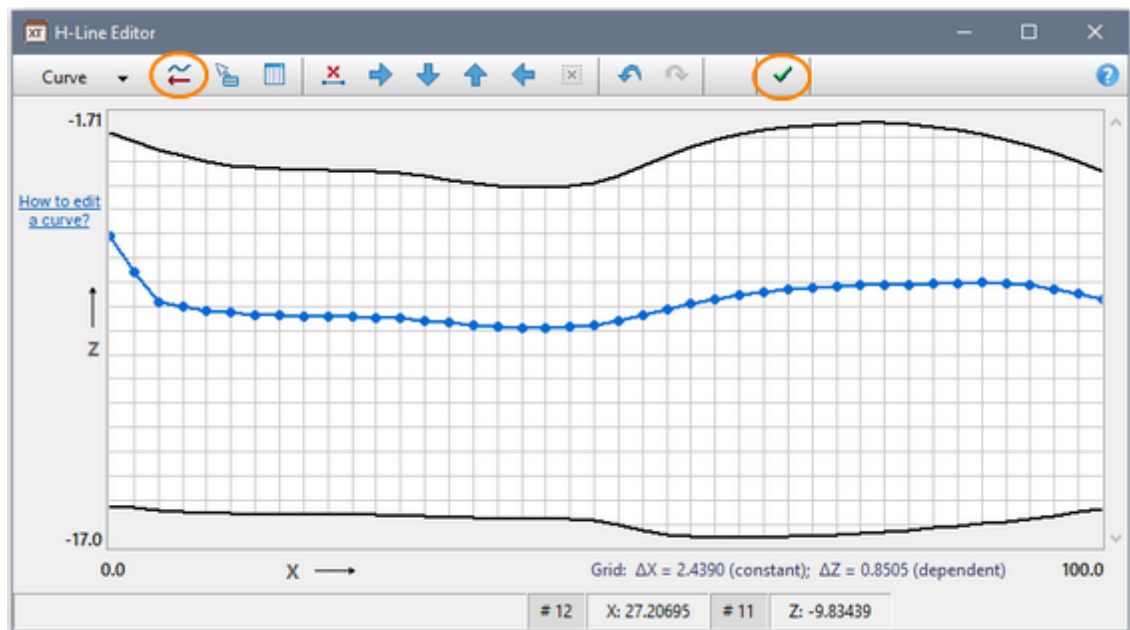


Fig. 3. H-Line Editor's main window. The new *Import curve* and *OK* buttons are circled.

The plotter depicts GPA and inside it – the uniform UMG grid and the h-line to be edited. A number of x-net nodes is equal to $\min(N, M_{\max})$, where N – is the number of verticals in the model grid, M_{\max} – maximal number of nodes allowed by UMG. Thus, the h-line is transferred to UMG exactly only in some cases, but the error of redefinition it on another x-net is acceptable for "manual" editing. There is only one menu button *Curve* on the toolbar, but with the two new buttons (*Import Curve* and *OK*), the entire editing tool kit is available. To cancel editing, close the window with the system button; to save the result, click on the OK button. In both cases, the user returns to Model Editor, and sees the saved curve implanted in GPA. Note, that importing a curve from an MG file is regarded as editing option.

Importing an h-line

After GPA is defined, select the Import H-line command of the *Edit* menu or its duplicate on the tool bar. The command invokes the H-line Import dialog shown on fig. 4.

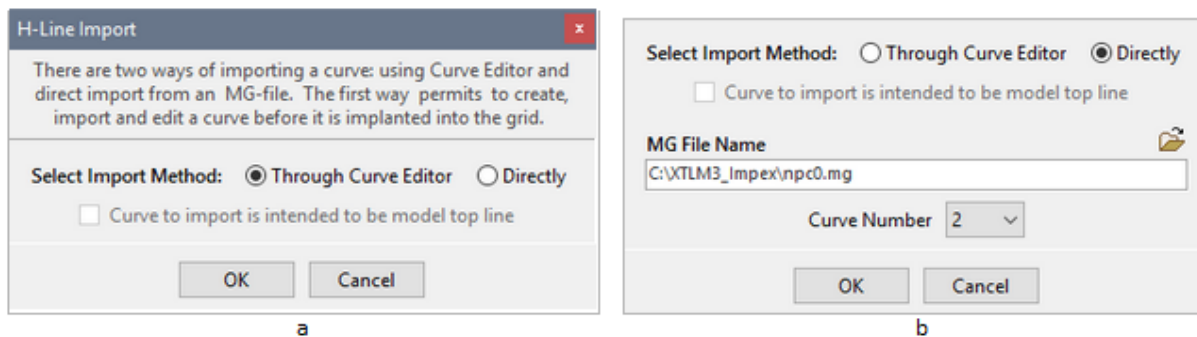


Fig. 4. H-line Import dialog. a) import through H-Line Editor; b) direct import.

By default, import is supposed to be performed through H-line Editor. That allows viewing and doing some editing before implanting the curve in the grid. A click on the *OK* button invokes H-Line Editor's window (fig. 3) with empty GPA. Now the user clicks on the *Import curve* button to start the import operation, then edits the curve if needed and, finally, clicks on the *OK* button to save the curve and insert it in the grid.

The second import option is *Directly*. It can be used when the curve has to be imported with maximal precision, avoiding intermediate redefinition on the UMG grid and possible loss of points. Of course, that makes sense if only the model grid itself is agreed with the x-net on which the curve is defined. Probably, the higher precision is required for importing leveling section or bottom contour (see below). Once the option is selected, the dialog drops down the additional section (fig. 4b) to define MG file and, after the file is loaded, to select the curve number.

Changing model top

Though the operation is, in a sense, special, it is carried out following common guidelines. GPA must contain the first h-line H_1 , and H_1 must be selected. Below are some specifics of UMG/HE (fig. 3):

- the upper wireframe border coincide with model rectangle top;
- before editing, the upper wireframe border may not be visible being overlapped by H_1 ;
- permissible area for H_1 is bounded by the upper wireframe border exactly, without tolerance.

If H_1 is to be replaced by import, then in the import dialog (fig. 4a), the box *Curve to import is intended to be model top line* must be checked.

Seismic horizons

Formally, a seismic horizon is an h-line with special features:

- it is referenced, at least, in one item of the horizon list;
- it cannot be modified or replaced automatically in the course of grid perturbation;
- it cannot be deleted by regular command for h-lines;
- it is marked out visually on the model image.

A horizon can be excluded from the horizon list and then it becomes a plain h-line. Inversely, an h-line can be turned into a reflector or refractor if only there is an appropriate wave in the project wave list. To do the transformation, select the h-line/horizon and choose in the plotter menu the *Convert...* command. To understand line-to-horizon conversion, remember that wave ID has the form HW, where H is horizon ID and W is wave type. An "appropriate" wave is required to have the necessary type and horizon ID that agrees with the h-line position in the grid (horizon IDs increase with depth). Model Editor does the analysis and either informs of conversion failure or displays a dialog with the list of waves to choose from.

Velocity and changing geometry

Let \mathbf{C} and \mathbf{C}' are sets of GPA cells before and after an operation of changing geometry. It follows from the features of remeshing algorithm, that there is a one-to-one mapping $\mathbf{C} \rightarrow \mathbf{C}'$. For example, each cell from \mathbf{C} can mapped on a cell in \mathbf{C}' with the same position relative to the GPA

top-left and, hence, to the model top-left. After this note, we can formulate the last property of the remeshing algorithm:

- 4) after any operation of changing geometry, velocity stay attached to cells with the same double indexes (row, column).

Such is the default velocity behavior at changing geometry. Sometimes, that is good, sometimes – not. Say, when moving a reflector, such behavior keeps velocity jump that generates reflection and so can be accepted. In other cases changing the depth-velocity dependency cannot be ignored. It is possible to restore it by replacing the changed velocity with velocity from the previous m-node of Processing Tree. Such technique is persistently applied in examples of building horizons in chapter TX-curve inversion.

Observations

1 Overview

In any [cycle of processing](#), work with observations starts after a model has been properly edited. If a model is stored in node Model *N* of Processing Tree, the first step is to create the child node Observations 1. To do that, select the Model *N* node and apply the main menu *Processing Tree/ Create O-Node* command or the matching tool button. The created o-node will store the first version of the observation system for the m-node. If necessary, other o-nodes can be created in the same way.

Importing and storing observations

The term *observation system* (in the user interface replaced with *spread*) is used for a storage containing information about sources, receivers and rays related to field observations (I-project) or to a modeling job (M-project). At this stage, "ray" is used instead of "source-receiver couple". According to [Introduction](#), observation data are originally delivered in ASCII files – *observation files*. Import of all such files, regardless of a project, is implemented by the SRT Port Manager module which services a standalone XTo-mo-LM component called *SRT Port*. In the port warehouse, data from different observation files are stored under unique names. On each o-node of the project Processing Tree, the user runs the SRT Data Extractor (SRE) module to extract the named data from the port warehouse and store them in the o-node database called *Ray Catalog*. SRE can extract all the data or a user-defined data sample. The Ray Catalog database is a data source for modules implementing ray-tracing and TX-curve inversion for diving, reflected and head waves.

Forward problem solver uses Ray Catalog in an o-node as input data and places output version of Ray Catalog in an f-node. The output version contains only rays successfully traced. For each ray, the output Ray Catalog stores traveltimes and – in I-projects – time residuals. Thus, Ray Catalog databases can be found both in o- and f-nodes.

The user can copy Ray Catalog from o- or f-node back to SRT Port. the aim of such operation is observation exchange between different projects. In particular, it allows using Ray Catalog in an f-node for modeling observations.

About this chapter

In [SRT Port: Manager](#) and the next three sections, the SRT Port component is explained in detail. The [Ray Catalog: Populating](#) section explains, how, in the course of processing, observation data are brought from the port warehouse to the Ray Catalog of the current o-node. In section [Ray Catalog: Viewing](#) the ways of viewing the Ray Catalog content are explained as well export to

ASCII files and SRT Port. The last section, [Ray Catalog in M-project](#), is devoted to a graphic module whose user interface allows creating a spread from scratch as an alternative to import.

2 SRT Port: Manager

SRT Port

The *SRT Port* component includes the data warehouse for storing observation files' content and the software for warehouse maintenance: data import, viewing export, removing, archiving. A storage unit of the port warehouse is a database (or *SRT database*) – an internal image of an observation file once imported into the system. A database is identified with its name and type. A database of *SRT type* is generated by observation files of the SRT and #DT formats, while SR or S+R files are mapped into databases of SR type (see [Introduction](#)). The port warehouse content is presented to the user as two-level data structure of a set of data stores (folders) containing databases (files). Once more: *the port warehouse* is a set of *data stores* containing *databases*. The previous sentence fixes the terminology. The default data store, *.Main* by name, is always available to start the work. Other data stores are created by the user when necessary. Data in the port warehouse cannot be edited. Databases of SRT type contains observed traveltimes and are used later in I-projects. Databases of SR types contain positioning data for M-projects. The SRT Port warehouse is located on disc in a fixed folder not connected in any way to working or project folders.

SRT Port Manager

The SRT Port warehouse and functionality is represented for the user by SRT Port Manager. It is launched by Project Manager main menu command *Tools/SRT Port*. Fig. 1 shows the module's main window.

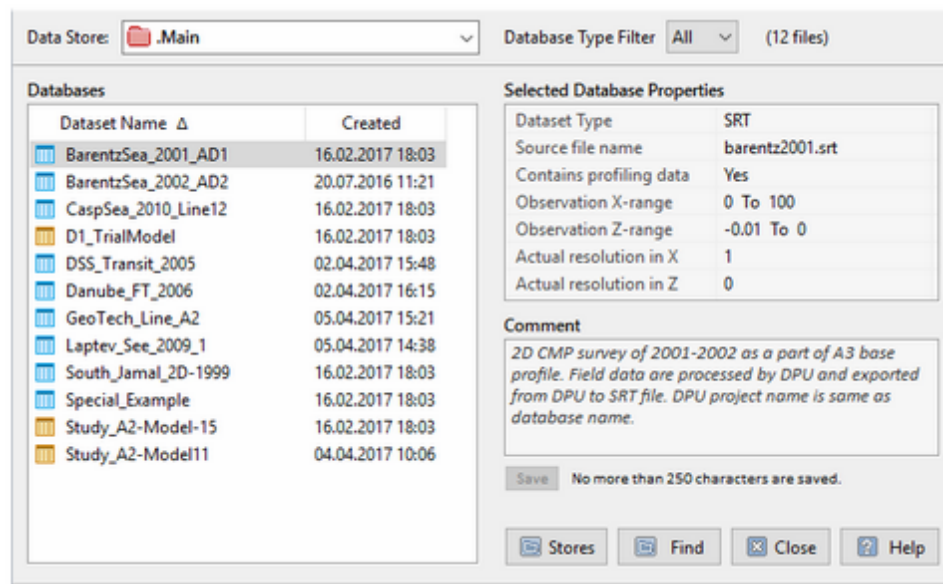


Fig. 1. SRT Port Manager main window.

At the windows's top, the drop-down list of data stores can be found. After module is started, the .Main data store is selected in the list. The left panel of the window bears the list of SRT databases stored in the selected data store. Databases of SRT type are marked with blue icons, while items of SR type have gold yellow icons. The list columns display database names and creation dates. By default, the list is sorted by name in increasing order (Δ). To change ordering, click on the column header. The right panel displays information relating to the database currently selected in the list – database properties: type, source observation file name; profiling data flag; observation domain (x-range, z-range); minimal distance between devices of the same kind (resolution in x, resolution in z). A separate field contains textual comment. This is the only piece of data the user can edit. To save changes, click on the *Save* button. The list allows multiple selection for group operations. If several items are selected, the right panel displays information pertaining to the first selected item. The user can display databases of one type only using the Database type filter drop-down list to the right of the data store field.

Managing databases

Work with databases of one data store is directed by the commands of the list database list popup menu described in Table 1.

Table 1. Module menu commands

Command	Description
<i>Import Observation file</i>	Reads input file, verifies it, and copies its content to a new SRT database with a user-defined name. Details are in the next topic .
<i>View as Table</i>	Allows viewing the content of the selected database in the form of numeric tables. Details are here .
<i>View as TX-curves</i>	Runs a graphic module for viewing the content of the selected database as a set of traveltimes curves. The command is accessible for profiling data only. Details are in this topic .
<i>Rename</i>	Allows changing the database name.
<i>Move To</i>	Moves all selected databases to another data store. The command displays the data store list (fig. 2a) in which the user double-clicks the target data store.
<i>Delete</i>	Removes selected databases from the port data warehouse.
<i>Archive</i>	Puts selected databases in one compressed archive file. Details are here .
<i>Extract from Archive</i>	Extract databases marked out by the user from an archive file to the port warehouse. Details are in this topic .
<i>Export</i>	Exports the content of the selected database to an ASCII file of one of input formats. Details are in this topic .

<i>Update Database List</i>	Creates the list anew reading off disc data.
<i>Data Warehouse Disc Size</i>	Displays current data warehouse disc size.

Managing data stores

The *Stores* button displays the *SRT Port Data Store List* dialog (fig.2a) for managing data stores. The list context menu contains commands for creating a new data store (Create), changing data store name (Rename) and removing (Delete) a data store with its content. The Rename and Remove commands relate to the store currently selected in the list. Data store name has no more than 25 characters allowed to use in file names. Data store name is unique (database name is unique within data store). Just after creating, a new data store can be selected in the Data store drop-down list on the top panel.

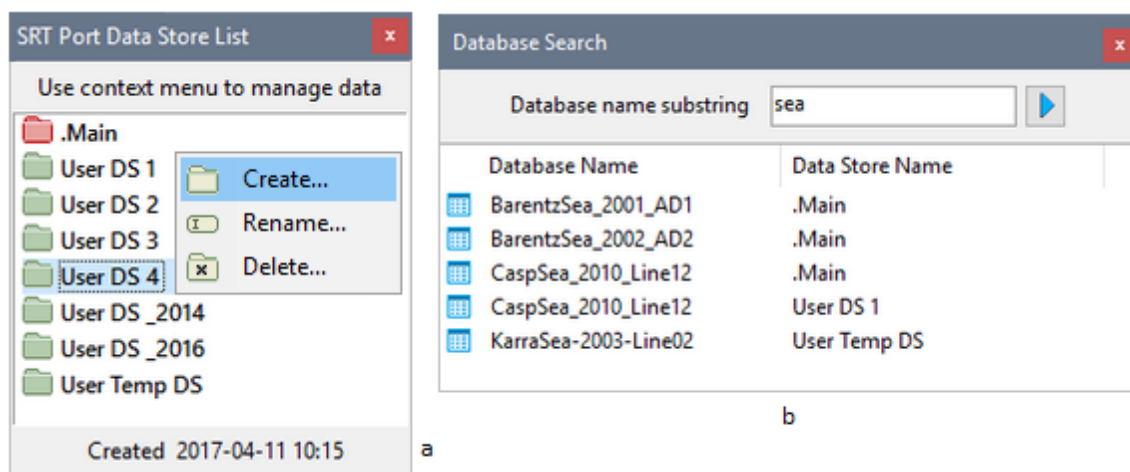


Fig. 2. Managing data stores. a) Data store list. b) Database search dialog.

The other managing tool is database search across all data stores. The *Find* button displays the Search dialog (fig. 2b). Enter a search pattern (a substring of the database name) in the edit field and click on the button. Search results are presented in the table with the *Database Name* and *Data Store Name* columns. To view a database found, double-click an item in the result table. In the main window, the data store needed will be opened and the required database selected in the database list. Now apply any view command in the database list menu.

3 SRT Port: Import

Importing an observation file

The *Import Observation file* (Table 1) starts the Importer module. For any file of permitted format, Importer does the following: (1) reads the file detecting all format errors; (2) verifies the file content; (3) creates a new database in the current data store. In the course of verification, data

unambiguity is thoroughly checked and duplicate rays uncovered. Unambiguity means that devices with the same IDs have the same coordinates and that, vice versa, no more than one source or receiver is located at the same point of the (x, z) plain. Importer detects all errors and logs them providing each with a reference to a file line number.

The main Importer window is presented on fig. 1a. First, click on the file button and set file filter for the type needed in the Open file dialog (fig. 1b).

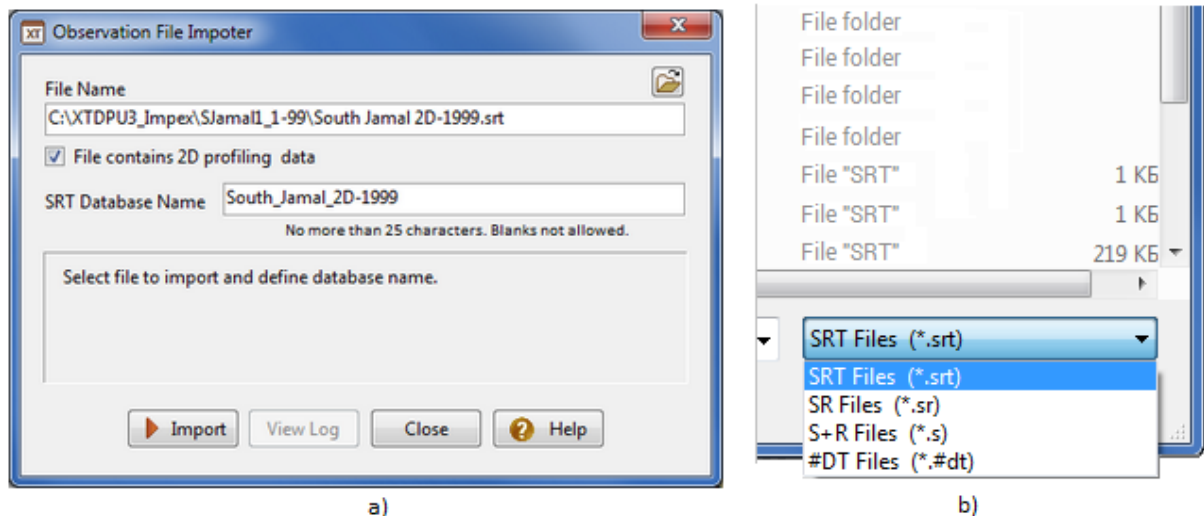


Fig. 1. Import module. a) main window; b) selecting format in the file filter drop-down list.

If file selected is of SRT or #DT format, set profiling data flag correctly taking into account the definition. Finally, define the new database name. By default, it is the same as file title with blanks replaced by the underscore character (blanks are not allowed). Maximal number of characters in the name is 25. To start import, click on the *Import* button. If the operation is completed successfully, Importer closes, and the new database name appears in the Manager database list. Otherwise, the user can view the log (the *View Log* button) and study the error list. The log is displayed in XTomo-LM text viewer (fig. 2) which allows saving log to a user-defined file, if necessary. The save command can be found in the viewer's popup menu.

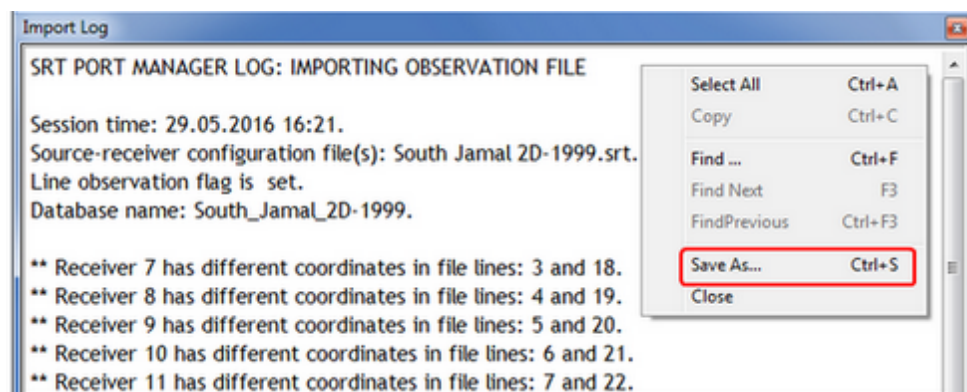


Fig. 2. A fragment of import log.

Viewing SRT database content

The *View as Table* command (Table 1) displays the selected SRT database content as three numeric tables (fig. 3): the source table, the receiver table and the ray table.

SRT Database Viewer [DSS_Transit_2005]

SID is source ID; RID is receiver ID. For a database of SRT type, Time is observed travel time of the wave selected in the Wave drop-down list. Double-clicking source or receiver list switches between sorting by ID and (X,Z). Ray list ordered by (SID, RID).

Sources (11)			Receivers (101)			"Rays" (1111)		
SID ↓	X	Z	RID ↓	X	Z	SID	RID	Time
1	0.000000	-0.010000	1	0.000000	0.000000	1	1	0.005000
2	10.00000	-0.010000	2	1.000000	0.000000	1	2	0.525000
3	20.00000	-0.010000	3	2.000000	0.000000	1	3	1.050000
4	30.00000	-0.010000	4	3.000000	0.000000	1	4	1.573000
5	40.00000	-0.010000	5	4.000000	0.000000	1	5	2.096000
6	50.00000	-0.010000	6	5.000000	0.000000	1	6	2.609000
7	60.00000	-0.010000	7	6.000000	0.000000	1	7	3.114000
8	70.00000	-0.010000	8	7.000000	0.000000	1	8	3.610000
9	80.00000	-0.010000	9	8.000000	0.000000	1	9	4.097000
10	90.00000	-0.010000	10	9.000000	0.000000	1	10	4.573000
11	100.0000	-0.010000	11	10.00000	0.000000	1	11	5.037000
			12	11.00000	0.000000			

Wave: * 0 Diving

Fig. 3. SRT Database Viewer. Database of SRT type is displayed.

Device tables are sorted by device ID. The ray table displays rays of one wave structured by blocks, each one containing all rays from one source. The wave is selected from the drop-down list under the table. The ray is represented by source and receiver IDs and traveltime. All said relates to a database of SRT type. The view of an SR database has neither wave list nor the time column in the ray table.

4 SRT Port: TX-curves

Viewing TX-curve sets

If an SRT database contains 2D profiling data, its content can be represented as a set of TX-curve sets. Such databases can be viewed graphically by the module TX-Curve Viewer. The module depicts the database content as a set of TX-curve plots on the (x, t) plain. The module is launched by Manager's menu command *View as TX-curves*. It works with the database selected in the list. The module allows detailed viewing TX-curve sets and, in a sense, analyzing apparent velocities. Fig. 1 shows the module's main window.

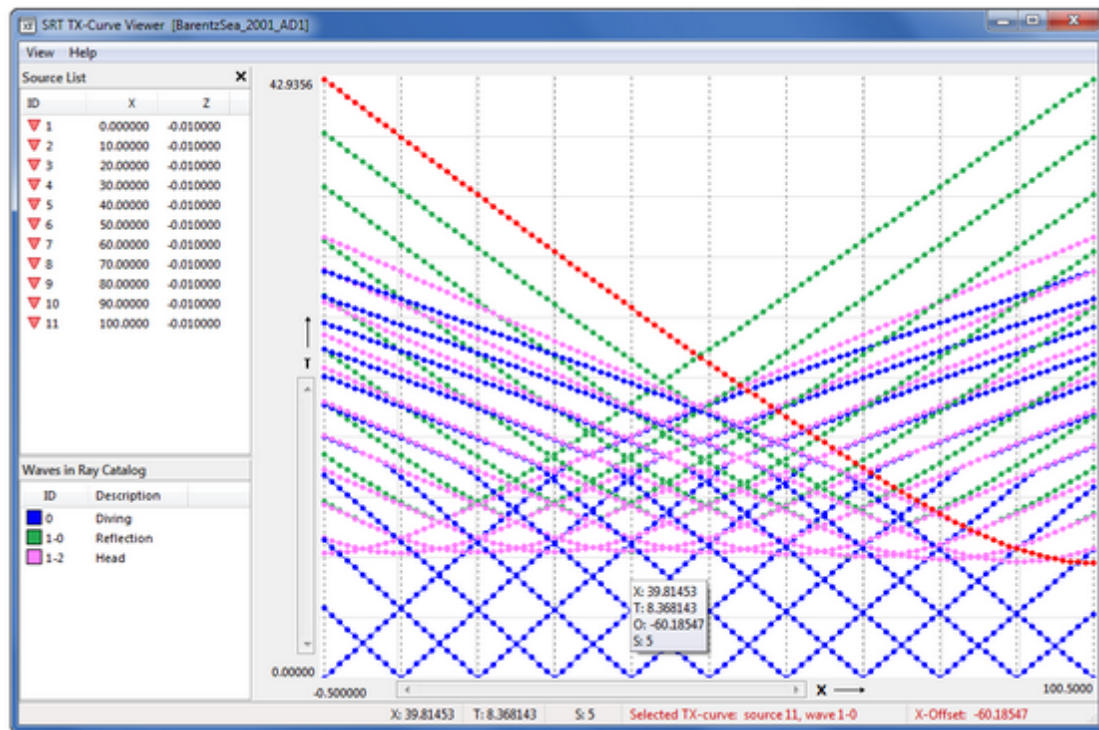


Рис. 1. SRT TX-curve Viewer's main window.

The selected curve is drawn in red; some info on the curve is printed in red on the status panel together with cursor coordinates.

Curves related to different waves are drawn with different drawing attributes, in particularly, in different colors. The program supports its own wave list enumerating waves found in the SRT database. It uses 10 reserved drawing attribute sets. If a number of waves in the database exceeds 10, the default attribute set is applied. The user can edit it.

The window contains three sections: Source List, Wave List and the plotter panel. Dotted verticals pass through source locations. The image can be magnified with the rubber-band. Selector works with TX-curves and their segments. To select a curve, stretch the rubber-band so that at least two curve points get inside. Then apply the *Select TX-Curve* command of the rb-menu. If points of several curves get inside the rubber band, the point nearest to the left contour side defines the curve selected; if there are several such points, the one nearest to the top side defines the curve. If there are several curves containing this point, the curve with less source ID is selected. To cancel selection, use plotter menu command or simply double-click the image.

Control

The user controls the program with the main menu and popup menus owned by source and wave list and the plotter. The list menu commands switch on/off the *visibility filters*. One can filter off curves generated by a subset of sources or/and waves. The commands of the list menus apply to selected list items. Both lists permit selection of multiple items. Commands of other menus are listed in Table 1.

Table 1. Commands of control menus

Command	Description
Main menu / View	
<i>Show X, T under Cursor</i>	A switch. Shows/hides the hint window attached to the plotter cursor with position-aware information. As minimum, the window displays real point coordinates. Near a source vertical, it displays, additionally, the source ID. If there is a selected curve, the current point offset relative to the TX-curve source is also shown.
<i>Show Source & Wave Lists</i>	A switch. Shows/hides the left panel with the source and wave lists.
<i>Show Source Verticals</i>	A switch. Shows/hides source verticals.
<i>Waves</i>	Displays a dialog for managing wave drawing attributes similar to Wave Manager .
Rubber band menu	
<i>Zoom in</i>	Stretches the rubber band content to the size of the plotter pad.
<i>Select TX-Curve</i>	Selects one of curves captured by the rubber band according to rule, stated above and: redraws the curve in the select color which is defined in the wave dialog.
<i>Apparent Velocity for Segment</i>	Adds a segment of a selected curve to the apparent velocity compare list (see below).
<i>Rubber-band Properties</i>	Displays the dialog for editing the rubber band properties.
Plotter menu	
<i>Zoom out</i>	Zooms out the image.
<i>Decrement Zoom</i>	Returns to the previous zoom level.
<i>Unselect</i>	Deselects the selected curve.
<i>TX-Curve Table</i>	Displays the selected curve as a numeric table (see below).

Curve apparent velocity (V_a)

Apparent velocity analysis is important for head wave study. The module provides information on apparent velocity of all TX-curves and a simple tool for comparing V_a of TX-curve segments. For a

selected TX-curve, the *TX-Curve Table* command of the plotter menu displays the dialog shown on fig. 2a.

