Code *PROFIT* for forward modeling and tomographic inversion based on active refraction seismic profiling data

Ivan Koulakov

Head of Laboratory for Forward and Inverse Seismic Modeling
Institute of Petroleum Geology and Geophysics, SB RAS,
Prospekt Akademika Koptuga, 3, Novosibirsk, 630090, Russia
e-mail: KoulakovIY@ipgg.nsc.ru
Phone: +7 383 3309201
Mobil: +7 913 453 8987

with active help of:
Heidrun Kopp (Kiel, Germany)
Tatyana Stupina (Novosibirsk)
Kirill Gadylshin (Novosibirsk)

Novosibirsk, Russia, December 2009

Table of contents:

1. Introductory remarks

We present a code for combined forward modeling and tomographic inversion based on active seismic refraction profiling data (*PROFIT* – Profile Forward and Inverse Tomographic modeling). The *PROFIT* code is created using the *FORTRAN-90* programming language and is designed in *MS Windows OS* (alternatively, compilation for *LINUX* is possible). The code is simple in operating and optimizes computation time (for example an outdated laptop of 700 Mhz of CPU speed performs the inversion for ~2000 rays in 9 iterations including 2D ray tracing and inversion in each step in about 30 minutes).

The code provides a possibility to visualize the results and intermediate steps (starting model, ray paths in each iteration, node distribution, results in absolute velocity and relative anomalies, as well as a plot with observed and modeled travel times). The code produces the files which are directly suitable for visualization in Surfer Golden Software.

The code can be freely downloaded from www.ivan-art.com/sience/PROFIT. All additional information can be obtained through personal communication with the author, Ivan Koulakov (KoulakovIY@ipgg.nsc.ru).

The main particular features of the *PROFIT* algorithm:

1. The algorithm allows defining complex 2D velocity models using a wide range of different tools. These velocity models can be used either as starting distribution for performing the inversion of observed data, or for synthetic modeling.

2. The ray tracing in a 2D velocity distribution is performed based on the Fermat principle (bending method). It is suitable for highly-complex velocity models and for any method of parameterization. This method is fast and absolutely stable.

3. The parameterization is performed based on nodes and accounts for variable ray distribution. The node spacing is always defined much smaller than the size of the expected anomalies, and the resolved anomalies are almost independent of grid configuration.

4. The stability of the solution is tuned by smoothing and amplitude damping parameters during tomographic inversion.

5. Intermediary and final results are automatically previewed and stored as bitmap images in PNG files. The output of the programs is adapted to visualizing with SURFER, which allows producing higher quality pictures.

The PROFIT code allows for performing several different schemes of forward modeling and inversion such as:

1. Inversion of the observed data. In this case there are several criteria for finding the best starting model and damping parameters.

2. The program blocks on velocity model construction and ray tracing can be used for performing the forward modeling. The purpose of this modeling is to construct a velocity model, which provides the minimal data misfit in the first iteration of the inversion.