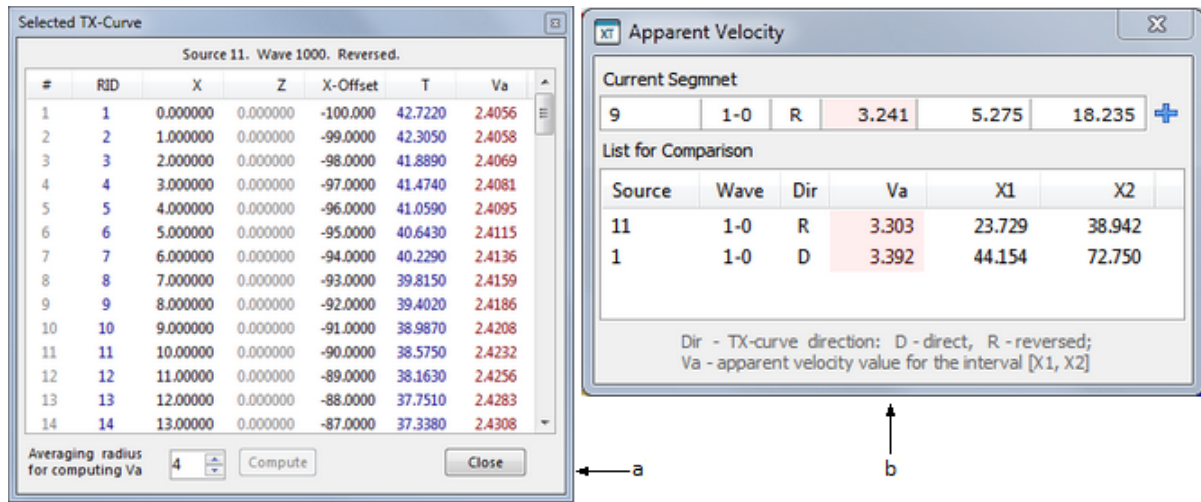


Fig. 2. Dialogs of SRT TX-curve Viewer.

a) The table representing reversed refraction TX-curve from source 11. b) VA compare dialog.

The table provides numeric representation of the selected TX-curve. Meaning of columns: observation ordinal number (#); receiver ID (RID); observation coordinates (X and Z); observation x-offset; wave traveltime (T); apparent velocity. V_a is computed by means of least square approximation of the curve with a linear function in a moving window. By default, window contains 8 points (averaging radius is 4). The user can recompute V_a using another value of the radius. He or she types it in the *Averaging radius* field and clicks on the *Compute* button.

The compare tool works this way. Just after start, the module creates the *compare list* of the form "segment - V_a ". V_a is calculated as the reciprocal of the dip of a straight line fitted to the curve segment. For the user, the list looks as shown on fig. 2b. To add a new item to the list, select a TX-curve and then define its segment by stretching the rubber band anew. In the rubber band menu select *Apparent Velocity for Segment*. The command displays the compare list. At the dialog top, the item to add is shown. It contains the following information on the segment: source ID; wave ID; TX-curve direction (Dir: D – direct, R – reversed); V_a ; x-coordinates of segment end points (X1, X2). Note that the rubber band can be positioned vertically in any way; only its horizontal position is essential. To add the new item to the list, click on the plus button. Comparing V_a of list items is aided by the proper list ordering. It can be sorted by source or wave IDs. The commands are in the list popup menu which includes also the command for removing the selected items.

5 SRT Port: Arhives. Export

Creating archive files

The SRT Port warehouse stores all databases that have been created after XTomo-LM 3.1.1 had been released minus those deleted by the user. The current disc size of the data warehouse can be displayed by the *Data Warehouse Disc Size* command of the Manager popup menu. To save disc space, some databases can be put to a compressed archive file. An archive file can be saved to any location, but we recommend to use the XTomo-LM [archive folder](#). Archive files can be also used for data exchange between the product users.

Support of archives is SRT Port Manager's responsibility. To create an archive, select the items to archive in the database list. In the popup menu select the *Archive* command. It displays the dialog shown on fig. 1a.

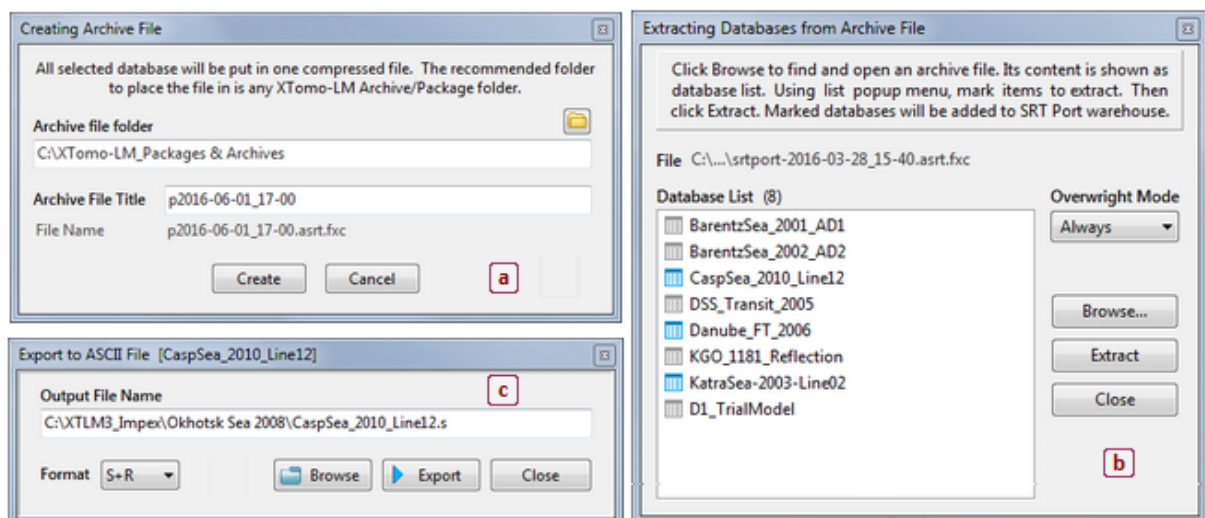


Fig. 1. Dialogs for creating an archive (a), extracting from archive (b) and export (c).

In the dialog, select the target folder clicking on the folder button which invokes the local folder browser. By default, it opens the folder lately used for this purpose. The archive file name is provided by the program. The user must not change it. Click on the *Create* button to start the operation.

Extracting databases from an archive

Apply the command *Extract from Archive* to display the dialog shown on fig. 1b. First, click on the *Browse* button to select an archive file. Its content is displayed as Database list. All list items are marked with discolored icons. Now select databases to copy from the archive to SRT Port warehouse. The list supports multiple selection and owns the popup menu with the commands of marking the selected items. To mark one item, double-click it. Marking makes item's icon colored. Now select the overwrite mode for duplicate database names from the drop-down list. Finally, click on the *Extract* button.

Export to ASCII Files

To export a database to an ASCII file, use the *Export* command of the Manager popup menu. It invokes the export dialog (fig. 1c). Select the target file name and type. Name is set in the Open dialog invoked by the *Browse* button. Type is selected from the *Format* drop-down list. Databases of SRT type can be exported to SRT, SR, S+R, #DT files. Click on the *Export* button to carry out the operation.

6 Ray Catalog: Populating

The Task

The empty Ray Catalog database is created automatically in a newly created o-node. Now it must be populated with data from the SRT Port warehouse. The operation is carried out by SRT Data Extractor. The module, however, does more than data copying. It performs the following meaningful tasks: (1) verifies that observations submit project restrictions; (2) extracts a user-defined data sample from an SRT database. Project restrictions:

- 1) project's observation geometry must match database profiling flag value;
- 2) at least, one wave from those found in SRT database must *be known*, i.e. contain in the project wave list;
- 3) sources and receivers must fall inside the model domain and must not cause resolution errors.

SRT Data Extractor

The module is launched by the command *Extract Data from SRT Port* of the Precessing Tree popup menu invoked on an o-node. The module main window is shown on fig. 1.

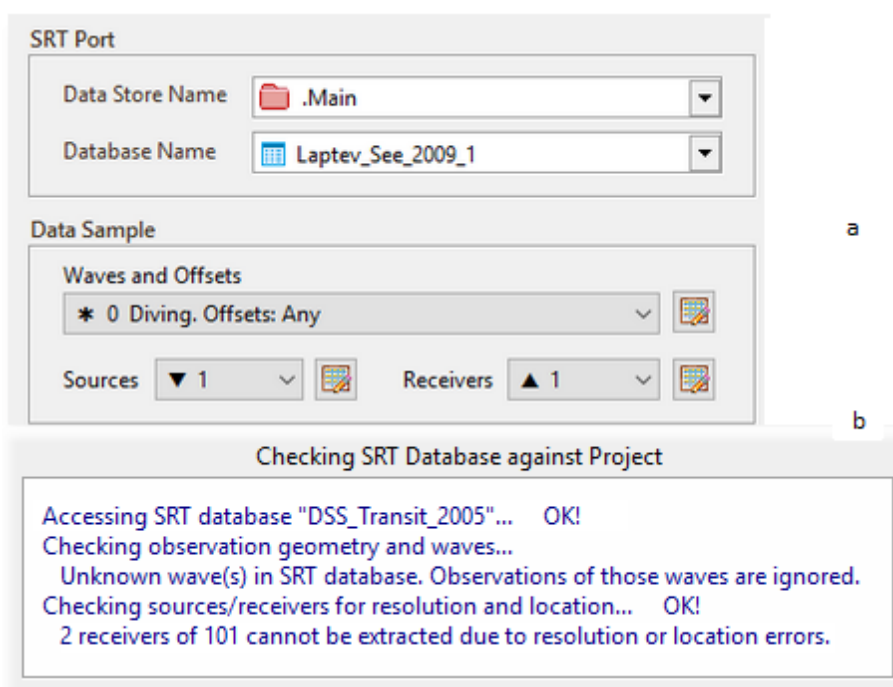


Fig. 1. SRT Data Extractor. a) main window; b) checking log.

The topmost window panel allows locating the source database in SRT Port. In the drop-down lists, select, first, data store name, then database name. In I-projects, the database list contains only databases of SRT type; in M-projects it contains databases of SR type. Just after the user selects the required database, the module begins task (1) – checking the data against the project requirements. The result is can be seen in the log dialog shown on fig. 2b. If the check is successful, all log lines end with *OK!* If data sampling is not required, a click on the *Copy* button finishes the operation.

Errors

SRT data verification may end with errors: observations geometry differs from that of the project (fatal); some waves are unknown (fatal, if all are unknown); some sources or receives do not meet condition 3 (fatal if there are too many such devices). If no fatal errors occurred, data extraction can be continued. Known waves and erroneous devices can be viewed by means of creating data samples.

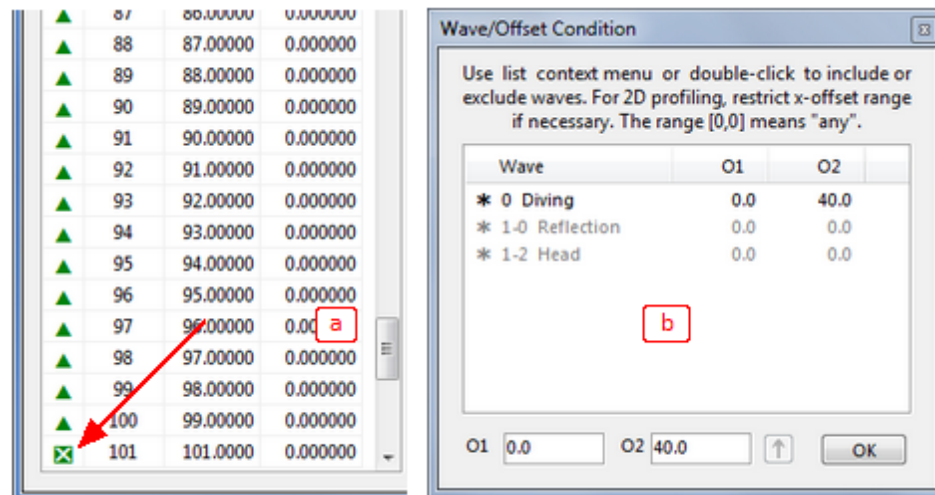


Fig. 2. Creating data sample: a - erroneous devices are marked with specific icons; b - known wave list for creating a wave/offset sample.

Use the *View and select* buttons on the *Data Sample* panel. For example, the one to the right of the *Receivers* field invokes the receiver list shown on fig. 2a.

Sampling

A sample from the SRT database is defined by subsets of waves, sources and receivers and offset ranges. In M-projects waves and offsets are not used at all. To create a sample, use the buttons to the right of each field. Fig. 2b shows the dialog for specifying waves and offsets. By default, all known waves are included in the sample, and all offsets are permitted. Visually: all list items of the wave list are printed in black, and offset range bounds in the O1 and O2 columns are zeros. To exclude/include a wave, double-click the item or use the popup menu. Excluded items are printed in light-gray. To edit an included item's offset range, select an item and enter minimal (O1) and maximal (O2) offset values in the edit fields under the list. Then click on the arrow-button to assign them to the item selected. On fig. 2b only diving wave is included in the sample and only rays with offsets no larger than 40. Sampling is completed by clicking on the *OK* button (accept) or the dialog system *Close* button (cancel).

Sampling sources and receivers is performed in the dialog *Source/Receiver Selector*, partly shown on fig. 2a. The full view is on fig. 3.

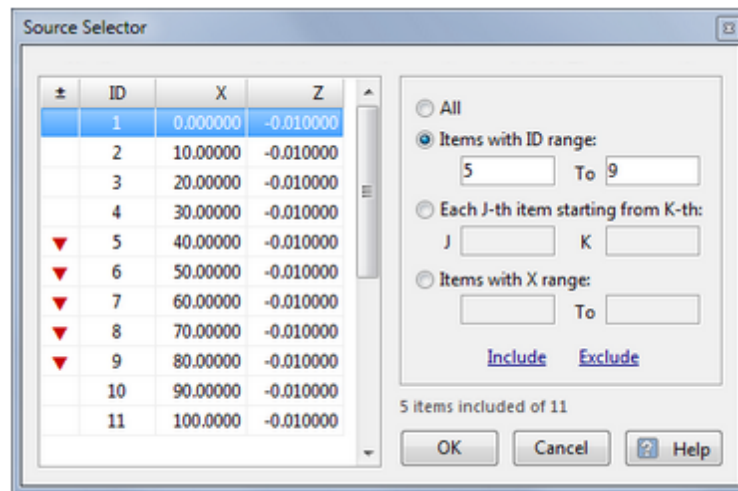


Fig 3. Sampling sources.

The left panel displays the source list, in which some items are marked with icons and the others – not. An item with the icon is included in the sample, an item without it is not. There may be items in the list excluded due to errors. They are simply ignored. To change an individual item's state, double-click on it. To change the state of a group of items, use controls on the right panel to perform the following group operations: (1) include/exclude all items; (2) include/exclude items with IDs from the specified range; (3) include/exclude each J-th item starting from the K-th one; (4) include/exclude items with x-coordinate falling into the specified interval. Click on the option radio-button, enter parameters (J, K, or range bounds), if necessary, and, finally, click on the *Include* or *Exclude* link. Mind that "Include" does not mean "include only": items not satisfying the condition might be selected earlier, and they stay included. For example, to get the selection shown on fig. 3, one should, first, click *All*, *Exclude*, then *Items with ID range*, then enter 5 and 9 and then click *Include*. After clicking on *OK*, the drop-down list *Sources* of the main window changes according to the sample.

After sampling waves, offsets, sources and receivers, click *Copy* to start copying the sampled data to Ray Catalog.

7 Ray Catalog: Viewing

In this section it is supposed that Ray Catalog of the o-node is already created and populated. The Processing Tree context menu, if invoked on this node, contains the command *View Observations* in I-project or *View Spread* in M-project. Each one drops down the submenu with commands listed in Table 1.

Table 1. Commands of viewing Ray Catalog.

Command	Project	Description
<i>On Model Image</i>	I, M	Runs the graphic module Spread Viewer for viewing source and receiver sets on the model image.
<i>As Ray Catalog Database</i>	I, M	Starts the Ray Catalog Viewer module representing the database in the form of numeric tables.
<i>As TX-Curve Set</i>	I, 2D profiling	Accessible for data of profiling only. Starts the TX-Curve Viewer module to view observation data as a set of TX-curves of different waves.

Spread Viewer

This graphic module allows viewing sets of sources and receivers on the model image. The devices are depicted as isosceles triangles: a source stands on its vertex, while a receiver – on its base (fig.1a). Position of a source is pointed by the triangle vertex; position of a receiver is the midpoint of the base. The shape of images cannot be changed, while their size and color can be adjusted. To do that, use the main menu command *Properties/Spread Drawing Attributes*. It displays the dialog shown on fig. 1b, quite similar to the [Wave Manager](#) window, similar both by appearance and functionality. On the right panel, one can edit the size, color and border of the image.

In the XTomo-LM user interface, the source image is a metaphor of observation system.

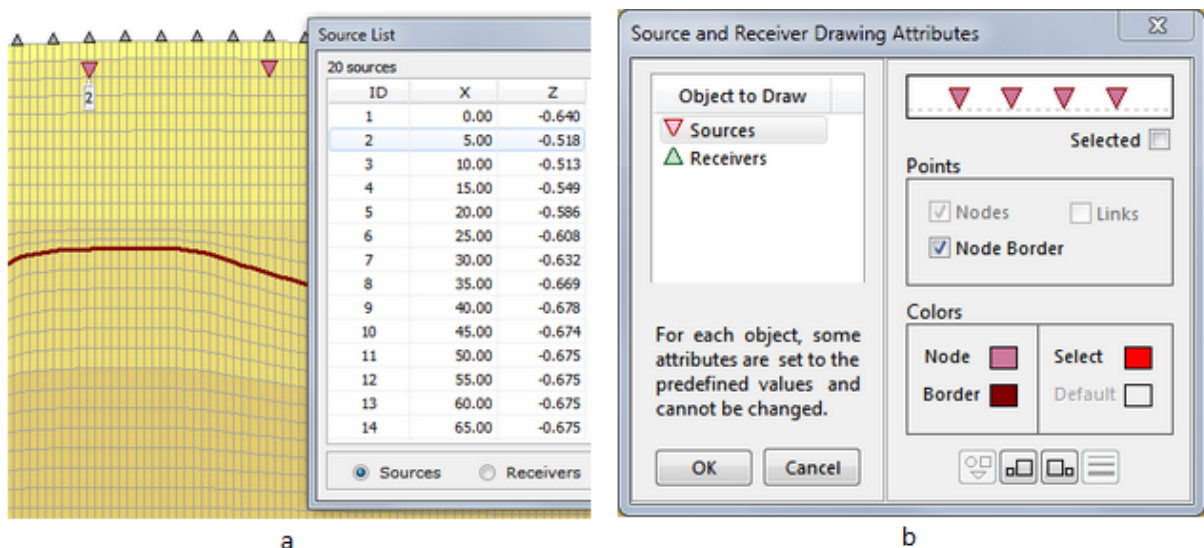


Fig. 1. Module Spread Viewer. a – list of sources/receivers; b – drawing attributes.

On fig. 1a, receivers and sources are arrayed along the surface line, and sources are buried. This is typical situation, though, in XTomo-LM no restriction on device location is imposed. The View/

Source/Receiver List command displays the window listing sources or receivers depending on the state of radio-buttons at the bottom. The image feels position of the table cursor in the sense that if an item is selected in the list, the corresponding source (receiver) image gets a tag with its ID. On the fig. 2a, ID = 2.

Ray Catalog Viewer (RCV)

RCV displays the database content in the form of numeric tables. On fig. 2, RCV is started on an o-node of an I-project.

RCV shows the part of the Ray Catalog database for the selected wave. For a selected source on the left grid, the Receivers table displays the list of responsive receivers.

Sources (11 of 11)			Receivers (101 of 101)			
ID*	X	Z	ID	X*	Z	To
1	0.000	-0.010	1	0.000	0.000	9.050
2	10.000	-0.010	2	1.000	0.000	8.687
3	20.000	-0.010	3	2.000	0.000	8.315
4	30.000	-0.010	4	3.000	0.000	7.933
5	40.000	-0.010	5	4.000	0.000	7.540
6	50.000	-0.010	6	5.000	0.000	7.137
7	60.000	-0.010	7	6.000	0.000	6.725
8	70.000	-0.010	8	7.000	0.000	6.301
9	80.000	-0.010	9	8.000	0.000	5.866
10	90.000	-0.010	10	9.000	0.000	5.421
11	100.000	-0.010	11	10.000	0.000	4.966
			12	11.000	0.000	4.501

Total number of: sources - 11; receivers - 101; rays - 2903.

Wave: 0 Diving

Buttons: Export, Help, About

Fig. 2. Ray Catalog Viewer main window for I-project.

The tables are not independent. The hierarchy of tables is controlled by the drop-down wave list. After a wave is selected, the source table (*Sources*) displays the list of all sources generating observations of the wave selected. Further, for a source selected in this table, the table *Receivers* displays the list of all receivers responsive to the signal from this source. It follows then, that both tables represent the set of the wave rays. *To* – is a standard denotation for *observed times*. The asterisk in column heading means that it is a table sort key.

When another source is selected in the left table, the right table is filled with new portion of data. When another wave is selected the left table updates. Updating is often unnoticed (but *To*), because sets of sources for different waves or sets of receivers for different sources are often the same or close to each other.

In the case of M-project, the *To* column of the *Receivers* table is absent as well as the list of waves.

RCV can be launched on an f-node too (the same command), i.e. after solving forward problem. In this case, the *Receivers* table has the new column *Tc* – *computed times*, and in an inversion project,

additionally, the columns Ea – absolute errors or residuals) and Er – relative errors (residuals). Notations Tc , Ea и Er always have the declared sense in the XTomo-LM user interface.

The *Export* button invokes the dialog for export of the Ray Catalog content to ASCII files. The user selects the export file format and option (export all waves or the selected one). Three formats are offered: **SRT**, **SR** и *All columns*. On an o-node, the SRT format can be selected only for I-projects. On an f-node, one can select SRT for an M-project too, with *computed times used instead of the observed*. The "All Columns" format suggests output of all columns of both tables in a text file (TXT).

TX-Curve Viewer

The module user interface does not, practically, differ from that of module **SRT TX-Curve Viewer**. The main difference is that TX-Curve Viewer works with the project data and uses the project wave list, while SRT Viewer maintains its own wave list. Additional command of the plot menu allows hiding/showing an TX-curve not resorting to filtration. So, for help address **SR Port: TX-curves**. The context help system directs the user straight to that section.

Copying to SRT Port

Ray Catalogs in o- or f-nodes can be copied to SRT Port. The operation is helpful for two aims: (1) using ray catalog content in other projects; (2) modeling observation with the help of forward problem solution. The operation is initiated from the Processing Tree menu. Invoke it on an o- or f-node and select the *Copy Ray Catalog to SRT Port* command. It starts the copying module whose main widow is shown on fig. 3.

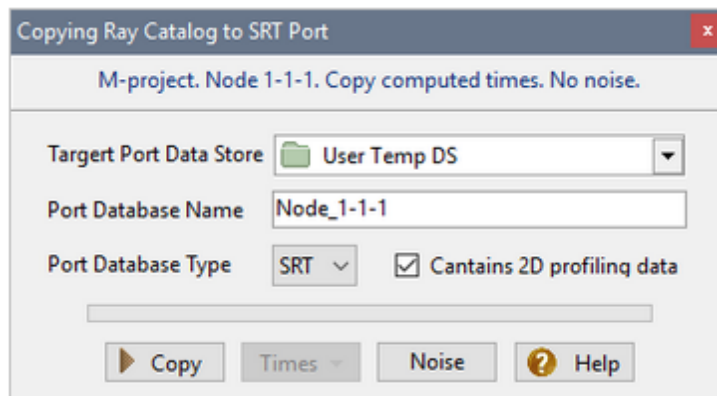


Fig. 3. Copying Ray Catalog to SRT Port.

The module displays project kind and node code on the top panel. Below is the drop-down list of the port data stores to choose the target one; below it, there are the target database name and its properties. Accessibility of property fields and control buttons depends on the project kind and node selected. For example, Ray Catalog in an o-node of an M-project can be copied to the database of SR type only. In other cases, type of the target database is selected by the user from the drop-down list. On an f-node of an I-project, the user has an option of choosing between observed and computed times. The *Times* button is just for that. Besides, the user can add to time values Gauss noise with specified mean value a and standard deviation σ (observation modeling). To do that, use the *Noise* button which displays a dialog to define a and σ . If $\sigma > 0$, times will be

noised. The default target database name is shown on fig. 3. It can be edited. For a port database, observation data type (profiling data or other) must be defined. In case of copying from an f-node of an M-project, the user defines data type with the help of the *Database contains 2D profiling data* check box. In all other cases, the flag is set automatically and cannot be changed. The *Copy* button starts the operation.

8 Ray Catalog: M-projects

In M-project, the observation system can be either defined in a text file and imported through SRT Port or drawn "manually" straight on the model image in graphic module Spread Editor. It is launched from the Processing Tree menu, invoked on an o-node without children. The module inherits the way of source/receiver representation and all the functionality of [Spread Viewer](#). The edit commands can be found in the rb-menu, plotter menu and in the Source/Receiver List window. To add a source or receiver, the user must provide its ID and position (coordinates). In all operations, ID are maintained automatically so that data integrity is guaranteed. Also, the module controls positioning and resolution errors. It is forbidden to place sources and receivers on the first grid column and the last row. Operation roll-back is implemented only in a few cases.

Commands and operations

Table 1. List of editing operations

Command	Location	Description
<i>Add Source, Receiver</i>	plotter menu	Accessible if there is a selected cell. A source or receiver is placed at the selected cell's vertex.
<i>Add Array</i>	plotter menu	Accessible if a row or a column is selected. Creates a source/receiver array along row roof or left column edge (see below).
<i>Add Inline Array</i>	rb-menu	Creates inline array of sources or receivers along edges or diagonals of the specified rectangle. Details .
<i>Delete Sources, Receivers or Both</i>	rb-menu	Removes sources or/and receivers captured by the rubber-band.
<i>Delete button</i>	list	Removes sources/receivers marked in the list.
<i>+Move button</i> <i>-Move button</i>	list	Moves sources/receivers marked in the list. Details .

Creating array along h- or v-line

Consider the case of h-line. Select a row, and choose *Add Array* in the plotter menu. It displays the dialog (fig. 1) that allows creating an array of equidistant sources or receivers depending on choice of radio-buttons on the *Array Item* panel.

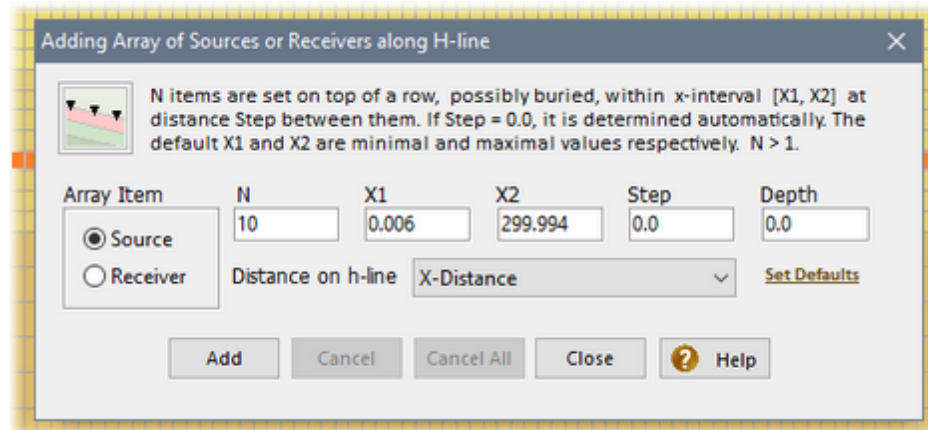


Fig. 1. The dialog for creating an array along an h-line.

Let the *Source* radio-button is checked. Two array dispositions are available.

1. N sources are positioned within the x-interval $[X1, X2]$; the user defines N, X1, X2 and sets Step = 0.
2. N sources are positioned within the x-interval $[X1, X2]$ with the specified step. The user defines N, X1, X2 and Step.

In case of 1 step is calculated by the program. In case of 2, when step is defined, array may contain less than N sources. In both cases, sources can be buried with respect to h-line, if the value of Depth is positive. Additionally, the user can specify, what does "distance" exactly mean: distance between x-projections of sources (X-Distance) or distance measured along the curve (*Distance on h-line*). Difference between the options is significant if h-line has significant curvature. If the user enters X1 or X2 falling beyond permissible area, the values are changed silently for the nearest area bound.

A click on the *Add* button starts the operation. On termination, the module informs about the number of actually created sources. At that, the dialog is not closed. Move it aside to see the result on the model image. If it is not satisfactory, click on *Cancel* to roll back the operation. On the other hand, the user can add another array on the same h-line, on another interval $[X1, X2]$, with other step. The *Cancel All* button removes all arrays added in this session.

Creating inline array

This operation allows creating an equidistant array along a line with arbitrary slope. The line is defined as the top or left edge or a diagonal of an appropriate rectangle. A rectangle is, initially, defined by the rubber-band. So, drag out the appropriate rubber-band and select *Add Inline Array* in the rb-menu. The command invokes the dialog shown on fig. 2a.