3. Synthetic modeling to reproduce velocity models of any complexity.

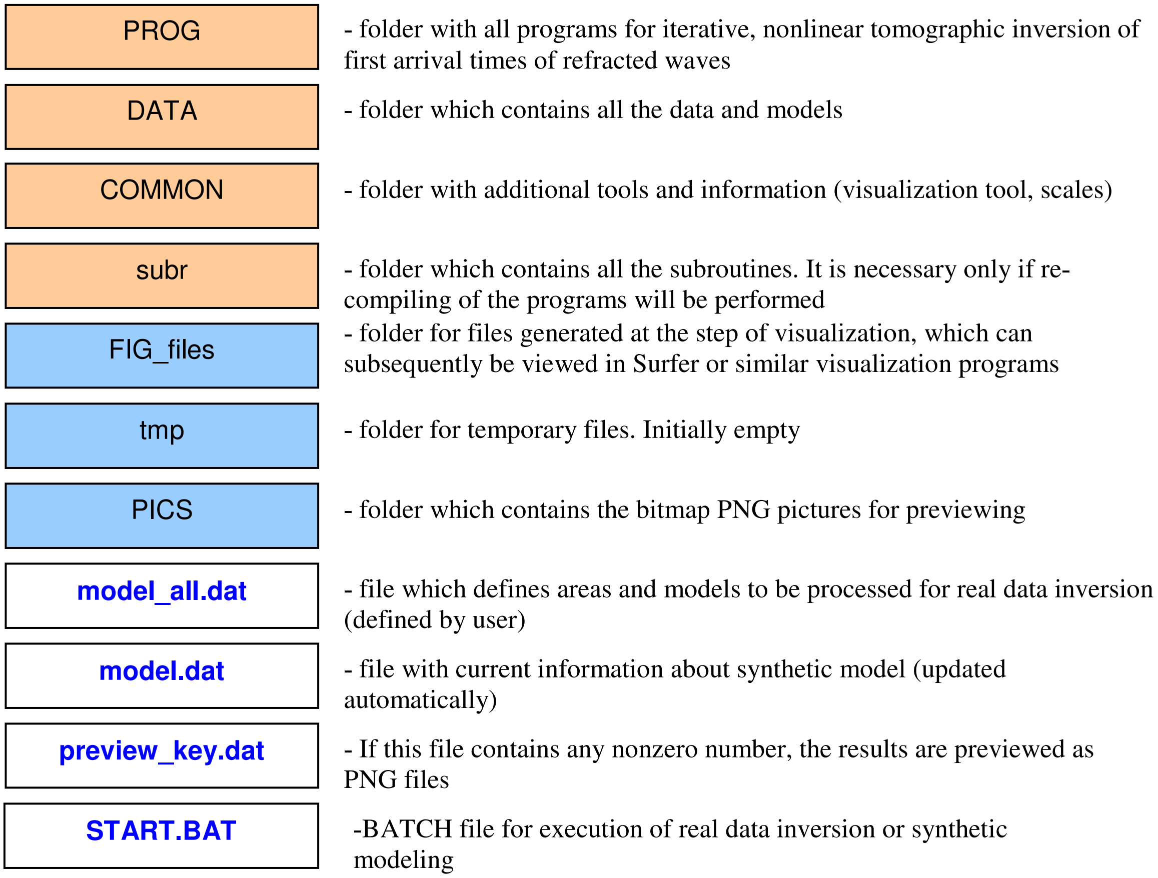
4. Concept of combined forward and inverse modeling (*FM&TI*). The construction of a synthetic model based on the results of forward modeling and tomographic inversion should reproduce an identical velocity structure as obtained after inversion of the observed data. In this case the inversion steps and all parameters should be identical for the cases of synthetic and observed data inversion. We propose that the synthetic model constructed in this way will adequately represent the real structures in the Earth.

The PROFIT code can be equally used for multiscale profiling studies starting from dozens meters to hundreds kilometers. It has already been used in different applied and fundamental studies. In this manual we present a case study using data from the Musicians Seamount range in the Pacific Ocean.

2. General structure of files and folders in PROFIT

2.1. List of folders in the root directory

The recommended file structure in the root directory with short descriptions is presented in the Figure 2.1.



PROG	- folder with all programs for iterative, nonlinear tomographic inversion of first arrival times of refracted waves
DATA	- folder which contains all the data and models
COMMON	- folder with additional tools and information (visualization tool, scales)
subr	- folder which contains all the subroutines. It is necessary only if re-compiling of the programs will be performed
FIG_files	- folder for files generated at the step of visualization, which can subsequently be viewed in Surfer or similar visualization programs
tmp	- folder for temporary files. Initially empty
PICS	- folder which contains the bitmap PNG pictures for previewing
model_all.dat	- file which defines areas and models to be processed for real data inversion (defined by user)
model.dat	- file with current information about synthetic model (updated automatically)
preview_key.dat	- If this file contains any nonzero number, the results are previewed as PNG files
START.BAT	-BATCH file for execution of real data inversion or synthetic modeling

Figure 2.1 Existing folders (pink boxes), folders created automatically after starting calculations (blues) and files (white boxes) in the root directory of *PROFIT*.

2.2. Structure of the DATA folder

The general structure of the `DATA` folder is shown in Figure 2.2. The `DATA` folder has a two-step hierarchy structure. The `DATA` contains the Area folders (e. g. “`HULA_p02`”, “`DATASET1`”, “`DATASET2`” etc). The name of the Area folder should consist of any 8 characters.

Each “Area” folder contains a mandatory subfolder “inidata” with initial data and several folders for observed data inversion and synthetic modeling (e.g. “R1_V1_A0” or “S1_V1_A1”).

In addition, the “Area” folder contains a mandatory file set_vis.dat, which contains parameters for visualization of the results, and an optional file config.txt for defining the preview parameters.

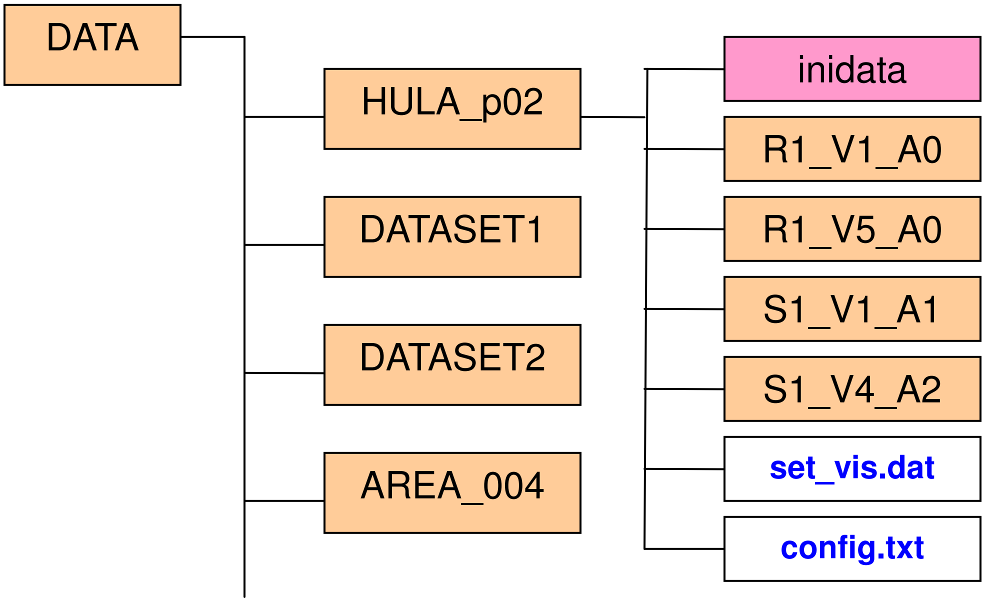


Figure 2.2. Structure of folders (orange and pink boxes) and files (white boxes) in the DATA directory.

2.2.1. *Organization of the input data in the "inidata" folder*

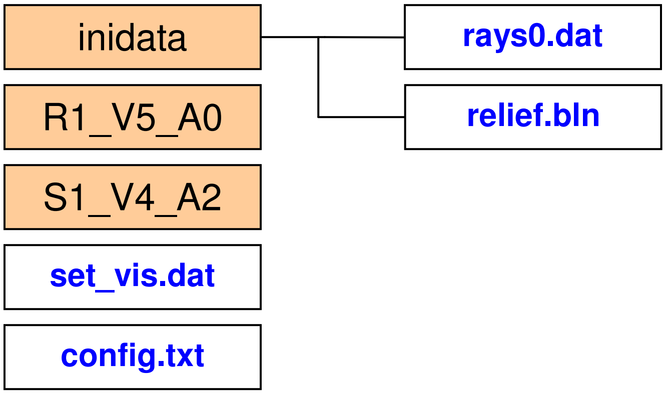


Figure 2.3. Structure of files (white boxes) and folders (orange boxes) in the “inidata” folder.

The input data are contained in the “inidata” folder, as shown in Figure 2.3 and include two mandatory files:

1. “rays0.dat”: list of all travel times
2. “relief.bln”: description of relief which may include seafloor bathymetry

Each line of "`rays0.dat`" is a description of one ray: ***xsrce, ysrce, xrecv, yrecv, time*** in free format.