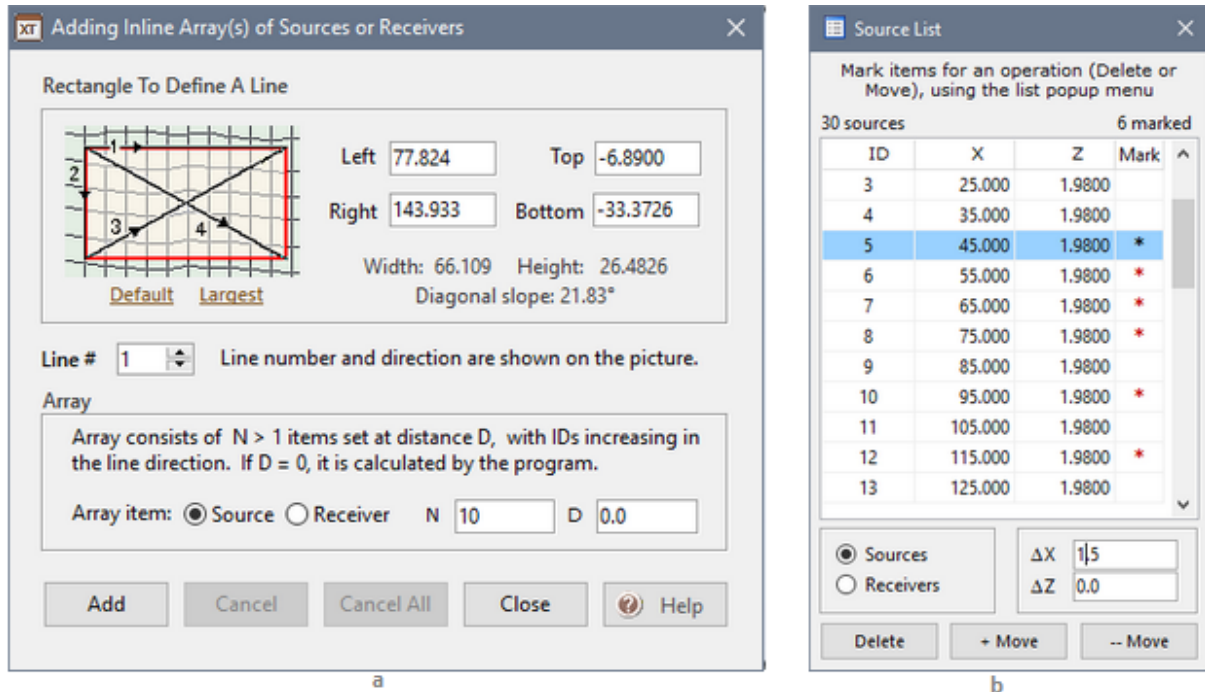


Fig. 2. Editing observation system. a – creating inline array; b – work from the source/receiver list.

Position of the rubber-band rectangle is shown on the top panel. It can be edited in the fields *Left*, *Right*, *Bottom*, *Top*. If the user enters a value falling beyond permissible area, the values are changed silently for the nearest area bound. Thus, rubber-band position and size does not predefine the line. Under those fields, rectangle width and height are shown as well as slope of its diagonals. The links *Default* and *Largest* set the initial rectangular and the maximal one. The sketch on the left side of the panel explains how the permitted lines are numbered and oriented. Orientation defines growth of device IDs. To select the line, use the spin-edit field *Line #*. Then define the item (source or receiver), the number of sources *N* and step *D*. If *D* is zero, the step between items is calculated automatically, else the actual number of sources may prove to be less than *N*. Clicking on the *Add* button creates the new array, but the dialog stays on screen, allowing to view the result on the plotter, or cancel the last addition (*Cancel*), or perform another operation. The *Cancel All* rolls back the dialog session entirely.

Deleting sources/receivers

There two ways of removing devices. The first is to capture them by the rubber-band and select the *Delete* command in the rb-menu, which has three options: sources, receivers and both devices. The other way is described in the next section.

Work with the source/receiver list

Spread Editor uses the list as an edit tool and performs deleting or moving arbitrary groups of objects. A group is specified by way of marking out list items. A marked item has the asterisk in the *Mark* column (fig. 2b). Commands of marking can be found in the list context menu.

Table 2. Commands for marking items in the source/receiver list

Command	Key	Description
<i>Mark/Unmark</i>	Ctrl+Click	Marks/unmarks the selected item.
<i>Mark Range</i>	Shift+Click	Marks range between previously marked and the current item.
<i>Mark All</i>	Ctrl+A	Marks all items.
<i>Unmark All</i>	Ctrl+U	Unmark all items.
<i>Filter Marked</i>	Ctrl+F	Switches on/off the filter that hides all items but the marked.

Object of an operation is a group of marked list items. There are two such operations: deleting (the *Delete* button) or moving up or down (the \pm *Move* button). The *Move* buttons are accessible if only, at least, one of the ΔX and ΔZ fields is not empty and its value is positive. This value is a move step by which the object position shifts at each click on the Move button. If the exact x- and z-shifts value are known beforehand, they can be entered into the delta fields, – and one click completes the job. The program tracks resolution errors and coming out of the allowed domain. The list window is not a dialog, it does not block access to the main window, but when the list is on screen all editing operations in the main window are forbidden.

Creating Default Ray Catalog

When the user quits the module, it saves sets of sources and receivers but only when each of them is not empty. Additionally, the module rebuilds *Default Ray Catalog* (containing all source-receiver couples). Default Catalog is also created if observation system is copied from SRT Port.

Removing redundant rays

Default Ray Catalog may not fit the modeling problem. Redundant rays can be removed from Catalog with the help of the Ray Catalog Editor module (RCE). It is launched by the *Edit Ray Catalog Database* command of the Processing Tree menu, invoked on an o-node. The RCE main window looks like that of [Ray Catalog Viewer](#), and RCV functionality is kept. However, the *Receivers* table in RCE owns a context menu. Besides, it has a left margin, on which the current database record is marked with an arrow. Additionally, the table is enabled with multiple selection of records. A selected record is marked on the margin with a black bold point (unlike selecting items in lists). To select an item, click on it with Ctrl pressed. The list of menu commands is in Table 3.

Table 3. Commands of removal and restoration of the *Receiver* table menu.

Commands	Description
<i>Delete Selected Rays</i>	Removes from the table all rays coming to to the selected receivers.

<i>Delete on Offset Condition</i>	Removes rays to receivers satisfying the offset conditions (see below).
<i>Delete Each Second Ray</i>	Deletes each second receiver starting from the first.
<i>Delete All Rays for this Source</i>	Removes all rays, generated by the source selected in the <i>Sources</i> table.
<i>Recreates Default Ray Catalog</i>	Restores Default Ray Catalog.

Deleting rays on offset condition

The command displays the dialog shown on fig1. It allows the user to compose the condition

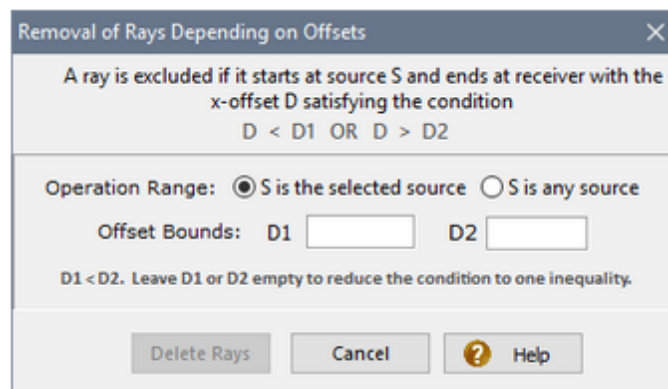


Fig 1. Dialog for setting offset condition.

of the form $D < D1$ or $D > D2$, where D is x-offset (distance between source and receiver projections onto the X axis). One of the fields can be left empty which means that one of inequalities is absent. The operation can be applied either to the currently selected source or to all sources, depending on the radio-button checked. The *Delete Rays* button starts the operation.

Exit

Before the module shuts down, it checks if after all removals some sources and receivers belongs to no ray. Such objects are removed.

About Ray Catalog tables

Ray Catalog Viewer and Editor represent relational database tables with a special GUI component *db-table*. It looks like the Windows *list* control, but it differs significantly, which requires some comment. A *db-table* is an image of a database table and implements two-way communication with it. In particular, the *db-table* cursor (color marker) points to the *current record* of the database table. In RCE, where multiple selection is used, there is the additional indicator on the left margin. It allows to determine which of the selected records is the current one. A selected *db-table* item corresponds to a *bookmark* on the data level. The indicator has three forms: an arrow for the current record: ; a bold point for a bookmark: ; an arrow with a point for a

bookmark pointing to the current record. To create the first bookmark, click on the matching db-table item. To create the next bookmarks, use click + Ctrl. At that, a click always makes the matching record the current one. A repeated click + Ctrl removes the bookmark. Thus, multiple selection means creating a set of bookmarks. Note that applying an arrow-key moves the current record pointer and removes all existing bookmarks.

Forward Problem

1 Overview

Solving kinematic forward problem and exploring its solution form the pivot stage of processing. Getting forward problem solution is the most resource and time consuming operation.

In projects of both types, input data for ray-tracing consist of the model in an m-node and the content of the Ray Catalog in an o-node. The user selects the needed o-node of Processing Tree, invokes the context menu and selects the *Solve Forward Problem* command. The command creates the new child f-node and launches Forward Problem Solver (**FPS**). In its main window the user creates the task or job for ray-tracing. If the user cancels the operation or getting solution fails, the newly created f-node is automatically deleted. If the operation succeeds, the f-node folder contains the results: updated version of Ray Catalog and the ray path file. To view and study the solution, run Forward Problem Solution Viewer (**FPV**) on the f-node. It demonstrates the ray picture, TX-curves and offers different tools for solution study. Besides, it makes preparations for tomography inversion. The shortcuts FPS and FPV will be used further without comment.

2 The Forward Problem Solver Module

FPS session includes four steps:

- a) creating task;
- b) start and execution;
- c) reading the operation log;
- d) saving the solution to the output Ray Catalog.

The user's main job is to compose the task, that is:

- create wave sample for ray-tracing (always);
- prepare description of converted waves (if they are included in the sample);
- adjust ray-tracing precision (very rarely);
- adjust execution parameters (sometimes).

Selecting waves

Just after it has started, FPS displays the list of waves for which solving forward problem makes sense. Not all waves from the project wave list get in the list. In M-project, a wave is in the list if it is diving or it is associated with a model horizon. In I-project, a wave is in the list if its observations are detected in Ray Catalog; if it is a reflection or refraction, it is associated with a model horizon.

The phrase "associated with a model horizon" has the following exact sense: horizon ID participating in a wave code can be found in model horizon list (see [Waves](#), [Horizons](#)). A fragment of FPS main window with wave and horizon lists is shown on fig. 1.

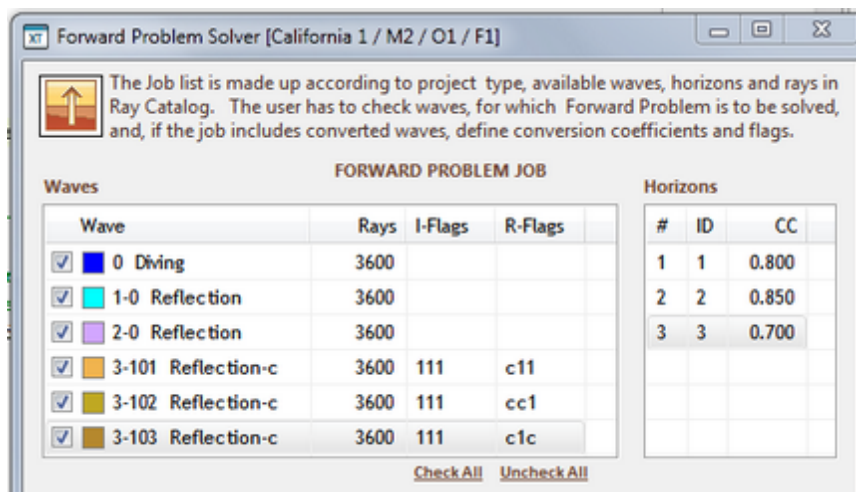


Fig. 1. FPS task: wave and horizon lists.

Each item of the wave list includes a check box to include a wave in the task, wave description, a number of rays in Ray Catalog and conversion flags (see below). The horizon list for each item displays its ordinal number in model (counting from top), its ID and conversion coefficient (CC). By default, CC = 1 (no conversion). Use the *Check All* and *Uncheck All* links for creating the wave sample.

Preparing information on converted waves

Wave ID shows whether a wave is converted or not. For each converted wave, *conversion flags* describe its behavior when passing through a horizon or reflecting from. One flag relates to a horizon, and its value is either "c" (there is conversion) or "1" (no conversion). If a wave meets N horizons on its path, one has to define N flags. They form a string, say, "11c1c1...1c". Character position in the string is equal to horizon ordinal number in the model counting from top. If the k-th flag is "c", velocity of wave propagation in the k-th layer is $CC_k \cdot V$, where CC_k is conversion coefficient for the k-th horizon in the horizon list, V is model velocity. Conversion flags are defined separately for incident wave and reflection/refraction. Thus, each converted wave is assigned two strings of flags with length equal to horizon number on which the wave is formed. If a wave undergoes conversion on the k-th horizon, its CC in horizon list on fig.1 must differ from 1.

For setting and editing conversion flags and coefficients the user has to display the *Conversion Specification* panel clicking on "+" (fig. 2).

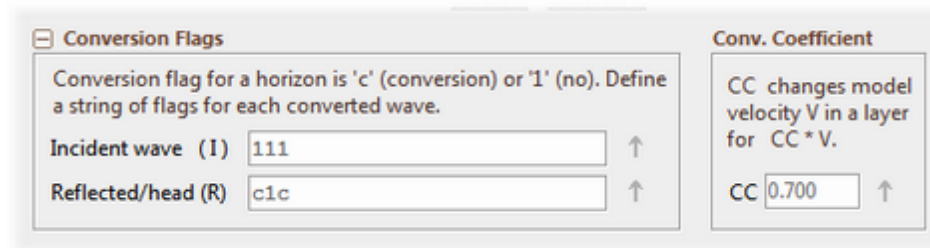


Fig. 2. FPS Task: conversion flags and coefficients.

Strings of flags are edited in the fields *Incident wave* и *Refracted/head wave*. The fields are accessible if only the converted wave is selected in the wave list. Each position of either field accepts either "1" or "c". Editing ends with a click on the Up arrow button. The most convenient way of editing is to use the following key combinations:

- 1) Shift + Down-arrow moves focus from the selected list item to the field *Incident wave*;
- 2) Shift + Up-arrow moves focus from any edit field back into the list;
- 3) pressing Up-arrow key in any edit field is equivalent to clicking the arrow button.

In the same way CC in the horizon list are edited.

Ray-tracing precision

RTP is an algorithm parameter. Changing its default value may affect ray picture smoothness. If the grid is thick enough, there is no need in change. The more so, that if RTP grows, computation time increases significantly. RTP is connected with resolution, so it may happen that it is impossible to increment RTP. Technically, to do the job, one has to unfold the *Ray-Tracing Precision* panel with the "+" button. Adjusting RTP in terms of "increase/decrease" is carried out by moving the thumb of a track bar. If a change is blocked, the thumb is disabled. The default value is restored by the *Default* link. RTP is defined for X and Z separately. To switch between the variables, use radio-buttons.

Execution

A click on the *Start* button starts the operation. FPS divides the entire set of rays to trace in groups. One group contains rays of one wave generated by one source. Tracing rays of one group is implemented by the module Ray Tracer (RT). This name is used in the log records. RT has no window, and its activity is not visible. FPS launches as many RT instances as the system has logical processors, and each logical processor runs one RT at a time. Thus, ray-tracing is performed in parallel, and the parallelism degree is defined by a number of logical processors available in the system.

Parallelism affects computation time (see the example in the section [What's new in version 3?](#)), but not in linear proportion to a number of processors. The reason is in the system overhead and disc controller overhead. Important: FPS take up most of system resources, so it does not make sense to run FPS along with other resource consuming applications. Project Manager allows running only one FPS instance at a time.

The execution is tracked in the FPS main window by the progress bar (fig. 3). Above it, the list of ray groups being currently traced is displayed. A group is represented by a string <wave ID>/<source ID>. Execution can be interrupted by the *Abort* button.

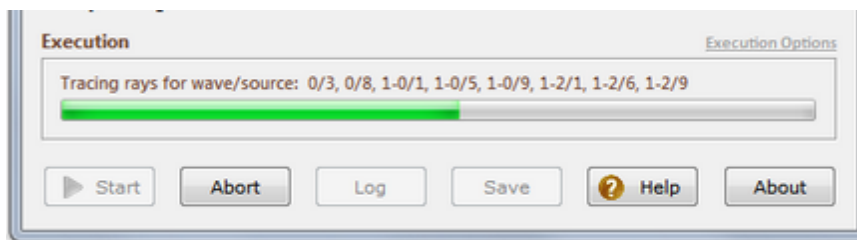


Fig. 3. Execution state depiction.

Log. Lost rays

After ray-tracing is over, the user has to study the operation log. It contains the task and information on every ray group that was traced. The fact of normal ray-tracing termination does not yet mean a success. It depends on acceptable percentage of lost rays. Remember that a source-receiver couple in the input Ray Catalog only declares the *intention* to trace the ray. Whether this intention is implementable is determined in the course of tracing. The ray can miss the source at given velocity distribution or it can leave the model domain before achieving the surface. Large percentage of lost rays means discrepancy between the model and the observation system. In the case of I-project, it is caused by unsatisfactory data preparation, but, rather, by wrong choice of the model. Studying the log is helpful for getting the right idea.

Saving solution

If the solution is found acceptable, it must be saved. The *Save* button starts transferring the data from a temporary storage to the output Ray Catalog and ray path file. The execution log is stored in the target f-node and can be read at any time by the *View FPS Log* command of the Processing Tree menu. Ray Catalog can be viewed (in numeric form) by the command *View Catalog of Traced Rays* on the target f-node after FPS is shut down.

If the solution is rejected, FPS exits without saving the data obtained. In this case, or because of a fatal error, or user interruption, the new f-node is removed from Processing Tree, but the log is stored in the parent o-node. The *View FPS Log* command called on an o-node displays list of logs of all failed attempts of solving forward problem. In the list, items are tagged with their creation time. The list menu allows viewing and deleting logs. Viewing can be initiated also by double-clicking a list item.

Execution parameters

Execution process described can be slightly modified by changing two parameters in the dialog invoked by the *Execution Options* link. If the *Auto-save* flag is set, FPS saves the result itself immediately after termination of ray-tracing. The second parameter is *Concurrency index* – number of logical processors used for solving the problem. By default, it is equal to a number of logical processors in the system. Formally, the index can be assigned a greater value, but it will


hardly boost computation time because processors are practically busy all the time of getting solution.

3 Ray Picture

Imaging traced rays is the function of graphic module Forward Problem Solution Viewer (FPV). In I-project, it also helps explore the solution. In the first place, it performs time residual analysis for making decision whether the model used for ray-tracing fits the observed traveltimes. If the decision is positive, the model is taken for the final result. If not, further steps are to be worked out. If velocity distribution is about to be refined by tomography inversion, FPV freezes velocity in user-defined subsets of cells and creates ray samples for inversion. If layered model is being studied, the above said may relate to an interval of depth.

The rest of the chapter is devoted to learning FPV which evolves in the following order:

- imaging rays and TX-curves;
- creating ray samples;
- residual statistics and distributions;
- ray coverage density;
- preparation to tomography inversion;
- export to ASCII and graphic files.

As a graphic module, FPV differs from all the others by owning two plotters. M-plotter displays model and rays, while T-plotter displays computed TX-curves and observed TX-curves in I-projects. T-plotter can be either docked to M-plotter in the main window or live in a separate window. Respectively, the *View* menu contains new commands *Show TX-Plot* и *Dock TX-Plot*, and the tool bar bears the duplicating buttons. Ray rendering is controlled by the *Rays* menu and the  button with its drop-down menu, which will be referred to as **r-menu**.

Rendering rays

When FPV main window appears on screen, it displays M-plotter with the model view. To draw rays, select the *Display All Rays* command in r-menu and wait until drawing is over and the entire ray-picture appears on M-plotter. Fig. 1 shows ray picture for a ray sample. A number of rays can be found on right side of the status bar. Together with rays, their sources and receivers can be viewed. R-menu contains the switch commands *Show XXX*, where XXX stands for rays, sources, receivers and "all". These switches help control the image. When rays are shown, mouse wheel is blocked, use drag + Ctrl for scrolling.

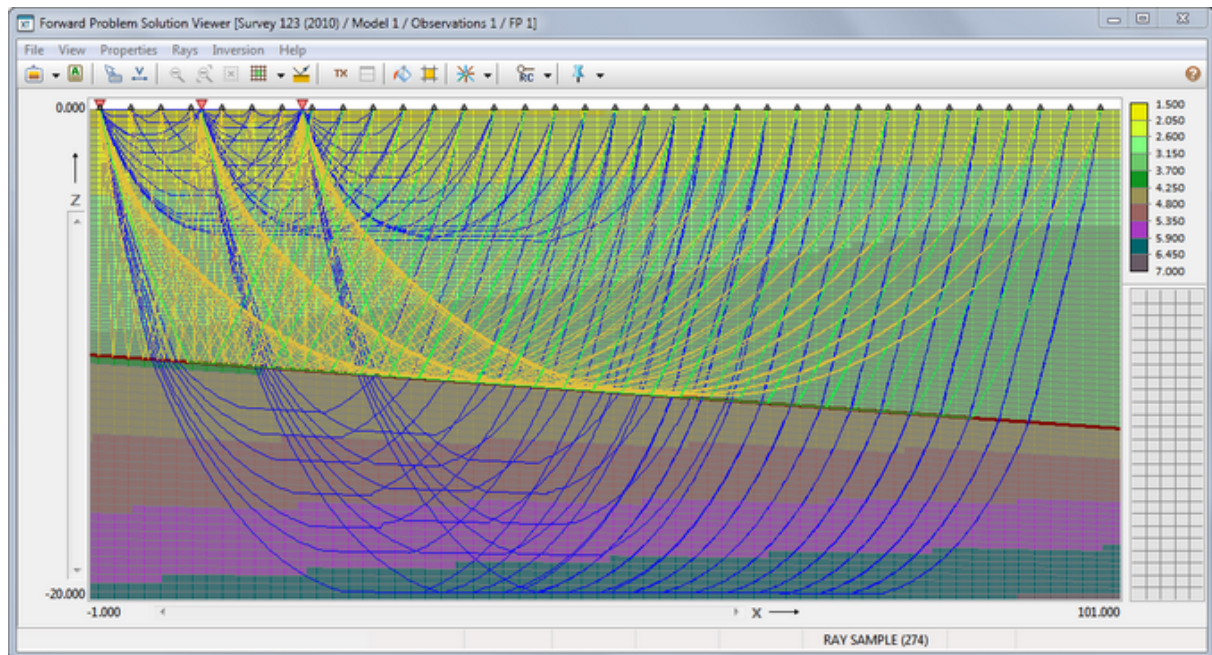


Fig. 1. FPV: M-plotter with ray sample.

Remember that drawing attributes of rays and computed TX-curve are defined by the wave they belong to. They are controlled by [Wave Manager](#). Width of ray lines, however, is controlled by FPV itself. By default it is 1. To change ray width as well as drawing attributes of sources, receivers and observed traveltimes, use the main menu command *Properties/Drawing Attributes*. The additional command *Special Grid Colors* of the *Properties* menu allows changing in one place all colors important for FPV.

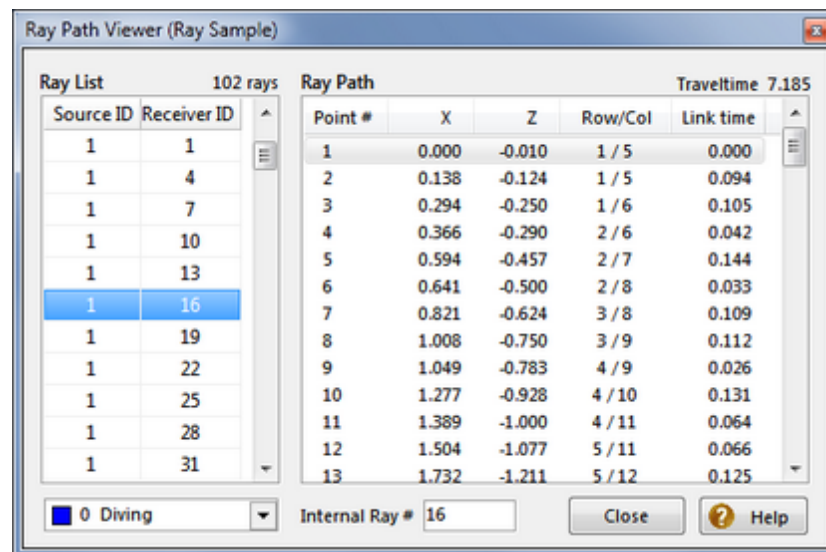


Fig. 2. FPV: Ray paths in numeric representation.

The main menu command *Rays/Ray Paths as Tables* displays ray paths as a hierarchy of tables (fig. 2). The main key is wave which is selected from the drop-down list at bottom-left of the dialog.

The left table *Ray List* lists all rays of the wave selected. A ray is identified with source and receiver IDs. The right table displays the trajectory of the ray selected in the left table. Ray points are numbered in the direction from source to receiver. For each point, there are columns with coordinates, cell double index (Row/Col) and traveltime along a link between this and previous points. Full traveltime is shown above the table.

A great amount of rays

Ray paths are stored in the file whose size for real problems can amount to tenths and hundreds megabytes. Rendering time for such volume of information may be too great. It depends on productivity of central and graphic processors and disc system. The situation is aggravated by necessity to repaint window at each operation changing the view: subgrid selection, window resizing and so on. On that reason, the user is recommended to work with ray samples of reasonable size (see the next section). If, nevertheless, the whole ray picture is to be imaged, there is a helper that stops rendering when convenient. If the total of rays is more than 5000, then after rendering has started a stop button appears on the tool-bar. It looks like this: ✖ STOP. A click on the button interrupts the process and resets the switch *Show rays* in the r-menu. The further work goes on without drawing rays until the user switches *Show rays* on. While rendering is on, resizing window and other operations causing repainting of the plotter content are blocked.

Drawing TX-curves. T-plotter

After all rays or ray sample are displayed for the first time, the command *View/Show TX-Plot* and the TX tool bar button become enabled. A click on the TX button rebuilds the FPV window: M-plotter stay at the bottom, while T-plotter appears at the top (fig. 3a). In I-project T-Plotter draws both computed TX-curves and observed TX-curves (in the background). The beveled line dividing plotters can be dragged within certain limits changing relative plotter sizes. The plotters are synchronized by ray sample. That means that T-plotter displays traveltimes of rays that are currently displayed on M-plotter.

T-plotter has Zoom and Selector tools. A TX-curve can be selected by the rubber-band that determines the curve by the point nearest to the band left side; if there are several such points, the one nearest to the top side is the determinant. The selected curve can be viewed in numeric form by the *Selected TX-curve in Detail* of the T-plotter menu. User control over T-plotter is fully implemented in the plotter and rubber-band menus.

Both plotters are X-synchronized. That means that using Zoom in one of them affects the other in the same way. Drawing TX-curves is incomparably faster than rendering rays: interruption of drawing TX-curves is not provided.

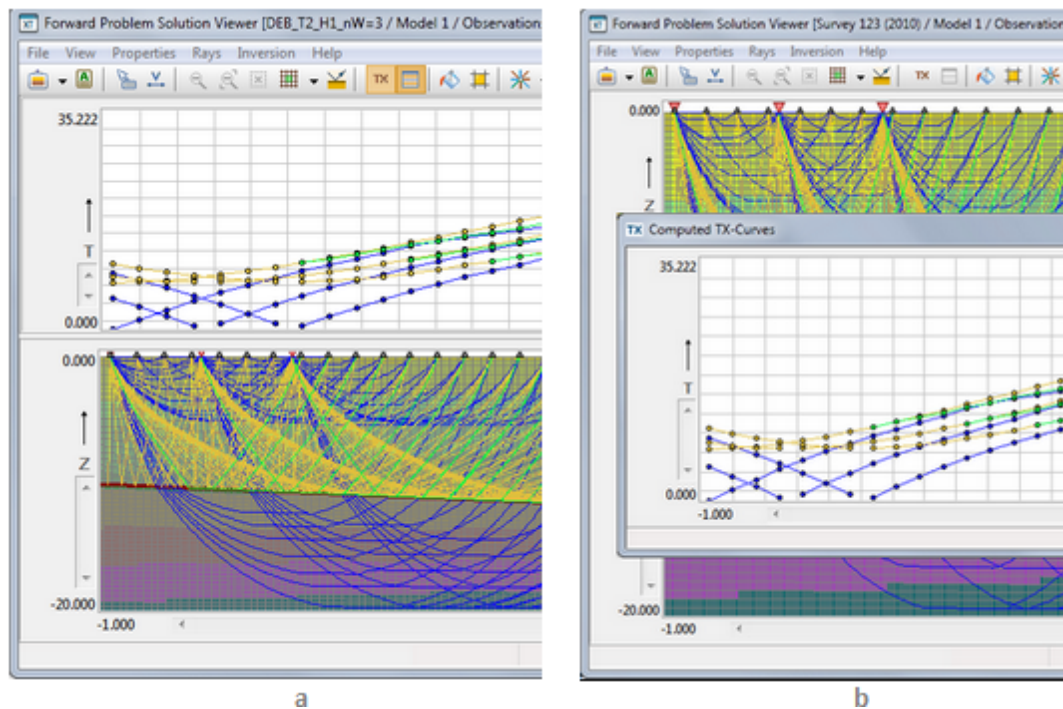



Fig. 3. FPV: T-plotter. a – docked; b –undocked.

After the first appearance of T-plotter in the FPV session, the switch command *View/Dock TX-Plot* (and corresponding tool bar button) become enabled. The initial switch position is "Docked". If one applies the command in this position, T-plotter appears in a separate window, while M-plotter takes up the entire plotter frame (fig. 3b). Specific features of the undocked state are:

- the *Dock* command appears in T-plotter context menu;
- the main menu command *View/Show TX-Plot* shows or hides T-plotter keeping it undocked;
- the main menu *View/Tile horizontally* command positions the main and T-plotter window side by side filling the entire screen;
- the command *Sync Grid and TX plots*, which can be found in the *View* section of the main menu and the T-plotter menu, switches on/off X-synchronization of the two plotters.

4 Ray Samples. Creation and Saving

Ray samples are required for examining and studying details of ray picture, especially when total number of rays is too large. The other aim of sampling rays is selection the right ray subsets for tomography inversion. Here "right rays" may mean "rays optimally illuminating" the interval of the section being studied; or rays with time residuals within reasonable range and the like. Building ray sample is started by the *Rays/Create Ray Sample* main menu command or by the button . The command launches Ray Sample Builder. At that, FPV main window gets locked.

Sample criterion

Ray Sample Builder's main window fields (fig. 1a) display conditions, from which the sampling criterion is composed using logical operator AND.

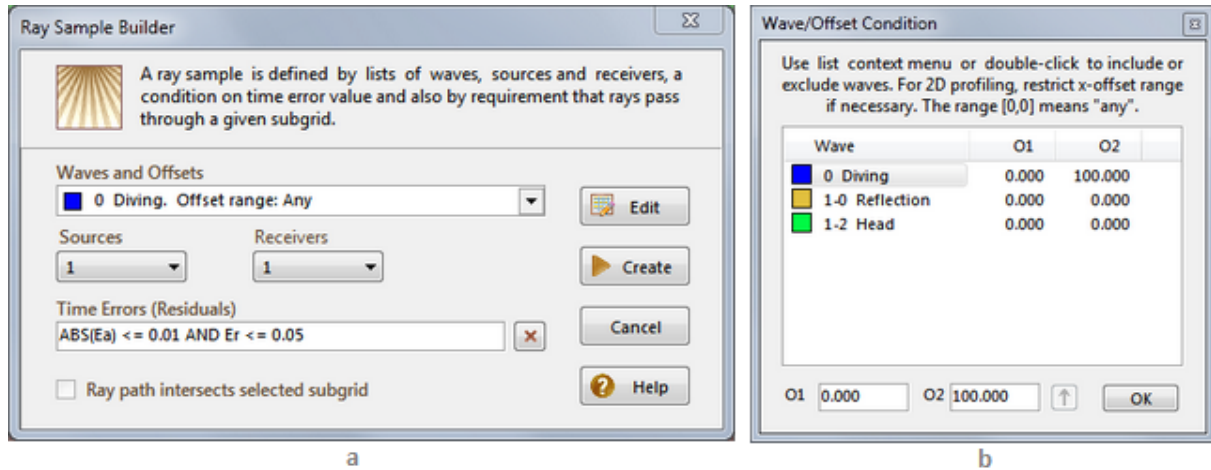


Fig. 1. Ray Sample Builder. a – main window; b – "wave and offsets" condition.

The *Waves and Offsets* condition enumerates waves included in the sample with possible restrictions on receiver x-offsets. The *Sources* and *Receivers* conditions list sources and receivers permitted in the sample. The *Time Errors* condition restricts ranges of absolute or relative time residuals. This condition makes sense only in I-projects. The *Ray Path...* flag, if set, includes in the sample only rays passing through the subgrid that had been selected on M-plotter, before running Ray Sample Builder.

Creating and changing conditions

The *Edit* button drops down menu for selecting condition type. Each command invokes an appropriate dialog. Below they are considered one by one. The dialog displayed by the *Waves* command (fig. 1b), lists all waves found in Ray Catalog, but only those included in the sample have colored icons. To include a wave to the sample or exclude it, double-click the list item or use the first command of the context menu. The other two menu commands include or exclude all waves. The columns O1 and O2 contain bounds of the offset range. Their values for the selected item are edited in the fields under the list. After the fields are edited, press the Up-arrow key or click the arrow-button. Then the new values get into the list. The range [0, 0] means that no restriction on offsets is imposed. After a click on *OK*, the dialog closes, and new condition appears in the *Waves and Offsets* field of the main window.

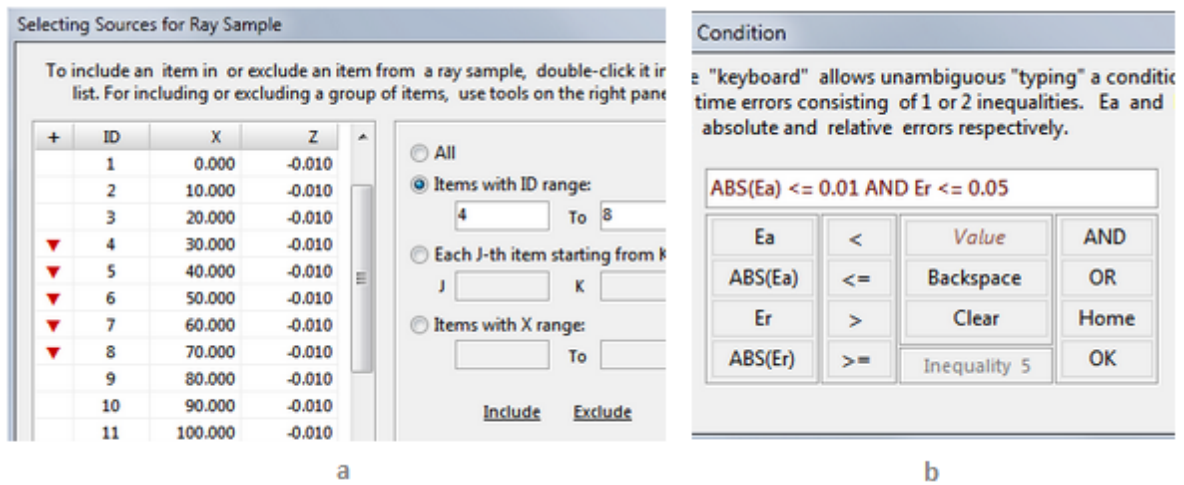


Fig. 2. Ray Sample Builder. a – composing condition on sources; b – composing condition on time residuals.

The *Sources* command of the *Edit* menu invokes the dialog shown on fig. 2a. The left part of the dialog is taken up by the list of all sources found in Ray Catalog. Those included in the sample have the source icon in the first column. Double-click is the include/exclude switch. This is an operation on an individual source. Group operations are presented on the right panel. A group can be defined with one of the four ways:

- all sources;
- sources whose IDs fall into a specified range;
- each J-th source starting from the K-th;
- sources with X-coordinate from a specified range.

To select a group, click on its radio-button and type the values needed in the edit fields. Then click *Include* or *Exclude*. The source list changes accordingly. Mind that *Include* does not mean *Include only*: the items included before the operation stay included. Now click *OK* to form the condition in the main window. This condition is simply a list of IDs of the sources included. Compose the receiver condition in the same way.

The *Errors* command of the *Edit* menu displays the dialog to make up a condition on time errors. The condition has the form of inequality or two inequalities joined with logical AND or OR. The dialog imitates a calculator, on whose keyboard one can select inequality components: error expression (Ea, Er or their absolute values), inequality sign, value and logical operator. The *Value* field becomes an edit field after a click. The label *Inequality N* (N = 1 or 2) shows which of two inequalities has currently input focus. The *OK* button ends editing.

Creating and Plotting a Sample. Current Ray Set

After the conditions are composed, click on the *Create* button. Ray Sample Builder performs sampling, saves the sample and shuts down. Then the sampled rays are rendered on M-plotter. If T-plotter is active, it draws the corresponding TX-curves. Sample volume can be seen on the status panel. The sample is stored in its own database, separately of Ray Catalog. The user can visualize both storages in turn using r-menu commands *Display All Rays* or *Display Last Ray Sample*. The current state corresponds to the command marked with bold dot on the left. The selected set of

rays (All rays or Sample) is called the *current ray set*. The ray picture on screen always images the current ray set. The status panel informs the user of which set is the current. To view the last sample criterion, start Ray Sample Builder and look at its main window, or apply the *Last Sample Description* command of the *Rays* menu or *r*-menu. These tools display the text description of the sample.

Saving ray samples for tomography inversion in I-projects

If a ray sample is created for the aim declared in the heading, save it in following way:

- close FPV and activate Project Manager;
- In Processing Tree select the f-node, in which the sample has just been created;
- click on the tool bar button *Copy* or apply the *Processing Tree/Copy Node* command.

Project Manager creates a new sibling f-node in Processing Tree and puts necessary data into it. This new f-node is called *virtual*. Virtual f-nodes are not created to store forward problem solutions as *true* f-nodes do, but to store ray samples from true f-nodes. Virtual f-nodes are created to solve tomography inversion problem on sampled data.

It is possible to save several ray samples from a true f-node, i.e. create several virtual sibling f-nodes. A virtual f-node has a modified icon to be recognized in Processing Tree. If one applies the *View FPS Log* command on a virtual f-node, one gets a text with information of the source f-node and sample description instead of a log. A virtual f-node can be created only from a true f-node.

5 Studying Time Residuals in I-project

Statistics

The *Error Statistics* command can be found T-plotter context and rubber-band menus. In the plotter menu it relates to the entire *current ray set*; in the rb-menu – to the set of (X, T) points inside rubber-band. Results of calculation are displayed in the dialog shown on fig. 1. If it was invoked from the rb-menu, the rubber-band domain is marked with color.

The following statistics are computed: a number of rays in current ray set; mean value; minimal and maximal values; median; standard deviation. Statistics are calculated for absolute residual $E_a = T_o - T_c$ (upper table) and relative residual $E_r = E_a/T_c$ (lower table). Here T_o and T_c – observed and computed traveltimes. Moreover, the statistics are computed for all rays (column 1) and for ray subsets with positive and negative errors (columns 2 and 3). Results in columns 2 and 3 point to how velocity should be changed to decrease residual values. The results can be exported to an ASCII file with the *Save* button. The file will contain information on the current ray set and on the rubber-band domain if it was selected.

Absolute Error $E_a = T_o - T_c$			
Statistic	All Rays	$E_a \geq 0$	$E_a < 0$
Number of rays	2903	1754	1149
Mean value	0.00022	0.00056	-0.00031
Minimal value	-0.00155	0.00000	-0.00155
Maximal value	0.00267	0.00267	0.00000
Median	0.00012	0.00039	-0.00026
Std deviation	0.00062	0.00054	0.00026
Relative Error $E_r = E_a / T_c$			
Mean value	0.00009	0.00024	-0.00015
Minimal value	-0.05380	0.00000	-0.05380
Maximal value	0.08960	0.08960	0.00000
Median	0.00001	0.00003	-0.00002
Std deviation	0.00302	0.00350	0.00208

Fig. 2. Dialog for displaying residual statistics.

Mean value and standard deviation of E_a and E_r for points of a selected TX-curve can be found in the dialog displaying TX-curve table (the *Selected TX-Curve in Detail* command).

Relative Residual Distribution

The variable of importance studied here is absolute value of relative residual $e = |E_r| = |T_o - T_c|/T_c$. As the statistics, distributions of e are computed for three ray sets: A – all rays of the current ray set; P – subset of rays with positive residuals; and N – subset of rays with negative errors. Each distribution is represented by two plots: that of density function $D(e)$ and that of complementary cumulative distribution function $C(e)$.

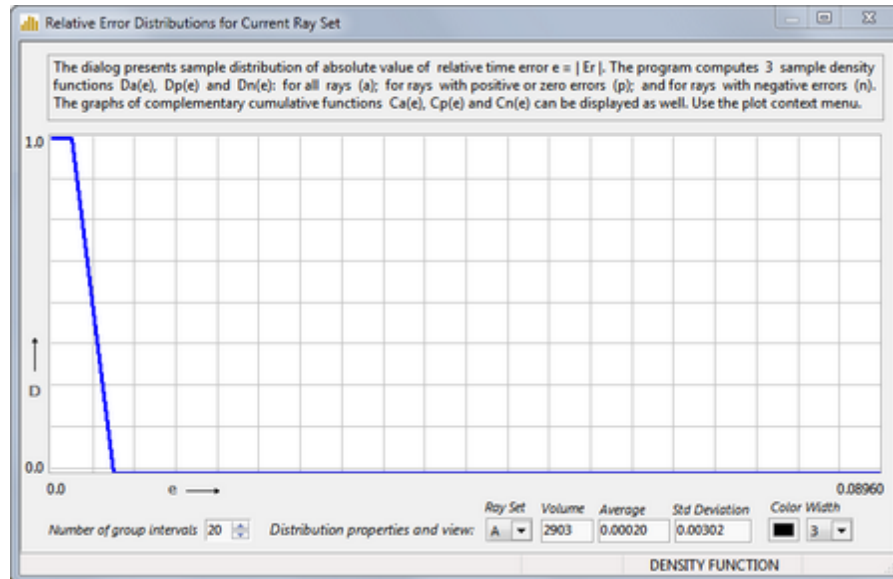



Fig. 2. Relative residual distributions.

To display the distributions, select the *Relative Error Distribution* command in the T-plotter menu. The displayed dialog shows the graph of $D(e)$ for the ray set A on its plotter. The plotter is managed by the controls at the dialog bottom and by the plotter context menu. The drop-down list *Number of group intervals* allows selecting an appropriate number of group intervals for building sample distributions. The *Ray Set* drop-down list switches between the sets A, P and N. Next three fields display distribution statistics. The two last controls allow changing the plot drawing attributes. All info displayed relates to the currently selected ray set. The plotter context menu contains switches between density and cumulative functions and flags for selecting a graphs to display.

6 Ray Coverage Density

Ray Coverage (RC) characterizes the degree of coverage of the model with rays. RC can be computed in projects of both types but it is really important for I-projects. It predicts how sensitive the solution of tomography inverse problem is to changing velocity in different model domains. Sensitivity is high if the domain is intersected by a large quantity of rays and is still higher if rays pass in different directions. On the other hand, if domain is crossed with no rays at all, velocity change does not affect the solution at all. Sensitivity is directly connected with reliability of the solution: the better the domain is illuminated by rays, the more reliable will be the velocity refinement.

FPV computes three density functions $RC(c)$, $RCH(c)$, $RCV(c)$, where c is a grid cell. $RC(c)$ is a total amount of rays passing through c , RCH is a number of subhorizontal rays, RCV – a number of subvertical rays. A ray intersects a cell *subvertically* if the angle between its link in the cell and the vertical is less than $\pi/4$ by absolute value. Otherwise, a ray passes through a cell *subhorizontally*.

All three functions are computed once and then stored. They can be displayed on M-plotter as color maps and exported to graphic file. Work with ray coverage is managed with the tool bar button  and its drop-down menu (*rc-menu*). The menu commands are accessible only after RC is computed. To do that, click on the button and in the *Ray Coverage Functions* dialog (fig. 1a) click on the *Compute* button.

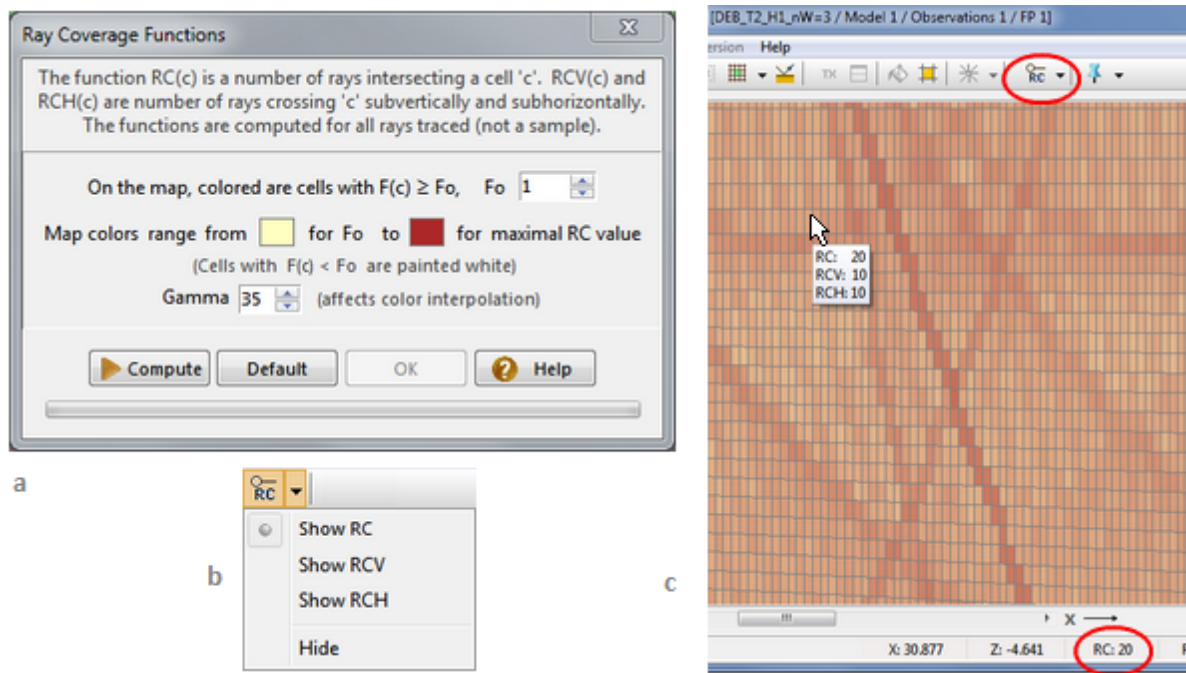


Fig. 1. Calculation and visualization of the ray coverage functions.
a – dialog for computing and setting map parameters; b – rc-menu; c – RC map.



After computation, ray coverage functions can be visualized by commands of the rc-menu (fig. 1b). The dialog fields are destined for setting map parameters. Let $F(c)$ is one of the density functions. On the $F(c)$ map, painted are cells for which $F(c) \geq F_0$, where parameter F_0 is specified in the spin-edit field. The colors used for cells with minimal and maximal values of $F(c)$ are the colors of the two rectangles on the next dialog line. Adjust their colors clicking on them (click invokes the Windows Color dialog). The color of cells with $F(c)$ values between the minimal and the maximal are calculated by means of interpolation between the rectangle colors. Parameter Gamma controls the interpolation and can be adjusted experimentally.


Studying ray coverage maps, note that name and value of the selected function is displayed on the status bar velocity field. Selecting the hint window mode (*View/X,Y,V under Cursor*), the user can observe values of all three functions just under the cursor (fig. 1c). To return to ray picture, select *Hide* in the rc-menu.

7 Preparing to Tomography

In the complicated interpretation process, the following important problem occurs quite often: how to refine velocity distribution with tomography inversion and ensure that velocity in certain model subdomains stay intact. This problem rises routinely at the layered model interpretation where layers are studied one by one. XTomo-LM provides such opportunity. The only thing the user has to do is to describe the set of grid cells in which velocity must not change. This task is

called *freezing* or *fixing* velocity. Instead of the phrase "a cell with frozen (fixed) velocity", the shortcuts "fixed cell" or "f-cell" is used in the user interface and the documentation. Fixing cells is a function of FPV.

The commands relating to freezing are gathered in the *Inversion* section of the main menu and the drop-down menu of the tool button , which is called **f-menu** for short. The first operation to be done always is to switch on the fix mode by clicking on . The button became red to indicate the fix mode is on. Additionally, the FIX MODE label appears at right end of the status bar. If before switching the mode on, the set of f-cells has already been non-empty, all f-cells are painted in the reserved color.

To make the cells of a selected grid subset fixed, use the M-plotter menu command *Add to Fixed Cells*. To release the selected cells, use the *Release Fixed Cells* command. The *Release All* command can be found in the *Inversion* menu or in the f-menu. It releases all fixed cells. The color for f-cells can be changed by the f-menu command *Change Fix Color*. To exit fix-mode, click on  again. The set of fixed cells is kept after exit the fix-mode and between FPV sessions.

The fixation mechanism allows freezing velocity in cells depending on ray coverage. For example, the user can block changing velocity during tomography inversion in all cells with zero or too small value of an RC function. The operation is started by the *RC Conditioned Fix* command of the f-menu. It displays the dialog for defining the condition (fig. 1).

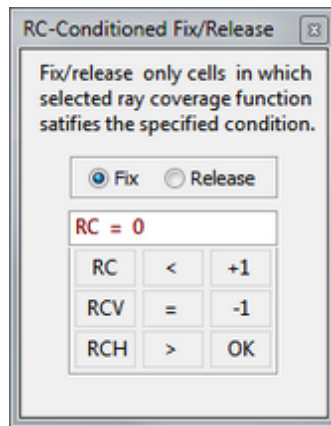


Fig. 1. Composing condition for fixing cells depending on ray coverage.

Click one of radio-buttons to select the operation (fix or release). The default condition is $RC = 0$. One can change in it: (1) density function; (2) compare sign; (3) value. Use the keyboard for that. Value can be typed straight in the field or using the keys $+1$ и -1 to increment or decrement the current value. A click on *OK* starts the operation.

8 Export

Graphics

FPV can export the content of M- and T-plotters to graphic files in the same sense as [Model Viewer](#). The *File/Image Export* command has the submenu for choosing an export object: *Model*, *Rays*, *Ray Coverage*, *TX-Curves*. The same submenu is dropped down by the corresponding tool bar button. The *Ray Coverage* command is accessible only when M-plotter displays ray coverage map. At that, exported is the map of the function currently displayed.

ASCII files

Text export is performed by the FPV ASCII Exporter module. It can be launched, after the first ray rendering, by the *File/ASCII Export* command. The module main window looks like the export dialog of Model Viewer (fig. 1a). The export object is selected from the drop-down list *Data to Export*. The list includes the following items:

1. Export of velocity function under a condition on ray coverage. $V(x, z)$ is exported to a DAT file only for those (x, z) which serve as vertices of cells with a specific ray coverage value.
2. Export of ray paths to a [BLN](#) file for a specified ray set.
3. Export of a specified ray coverage function to a DAT file.
4. Creation of an SRT file for a specified ray set with optional Gauss noise added to the *computed* times.

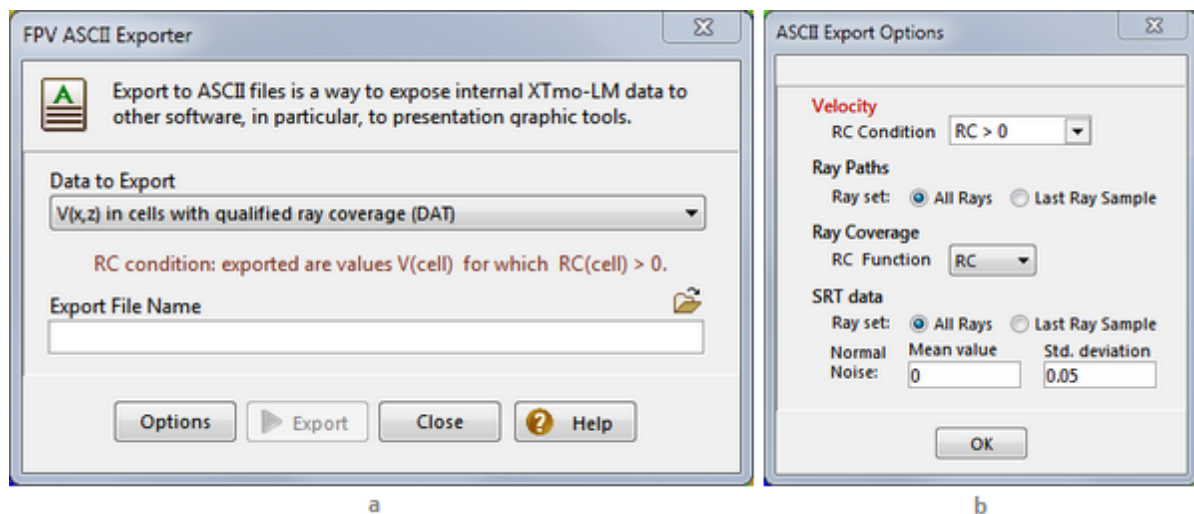


Fig. 1. Module FPV ASCII Exporter. a – main menu with list of options; b – options dialog .

In each export option, one or more parameters need to be specified. Use the *Options* button (fig. 1b) to do the job. The heading of a parameters group pertaining to the selected object is printed in red. In case of 1 (Velocity), the type of ray coverage function is chosen by the down arrow, while the threshold value is entered into the field. In case of 2 (Ray paths), click radio-button to choose between all rays and last ray sample. In case of 4 (Observation data), the first parameter is the kind of ray set (radio-buttons), the next two are the parameters of the Gauss distribution: mean value and standard deviation. To kill noise, set standard deviation to 0.

Tomography

1 Theory

Tomography problem

In [Introduction](#), the inverse tomography problem was stated as Linear Least Square Problem

$$(3a) \quad |\Delta \mathbf{T} - \mathbf{D} \cdot \Delta \mathbf{V}|^2 \rightarrow \min.$$

Here $\Delta \mathbf{V}$ is the desired vector of corrections to the current (initial) velocity $\mathbf{V}_0(x, z)$. Velocity is defined by its values at grid nodes, therefore, it is a vector whose dimension is equal to a number of grid cells m . $\Delta \mathbf{T}$ is vector of time residuals, i.e. differences between observed traveltimes and traveltimes along rays computed as forward problem solution for velocity function \mathbf{V}_0 . Its dimension is equal to a number k of rays in Ray Catalog of the current f-node. Finally, \mathbf{D} is matrix with k rows and m columns. The matrix element at i -th row and j -th column is partial derivative of traveltime of i -th ray by velocity in j -th cell. The derivative is evaluated at $\mathbf{V} = \mathbf{V}_0$. $|\mathbf{X}|$ denotes Euclidean norm of a vector \mathbf{X} (square root from the sum of squared vector components). Formula (3a) differs from (3) in Introduction in two inessentials: norm sign has no dimension subscript; norm is squared. In our case dimension is known (k) and is omitted. Square does not change the essence but simplifies formulas. New approximation to (or refinement of) model velocity is $\mathbf{V}_0 + \Delta \mathbf{V}$.

Reduction of general inverse problem to the tomography inverse problem was carried out under assumption that corrections $\Delta \mathbf{V}$ are small. The assumption allowed replacing of highly complicated operator of calculation traveltime along the traced ray with its linear approximation \mathbf{D} . Moreover, elements of matrix \mathbf{D} can be calculated under the same assumption. The difficulty here is that velocity change implies change in ray trajectory. However, it can be proved that changes in traveltime, caused by change in ray path are negligibly small to compare with change of velocity itself. Under the assumption that $\Delta \mathbf{T}$ depends only on $\Delta \mathbf{V}$, the elements of \mathbf{D} can be calculated explicitly through the cell traveltimes and components of \mathbf{V}_0 .

Regularization

In optimization theory, the function to be minimized is called the *objective function*. The trouble with the objective function in (3a) is that the problem is *ill-posed*. That means that It is not, generally speaking, uniquely solvable and its solutions are not robust with relation to small variations of input data. To guarantee existence of unique minimum, the objective function must satisfy the known condition: it must be a *convex* function, which, generally, does not take place. Explanation for functions of one variable is given on fig. 1.

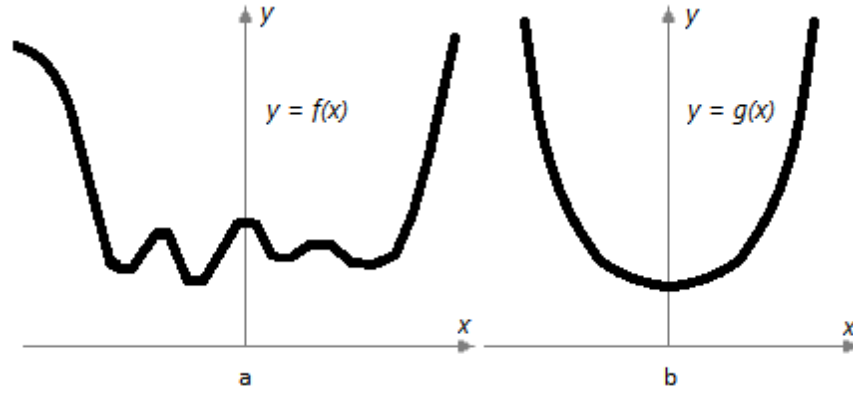


Fig. 1. If a function of one variable is convex it, surely, has a unique minimum at the stationary point. The concept of convexity is generalized to functions of multiple variables and takes the central place in the optimization theory. a – multiextremal non-convex function with many local minima; b – convex function with a unique minimum.

The theory of ill-posed problems offers a method of *regularization* for problems of the type (3a). According to it, (3a) is replaced with a similar problem which is close to (3a) in a certain sense and, at the same time, is well-posed or robust, i.e. has the unique stable solution. In our case it has the form

$$(4) \quad |\Delta \mathbf{T} - \mathbf{D} \cdot \Delta \mathbf{V}|^2 + \alpha \cdot R(\Delta \mathbf{V}) \rightarrow \min,$$

where R is a convex function of k variables. The real $\alpha > 0$ is called the *regularization* parameter. Its meaning is clear: on the one hand it must be as small as possible to keep (3a) and (4) close; on the other hand, it must secure that the new objective function be convex, which is possible if only it is large enough. The function R is called a *regularizer* or *stabilizer* for the problem (3a).

Additionally to securing unique solvability, *regularization* allows, in a sense, to make its solution "more physical". The mathematical minimization process is directed by the only target: to get to point of minimum *at any price*. From the physical point of view, the price may prove to be too high. For example, velocity values in some cells may be too large or too small, But the common case is that velocity proves to be oscillating. Both artifacts have no physical sense. An appropriate choice of R helps eliminate such extremities.

Stabilizers

XTomo-LM uses three built-in stabilizers, which are found to be appropriate to tomography applications. The standard stabilizer is of the form

$$R_d(\Delta \mathbf{V}) = |\Delta \mathbf{V}|^2.$$

This stabilizer secures existing of the unique solution if α is large enough. Moreover, time residuals are minimized in parallel with minimizing values of velocity corrections. The mentioned above situation with abnormal velocity values is excluded. The R_d stabilizer is called *damper* in the user interface.