xsrce, ysrce: coordinates of the source, in km (in the presented example: airgun shots on the sea surface);

xrecv, yrecv: coordinates of the source, in km (in the presented example: ocean bottom stations). Below sea level, y is negative;

time: travel time, in seconds

An example of the initial data is presented below

`/DATA/HUL1_p02/inidata/rays0.dat`

```
     178.5790      0.0000000E+00   52.95900      -5.850410       20.68900
     177.8180      0.0000000E+00   52.95900      -5.850410       20.60100
     176.8960      0.0000000E+00   52.95900      -5.850410       20.53900
     175.6520      0.0000000E+00   52.95900      -5.850410       20.38500
     174.4290      0.0000000E+00   52.95900      -5.850410       20.23400
     173.1850      0.0000000E+00   52.95900      -5.850410       20.08100
     171.9380      0.0000000E+00   52.95900      -5.850410       19.92600
     170.6970      0.0000000E+00   52.95900      -5.850410       19.77300
     169.5380      0.0000000E+00   52.95900      -5.850410       19.62900
     168.3830      0.0000000E+00   52.95900      -5.850410       19.48500
     167.2360      0.0000000E+00   52.95900      -5.850410       19.40100
     166.0580      0.0000000E+00   52.95900      -5.850410       19.25500
     164.5470      0.0000000E+00   52.95900      -5.850410       19.06600
     163.3680      0.0000000E+00   52.95900      -5.850410       18.95400
     162.2120      0.0000000E+00   52.95900      -5.850410       18.83700
     161.0960      0.0000000E+00   52.95900      -5.850410       18.69900
     159.9460      0.0000000E+00   52.95900      -5.850410       18.55600
     158.8210      0.0000000E+00   52.95900      -5.850410       18.41500
     157.6850      0.0000000E+00   52.95900      -5.850410       18.27200
     156.5430      0.0000000E+00   52.95900      -5.850410       18.12900
     155.3780      0.0000000E+00   52.95900      -5.850410       17.98300
     154.2340      0.0000000E+00   52.95900      -5.850410       17.84000
     153.0450      0.0000000E+00   52.95900      -5.850410       17.68900
     151.9000      0.0000000E+00   52.95900      -5.850410       17.54600
     150.7610      0.0000000E+00   52.95900      -5.850410       17.40100
     149.6190      0.0000000E+00   52.95900      -5.850410       17.25600
     148.4680      0.0000000E+00   52.95900      -5.850410       17.10100
     147.2980      0.0000000E+00   52.95900      -5.850410       16.90800
. . . . . . to be continued
```

File "`relief.bln`": contains the information on the relief which may include the seafloor bathymetry. Example of the file is presented below:

`/DATA/HUL1_p02/inidata/relief.bln`

```
    1441 0 0.0001 1.5 0.4      N_nodes, Sea_level, V_air, V_sea, V_min
      0.0000000E+00  -5.890000   xrlf, yrlf
      0.1250000      -5.892500
      0.2500000      -5.895000
      0.3750000      -5.897500
      0.5000000      -5.900000
      0.6250000      -5.902500
      0.7500000      -5.905000
      0.8750000      -5.907500
       1.000000      -5.910000
       1.125000      -5.911250
       1.250000      -5.912500
       1.375000      -5.913750
       1.500000      -5.915000
       1.625000      -5.916250
       1.750000      -5.917500
       1.875000      -5.918750
       2.000000      -5.920000
       2.125000      -5.922500
       2.250000      -5.925000
       2.375000      -5.927500
       2.500000      -5.930000
       2.625000      -5.932500
       2.750000      -5.935000
       2.875000      -5.937500
       3.000000      -5.940000
       3.125000      -5.945000
       3.250000      -5.950000
 . . . . . . to be continued
```

First line of this file includes the following parameters:

N_nodes: number of points for definition of the relief.
Sea_level: Sea level altitude in the current coordinates
V_air: Velocity in the air. Usually it is presumed that rays do not travel in air; thus this velocity is usually defined very small (0 value is not recommended).
V_sea: Velocity in the sea water.
V_min: Minimal velocity which is allowed in the target area.

Next lines: **xrlf, yrlf**, coordinates of the relief (same coordinate system as in "`rays0.dat`").

2.2.2. Organization of the MODEL folder

The MODEL folder is created either for inversion of the observed data or synthetic modeling. The name of the MODEL folder should contain any 8 characters (e.g. "`R1_V1_A0`" or "`S1_V1_A1`").