If $\Delta \mathbf{U}$ is the solution of (4) c $R = R_d$, then the components of vector $\Delta \mathbf{U}$ corresponding to cells that are crossed by no ray are zeros. Otherwise, vector $\Delta \mathbf{U}'$ with zero components in those cells would not change the first member in (4) but would make the second one less. This contradicts to uniqueness of solution. Hence, R_d does not change velocity in cells not illuminated with rays.

Damper is good in excluding abnormal values but it does nothing to fight velocity oscillations. To proceed with this problem, consider Smoothing Damper R_s – the stabilizer that for the smooth vector function $\Delta \mathbf{V}$ looks like this:

$$R_s(\Delta \mathbf{V}) = R_d(\Delta \mathbf{V}) + |(\Delta \mathbf{V})_x|^2 + |(\Delta \mathbf{V})_z|^2 = R_d(\Delta \mathbf{V}) + |\text{grad}(\Delta \mathbf{V})|^2.$$

Here, subscripts x and z mean partial derivatives. Thus, R_s includes in the minimization process $\Delta \mathbf{V}$ gradient ; therefore, oscillations will be minimized in parallel with time residuals. In discrete case, derivatives are replaced with finite differences. Unlike Damper, Smoothing Damper may change velocity in any cell, even in those crossed by no ray. If this effect is undesirable, freeze velocity in cells with zero ray coverage beforehand (in FPV).

In geophysical applications, especially, in deep seismic investigations, Smoothing Damper is not quite relevant because velocity horizontal gradient is small to compare with the vertical. The following stabilizer (Evening Damper) is more appropriate in such cases:

$$R_e(\Delta \mathbf{V}) = R_d(\Delta \mathbf{V}) + \beta \cdot |(\Delta \mathbf{V})_x|^2 + |(\Delta \mathbf{V})_z|^2.$$

It differs from R_s by the factor β before the norm of partial derivative by x. Choice of β allows taking into account domination of vertical ($\beta > 1$) or horizontal ($\beta < 1$) $\Delta \mathbf{V}$ gradient (the greater member is minimized in the first place) .

Note, that beneficial effects of stabilizers are not unconditional. They are stipulated by the value of the regularization parameter α .

Constraints

There is one more important means of control over the solution: constraints imposed on components of vector $\Delta \mathbf{V}$. The minimization problem (4) is unconditional (unconstrained). Formally, its solution is searched in the set of all m-vectors (though we came to it assuming smallness of corrections). It is quite probable that the solution would come out of the reasonable limits. XTomo-LM allows solving the constrained problem (4) too.

First, consider *interval constrains*:

$$\mathbf{d}_1 \leq \Delta \mathbf{V} \leq \mathbf{d}_2$$

where \mathbf{d}_1 and \mathbf{d}_2 are known vectors. More precisely, XTomo-LM offers two kinds of special interval constraints (5):

$$(5a) \quad v_1 \leq (\mathbf{V}_0 + \Delta \mathbf{V})_j \leq v_2,$$

$$(5r) \quad p_1 \cdot \mathbf{V}_0 \leq \Delta \mathbf{V} \leq p_2 \cdot \mathbf{V}_0,$$

where v_1, v_2, p_1 and p_2 are positive real numbers, j is any vector component. Constraints (5a) sets absolute bounds for the future refined velocity vector. Of course, these bound must not narrow the range of \mathbf{V}_0 , otherwise, the objective function might increase. Constraints (5r) restrict relative change of \mathbf{V}_0 in the course of minimization. Only one of conditions (5) can be used.

The second kind of constraints is nullifying some components of vector $\Delta\mathbf{V}$ (zero correction constraints):

$$(5f) \quad (\Delta\mathbf{V})_j = 0, \quad j \text{ spans a subset of } \{1, 2, \dots, m\}.$$

It is easy to see that (5f) is equivalent to freezing velocity on a subset of grid cells. Constraints (5f) can be combined with the interval constraints.

Solving the problem. Recommendations

The process of getting solution to the constrained problem (4) is hidden from the user. It is enough to know the following. Methods of solving of optimization problems are iterative. XTomo-LM makes use of one of the relaxation (descent) methods. The stopping criterion is defined by the specified value of solution accuracy. If accuracy is ϵ , and new iteration changes the objective function by a value less than ϵ , the process is stopped. A number of iterations for the unconstrained problem is comparatively small, if only not to approach the critical situation, when the problem becomes ill-posed (α is too small). In this case number of iterations goes up sharply, but the required accuracy is not attained. The problem gets caught in an endless loop. To avoid that, the limit number of iterations must be specified together with the other parameters. Do not heighten the accuracy unreasonably; that can lead to the same situation as lowering α .

The constrained problem is much more complicated. When a sequence of iterations meets the domain border, which is defined by the constraints, the descent to the solution starts as though from the beginning. The total number of iterations increases. It is recommended to start with unconstrained problem (with fixed cells, if necessary) and look at what is the solution like. If using constrains is, actually, necessary add them and solve the problem once more. In any case, there are several parameters to be specified (α , β , accuracy and so on), so comparing several variants of solutions is a normal practice.

2 Inverse Problem Solver

The Inverse Problem Solver (IPS) module is launched by the *Solve Inverse Problem* command on an f-node of Processing Tree. Just after launch, Project manager creates a new i-node, for storing the results of inversion. IPS implements getting solution of tomography inversion problem exactly as it is described in "[theory](#)". The user interface uses the notations from the previous section.

Task

When started, the module offers the default task. The user edits it in the following way. The module main window has three tabs. The first one, *Stabilizer*, is for selection of stabilizer type.

There is a check box at the tab bottom for selection the simplified estimating inversion methods (back projections). It is kept only for old users. The contemporary work station productivity requires no preliminary estimation.

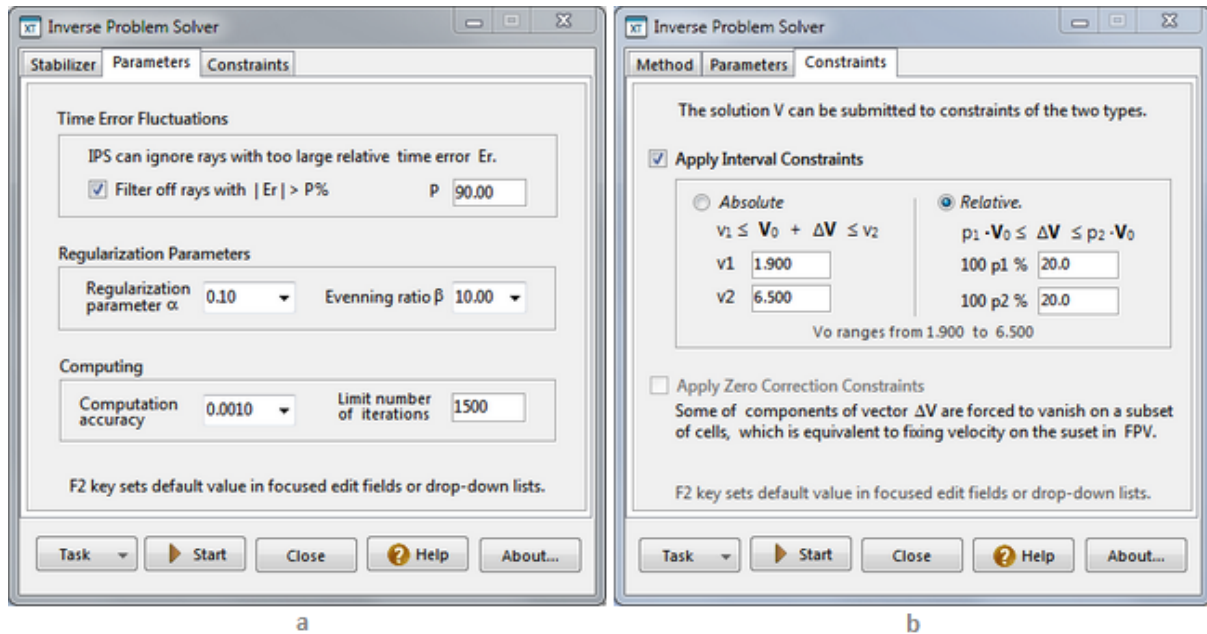


Fig. 1. IPS main window: a – Parameters tab ; b – Constraints tab.

On the *Parameters* tab (fig. 1a) the inversion parameters are to be defined. The *Time Error Fluctuations* panel allows excluding rays with large relative residuals. The *Regularization Parameters* panel is for specifying the regularization parameter α . If Evening Damper is selected as a stabilizer on the first tab, its parameter β is specified here too. The *Computing* panel is destined for specifying the solution accuracy and limit number of iterations.

On the *Constraints* tab (fig. 1b) the user defines constraints. In the default task, no constraints are set. To specify interval constraints, check the *Apply Interval Constraints* box. Then choose constraint type using radio-buttons and define the parameter values. In the case of absolute constraints, the range $\mathbf{V}_0 + \Delta\mathbf{V}$ is, by default, the same as the \mathbf{V}_0 range. It can be only widened. The relative constraints parameters are specified as percentage.

The *Apply Zero Correction Constraints* check box is accessible if only velocity had been frozen on a set of cell in **FPV**. If the box is checked this information will be taken into account, else it is ignored.

The composed task can be saved to a user file. To do that, click on the *Task* button and in the dropped-down menu select *Save as*. To load an earlier saved file, use the *Load* command of the same menu. Whether the user saves a task or not, the last task in the module session is saved automatically and can be loaded by the *Load Last* command of the button menu. The *Default* command sets the default task. To change an individual field for its default value, press F2 when a field has input focus.

Execution

After clicking on the *Start* button, the dialog representing execution process appears on screen. (fig. 2). It contains two progress bars.

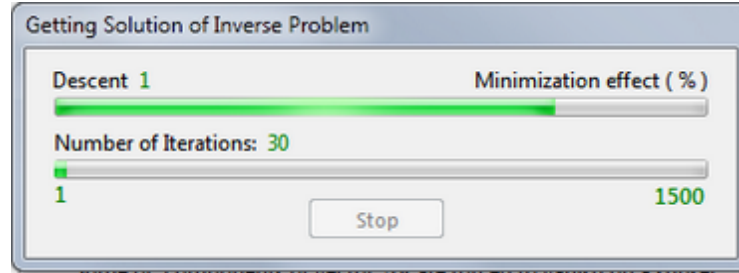


Fig. 2. Depiction of execution progress.

First, the top progress bar reflects the data preparation stage. Then both image process of getting solution. Special module FTI solves the Regularized Least Square Problem. The name FTI appears in some messages. The top progress bar shows the degree of dropping of the objective function during one descent. The bottom one reflects the growth of total number of iterations. The *Stop* button interrupts execution. Execution ends with a message informing the user of the result. Below typical termination scenarios are listed.

1. Success. Relative drop of the objective function is displayed.
2. Solution is found, but change of objective function is insignificant.
3. The initial value of the objective function is close to zero. Minimization is not needed.
4. The process is stopped because the user-defined limit number of iterations was exceeded, while the specified accuracy was not attained.
5. Fatal error occurred, or the solution makes objective function greater.



In the cases of 1–3 the refined velocity is stored together with the information about the task and solution. To look at it, use the *Info on Inversion* command of Processing Tree menu on the newly created i-node.

The other cases are treated as a failure. When the user quits IPS, the created i-node is removed from the Tree. The case of 4, when limit number of iteration is great (the first thousands), indicates the critical situation: the problem is ill-posed. The user must increase α and, possibly, decrease accuracy and try again without exiting IPS. When any of the task parameter changes, the *Start* button becomes active again.

3 Viewing solution

Viewing tomography inversion result is implemented by the Inverse Problem Viewer (IPS) module. It is launched by the *View Inverse Problem Solution* command of the Processing Tree menu invoked on an i-node. In this section we return vector notations to the functional and regard functions V_0 and $V = V_0 + \Delta V$ depending on a grid cell c . The plotter of graphic module can display initial model $V_0(c)$, refined model $V(c)$ and the *difference map* $\delta V(c)$:

$$\delta V(c) = 100\% \cdot \Delta V(c) / V_0(c).$$

$\delta V(c)$ is percentage wise relative value of $\Delta V(c)$. After the module is started, the plotter displays the refined velocity $V(c)$. Comparing to Model Viewer, IPS has two new buttons on the tool bar:  and . The former controls the plotter content, the latter launches velocity profiles viewer which is a part of the Velocity Comparator utility (details are [here](#)). Both buttons have their sibling commands in the main menu.

The first button mentioned drops down a menu when clicked on the arrow. The menu is a switch between three plotter options: to show the refined velocity, or the initial velocity, or difference map. When clicked on the icon, the button invokes the dialog for adjusting difference map settings (fig. 1a).

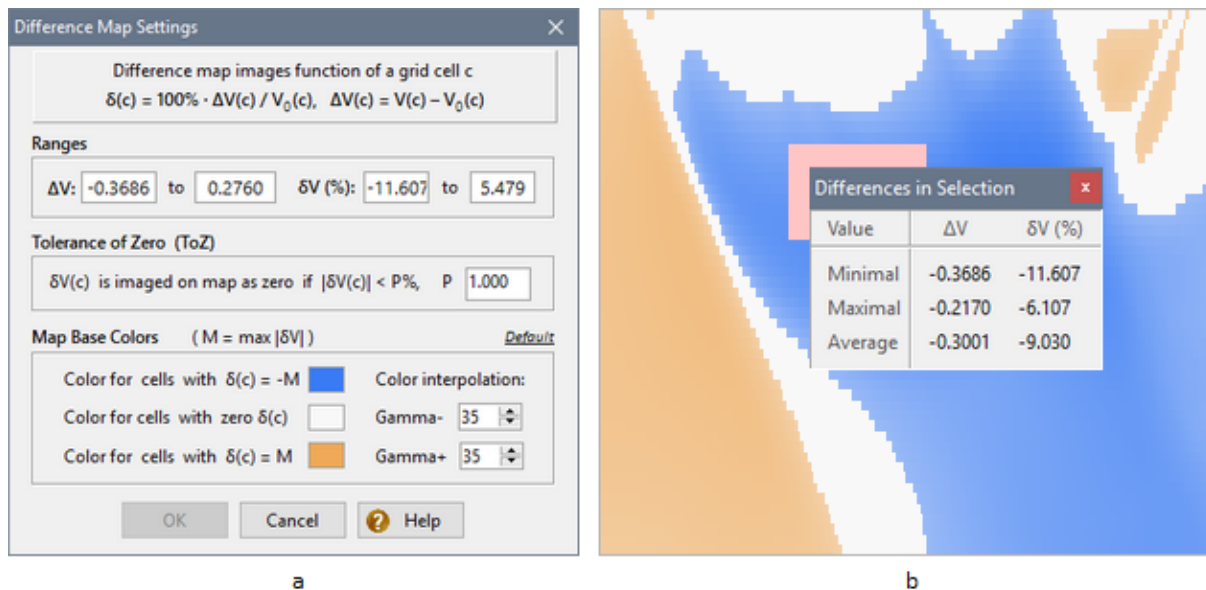


Fig. 1. Inverse Problem Viewer.

a – dialog for adjusting difference map; b – a fragment of difference map with a selection.

At the dialog's top, ranges of absolute corrections ΔV and relative corrections δV are shown. Below is *tolerance of zero* or ToZ. This value defines when δV is imaged on the map as 0. The user can adjust it in order that the map display only significant velocity change.

The three color parameters are *base colors*. If $M = \max |\delta V|$ over all grid cells, then the first color C_1 is used for those cells in which $\delta V = M$; the second color C_0 is for cells with $\delta V = 0$ counting with ToZ; the third color C_{-1} – for cells with $\delta V = -M$. Color of an arbitrary cell is determined by interpolation between C_0 and C_1 or between C_{-1} and C_0 depending on δV sign. Interpolation can be controlled by parameters *Gamma+* and *Gamma-*. Changing Gamma shifts color towards one of the base colors.

If a subset is selected on the grid, then, if V or V_0 is currently imaged, the *Velocity in Selection* command of the plotter menu displays velocity range and mean value of V or V_0 on the subset; if δV is mapped, the command changes its name for *Difference in Selection* and displays ranges and mean values of both ΔV and δV in the selected domain, as on fig. 1b.

The difference map can be exported to a graphic file by the *File/Export Difference Map*.

4 Termination of interpretation cycle

Tomography inversion is the last step of a full *cycle* of the process of kinematic interpretation, but it is NOT a point of exit. The obtained solution is to be estimated. Percent of objective function drop, which IPS displays after inversion, is a technical indicator and is no more than evidence of successful solving of the mathematical problem. The only true numeric estimations of the solution found are statistics of residuals between the observed traveltimes and traveltimes computed in the course of ray-tracing on the *refined model*.

Thus, after getting the solution of tomography inversion problem, the user begins a new cycle of interpretation following these directions:

1. Make a copy of i-node with the obtained inverse problem solution using Project Manager's tool bar button. The i-node copy is a new Model node with the *refined velocity*.
2. Select the o-node, which is the ancestor of the i-node in question and copy it as a child to the newly created m-node.
3. On the just created o-node, launch FPS to get forward problem solution for the same task as in previous cycle.
4. Explore the forward problem solution, in particular, residual statistics and compare them with those in the previous cycle.
5. Using results of step 4, make decision on termination the process or moving on.

TX-Curve Inversion

1 Overview

The chapter is devoted to the XTomo-LM tools for travelttime curve inversion aimed at reconstruction of different model elements. Diving wave TX-curves inversion provides an initial approximation to the true velocity distribution. Reflection or head wave curve inversion means building a seismic horizon on which the wave was formed. Clearly, these problems are set only for 2D profiling data only. The strict problem definitions are stated in the chapter topics.

In the [Building Initial Velocity](#), the well-posed inverse problem for diving wave TX-curves is stated and then the user interface of the module which implements getting solution to the problem is described. The resulting velocity section can be used as initial in tomography inversion.

In section [Building Horizons](#), the inverse problems for TX-curve set of reflected and head waves are stated together with general approach to getting solution. The next section [Selecting TX-curves](#) describes how to select optimal subset of curves for building horizons.

In the sections [Building Reflectors](#) and [Building Refractors](#) the user interface of the modules implementing the inversion algorithms is explained. They are followed by the [Samples](#) topic containing a comment to sample projects supplied with the product.

2 Building Initial Velocity

[The Idea](#) – [Comment](#) – [Module DWI](#) – [TX-curve Sample](#). [Auto-sampling](#) – [Velocity Range](#) – [Program Control](#) – [Using Results](#). [Export](#)

Building velocity section is the aim of seismic tomography, and its first step is to choose an appropriate initial velocity section. If the initial approximation is chosen reasonably, the successive refinements lead to the physically sensible result. If not, one may move in wrong direction. In case of 2D profiling, diving wave TX-curves enable one to get the initial velocity that is certainly "reasonable". The less the effect of inversion layeres is, the better the approximation is to the true velocity distribution.

The Idea

Problem 1. Let the observation line coincide with the X-axis. Let \mathbf{M}_1 be a class of one-dimensional velocity models $V(z)$ with decreasing functions V (decreasing in z means increasing in depth). Let $T_v = T_v(z)$ be forward problem solution for model V from \mathbf{M}_1 . Consider the following inverse

problem: for any observed diving wave TX-curve $T = T_{\text{obs}}(x)$ it is required to find a model V_0 in \mathbf{M}_1 which minimizes RMS deviation T_v from T_{obs} . According to XTomo-LM terminology, a solution of Problem 1 is a velocity column (VC). Suppose for the time being, that problem 1 has a unique solution.

Before moving to two-dimensional problem, let us introduce some new terms. If a TX-curve curve is generated by the source S on the observation line, then x_s denote x -coordinate of S and x_{mo} – x -coordinate of the curve point with maximal offset. The interval $[x_s, x_{\text{mo}}]$ (or $[x_{\text{mo}}, x_s]$ for a reverse curve) is called TX-curve *domain*. TX-curve *midpoint* and *base* are, respectively, the domain center and its length.

Now consider diving wave TX-curve set $\{T_{\text{obs},k}; k = 1, \dots, n\}$, and denote by $V_{0k}(z)$ solution of the problem 1 for the k -th TX-curve. Let us attach $V_{0k}(z)$ to the k -th TX-curve midpoint x_k . Let \mathbf{M}_2 designates class of two-dimensional velocity distributions $V(x, z)$ with function V decreasing in z for each x fixed. There is an algorithm of building two-dimensional velocity distribution $V_0(x, z)$ from \mathbf{M}_2 such that $V_0(x_k, z) = V_{0k}(z)$, $k = 1, \dots, n$. It is V_0 that is about to be used as the initial velocity distribution for the inverse tomography problem.

Comment

The content of the previous section requires some comment. First, Problem 1 is not *correct* in mathematical sense: unique existence of its solution cannot be guaranteed and, therefore, there is no robust algorithm for getting solution. However, if we make class \mathbf{M}_1 slightly narrower, unique solvability of the problem can be proved and a stable algorithm built. It is enough to suppose that for any V from \mathbf{M}_1 there exists such $\varepsilon > 0$ that $z_1 > z_2$ implies $V(z_1) < V(z_2) - \varepsilon$. That means that a decreasing function $V(z)$ is separated from a constant function by ε on any z -interval. With this refinement, Problem 1 becomes well-posed.

The choice of points x_k is motivated by the following considerations. Examine a diving wave ray running from the source point $S(0, 0)$ to the observation point $(X, 0)$ in a model from \mathbf{M}_1 . The ray point with the maximal depth has x -coordinate equal to $X/2$. Hence, it is natural to attach the velocity column $V(z)$ obtained by inversion of a TX-curve to its midpoint.

Building $V_0(x, z)$ includes two steps: solving problem 1 (the true problem) and processing the computed VC set (a technical task). It was silently supposed that each curve from a given TX-curve set is inverted to contribute to $V_0(x, z)$. In practice, however, it does not make sense because of significant variation in curve bases. The deepest velocity column is obtained by inversion of the TX-curve with maximal base. This is true, at least, for media with relatively small horizontal gradient. For such media it is enough to use this one column as initial approximation. Generally, one might want to catch velocity changes along the line by inversion the TX-curves from an appropriate sample.

Diving Wave TX-curve Inverter (DWI)

The DWI module is launch by the *TX-curve Inversion/Build Initial Velocity Distribution* command of the Processing Tree menu invoked on an o-node. The module's main window displays the list of diving wave TX-curves found in Ray Catalog of the node (fig. 1).

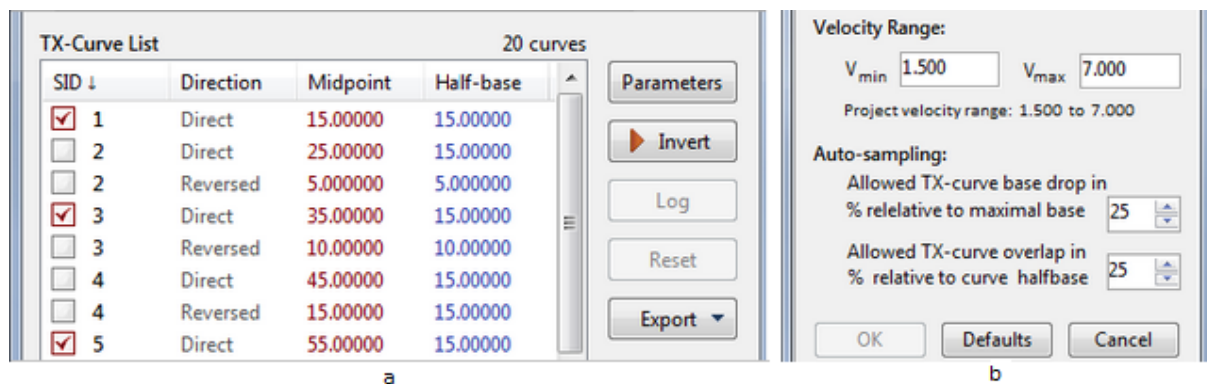


Fig. 1. a – DWI main window; b – dialog for adjusting program parameters.

The column named SID contains source IDs; meaning of other columns is evident. By default, the list is sorted by SID. To resort the list, click on another column's header. Each list items begins with a check box to mark whether the item is intended to be inverted.

TX-curve Sample. Auto-sampling

One can sample curves for inversion the way one prefers, clicking on item check boxes or/and using the context menu commands. Having the [view](#) of the TX-curve set on screen is helpful for the job. Alternatively, one can entrust the task to the program which performs automatic sampling (*auto-sampling*). Auto-sampling results in the curve subset with maximal bases and controlled length of overlapping. One uses two parameters to direct auto-sampling. The first parameter is permitted drop of curve base relative to the maximal base. The second parameter controls maximal overlapping of neighboring curves as percentage of the half-base. Here "neighboring" relates to ordering by midpoints. Parameters of auto-sampling are set in the *Parameters* dialog (fig. 1b). To display it, click on the top button of the main window.

Velocity Range

The third module parameter is permissible range of velocity in velocity columns. The module signals when, in the course of TX-curve inversion, a velocity value in a VC falls beyond the user-defined interval. The range should be selected basing on physical and geological considerations. Velocity range is set in the *Parameters* dialog. The default range is borrowed from the project properties.

Program Control

Program work session includes three steps: (1) sampling TX-curves; (2) inversion; (3) analyzing the results. After step 2 is completed, one can view the inversion log and the built velocity columns in graphic or numeric form. At that, the input data (TX-curve sample and parameters) cannot be

changed until the *Reset* button is clicked on. The click ends inversion operation and enables the user to begin a new one. The TX-curves whose inversion failed are still checked, but the check box color is changed for gray. The menu commands are described in Table 1.

Table 1. Menu and button commands.

Command	Description
	<i>Popup menu</i>
<i>Check All</i>	Checks boxes of all items (includes all curves in the sample for inversion).
<i>Uncheck All</i>	Unchecks boxes of all items (excludes all curves from the sample).
<i>Auto-sampling</i>	Starts the auto-sampling procedure.
<i>Show VC Plot</i>	Displays the floating window with the velocity column plot. The command is accessible if only the selected TX-curve is successfully inverted. The window header identifies the curve and displays its midpoint (fig. 2a). Because the window is floating, it does not block access the main one, therefore, several windows may be displayed to compare VC plots .
<i>Show All from Here</i>	Performs the previous operations for all inverted curves starting from the selected (which must be checked).
<i>Sort Plots by X</i>	Rearranges the displayed plots in the increasing order of the curve midpoints.
<i>Close All Plots</i>	Closes all plot windows.
<i>Show VC as Table</i>	If the selected list item is inverted, the command displays the dialog representing the corresponding VC as a numeric table (fig. 2b). Actually, the dialog allows navigating the set of built velocity columns. There are two navigation tools: the drop-down VC list at the dialog top and arrow-buttons at the bottom.
	<i>Main window buttons</i>
<i>Parameters</i>	Invokes the dialog for adjusting program parameters (fig. 1b).
<i>Invert</i>	Starts the inversion process. If the sample contains more than one TX-curve, the operation is performed concurrently for N curves, where N is a number of logical processors in the system.
<i>Log</i>	Displays the inversion log containing records for each TX-curve from the sample. Records are not necessary ordered. To save the log, right-click it and select <i>Save As</i> in the popup menu.
<i>Reset</i>	After the inversion is completed, changing input data is blocked until the <i>Reset</i> button is clicked. After that, the user can change data to start new

	inversion operation.
<i>Export</i>	The module can export the built velocity columns to a VC file (for use in XTomo-LM) or the DAT format (for external applications). The button drops down the menu for choosing the format. About export to a VC file, see below for more details.

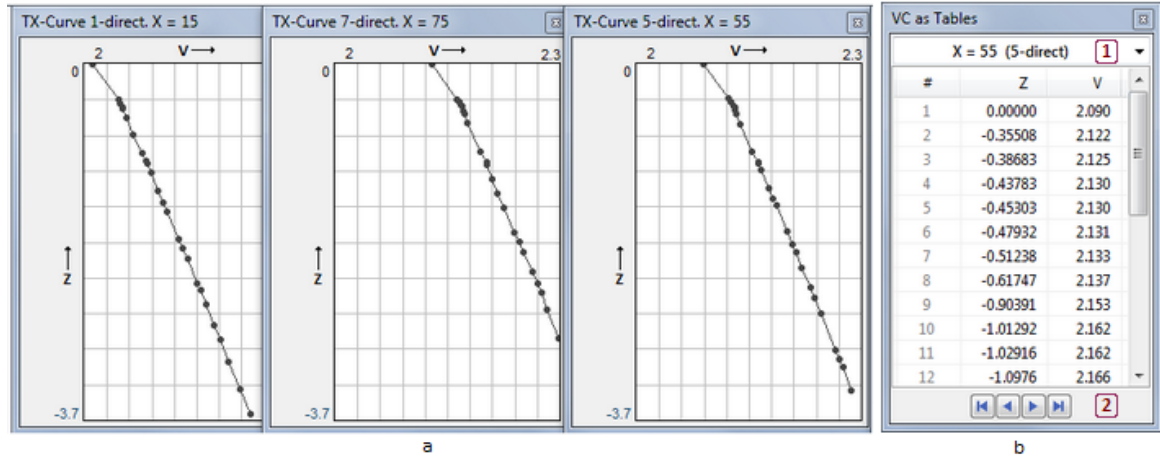


Fig. 2. Displaying velocity columns: a) VC plots; b) VC in the form of numeric table; 1, 2 – VC navigation tools.

Using Results. Export

The built velocity columns can be used either for creating the [starting model](#) of a new project or for [replacing velocity](#) in a new m-node of the current project. In both cases, XTomo-LM applies the built-in algorithm shortly described [here](#). At this point, there two issues must be explained.

1. When importing a VC file, the user has to choose how to treat velocity columns: as step functions or as piecewise linear functions. In our case, the answer is evident: the built velocity columns are interpreted at import as piecewise linear continuous functions (see more [here](#)).
2. If depth of a velocity column is less than that of lower border of the model rectangle, the import algorithm continues the column downward with constant velocity value. Thereby, the column is thrown out of \mathbf{M}_1 and velocity defined by velocity column set – from \mathbf{M}_2 . This problem is worked around by a preprocessing of each column in the course of export.

Suppose that N velocity columns have been built. The k -th VC is a set of couples $\{(Z_{ki}, V_{ki}), i = 1, \dots, m_k\}$. Let Z_{\min} be minimum over all Z_{ki} and V_{\max} be maximum of all V_{ki} , $k = 1, \dots, N$, $i = 1, m_k$. Just before exporting a VC, the program adds to it the additional bottom couple (Z^*, V^*) satisfying the conditions $Z^* < Z_{\min}$, $V^* > V_{\max}$. The same point is added to all VCs. This action defines all VCs on the same z -interval $[Z^*, 0]$ with the last velocity value V^* . The exact values of the parameters Z^* and V^* are defined by the user. In order that the starting model of a new project belong to \mathbf{M}_2 , the user must define Z^* so that it is less than the lower border of the grid rectangle.

Technically, to carry out VC export, one clicks on the *Export* button, then clicks on the *To VC file* menu command, then selects VC file name and location in the Save File dialog. After that, the module displays the *Export Parameters* dialog which shows values of Z_{\min} and V_{\max} and requires to edit fields for Z^* and V^* .

3 Building horizons

Reflections and refractions bear different information about a seismic horizon. A reflected wave is, actually, a local horizon scanner (at small offsets). On the contrary, a head wave presents integral information on a boundary as a result of "sliding" along it over large extents. Under certain conditions, both problems are posed and solved using the same method of TX-curve backward continuation (migration) within the ray approximation to wave propagation.

The TX-curve inversion procedures are hidden from the user. Only one important notion concerning these procedures reveals itself in the user interface: *eikonal*. Eikonal is a function $T_R(x, z)$ whose value at point $M(x, z)$ of the model is wave traveltimes from a fixed point R to M . R should be considered as a parameter. TX-curve backward continuation and computing eikonals for every receiver R are very close problems.

Reflection

For a reflection, the problem is posed this way. Given are: reflection TX-curve observed on the surface line; the source position; velocity distribution in the covering medium. Searched for is a reflector segment. To get the solution, two processes synchronously developing in time are studied: propagation of the direct (incident) wave from the source and TX-curve migration. Analyzing kinematics of both processes, it is possible to determine position and shape of the reflector segment. The generalization of the problem consists in multiplying sources and TX-curves to illuminate the whole reflector under observation line. The input data include a TX-curve set illuminating the reflector, possibly, with (multiple) overlaps.

In the kinematic analysis overlaps are not, actually, used (unlike in seismic record migration where that is the key point). They only offer possibility of choosing the best variant. Moreover, there is a special module TX-Curve Selector designed for selecting from a TX-curve set in Ray Catalog a sample of curves providing the optimal result. When selecting a curve sample, the user is enabled to limit offsets to reduce redundant overlaps. At that, the user keeps in mind that the reflector must be fully illuminated by the TX-curve set.

Head wave

The "classic" problem for refraction is considered. Given are: a *reciprocal couple* of TX-curves and velocity distribution in the covering medium. A reciprocal couple consists of the direct curve $T(R)$ from the source S and the reversed curve $T^*(R)$ from the source S^* ; R is a receiver. It is supposed that there exists a refractor segment illuminated by both curves which is to be determined. The idea of getting a solution originates from the famous traveltime equation for *reciprocal points* explained on (fig. 1).

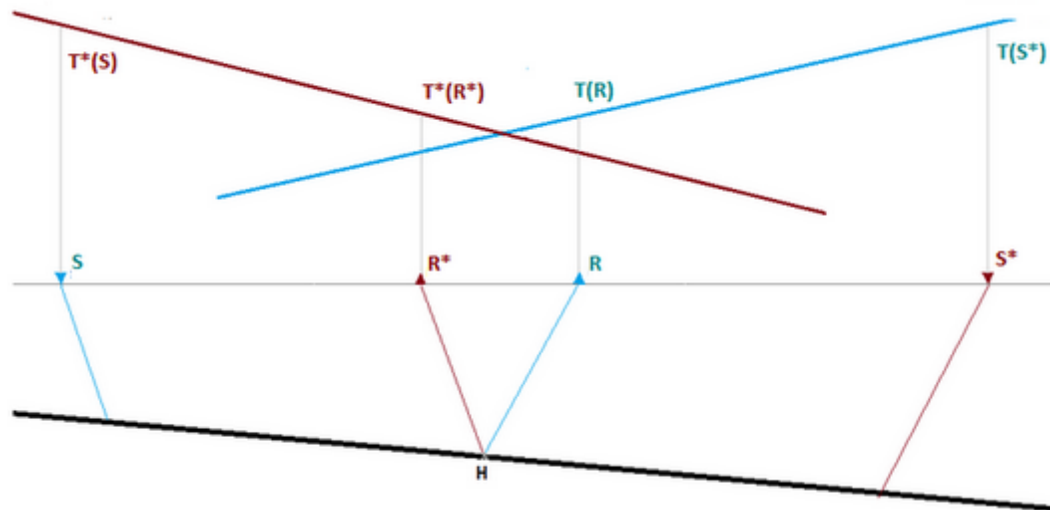


Fig. 1. Ray diagram explaining the notion of *reciprocal points*.

Supposing that head wave slides along the refractor, for any point H the following equality holds:

$$T_{HR^*} + T_{HR} = T(R) + T^*(R^*) - T(S^*),$$

where the left part is the sum of traveltimes along rays HR and HR^* coming out of a point H at critical angle. According to reciprocity principle, $T(S^*) = T^*(S)$; this value is called *reciprocal time*. Receivers R and R^* are called *reciprocal points*, while distance between them – *reciprocal points base*.

The ray diagram shows that the problem of building refractor can be restated as the problem of finding for each point of the direct curve its reciprocal point on the reversed. The algorithm used in XTomo-LM implements solving the latter problem. For crude estimation of reciprocal points base, the initial TX-curve points can be used: the true initial curve point is the reciprocal to the source point.

The head wave TX-curve set consists of subsets of catching up direct and reversed curves. With the help of TX-Curve Selector, the user builds the optimal TX-curve sample.

Determination of the refractor shape is, in principle, an approximate problem because it is never clear in what degree the XTomo-LM physical model (or any other such model) of head wave propagation fits the real conditions. For flat or almost flat boundaries different physical models

lead to similar results. That is just the horizons with small and slowly varying curvature that can be reconstructed by inversion of observed TX-curves.

Inversion algorithms

The XTomo-LM inversion algorithms work under condition that *velocity is constant or only has small horizontal gradient in the area where the would-be horizon is located*. Clearly, they are applicable for constant velocity in the covering layer. Similarly, they are applicable, if a boundary is horizontal and velocity is variable in the overlay. The algorithms seem inapplicable, if a horizon dips significantly or includes structural elements, that is, takes up a part of the model of significant thickness. However, curvilinear grid technique makes the problematic case quite similar to the case of horizontal boundary. The technique is demonstrated in the [Example](#) topic. Thus, inversion algorithms prove to be applicable to the general case of variable velocity in the covering layer which is regular enough and whose horizontal gradient is comparatively small.

Inversion, storing, viewing

Both modules, Reflector Builder and Refractor Builder, present the result as a pair of iterations of the horizon curve: the initial and the final (iterations 0 and 1). The distinctive feature of the initial iteration is its robustness. Its refinement, however, uses unstable operations which can produce fluctuations or loss of curve points. In this case, the initial iteration becomes more reliable solution, though it is an approximate solution. For flat horizons both iterations coincide. In general case, the initial iteration's points are shifted dip-ward, but general shape of the horizon is retrieved quite satisfactorily.

Building horizons is carried out on the Observations node of Processing Tree. All horizons built on the node are stored in a common storage which is referred to as *horizon database (HDB)*. It stores all builds of both iterations, so that the user can compare them. In the horizon database, the curve is identified with wave code, build number and iteration number (0 or 1). The contents of the HDB can be managed and viewed by the graphic module Horizon Previewer which is launched by each Horizon Builder.

4 Selecting TX-curves

Preparing TX-curve set for inversion is performed by the TX-Curve Selector module. It is launched from Processing Tree menu invoked on an o-node with the *TX-curves Inversion/Select TX-Curve...* command. The module maintains all functionality of [SRT TX-Curve Viewer](#). Additional tools are accessible under the following conditions: (1) the source filter is off: all sources are visible; (2) wave filter passes only one wave – the wave generated by the horizon to build (use the *Hide All but Selected* command in the wave menu). The user's task is to build a TX-curve sample optimal for building the horizon (in the users opinion). Description of the sample should be saved. The program puts it to a file whose name coincide with the wave code for later use by horizon builder.

Reflection

In TX-Curve Selector, a reflection TX-curve is not divided into direct and the reversed. A source is supposed to generate one TX-curve which can have left or/and right branch. Preparation of a TX-curve set includes sampling optimal subset of curves and setting limitation for offsets. The sampled TX-curves must provide some overlapping of the corresponding reflector segments. After inversion, Reflector Builder sews together the segments to get the reflector curve. At that, overlaps are processed in special way granting preference to the horizon points which correspond to receivers with less offsets.

Wave Filter. Using the wave list menu, filter off all waves but the one in question. In the plotter menu select *TX-Curve Sample for Inversion*. The sample dialog appears on screen (fig. 1). Its central part is taken up by the list called *sample description*. For each TX-curve the list columns display: source ordinal number in the list; source ID; source x-coordinate; maximal allowed offset. The list is sorted by source ID and provided with the context menu.

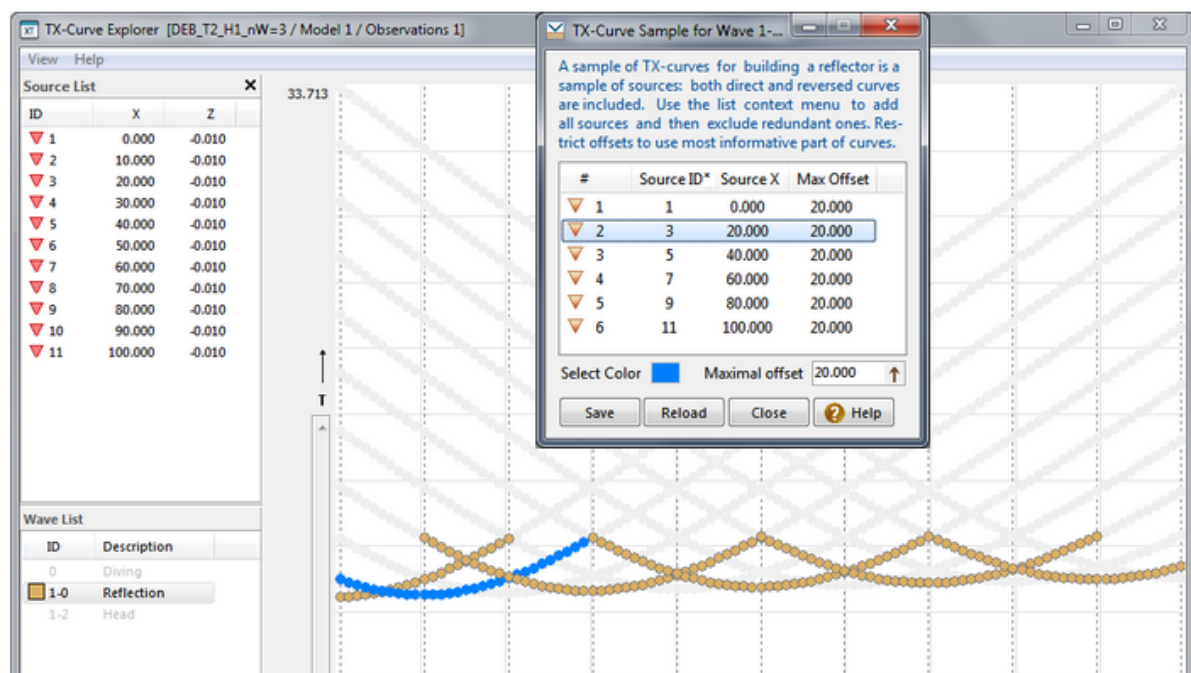


Fig.1. Description of the TX-curve sample for inversion.

Sample. When the dialog is invoked for the first time, the list is empty. Apply the *Add All* menu command to fill the list with all curves available. Then, using multiple selection and the *Exclude Selected* menu command, remove unnecessary items from the list.

Changing sample affects the plotter's picture: the curves from the sample are drawn with usual attributes, while the excluded are drawn in the background in light-gray. Thus, the user sees distinctly what he or she has chosen. The curve currently selected in the list is drawn in "Select color" which can be adjusted straight in the dialog. This mode of rendering is switched off after

the dialog is closed. However, it can be switched off/on by the list menu command *Sample Display Mode*.

The other way of adding a curve to the sample is to select it on the plotter and apply the *Add to Sample* command of the plotter menu.

Offsets. Offsets can be limited by a user-defined maximal value, so that points with greater offsets are removed from the sampled TX-curves. In the list, select items with the same offset limit and click on the *Maximal offset* field under the list. Enter the limiting value and click on the arrow button. The value will be assigned to the last column cell of all selected items. Zero means "any offset".

Saving. The *Save* button stores the sample description in a file inside the current o-node. Now, when invoked, the dialog will load the description corresponding to the wave chosen. To cancel changes made after calling the dialog, use the *Reload* button or exit the dialog without saving. Reflector Builder can be launched if only a sample description file for the reflection is discovered in the o-node folder.

Head wave

TX-curve sample

Head wave TX-curve sample consists of subsets of direct and reversed catching-up curves. A subset of direct curves must meet the following conditions:

- 1) TX-curves are sorted by source x-coordinate;
- 2) every TX-curve, but the first, has overlap with the previous curve containing no less than m points in it;
- 3) x-coordinates of the last curve points increase monotonously.

For reversed curves, condition 1 is valid, while the other two are replaced with:

- 2) every TX-curve, but the last, has overlap with the next curve containing no less than m points in it;
- 3) x-coordinates of the first curve points increase monotonously.

Value of m is set to 6.

Creating

The dialog *TX-Curve Sample* is shown on fig. 2. It contains descriptions of the subsets as two lists, which are empty at first invocation for a given head wave.

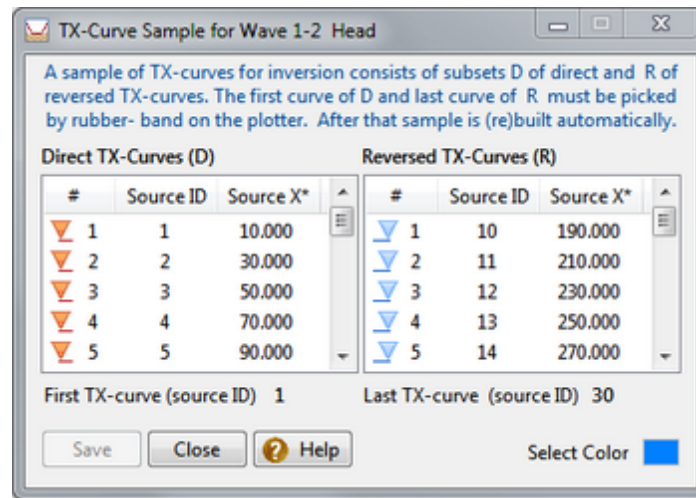


Fig.2. Head wave TX-curve sample for inversion. The asterisk in column title points to the sort key.

Samples of direct and reversed curves are built automatically and independently. The user can only exclude some of curves at the flanks of the line. To build the sample of direct curves, invoke the dialog, then select a direct curve on the plotter. In the plotter menu choose the *Set as First Direct curve* command. The sample list will be built, and it will start with the selected curve. The list of reversed TX-curves is built similarly, after the user selects on the plotter a reversed curve to be the last one in the sample. The described procedure can be used to rebuild the sample when necessary. After the sample is built, click on the *Save* button to store it on disk as a head wave TX-curve description file.

5 Building Reflectors

Reflector Builder works with the curve sample description created by TX-Curve Selector. Use the *TX-Curve Inversion/Build Reflector* command of Processing Tree menu to start the module. Fig. 1 shows the central part of the module's main window.

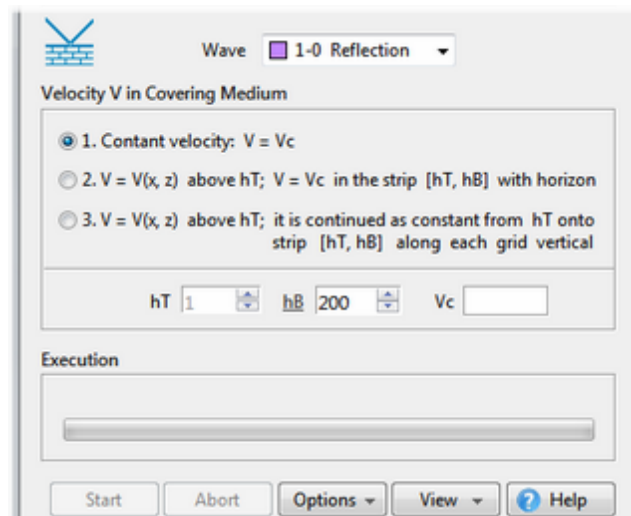


Fig. 1. Reflector Builder main window

Velocity V' in overlay

Velocity definition is the main user's care. There are three options of velocity definition, and they are stipulated by the requirement for velocity to be vertically constant in the vicinity of a horizon (see [algorithm](#)). The question how to apply them is treated in [Examples](#). Here a formal description is presented. Below M is grid node; $V(M)$ is model velocity; $V'(M)$ is velocity to be used for inversion; $[h_1, h_2]$ denotes grid horizontal strip bounded by h -lines # h_1 and # h_2 in which the horizon, supposedly, lies; hB_{\max} is bottom line number minus 1. Velocity V' in covering medium can be defined in the following ways:

1. Constant velocity: $V'(M) = V_c$ on $[1, hB]$. By default, $hB = hB_{\max}$. A click on the hB link sets the default value. Setting hB less than hB_{\max} reduces computation time. Average velocity in $[1, hB]$ is a good candidate for V_c . Values of hB , hT and V_c are entered by the user into the edit fields.
2. $V'(M) = V(M)$ on $[1, hT]$; $V'(M) = V_c$ on $[hT, hB]$. Values of hT and V_c are entered by the user.
3. $V'(M) = V(M)$ on $[1, hT]$; $V'(M) = V(M^*)$ in $[hT, hB]$, where M^* lies on line # hT and on the same vertical as M . In other words, V' is continuous continuation of V from line # hT onto strip $[hT, hB]$ as constant on each vertical. The continuation is carried out by the program. Values of hB and hT are defined by the user, V_c is not used. V' is not constant on $[hT, hB]$ if only $V(x, z)$ has non-zero, though small, horizontal gradient. .

In the above definitions, the strip $[hT, hB]$, in which the horizon surely lies, is to be regarded as a priori information. It can be obtained, for example, by the Reflector Evaluator explained in the end of the topic.

Creating task

Velocity definition is the main but not the only thing to do before starting inversion session. Here is the full list:

1. *Select wave* from the drop-down list at the window top. It contains those reflections for which the TX-curve sample descriptions are discovered in the o-node folder.
2. *Define velocity V'* in the covering layer. Enter hT , hB and V_c fields, when necessary.
3. *Change sample* for the current session if necessary. To do that, click on the *Options* button and in the dropped down menu select *TX-Curve Set*. The command invokes then same dialog *TX-Curve Sample* as in [TX-Curve Selector](#). Edit the list the in the same way.
4. *Adjust execution parameters*. In the *Options* menu select *Parameters*. The command displays the dialog shown on fig. 2. On the *Algorithm* panel the user defines parameters affecting inversion algorithm. The first one stores a priori information about horizon slope (both local and global). For the first inversion always use the default value 20° , the more so that it usually

suits. A change is justified if only the result obtained witnesses that the horizon shape is distorted by the slope restriction. The choice of greater value may help but also increase possibility of fluctuations. The averaging radius for computing derivative, as a rule, can be left as is. The parameter on the *Execution* panel is explained below.

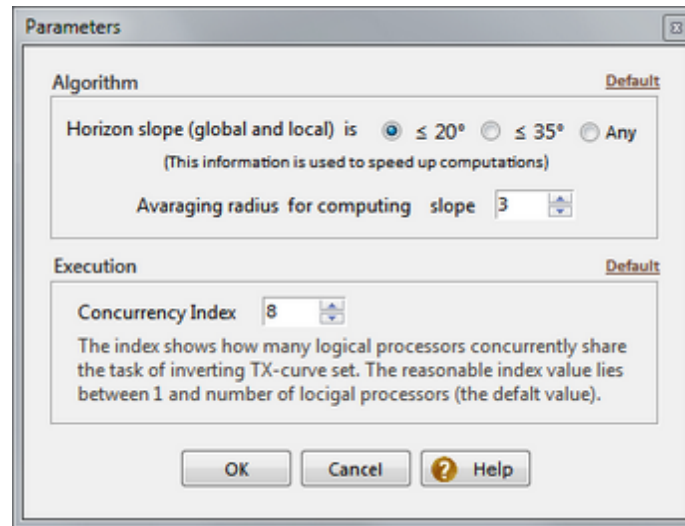


Fig. 2. Dialog for setting execution parameters.

Execution

The inversion session is started by click on the *Start* button. It builds a version or *build* of the reflector for the given task. Builds are numbered automatically. The build number is displayed on the *Execution panel* after clicking the *Start* button.

Reflector Builder treats inversion of one TX-Curve as an elementary job and runs for that the special module: *Inverter*. Its product is a segment of the reflector. Inverter is attached to one logical processor of the system. Reflector Builder runs simultaneously N inverters, where N is the value of the *Concurrency index* parameter (fig. 2). By default, N equals the number of available logical processors. When the session is under way, source IDs of elementary jobs currently being processed are displayed above the progress bar. Segments obtained by Inverters are sewed together by Reflector Builder to get the entire horizon.

Viewing results and export

The results include build log and, if session was a success, two iterations of the horizon curve stored in the horizon database. The *View* button drops down menu that provides access to the results. All logs are stored in the o-node folder, so the *View/Logs* commands displays the list of all logs. Log name begins with the creation time and ends with wave code and build number. The list is sorted by creation time. It owns context menu with the *View* and *Delete Selected* commands. A log can be opened by double-clicking its name as well.

To preview the horizon, the *View/Horizon* command launches the Horizon Previewer module (HPV). This is a graphic module displaying model image and capable of drawing over it any curve from the horizon database. Fig. 2 shows a fragment of the HPV main window.

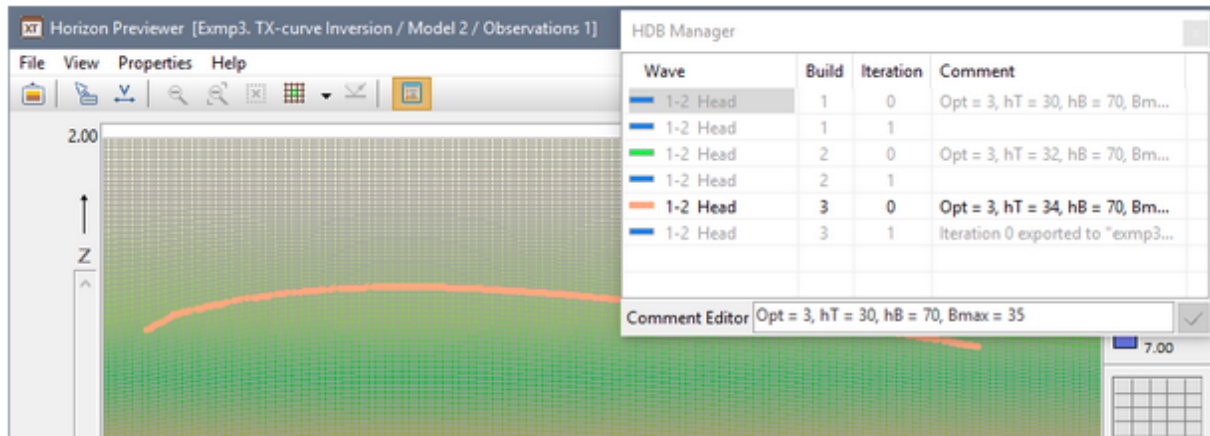


Fig. 3. HPV main window with horizon database manager.

At the top, the plotter is overlapped by the window of Horizon Database Manager (HDB Manager). If the window is closed, it can be displayed again by the command *View/Horizon Database Manager* or the matching tool bar button. Manager displays HDB content as a list of horizons – all ones built in the node, both reflectors and refractors. Each item shows wave description, build and iteration numbers and comment. The selected item comment can be input or edited in the field below the list. Length limit is 40 characters. Editing must be finished by click on the tick button. On fig. 3, comment is used for task parameters. An item is displayed in black if only the corresponding curve is visible on the plotter. Otherwise, it is printed in gray. The list permits multiple selection to operate on a groups of items. Operation menu commands are presented in Table 1.

Table 1. HDB Manager menu commands

Command	Description
<i>Draw</i>	Forces the plotter to draw all selected horizons. Double-clicking an item shows or hides the item curve.
<i>Hide</i>	Plotter hides all curves selected in the list.
<i>Boundary Velocity</i>	The command is available if only the selected horizon is a refractor. It invokes a dialog with the plotter displaying boundary velocity graph and containing a popup menu with the commands for smoothing velocity and export to an ASCII file.
<i>Change Color</i>	Sets the color picked from the Windows Color dialog to all curves selected in the list as the drawing color.
<i>Heavy Line</i>	Makes the plotter to draw all selected curves with heavy lines.
<i>Ordinary line</i>	Makes the plotter to draw all selected curves with ordinary lines.
<i>Display Build Log</i>	Displays build log of the selected horizon.

<i>Refresh List</i>	Updates the list after reading HDB anew.
<i>Export to MG File</i>	Exports the selected curve to MG file.
<i>Delete Selected</i>	Removes the selected curves from HDB.

While Reflector Builder is active, HPV may stay active too. To view a newly built horizon, apply the *Refresh List* command of HDB Manager. HPV can be launched straight from Processing Tree menu invoked on an o-node with the *Preview Built Horizons* command.

Crude estimation of reflector curve

When Reflector Builder is used for variable velocity in overlay (see Examples further in this chapter), it might be helpful to get a simple crude estimation of the reflector curve. The Reflector Evaluator module provides such estimation at each source point under assumptions that velocity in overlay is known, receiver R_0 nearest to a source has zero offset and the ray coming up to R_0 is vertical. The module is launched by the *TX-Curve Inversion/Evaluate Reflector* command on an o-node. Its main window is shown on fig. 4 below.

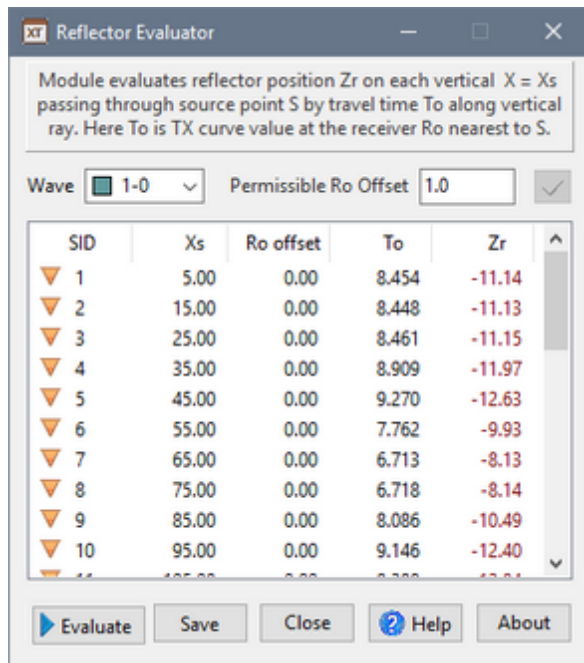


Fig. 4. Reflector Evaluator. Main window.

First, select the required wave in the drop-down list *Wave*. Then enter maximal permissible offset of R_0 and click on the tick button. Now the module is sampling all TX-curves for which R_0 offset does not exceed the defined the bound. The operation results in TX-curve list with first four columns filled: source ID and its x-coordinate, R_0 offset and TX-curve value T_0 at R_0 . Now click on the *Evaluate* button to fill the Z_r column, representing reflector z-coordinates at source points. The result can be saved to a text file of MG format (the *Save* button).

6 Building Refractors

Refractor Builder is started by the *Build Refractor* command of the Processing Tree on an o-node. Though head wave TX-curve inversion differs essentially, the user interfaces of Refractor Builder and Reflector Builder as well as user actions are very similar. In particular, velocity definition in covering medium does not differ at all. One exception relates to using average velocity as V_c : the

advise is valid only for reflections. There is the additional panel **Reciprocal Points Base** (RPB) in Refractor Builder's main window (fig. 1).

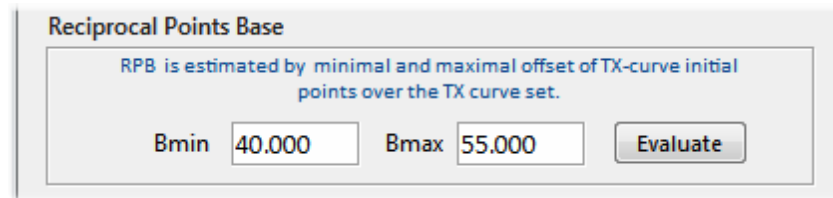


Fig. 1. Panel for estimate of reciprocal points base.

According to **theory**, the heart of the inversion algorithm is search for the reciprocal point of the reversed TX-curve for each point of the direct. For that, it is important to know a priory reciprocal point base estimation. The information can be obtained by scanning TX-curve initial point offsets. Though the visible initial point on seismic record is usually found at greater offset, this is the only simple way to get the estimation. The *Evaluate* button does such estimation over all participating curves. If the user can define them more precisely, he or she may simply type in the necessary values. In some cases, bad estimations for the given velocity are discovered in the course of inversion, and the user gets recommendation for correction.