ADVICE: We recommend fixing the names of the models according to the description of starting or synthetic model. For example, "`R1_V1_A0`" refers to the first trial of observed data inversion. The basic velocity model in this case is defined with the key "1" (1D model) and no anomalies are defined. For "`S1_V4_A2`" we have a first trial of a synthetic model in which the basic synthetic model is fixed with the key "4" (parameterization with lines) and anomalies have the key "2" (free shaped polygons). The definition of the velocity models is described in more detail in Section 5.

The structure of the MODEL folder for performing the inversion of the observed data with brief description of the main files and folders is shown in Figure 2.4. The Starting velocity model is described by two files with basic velocity model (`velocity_start.dat`) and velocity anomalies (`anomaly_start.dat`). The definition of velocity models is described in more detail in Section 5.

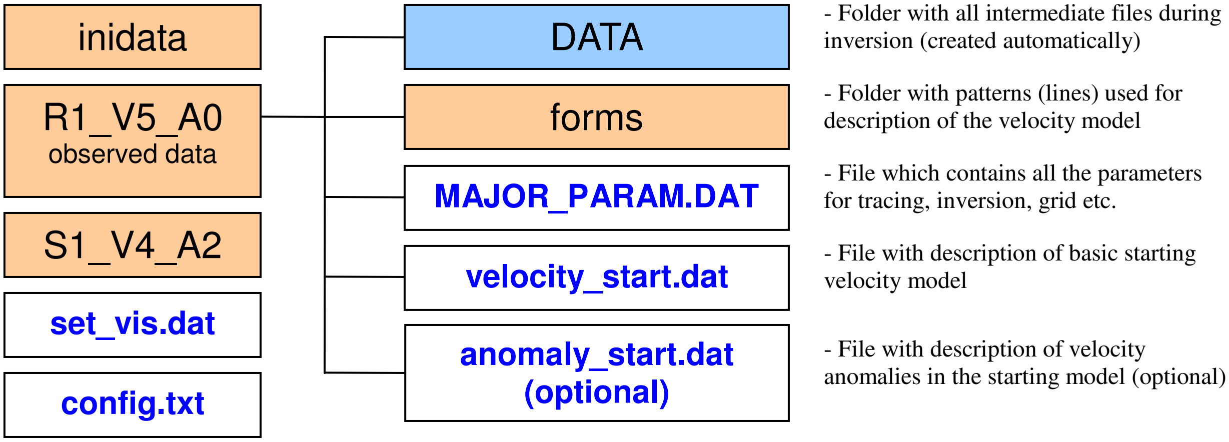

inidata
R1_V5_A0
observed data
S1_V4_A2
set_vis.dat
config.txt
DATA
forms
MAJOR_PARAM.DAT
velocity_start.dat
anomaly_start.dat
(optional)
- Folder with all intermediate files during inversion (created automatically)
- Folder with patterns (lines) used for description of the velocity model
- File which contains all the parameters for tracing, inversion, grid etc.
- File with description of basic starting velocity model
- File with description of velocity anomalies in the starting model (optional)

Figure 2.4. Structure of files and folders in the folder corresponding to observed data inversion

In the case of synthetic modeling (Figure 2.5), the structure of files and folders remains the same, except for two additional files (`velocity_syn.dat` and `anomaly_syn.dat`) which determine the synthetic velocity model. Note that both synthetic and starting models are defined by two files each, using similar formats which are described in Section 5.

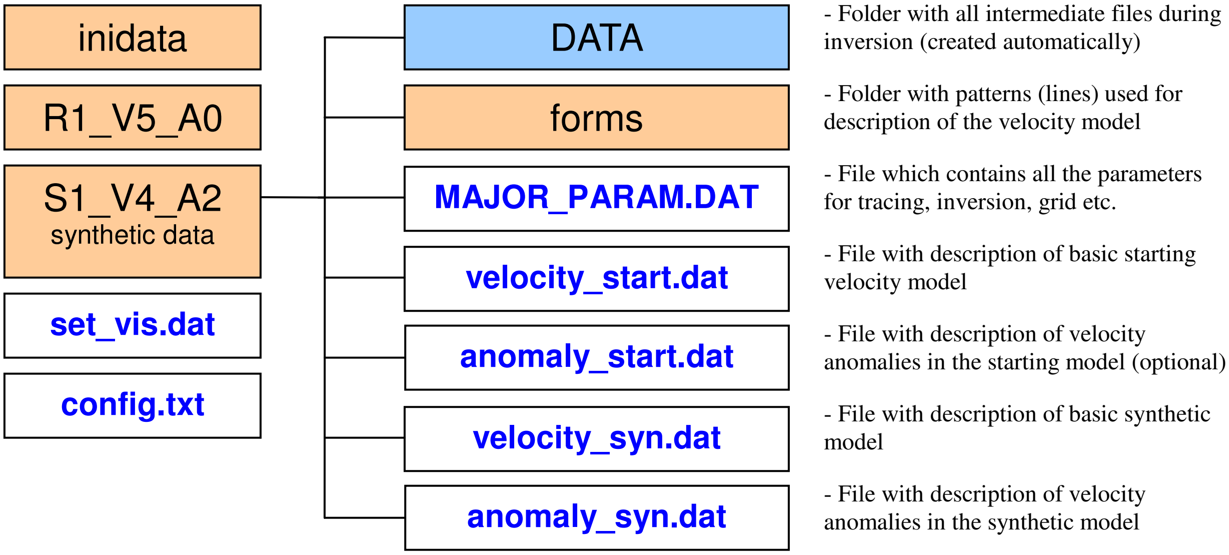

inidata
R1_V5_A0
S1_V4_A2
synthetic data
set_vis.dat
config.txt
DATA
forms
MAJOR_PARAM.DAT
velocity_start.dat
anomaly_start.dat
velocity_syn.dat
anomaly_syn.dat
- Folder with all intermediate files during inversion (created automatically)
- Folder with patterns (lines) used for description of the velocity model
- File which contains all the parameters for tracing, inversion, grid etc.
- File with description of basic starting velocity model
- File with description of velocity anomalies in the starting model (optional)
- File with description of basic synthetic model
- File with description of velocity anomalies in the synthetic model

Figure 2.5. Structure of files and folders in the folder corresponding to synthetic modeling.

2.2.3. Description of the main initial parameters

Most of the parameters for ray tracing, parameterization and inversion are defined in file '`MAJOR_PARAM.DAT`'. The content of this file is organized by rubrics. Each rubric starts with a key line. For example:

```
GENERAL INFORMATION :
REG.BENDING PARAMETERS:
GRID_PARAMETERS:
```

```
INVERSION PARAMETERS :
2D GRID PARAMETERS:
```

etc.

Example of the “`MAJOR_PARAM.DAT`” file is given below (names of rubrics are indicated in red):

```
********************************************************
GENERAL INFORMATION :
1   KEY 1: REAL; KEY 2: SYNTHETIC
6   Number of iterations

********************************************************
Parameters for bending tracing in 2D model
********************************************************
REG.BENDING PARAMETERS:
1 step=10: step along ray to compute travel time
15 segm_min=50: minimum segment to be bended
0.2 bendreg_min=3: minimum value of bending
5 bendreg_max=100: maximum value of bending
10          frequency for printing
3 maximum residual
3   1       sec
5 minimal ray path on bottom

********************************************************
Inversion parameters
********************************************************
INVERSION PARAMETERS:
80          LSQR iterations
15.         smoothing
15.0               regularization


********************************************************
Parameters for grid construction
********************************************************
GRID_PARAMETERS:
0 180 2.           x1,x2,dx for ray density calculation
1 -20 -0.5                 y1,y2,dy for ray density calculation
5   divide the segments in smaller parts
1                  ! Grid type: 1: nodes, 2: blocks
0.5                !min distance between nodes in y. direction
0.001 3.0  !plotmin, plotmax= maximal ray density, relative to average
0.01                       !dy= step of movement along y
0 kod_surf: if not 0, allow sharp contrast on interfaces


********************************************************
Parameters for 2d regilar grid
********************************************************
2D GRID PARAMETERS:
0 180 2.           x1,x2,dx for ray density calculation
 1 -20 -0.2        y1,y2,dy for ray density calculation
5 0.7 dxmax, dymax: max distance to node

********************************************************
Parameters for source correction (optional)
********************************************************
SOURCE CORRECTION:
1.0 weight for source correction in inversion
```

Following the key line in red, a description of parameters for the current group with a fixed format is given. The order of groups and number of empty lines between groups are free.