Reciprocal point search requires scan of great amount of combinations. To make the search more efficient, Refractor Builder computes **eikonals** $T_R(x, z)$ for all participating receivers R beforehand and distributes this time-consuming job between the available logical processors. Computing eikonals takes most part of the computation time, which is divided between logical processors of the system. It is worth noticing that to save system resources, each eikonal T_R is computed only at nodes of a vertical band B_R around receiver R . Width of the band is defined by value of B_{max} .

Refractor Builder, additionally, computes boundary velocity and stores it in the horizon database.

The parameter dialog of Refractor Builder allows adjusting two parameters only: averaging radius for computing derivative and concurrency index, which is used on the stage of computing eikonals. Both parameters usually keep their defaults values.

Termination of the program with code 616 means that the band B_R is too narrow. The user may try to increase B_{max} and start building again. Repeated errors 616 rather witness of wrong choice of the horizon strip $[hT, hB]$ or wrong velocity distribution in the overlay.

After inversion is terminated, the user acts exactly in the same way as in the case of reflection. Horizon Database Manager has the special **command** for viewing and exporting boundary velocity.

7 Examples

This section is a commentary on the sample projects supplied with the product. The aim of the sample projects is to demonstrate how to work with XTomo-LM in general, and in solving some problems of layered model interpretation, in particular.

Installation of sample projects

If in the setup program, the user selects option "XTomo-LM 3.2 + Sample projects", the program creates folder [S:\XTLM 3.2 Examples](#) on the system disc S: and puts in it the archive RAR file with the same name as the folder. The file contains three common folders: working ([Works](#)), archive ([Archive](#)) and import-export ([Impex](#)). The sample projects lie in the working folder. Unpack the file into the same folder. Then run Project Manager and define three new common folders: Works, Impex and Archive. After clicking on OK, the sample projects appears in the project list. They are ready for work if only S: = C:, i.e. if C: is the system disc. If not, paths to projects' import-export folders is wrong (it contains C:). For each of six project, enter properties window and select import0export folder anew. After that, the projects can be smoothly used. The setup program, additionally, adds the new data store [XTLM 3.2 Examples](#) with databases to SRT Port. They are used in the sample projects. The archive file can be unpacked into any other location with similar adjustments in Project Manager.

Problems

The sample projects are created for solving three problems of TX-curve inversion, for which observation data are generated by modeling. Names of modeling projects are [ExmpN. Modeling TX-curves](#), names of inversion projects are [ExmpN. TX-curve Inversion](#), where N is problem number. The problems (of teaching type) are stated as follows:

Problem 1. Build interfaces of 3-layered model with known constant layer velocities by inversion of two reflection TX-curve sets.

Problem 2. Given are reflection TX-curve set from the floor of a layer with velocity $V(x, z)$. Build the reflector curve.

Problem 3. Given are head wave TX-curve set from the floor of a layer with velocity $V(x, z)$. Build the refractor curve and evaluate boundary velocity.

The projects contains all steps of getting solutions to the above problems. The user can repeat all actions himself or herself by creating "parallel" nodes and performing necessary operations described in this topic and in comment fields of Processing Tree nodes.

Problem 1

Observations. The model for generating observation can be found in node M1 of project [Exmp1. Modeling TX-curves](#). Invoke Model Viewer to see the model. It was borrowed from a literary source. See comment on M1 for layer velocity values. Two horizons with IDs are 2 and 3 (see Horizon list) match two reflections 2-0 and 3-0 defined in the project wave list. Observation system in node M1/O1 is created in graphic module [Spread Editor](#). For further use in other examples, the spread created was copied to the [XTLM 3.2 Examples](#) data store of SRT Port as

database [exmp1_spread](#). Forward Problem Solver was launched on M1/O1. Diving wave was excluded from its task, so that rays were traced only for reflections. Rays and TX-curves can be seen in Forward Problem Viewer invoked on node M1/O1/F1. To use the TX-curves computed as "observed", Ray Catalog was copied to SRT Port by the *Copy Ray Catalog to SRT Port* command of Processing Tree menu. In copying module, the target data store was selected as [XTLM 3.2 Examples](#), database name – as [exmp1_tx-curves](#), database type – SRT, profiling data flag was set. The result of copying can be viewed by SRT Port Manager.

Inversion project is [Exmp1. TX-curve Inversion](#). The starting model in M1 uses orthogonal grid in which number of rows is twice as many as in the modeling project. Velocity values are assigned to three depth interval, but it is not necessary: we could start with constant velocity or any other. Empty node M1/O1 was created, the command *Extract Data from SRT Port* applied to extract data from [exmp1_tx-curves](#). In SRT Data Extractor, no sampling was done, the *Copy All* command used. Then, in TX-curve Selector, we set wave filter to pass only wave 2-0 and included in the sample all curves, but only observations with receiver offsets not exceeding 20. The next step is starting Reflector Builder. We selected wave 2-0 and option 1 for velocity in overlay: constant velocity, $V_c = 3$. To view the inversion result, launch Horizon Previewer by the *View/Horizon* command. Pay attention to difference between iterations 0 and 1. We exported iteration 1 to [exmp1_h1.mg](#) for implanting into the model. That was done in node M2 – a copy of M1. In Model Editor, h-line import was performed through H-Line Editor. Imported curve proved to be the h-line # 67, and it was immediately converted to horizon corresponding to wave 2-0. The first layer was selected and its velocity was assigned value 3. That ended building of the first interface.

Node M2/O1 was created as a copy of M1/O1. In TX-curve Selector, wave filter was set to pass only wave 3-0, the TX-curve sample was composed similarly, but maximal offset was set to 30. In Reflector Builder, wave 3-0 was selected and velocity option 2 chosen with $hT = 67$, and $V_c = 4.5$.

Thus, above h-line #67, velocity for inversion V' coincided with the model velocity (i.e. equalled 3) and below it $V' = 4.5$. After inversion, in Horizon Previewer iteration 1 was exported to file to [exmp1_h2.mg](#), which then was imported to node M3 (a copy of M2). After that, velocity in layers 2 and 3 was set to 4.5 and 6, which ended getting solution to problem 1.

To compare the solution with initial model, one can use the Velocity Comparator utility which, in our case, images only differences in horizon geometry. Select node M3 and run the utility from Tools menu. In its main window, select node M1 from project [Exmp1. Modeling TX-curves](#). Select appropriate grid frequency, say, $M = 200$, $N = 300$, then start comparison and examine the difference map.

Problem 2

The aim of this example is to show, how to deal with variable velocity in overlay. Here we comment only on the actions specific for this case. First goes observations modeling. In node M1 of the project [Exmp2. Modeling TX-curves](#), velocity distribution for overlay is defined as function with constant vertical gradient. In node M2 (copied from M1), two-layered model is formed by implanting the curve of horizon 1 from Problem 1, which had been stored in file [exmp1_h1.mg](#). The curve was imported, then converted into a horizon. We know that changing grid geometry

modifies velocity distribution (details are [here](#)), so we replaced it with the initial one from node M1 (Model Editor's command *Edit/Replace velocity with/ Velocity from M-node*). Then, to provide velocity jump, we increased it by 5% in the row underlying the reflector and then interpolated it between that and the bottom row of the grid.

The empty M1/O1 node was created and observation system extracted from the [exmp1_spread](#) database of SRT Port. Forward problem was solved for wave 1-0 (diving wave excluded), and Ray Catalog copied to the SRT Port database [exmp2_tx-curves](#) with the same settings as in Problem 1.

Inversion. When creating I-project [Exmp2. TX-curve Inversion](#), the starting model was copied from node M1 of the M-project. Thus, M1 stored velocity in the future overlay. Observations in M1/O1 were extracted from SRT Port database [exmp2_tx-curves](#), created in M-project. Data sample for inversion included observations of all curves with receiver offsets not exceeding 20. We used module Refractor Evaluator to estimate a horizontal grid strip containing the reflector. According to evaluation, interval [-24, -8] contained z-coordinates of all reflector points with some reserve. It matched interval [20, 70] of h-line numbers. In Reflector Builder, we chose option 3 for velocity with $hT = 20$, $hB = 70$. Thus, velocity in overlay grew, as in the model, from top down to h-line 20 and below it became constant to satisfy [usability condition](#) of the of inversion algorithm. Horizon Previewer showed that the built horizon was slightly flattened and shifted upwards. The result was expected due to fall of average overlay velocity. Nevertheless, we intended to implant the curve into the model. To do that, iteration 1 was exported to [exmp2_1.mg](#).

In Model Editor invoked on the M2 node (a copy of M1), the curve was imported through H-line Editor. At that, to exclude some ripples, we smoothed the curve by moving average with the parameter value 2. The imported curve became h-line #28 (select row #28 to view it). After that, velocity was restored by replacing it with velocity from M1. The imported curve changed grid geometry, and that was exactly the aim of the first step. Now h-lines became almost parallel to the future horizon, and, therefore, we would be able to narrow a band of freezing velocity drastically. That's why the first step of inversion can be called "adjustment of grid".

The second step was simply repetition of the first one with the new "native" grid. The M2/O1 node was created as a copy of M1/O1. In Reflector Builder we selected option 3 with $hT = 27$ and exported horizon iteration #1 to [exmp2_2.mg](#). Then node M3 was created by copying M2; the curve in that file imported as h-line #30 and converted to a horizon; velocity was restored from M1. That ended building of the reflector. Velocity in overlay was changed in quite narrow strip.

In order to estimate difference between the built boundary and its prototype we used Velocity Comparator with some preliminary actions aimed at eliminating velocity divergence. More precisely, we created node M4 as a copy of M3 and in M4 we made velocity constant in both layers (3 и 4.5, figures are not essential). Then we created similar node M3 in M-project. Now, that velocity had been made equal, we launched Velocity Comparator with the same grid as in problem 1 and got the required estimation.

Problem 3

Approach to problems 2 and 3 is the same. We comment only on the points specific for head waves. First, the physical model of head waves used in XTomo-LM makes sense only for seismic

boundaries of small curvature. This fact motivated choice of horizon shape in node M2 of the project [Exmp3. Modeling TX-curves](#) (node M1 contains velocity model in overlay, as in problem 2). The curve was obtained by editing of an h-line of the initial orthogonal grid.

Modeling of head wave observations is somewhat more complex, because the set of traced rays of head wave contains, additionally, rays of subcritical reflections (see ray picture in Forward Problem Viewer (FPV) on the node M2/O1/F1). Traveltimes along such rays must not participate in head wave TX-curves, therefore we need a subtler means than copying the entire Ray Catalog to SRT Port. Examination of the ray picture shows that subcritical reflections are cut off at offsets 35 and more. In FPV, we built ray sample containing only rays coming to receivers with offsets no less than 35 and exported TX-curves of the sample to SRT file [exmp3_tx-curves](#) (*Export/ASCII Export/Observation data*, option *Last Ray Sample*). Then we started SRT Port Manager and imported the file to data store [XTLM 3.2 Examples](#). The obtained database with the same name contains the required head wave observations.

Inversion. Project [Exmp3. TX-curve Inversion](#) was created with the starting model from node M1 of the above M-project. Observation were extracted from SRT database [exmp3_tx-curves](#) to node M1/O2. In TX-curve Selector, the sets of direct (starting from the leftmost) and reversed (ending with the rightmost) TX-curves were composed and saved. A crude estimation of the refractor depth can be made using initial TX-curve points to which critical reflections come. Basing on the estimation, we came to conclusion that the refractor lied below h-line #28. In Refractor Builder we chose option 3 for velocity in overlay with $c_{HT} = 28$, $h_B = 70$. In case of head wave we work with 0-th iteration only as was recommended. In the image displayed by Horizon Previewer, pay attention that the ends of the built horizon are located rather far away from model bounds. The value of refractor "margins" is approximately equal to x-shift of the leftmost and rightmost critical rays. While the curve was being imported into the model at node M2, it was continued to model bounds with horizontal segments (that is the default action in XTomo-LM). On order to avoid undesired effects, we replaced the horizontal curve flanks with the tangent ones by manual editing. After that, the curve was implanted into the grid as h-line #36, and then velocity was restored from M1.

During the second step (node M2/O1) we built three approximation to the refractor setting successively $h_T = 30, 32, 34$ (to be more instructive). So we carefully approached h-line #36 from above with horizon moving slightly down at each attempt. We took the last variant for the final and exported iteration 0 to MG file. In node M3 the curve was imported and implanted into the grid as h-line #38 after editing its flanks as at the first step. The final action was restoring velocity. Node M4 was created to compare the result with the prototype horizon in M-project, exactly in the same way as it was done in problem 2.

Refractor build contains estimation of boundary velocity $V_b = V_b(x)$. Select node M2/O1, run Refractor Builder and then Horizon Previewer. In horizon list, select iteration 0 of the last build, invoke context menu and choose the *Boundary Velocity* command. It displays a dialog with V_b plot. Usually it is oscillating. Apply the *Smooth* command of the plot popup menu twice. The V_b graph still has a sharp spike at the left flank. Note that the left end of the horizon curve also looks

too bent down. Both irregularities are consequences of an "edge effect". Reciprocal point search goes right to left and is hampered by the end of leftmost TX-curve. Starting from $X = 23$, V_b stays within the range $[4.53, 5.00]$. Those figures are taken from file [exmp3_velbnd.bln](#), to which V_b has been exported. Leaving the plot on screen, select [Exmp3. Modeling TX-curves](#) in the project list, invoke context menu, choose the *View Model* command and then – Model 2. The initial model is displayed in Model Viewer. Select the row underlying the horizon curve and invoke horizontal velocity profile from the *View* menu. Compare it with the V_b plot on the common domain without the support of the spike. They must be close to each other.

Utilities

1 Comparing velocities

The Velocity Comparator utility compares velocity distributions stored in two m-nodes of arbitrary projects. Comparing velocities in different m-nodes of a project allows tracking changes in the course of interpretation. When more than one project is created for processing of the same data (alternative approaches or M- and I- projects), comparing velocity functions from different projects becomes insistent. The utility let the user evaluate difference of velocity distributions both visually and in numeric form.

Posing the Problem

Speaking of comparing, one has to take into account that velocities in different m-nodes are defined, generally, on different grids. The grids in question may be defined for different domains, have different h- and v-lines, top h-lines in particular. That's why the compare problem needs to be accurately stated.

Let velocity functions $V_1(x, z)$ and $V_2(x, z)$ are defined at grid nodes with model domains D_1 and D_2 . Each domain is a rectangle, possibly, with curved top. Comparison is permitted provided that domain $D = D_1 \cap D_2$ is not empty and, moreover, is a significant subset of both D_1 and D_2 .

If the condition is fulfilled, $V_1(x, z)$ and $V_2(x, z)$ are compared at nodes of an orthogonal grid G defined on the rectangle inscribed in D . Frequency of grid lines is adjusted by the user. For each cell $c \in G$, the utility computes absolute ΔV and relative δV velocity differences:

$$\Delta V(c) = V_1(c) - V_2(c), \quad \delta V(c) = 100\% \cdot \Delta V(c) / \max(V_1(c), V_2(c)),$$

and let the user view ΔV and δV maps as well as vertical profiles of $V_1(*, z)$ and $V_2(*, z)$ at any choice of the first argument.

Choosing velocities. Calculation

Velocity Comparator is launched from Project Manager's *Tools* menu. Utility's main window is shown on fig. 1.

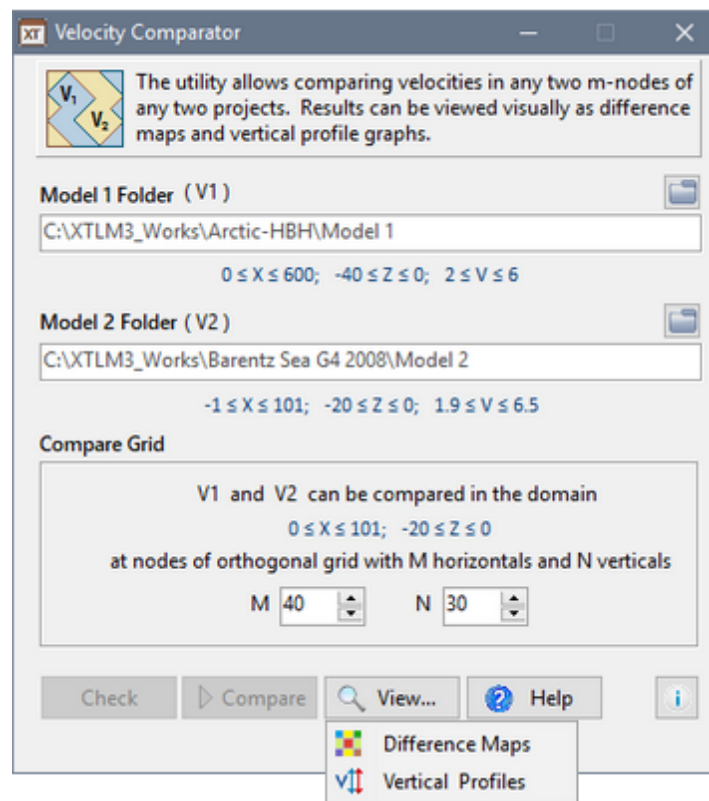


Fig. 1. Utility's main window.

Snapshot is made after m-nodes with velocity functions V_1 and V_2 passed the comparison check.

The first user's action is to select m-nodes. The *Browse* buttons invokes local folder browser in which the cursor is set on the current working folder. Enter the folder, then a project folder and set the list cursor on the desired m-node. Click on the browser's *Select* button to finish the choice. Path to the node appears in the Model Folder field, while under the field, model rectangle and velocity range are displayed. Do the same to define the second m-node. If, before launch, an m-node was selected in Processing Tree, it is, by default, the first compare operand. Now click on the *Check* button to verify comparison condition. After that, additional information is displayed on the *Compare Grid* panel. It contains the domain of comparison and the default h- and v-line frequencies. Adjust them as needed, and click on the *Compare* button. After the calculations are done, use the *View* button to study the results. The buttons drops down the menu with two options: maps and profiles.

Difference maps

The maps are displayed by a separate graphic module representing ΔV and δV values by color shades. The new tool button (fig. 2a) allows tuning maps and switching between them: the arrow, when clicked on, drops down the switch menu. The module has Zoom and Selector, supports text and graphic export. Map tuning is carried out in the dialog shown on fig. 2b.

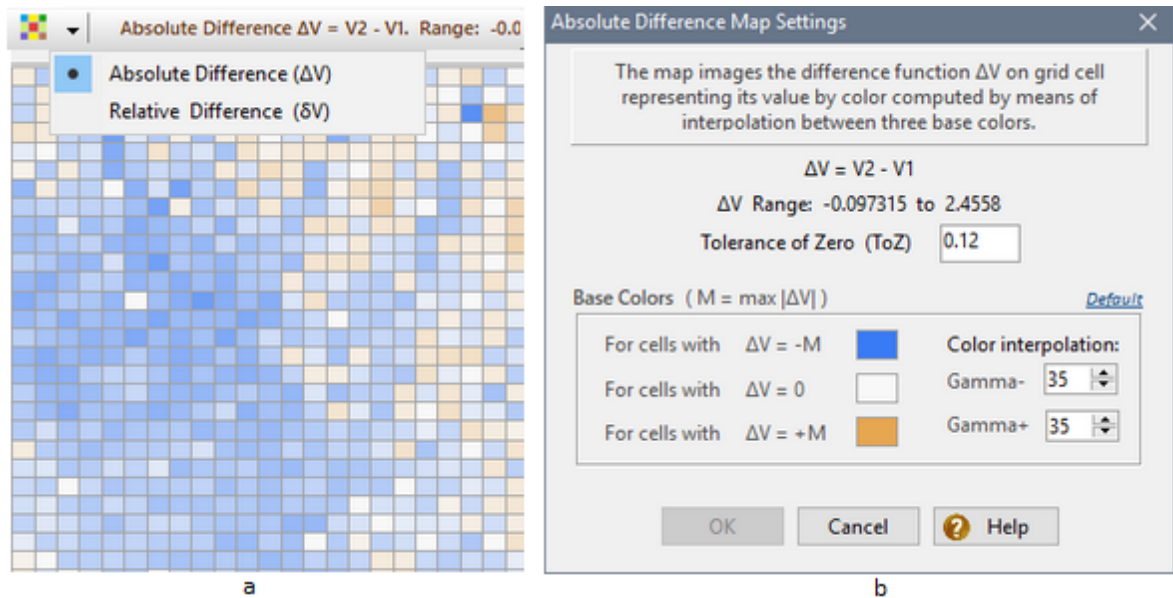
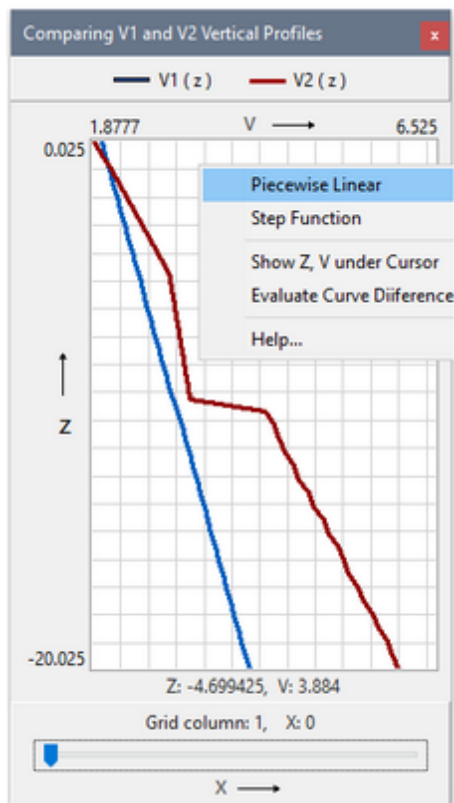


Fig. 2. Difference map. a) map view; map settings button is at the snapshot's top-left; b) map settings dialog.

One can adjust Tolerance of Zero (ToZ) and color parameters. ToZ is a threshold for absolute value of ΔV or δV to be displayed as zero. The color parameters include three base colors. If $M = \max |\Delta V(c)|$ over all grid cells, then the first color C_{-1} is used for cells in which $\Delta V = -M$, the second color C_0 is for cells with $\Delta V = 0$ counting ToZ, the third C_1 is for cells with $\Delta V = M$. Color in an arbitrary cell is calculated by interpolation between C_{-1} and C_0 or between C_0 and C_1 depending on ΔV sign. Interpolation can be controlled by parameters Gamma+ and Gamma-. Changing Gamma moves color towards one of the base colors. If a subset is selected on the grid, the *ΔV in Selection* command of the plotter menu displays ΔV range and mean value in the selection. Each map can be tuned in its own way, and settings are stored on disc.



Vertical velocity profiles

The *Vertical Profiles* command of the *View* menu displays the window shown on the left snapshot. It contains a plotter with graphs of functions

$$v_1(Z) = V_1(X, Z), \quad v_2(Z) = V_2(X, Z)$$

for a fixed value of the X-coordinate. Changing X is modeled by moving a slider along the track bar on the bottom window panel. Coordinates of a point of the (Z, V) plane corresponding to the cursor position are displayed under the plot.

Applying a command of the plotter menu, the user can do the following:

- switch the way of representing vertical profile: as piecewise linear or step function;
- display Z- and V coordinates in the hint window immediately under the cursor;
- evaluate curve difference using numerical characteristics computed for each X.

2 Model Geometry

[Grid](#) – [Creating a wireframe](#) – [Inserting a curve](#) – [Current curve](#). [Permissible area](#) – [Pointwise editing](#) – [Curve table](#) – [Curve transformations](#) – [Segments](#) – [Settings](#) – [Changing the wireframe](#) – [MG files](#) – [H-line editor mode](#)

The MG File Editor (UMG) utility is a graphic tool for editing model geometry. Utility's name is motivated by the fact that model [wireframe](#) is stored in an ASCII file of the MG format.

Grid

Fig. 1 shows a wireframe consisting of three curves drawn by the UMG plotter. Curves are defined on an uniform x-net. The curves are simple, that is, each of them is a graph of a function $z = f(x)$. [MG](#) files can store curves, defined on different x-nets, and not necessary uniform. UMG can read such files, but redefines all curves on a common x-net. From the practical point of view, that is the optimal way of representation for editing.

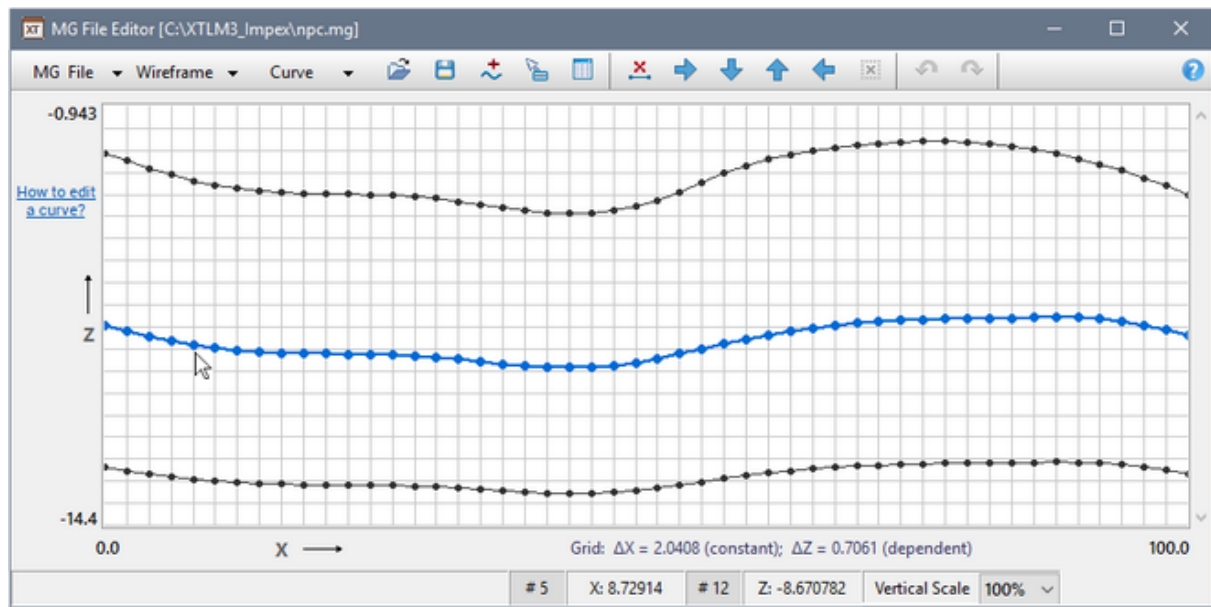


Fig. 1. UMG main window.

On fig. 1, the plotter surface displays the 2D grid with verticals passing through the nodes of x-net. Grid horizontals are virtual in the sense that grid cells are always square – just for convenience. While the x-net step is a constant for the wireframe, real distance between verticals (i.e. expressed in length units) depends on the x-net step, domain real height and window height. At any moment, real values of grid steps ΔX and ΔZ can be seen under the plotter.

Creating a wireframe

Utility's control tools are collected, mainly, on the tool bar. The first tree buttons drop down menus. To create a new wireframe, click on the *Wireframe* button and select *New Wireframe*. The command displays the *Definition of Grid* dialog (fig. 2).



Fig. 2. Grid definition dialog.

The user has to define the number of x-net nodes N and type in bounds of the rectangular *wireframe domain* which contains all the curves. N is chosen within the range 25 to 160; N is a wireframe constant. The domain can be extended later but only vertically. After a click on *OK*, the plotter displays the empty wireframe showing the grid with N verticals.

Curve points can sit only on the grid verticals. When the cursor is moving on the plotter, the status bar displays the real coordinates X and Z of the cursor point. To the left of X , the nearest vertical number is displayed, to the left of Z – nearest horizontal number. If the button is downed,

vertical number and coordinates are shown in the cursor's hint window. The image can be magnified vertically by selecting the scale coefficient from the drop-down list *Vertical Scale* on the status bar.

Inserting a curve

A curve can be inserted in any place of the wireframe domain. The button  displays a message box with information for the user: he or she has to double-click the point through which a curve to insert will pass. The point defines new curve's position relatively the other curves or upper/lower wireframe borders. After the message is closed with the *OK* button, the program stays in the insert mode (the button still downed) waiting for a double-click. After the user double-clicks the spot needed, the plotter images the *default curve*. To cancel the operation before double-click, use the *Cancel* button in the message box or click on the the downed button . Now the default curve must be edited.

Current curve. Permissible area


The curve added becomes the *current* or *selected*. It differs from the other curves in *drawing attributes* (fig. 1). The current curve is in focus of editing in the sense that it is the object of all editing commands. If there are several curves in the wireframe, the user can change focus by *selecting* another curve. Capture the target curve with the rubber-band and choose the *Select Curve to Edit* command in the rb-menu. The target curve becomes the current, but if only the previous current curve was defined at all x-net nodes (see below).

The wireframe can contain only nonintersecting curves. Moreover, they cannot come closer to each other than a certain distance h called *intersection tolerance*. This requirement defines for each curve its *permissible area*, within which it can be changed. The curve permissible area are defined by the neighboring curves or wireframe top or bottom border. Drop the upper neighbor by h , then lift the lower neighbor by h – the resulting strip is the current curve permissible area. The program itself adjusts the appropriate value of h and tracks when the curve comes out of its permissible area.

Pointwise editing

Here it is shown how to create a curve quickly using the following built-in features:

double-click on a point M of the plotter surface makes its projection M' on the nearest vertical V a point of the current curve on V ; double-click on a plotter point with Shift pressed removes from the nearest vertical the point of the current curve.