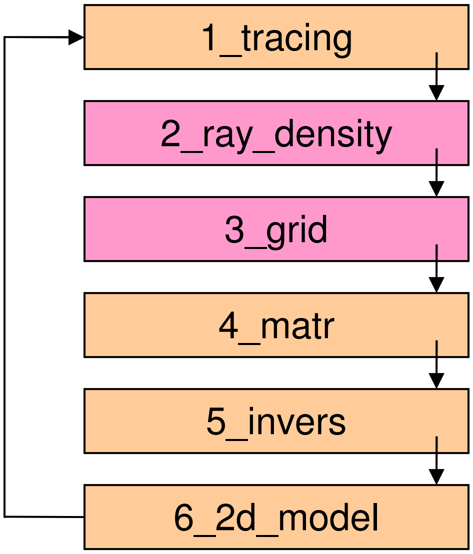
For example, the group presented below indicates the way of modeling (synthetic or real data inversion) and number of iterations.

```
********************************************************
GENERAL INFORMATION :
1   KEY 1: REAL; KEY 2: SYNTHETIC
6   Number of iterations
```

The meaning of other parameters will be explained in the description of the main steps.

3. Description of the tomographic inversion using the PROFIT code

Iterative inversion consists of consequent execution of the following programs:



- Program for ray tracing in a current velocity model

- Program for computing the ray density (only in 1st iteration)

- Program for constructing the parameterization grid (only in 1st iteration)

- Program for computing the matrix of first derivatives

- Program for matrix inversion

- Program for computing the velocity model in regular 2D grid which is used as basic model in the next iteration

Programs indicated with pink are executed only during the first iteration.
We use the following indications:
'//ar//' is the AREA folder
'//md//' is the MODEL folder
'//it//' is the number of iteration

3.1. Ray tracing in 2D velocity model

Project: **\PROG\1_tracing\**
Input data:
in 1 iteration:
/DATA/**'//ar//'**/**'//md//'**/inidata/rays0.dat
in next iterations:
/DATA/**'//ar//'**/**'//md//'**/DATA/rays**'//it-1//'**.dat

The calculations are controlled by parameters in the file:
/DATA/HULA_p02/R2_V5_A0/MAJOR_PARAM.DAT

```
*******************************************************
Parameters for bending tracing in 2D model
*******************************************************
REG.BENDING PARAMETERS:
0.5    dstep, step along ray to compute travel time
10     seg_min, minimum segment to be bended
0.1    bend_min, minimum value of bending
3      bend_max, maximum value of bending
5      n_freq, frequency for printing
3      res_max1, maximum residual for rough selection
3 1.   iter_res, res_max2, parameters for finer selection
5      dis_bot_min, minimal ray path on bottom
```

Description of the main principle for the ray tracing

The ray tracing used in this code is based on the Fermat principle and consists of finding a path, which provides the minimum travel time between source and receiver. This idea is the basis of the bending method of ray tracing (e.g. Um & Thurber, 1987), which has been widely applied for decades and evolved as a standard in different practical codes. We have created our own version of the bending algorithm, which is schematically shown in Figure 3.1. Finding the path of minimum travel time consists of subsequent execution of several bending regimes. In the initial step (Figure 3.1A), the bounce point on the sea bottom (point *b*) is located just beneath the source *s*. We start from the straight line between the bounce point *b* and the receiver *r* and deform it to obtain the minimum travel time. In the first approximation, the deviation *A* with respect to the initial straight path is computed according to the following formula:

$$A = B\cos\left(\pi\frac{s - D_{tot}/2}{D_{tot}}\right) \qquad [1]$$

where *B* is the value of bending, s is the distance along the initial path, and D_{tot} is the length of the initial path. The value of B is adjusted to obtain the curve γ(B) which provides the minimum value of the integral:

$$t = \int_{\gamma(B)} \frac{ds}{V(s)} \qquad [2]$$

where V(s) is the velocity distribution along the ray. (*ds*=**dstep**). B is varied from a maximum value, **bend_max,** to a minimum value, **bend_min**.

In the second step (Figure 3.1B), we move the bounce point *b* along the sea bottom to obtain the minimum value of integral [2]. For land observations, this step will be omitted, because the location of s and b are identical. If the distance between b and r is less than **dis_bot_min**, this ray is rejected.