1. Insert a new curve in the proper wireframe area (fig. 3a).
2. Click on the button  to clear the default curve of all its points but the first and the last ones (fig. 3b).
3. Use an appropriate sequence of double-clicks to create the draft curve (fig. 3c).
4. Define the draft curve at all x-net nodes, applying linear or spline interpolation (fig.3d). The operations are carried out by the *Interpolate/Linear*, *Interpolate/Spline* commands of the drop-down menu *Curve*.

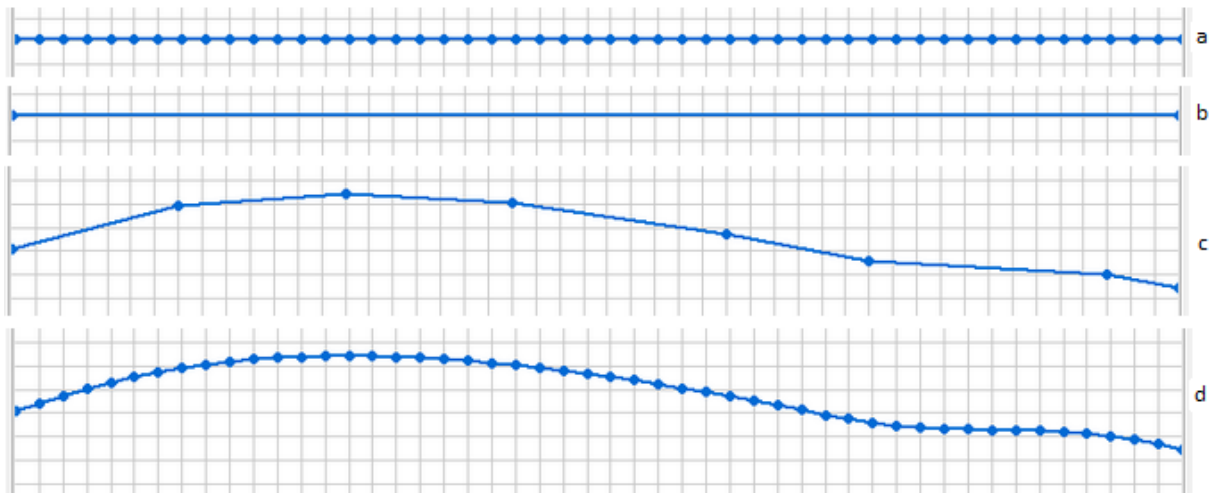




Fig. 3. Creating a curve. a) the default curve inserted in the wireframe; b) all point but the end ones are removed;
c) draft curve; d) final version after spline interpolation.

Several last changes can be canceled by clicking on the button . That relates to all editing operations. The next button to the right returns the canceled changed. In the course of building the draft curve, permissible area errors can occur, if double-clicked is a point too close to the neighboring curve or to the wireframe border, not to say of a point suggesting curve intersection.





Curve Table

The button  displays a window containing a numeric table representing equation of the current curve. The table's columns are: point number, its x- and z-coordinate. The window can stay in the foreground while editing is under way, and the table's content will change in step with curve changes made. The table length is equal to the number of grid verticals even if some points are removed. If so, the placeholder of z-coordinate is empty. If the box *Show point on the image* is checked, the curve point matching the selected item in the table is pointed to by a small red arrow.

Curve transformations

Some tool bar buttons and the *Curve* menu contain commands for different transformations of the current curve.


Table 1. Curve transformations.

Command	Operation
Buttons  and 	Vertical shift by the value defined in the <i>Parameters</i> dialog.
Buttons  and 	Horizontal shift: the entire curve is shifted to the right or to the left by the number of x-steps defined in the <i>Parameters</i> dialog. When shifted to the right, the curve loses points at the right end and acquires new points at the left end with the same z-coordinate as the first curve point had before shift. The symmetric picture is observed when the curve is shifted to the left.

<i>Interpolate</i>	When the current curve is defined at some nodes only, the command reconstructs it at the rest of nodes using linear or spline interpolation. Editing is not finished until the curve is defined at all x-net nodes.
<i>Approximate</i>	Replaces the current curve with a polynomial or a spline with equidistant nodes built so as to fit the curve in the sense of least squares. Polynomial degree and number of spline nodes are defined in the <i>Parameters</i> dialog.
<i>Smooth</i>	Smooths the current curve with moving average or median method. Half-width of sliding window is defined in the <i>Parameters</i> dialog.

Performing any of the listed operation may cause the permissible area error. In that case, the curve stays intact.

Segment

The operations of removing points, vertical shift, interpolation and smoothing can be applied not only to the entire curve, but also to a selected curve segment. To select a segment, capture it with the rubber-band and choose *Select segment* in the rb-menu. The segment is redrawn in the select color (red, by default) and becomes the object of the listed operations. Pointwise editing is not affected by selection. To cancel selection, use the button  or double-click the plotter.

Settings

The user can adjust curve drawing attributes (*Wireframe/Drawing attributes*) and operation parameters (*Curve/Parameters*). Each of those commands displays the appropriate dialog (fig. 4).

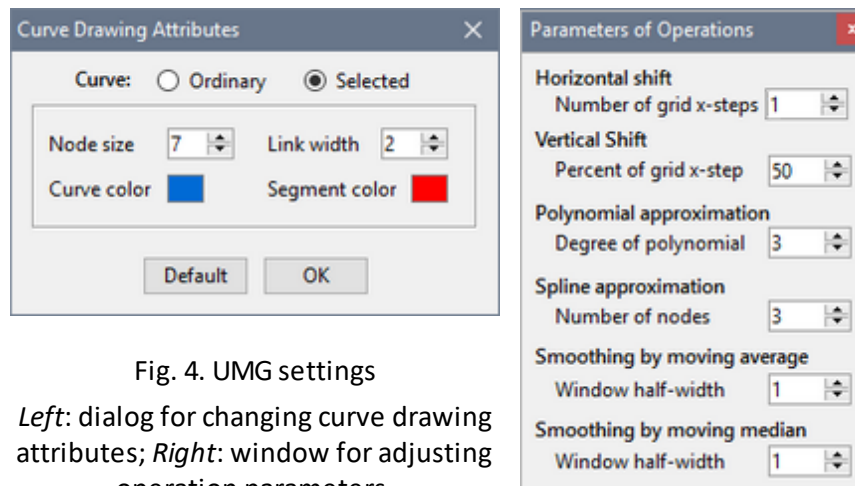


Fig. 4. UMG settings

Left: dialog for changing curve drawing attributes; *Right:* window for adjusting operation parameters.

Drawing attributes include: node size, link width, curve color and segment color. There are two sets of attributes: for the current curve and for the other curves. They can be accessed with the radio-buttons *Ordinary* and *Selected*. The Parameters window floats over the plotter and does not block access to it. The parameter are listed in table 2.

Table 2. Operation parameters


Operation	Parameter
-----------	-----------

<i>Horizontal shift</i>	The parameter is shift value measured in a number of the x-step.
<i>Vertical shift</i>	The parameter is shift value specified as percentage of x-step.
<i>Polynomial approximation</i>	The parameter is degree of polynomial. Cannot be more than a number of curve points minus 1.
<i>Spline approximation</i>	The parameter is a number cubic spline nodes <i>inside</i> the curve x-range. Cannot be more than a number of curve points minus 4.
<i>Smoothing by moving average</i>	The parameter is half-width of sliding window measured in a number of points. The window contains the current curve point, R points to the left of it and R points to the right of it. R is window half-width. z-coordinate of the current point is replaced with the average of z-coordinates of 2R+1 points inside the window.
<i>Smoothing by moving median</i>	The parameter is the same as in previous operation. Here median is computed instead of the average. This kind of smoothing is helpful for removing sharp curve fluctuations.

Changing the wireframe

Operations relating to changing the wireframe include inserting a curve, explained above, and operations from the *Wireframe* menu listed in table 3.

Table 3. Commands of the *Wireframe* menu.

Command	Operation
<i>New Wireframe</i>	Creates a new wireframe (see above).
<i>Remove Curve</i>	Removes the current curve from the wireframe. Do not confuse with the command  , which deletes points of the curve or its segment, but not a curve as a member of the wireframe.
<i>Expand</i>	Expands the wireframe rectangle vertically. The user defines new z-coordinates of the rectangle top or/and bottom.
<i>Clear</i>	Destroys created or imported wireframe in the computer's memory.
<i>Drawing Attributes</i>	Displays the dialog for changing curve drawing attributes (see above).

MG-файлы

Commands for loading and saving a wireframe are collected in the *MG File* menu and listed in table 4. Details will follow.

Table 3. Commands of the menu *MG File*.

Command	Operation
<i>Open</i>	Loads specified MG file.

<i>Save</i>	Saves the edited wireframe to the file it was loaded from, and if it is created anew – to the file specified by the user.
<i>Save As</i>	Saves the wireframe to the file specified by the user.
<i>Import Curve</i>	Read the curve with specified number from an MG file and insert it into the wireframe.

After the *Open* command is issued, the program verifies file structure and, if no error occurs, determines the wireframe domain *D* and optimal number of x-net nodes *N*. Then UMG displays *D* and *N* in the *Definition of Grid* dialog (fig. 2) permitting the user to increase *N* and extent vertical range of *D*. When the user closes the dialog with *OK*, the wireframe is displayed on the plotter surface.

When a curve is imported from an MG file into the current wireframe, UMG displays the list of curve numbers found in the file. The user selects the curve by its number. Before displaying the curve, UMG redefines it on the current x-net and checks for the permissible area errors. Then the curve is displayed on the plotter.

In the MG file created by UMG, when saving the current wireframe, the curves are defined on the x-net, which the user sees on the plotter, even if, initially, the wireframe was loaded from a file with other way of storing the curves.

H-line editor mode

UMG is launched by Model Editor in a special mode for editing or importing grid h-lines. In this mode, UMG functionality is reduced to the level required by the task. Specifics of the mode are described [here](#).

3 Static Corrections

The Problem

Applying tomography for computation of statics is well-known practice. First arrival tomography inversion is used here to determine velocity distribution in near-surface layer and reference observations to a new datum – horizontal line. The Static Calculator (USC) utility computes statics for a given set of stations (observations), using velocity distribution obtained in XTomo-LM.

Stations are defined with their numbers or IDs and their coordinates (in the coordinate system of a project whose model is used). A set of stations are provided in the form of an ASCII file of the **SLS** format (Station Locations for Statics). New datum is supposed to be one of horizontal lines $Z = h_k$ $k = 1, 2, \dots, n$. The user defines them in the utility main window. For each h_k , the utility creates a file of corrections of the **SC** format, which can have either ".sc" or ".txt" extension, on the user choice. To facilitate working with files, their name follow the pattern **<prefix>(<h>).<ext>**, for example, statics(-1500).txt. Each station in an SLS file contains a scale factor for datum levels to be integers (see [format description](#)).

User Interface

The utility is launched by the command *Tools/Static Calculator* of the Project Manager main menu after the root node containing the model in question is selected in Processing Tree. A part of the utility main window is shown on fig. 1.

The screenshot shows a window titled 'Static Calculator' with three main panels: **Input**, **Datum**, and **Output**.

- Input Panel:** Contains a 'Model Folder' text field with the path 'C:\XTLM3_Works\Barentz Sea G4 2008\Model 2'. Below it is an 'SLS File (Stations)' text field with the path 'C:\XTLM3_Impex\SLS\2134-67.sls' and a folder selection button on the right. At the bottom right of this panel, it says 'Scale factor for datum Z: -1000.'
- Datum Panel:** Contains a label 'Z = H; H sampled from [H1, H2] with step D; all numbers are integers'. Below this are three text fields: 'H1' with '-250', 'H2' with '-50', and 'D' with '10'. To the right of these fields is the constraint 'H1 < H2; D > 0'.
- Output Panel:** Contains three fields: 'SC file name prefix' with 'statics', 'Extension' with a dropdown menu showing 'TXT', and 'Text' with a dropdown menu showing 'Windows'. Below these is an 'Output folder' text field with the path 'C:\XTLM3_Impex\SC' and a folder selection button on the right.

Fig. 1. Specifying data for USC.

The window contains three panels: *Input*, *Datum* and *Output*. The *Input* panel displays the filled *Model Folder* field and the field for SLS file name. To select file, use the button on right side above the field. On the *Datum* panel, datum values are defined by specifying the interval [H1, H2] ($H1 < H2$) and step $D > 0$. The integer numbers are entered in their fields with the scale factor in mind.

The SC file is defined on the *Output* panel according to the pattern. If the file will be used on Unix, select the necessary item of the *Text* drop-down list. Finally, select a container folder for all SC files to be created. Use the local folder browser for that that is invoked by the folder button. Now everything is ready to start the operation with the *Start* button.

Appendix

1 Export from XTomo LM: Summary

Table 1. Options of export to ASCII files

Data	Format	Module
Velocity $V(x, z)$ at grid nodes	VFT, DAT	Model Viewer , Model Editor
Model seismic horizons	MG, BLN	Model Viewer, Model Editor
Observation data	SRT	SRT Port Manager , Ray Catalog Viewer on o-node
SR data (all or by waves)	SR	Ray Catalog Viewer on o-node
Ray Catalog after ray tracing	SR, SRT, TXT	Ray Catalog Viewer on f-node
Velocity $V(x, z)$ at vertices of cells for which $F(c) > F_0$, where F is any ray coverage function	DAT	Forward Problem Viewer
Ray paths	BLN	Forward Problem Viewer
Ray coverage functions	DAT	Forward Problem Viewer
Observation data with, possibly, noised computed times	SRT	Forward Problem Viewer

Table 2. Options of export to graphic files

Data	Module
Model	Model Viewer , Model Editor
Model with observation system	Graphic Spread Viewer
Model with traced rays	Forward Problem Viewer
Computed TX-curves (in I-projects with the observed)	Forward Problem Viewer

Ray coverage maps	Forward Problem Viewer
Velocity difference map after tomography inversion	Inverse Problem Viewer

2 Basic ASCII File Formats

Types (formats) of ASCII files used in XTomo-LM can be divided in two groups: *table-like* and *consecutive*. In a table format data are represented as a numeric table, usually with named columns. Numbers in line are delimited with spaces, comma or tab. Alignment is not required. As a rule, there is a *title line* in a table file. It is the first line containing column names, often in quotes. The title line is not interpreted by the software.

Consecutive format suggests that a file consists of blocks of the same structure following one after another. Usually, a block contains one or two headlines and a table. Below, the main XTomo-LM formats are listed alphabetically.

An empty line in a file is interpreted as end of file. In some formats comment-lines are allowed. They must start with double slash: "//". In the below list such formats are marked with the asterisk.

BLN

This is a consecutive format for a set of curves. It is an input format of the graphic system Surfer®. In XTomo-LM it is used for exporting horizons and rays. One block matches one curve (a horizon, a ray). A headline contains one number: number of curve points. The table has two columns "X" and "Z" for point coordinates.

DAT

The table format for representing a function $F(X, Z)$ of two variables, as velocity distribution or ray coverage. The format is one of input formats of Surfer®. The columns are: X, Z, F. The title line can look like that: "X", "Z", "V".

MG

This is an XTomo-LM table format for a a set of curves, for example. In particular, the format is used to store model wireframes. The name comes from "model geometry". Columns: "Curve number", "X", "Z". For all points of one curve, the value of the first column is the same. Curve points are ordered by X. Curve numbers growth with no gap from 1 or 0. If the first curve number is 0, then it is treated as model top line, when an MG file is used for creating the [starting model](#). See fig. 1a for an example.

"Horizon #", "X", "Z"					
1	0.000	-2.700			
1	2.041	-2.819			
1	4.082	-2.926			
1	6.122	-3.022			
1	8.163	-3.107			
1	10.204	-3.182			
.....					
2	0.000	-12.947			
2	2.041	-13.046			
2	4.082	-13.142			
2	6.122	-13.237			
2	8.163	-13.329			
2	10.204	-13.418			
2	12.245	-13.505			
2	16.327	-13.667			
2	18.367	-13.742			
2	20.408	-13.813			
2	22.449	-13.879			
.....					
a					

X = 4.500					
-0.010	1.93				
-0.422	1.93				
-0.427	2.02				
-0.668	2.06				
X = 9.500					
-0.010	1.95				
-0.298	1.95				
-0.327	2.02				
-0.525	2.05				
-0.710	2.09				
-0.906	2.13				
-1.122	2.17				
-1.357	2.22				
-1.613	2.28				
-1.880	2.33				
-2.154	2.39				
X = 14.500					
-0.010	2.00				
-0.209	2.00				
.....					
b					

Fig. 1. ASCII file formats. a – MG (a set of curves); b – VC (a set of velocity columns V(z)).

S+R*

This is a table format for lists sources and receivers in two separate files. The files have the same title but different extension: ".s" for sources, ".r" for receivers. Both tables have the same columns: ID, X, Z (fig. 2a).

SR*

The table format for description of observation system, i.e. source-receiver couples. The table has 6 columns: three for sources and three for receivers (fig. 2b).

"ID", "X", "Z"								
1	0.00000	-0.01000	"S_ID", "S_X", "S_Z", "R_ID", "R_X", "R_Z"					
2	10.00000	-0.01000	1	0.000	-0.010	1	0.000	0.000
3	20.00000	-0.01000	1	0.000	-0.010	2	1.000	0.000
4	30.00000	-0.01000	1	0.000	-0.010	3	2.000	0.000
5	40.00000	-0.01000	1	0.000	-0.010	4	3.000	0.000
6	50.00000	-0.01000	1	0.000	-0.010	5	4.000	0.000
7	60.00000	-0.01000	1	0.000	-0.010	6	5.000	0.000
8	70.00000	-0.01000	1	0.000	-0.010	7	6.000	0.000
			1	0.000	-0.010	8	7.000	0.000
a			b					

Fig. 2. ASCII file formats. a – S+R (sources and receivers); b – SR (observation system).

SRT*

The table format for description of observations. The first 6 columns are the same as in SR, then follow columns for observed time and wave code follow (fig. 3).

S_ID	S_X	S_Z	R_ID	R_X	R_Z	To	Wave
1	0.000	-0.010	1	0.000	0.000	0.005	0
1	0.000	-0.010	2	1.000	0.000	0.525	0
1	0.000	-0.010	3	2.000	0.000	1.050	0
1	0.000	-0.010	4	3.000	0.000	1.573	0
1	0.000	-0.010	5	4.000	0.000	2.096	0
1	0.000	-0.010	6	5.000	0.000	2.609	0
1	0.000	-0.010	7	6.000	0.000	3.114	0
1	0.000	-0.010	8	7.000	0.000	3.610	0
.							
1	0.000	-0.010	1	0.000	0.000	7.513	10
1	0.000	-0.010	2	1.000	0.000	7.528	10
1	0.000	-0.010	3	2.000	0.000	7.560	10
1	0.000	-0.010	4	3.000	0.000	7.611	10
1	0.000	-0.010	5	4.000	0.000	7.678	10
1	0.000	-0.010	6	5.000	0.000	7.762	10
1	0.000	-0.010	7	6.000	0.000	7.863	10
1	0.000	-0.010	8	7.000	0.000	7.978	10
1	0.000	-0.010	9	8.000	0.000	8.108	10

Fig. 3. ASCII file formats. SRT format for description observations in I-projects.

First arrivals tomography system Firstomo uses the #DT format for observation files. Unlike SRT, it does not contain "S_ID", "R_ID" and "Wave" columns.

VC

The XTomo-LM format for import/export of velocity columns $V(z)$. It is the consecutive format with blocks matching velocity columns. The block headline contains X-coordinate of velocity column in the form $X = \langle \text{number} \rangle$ (fig. 1b); it is followed by the "Z-V" table for the velocity column.

VFT

The table format for velocity distribution used by the Firstomo system and all XTomo and XTomo-LM version. It uses no title line and the following columns: cell number in the sense of Firstomo, X, Z, V. The format is used for data exchange between different versions of Firstomo, XTomo, XTomo-LM.

3 Other formats

Formats of file for statics

To apply XTomo-LM for introducing static correction, two ASCII files are required: an input file of stations (SLS format) and an output file of corrections (SC format). Details can be found [here](#). Formats of both types are shown on fig. 1.

ZScale = -1000			Reference datum elevation: -250	
"Station ID",	"X",	"Z"	"Station ID",	"Correction (ms)"
333	99.8764	0.0	999999999	9999999
334	99.0783	0.0	333	-114
335	98.2804	0.0	334	-114
336	97.4825	0.0	335	-114
337	96.6845	0.0	336	-114
338	95.8858	0.0	337	-114
339	95.0879	0.0	338	-114
340	94.2899	0.0	339	-115
341	93.4920	0.0	340	-115
342	92.6941	0.0	341	-115
343	91.8954	0.0	342	-115
344	91.0974	0.0	343	-115
345	90.2995	0.0	344	-115
346	89.5016	0.0	345	-115
347	88.7035	0.0	346	-115
348	87.9049	0.0	347	-116
349	87.1070	0.0	348	-116
350	86.3091	0.0	349	-116
.....			
a)			b)	

Fig. 1. File Formats for statics. a - SLS format for stations; b - SC format for corrections.

In the first line, SLS format bears a scale coefficient for representing reference data depth as integers. The scale coefficient looks like in SEG-Y: it is one of the integers S: -10000, -1000, -100, -10, 1, 10, 100, 1000, 10000. The scaled integer N and its true value R are relates this way: $R = N \cdot S$ for $S > 0$, and $R = -N/S$ for $S < 0$.

The second file line is a title line; it contains columns names, often in quotes. It is not interpreted, but must be present. Then the numeric table with named columns follows. A delimiter in a table line is space(s) or comma, or tabulation. Alignment not required.

The SC format used is adjusted to one of known seismic processing system. The first line contains datum z-coordinate (scaled with the coefficient from SLS). The second line is a title one, it names the two table columns: "Station ID" and "Correction". The third line contains two strings of '9'. Each string points to column position and maximal width. Correction is measured in milliseconds and *must be added* to station time.

Index

- 2 -

2D profiling 23

- A -

active module list 26, 41
apparent velocity 75
archive 48

- B -

black cells 46, 63
building
 horizon 126
 initial velocity 115
 reflector 125
 refractor 129

- C -

color set 46
column 21
common folders 29
comparing velocities 136
computations 25
concurrency 93
constarints 107
conversion coefficient (CC) 18
converted wave 18
coordinate system 21
cycle of processing 28
 termination 114

- D -

Data Preparation Unit 23
distance unit 25
Diving Wave TX-curve Inverter 115
docked 95, 97
DPU 16, 23
DWI 115

- E -

editing model 57

export

 to ASCII files 54, 84, 106, 147
 to graphic files 54, 55, 105, 147

- F -

f-cell 104

file

 ASCII 148, 150
 BLN 148
 CS 46
 DAT 148
 MG 24, 148
 observation 22
 S и R 149
 S+R 22, 148
 SC 150
 SLS 150
 SR 22, 149
 SRT 22, 148, 149
 VC 24, 148, 150
 VFT 148

first arrivals 18

Firstomo 22

fixed cell 104

fixing velocity 104

f-node 35

folder

 archive 29, 47
 common 29
 common import-export 29
 working 29

format string 25, 33

forward problem 14, 91

 log 94

FPS 91

FPV 91, 95

freezing velocity 104

- G -

GPA 63, 64

graphic module

 ASCII export 54
 based on a curve set image 26, 73
 based on model image 26, 49
 graphic export 54, 55
 hot keys 53

graphic module
 magnifying 53
 rb-menu 52
 rubber-band 52
 scrolling 53
 selection 52
 velocity profile 54
 zoom 53
 grid 21
 cell 21
 changing geometry 63
 chnging thickness 58
 column 21
 editing velocity 59
 h-line 21
 h-line, editing 63, 64
 h-line, import 63, 65
 h-line, moving 63, 64
 horizon 22
 horizon list 51
 inserting rows and columns 59
 node 21
 perturbation area 63, 64
 properties 51
 row 21
 seismic horizon 22
 selection 52
 smoothing 59
 strip 21
 top line, editing 66
 top line, import 66
 velocity and geometry 66
 vertex 21
 vertical 21
 v-line 21

- H -

head wave 18
 h-line 21
 h-line editor 64, 65, 145
 horizon
 database 128
 database manager 128
 ID 22
 list 22, 51
 Horizon Previewer 128
 horizontal strip 21

hot keys 53

- I -

initial approximation 14
 initial points 129
 initial velocity 115
 i-node 35
 Inverse Problem Solver 110
 inverse tomography problem 14, 107
 solution 110
 viewing solution 112
 inversion project 17
 I-project 17
 IPS 110

- K -

kinematic interpretation 14

- L -

Least Square Method
 regularized 107
 luminance 43

- M -

manual model fitting 14
 measurement units 25
 MED 57
 mesh 21
 MG File Editor 24, 139
 m-node 35
 model 20
 changing geometry 63
 conception 20
 domain 21
 editing 57
 geometry 24
 h-line, editing 63
 h-line, import 63
 h-line, moving 63
 initial 14
 inserting velocity column 57
 layered 14
 rectangle 21
 refined 14
 replacing velocity 57

model 20
 seismic horizons 66
 starting 30
 top line, editing 66
 top line, import 66
 top, editing 58
 velocity 14
 velocity function 20
 wireframe 24, 30, 139
 Model Editor 57
 Model Viewer 49
 modeling 14
 observation 68
 modeling project 17
 module 17
 active 26, 41
 active module list 26
 graphic 26
 list 41
 primary 26
 short name 41
 MOV 49

- N -

 New Project Constructor 31
 node 21
 numeric formats 25

- O -

 observation file
 import 71
 observation system 68
 observations
 file 22
 type 23
 o-node 35

- P -

 package 47
 permissible area (wireframe) 141
 plotter
 M- 95
 of graphic module 26
 T- 95
 PM 28
 Processing Tree 28

comment 36
 copying nodes 36
 creating nodes 36
 deleteing nodes 36
 f-node 35
 growth 36
 i-node 35
 menu 37
 m-node 35
 o-node 35
 profiling 23
 project
 adding to Working Folder 34
 archive 48
 archiving 34
 cloniong 34
 copying 34
 creating 30
 creating from a package/archive 34
 creating from version 2 48
 creating from version 2 data 34
 format strings 33
 list 28
 model rectangle 33
 package 47
 properties 33, 34
 removal 34
 renaming 34
 resolution 33
 starting model 30, 32
 velocity range 33
 Project Manager 26, 28, 34
 Prpcessing Tree
 workflow 36

- R -

 Ray Catalog 81
 copying to SRT Port 68, 84
 default 88
 export 84
 f-node 68
 in M-projects 85
 o-node 68
 populating 78
 Ray Catalog Viewer 83
 ray coverage density 103
 ray picture 95

- ray sample 98, 99
 - creation 100
 - saving in I-project 101
- ray-tracing
 - precision 93
- rb-menu 52
- RC 103
- RCH 103
- RCV 103
- reciprocal
 - points 120
 - points base 120, 129
 - time 120
 - time residual 122
 - TX-curves 120
- reflection 18
- reflector 125
- Reflector Builder 125
- refraction 18
- refractor 129
- Refractor Builder 129
- regularization 107
- regularized linear least square problem 107
- regularizer 107
- residuals 95
- resolution 25, 33
- resolution error 26
- row 21
- rubber-band 52

- S -

- samples 131
- seismic horizon 22
 - building 125
 - database 120, 125
 - refractor 129
 - reflector 125
- seismic horizons 66
- selector 52
- smoothing 107
- spectrum 34
 - color sets 46
 - continuous 43
 - creating 45
 - editing 44
 - editing lines 45
 - line 43

- velocity range 45
- spread 68
 - viewing 82
- Spread Viewer 82
- SRT Data Extractor 78
- SRT Port 23, 68
 - archives 77
 - data store 69, 71
 - database 69
 - databases 70
 - export 77
 - import 71
 - sampling data for processing 80
 - TX-curves 73
- SRT Port Manager 69
- SRT TX-curve Viewer 73
- stabilizer 107
- starting model 30
 - creation 32
- Surfer 17, 54

- T -

- time errors 95
- time residuals 95
 - distributions 102
 - statistics 101
- tolerance
 - zero 137
- tomography 110
 - first arrival 14
 - initial velocity 115
 - inverse problem 14, 107
 - regularization 107
 - restrictions 107
 - stabilizer 107
 - theory 107
- T-plotter 95
- TX-curve 97, 120, 129
 - base 115
 - direct 23
 - diving wave 115
 - domain 115
 - head wave 120
 - initial points 129
 - inversion 120
 - midpoint 115
 - reciprocal 120

TX-curve 97, 120, 129
 reflection 120
 reversed 23
 sample for inversion 117
 sampling 115
 set 23, 73, 115, 120, 125, 129
 set, sampling 122
 TX-curve inversion 14, 122
 diving wave 115
 head wave 120, 122, 129
 reflection 120, 122, 125
 TX-Curve Selector 122
 type of observations 30

- U -

UMG 24, 139
 utility 17
 MG File Editor 139
 Static Calculator 145
 Velocity Comparator 136

- V -

velocity
 black cells 63
 color spectrum 43, 63
 column 24
 column set 24, 30
 comparison 136
 continuation 59, 60
 copying 62
 difference map 137
 editing 59, 60
 fixation 104
 freezing 104
 in covering layer 126
 in selection 52
 in-place editor 61
 power function 60
 profile 49, 54
 range 33, 45
 refined 107, 114
 replacing 62
 smoothing 59, 60
 vertical profiles comparison 139
 version 2 data conversion 48
 vertical strip 21

- W -

wave
 adding 42
 converted 18, 42, 92
 diving 18
 diving wave 18
 drawing attributes 43
 export/import 42
 first 18
 head 18
 ID 19
 list 41, 42
 numeric code 18
 numeric codes 19
 refraction 18
 transient 18
 Wave Manager 34, 41
 wireframe 24
 permissible area 141
 workflow 28

- X -

XMF 16, 32

- M -

M-plotter 95
 M-project 17

- P -

рефлектор
 оценка 129

- T -

T-plotter 97