At the next stage, further deviations of the path between *b* and *r* are performed iteratively using a formula for bending values:

$$A = \frac{B}{2}\cos\left(2\pi\frac{s-(S_2-S_1)/2}{(S_2-S_1)}\right)+\frac{1}{2} \qquad [3]$$

where S_1 and S_2 correspond to the length along the path in the beginning and at the end of the current segment.

During the first iteration, the bending is performed for the entire segment *b*-*r* in a similar way as demonstrated in Figure 3.1A, but using formula [3]. In the second iteration (Figure 3.1C), the path is divided into two segments of equal lengths (*b-m1* and *m1-r*), and each of them is bended according to formula [3]. After determining the minimum time curve, the entire path is divided into three parts (Figure 2.1D), and the same approach of bending is performed for segments *b-m1*, *m1-m2,* and *m2-r*. Subsequently, this procedure is repeated for the path divided into four, five and more parts. The bending terminates when the length of the sections becomes smaller than a predefined value, **seg_min**.

The data selection is performed based on rough and finer criteria. In first iterations (usually 1 and 2) the residuals are rejected if they are larger than **res_max1**. After iteration **iter_res** (usually 3), the selection is based on a finer criterion with maximal residual limit at **res_max2**. **n_freq** is frequency of printing the results of tracing on console.

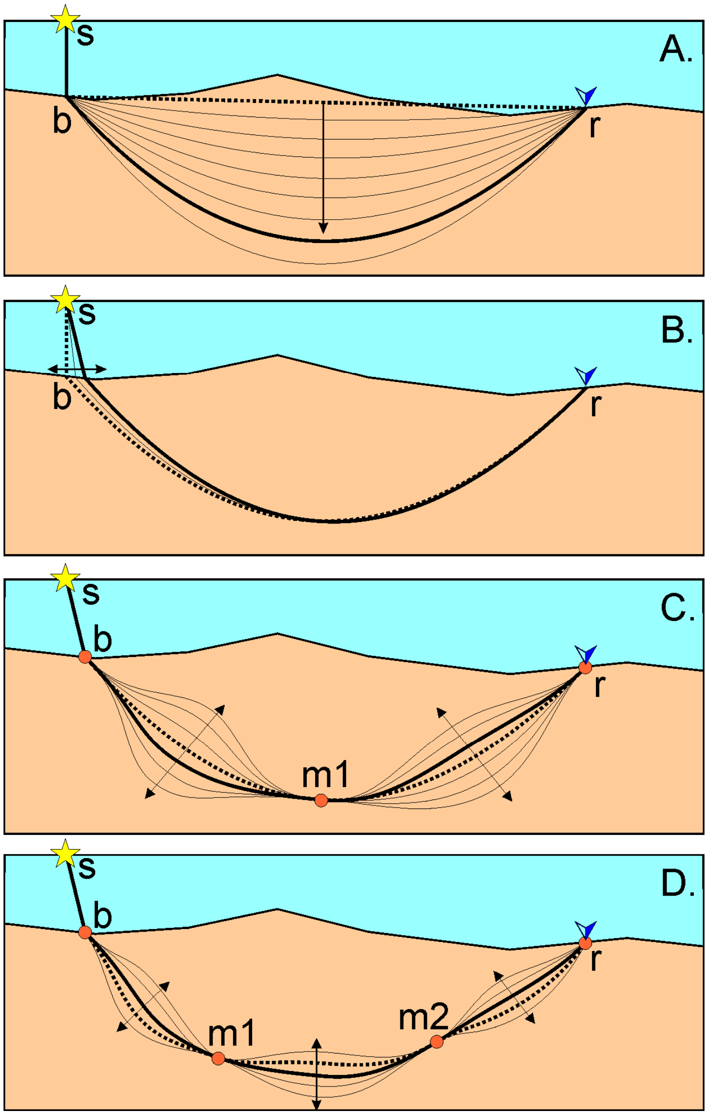


Figure 3.1. Sketch for explaining the principle of our version of the bending algorithm (see description in the text).

It is possible to visualize the rays computed in this step in two ways. (1) High quality images can be produced by Surfer or any other graphic tool. An example of ray distribution in the map view and in a cross section is shown in Figure 3.2. (2) Furthermore, the ray paths in each iteration are previewed automatically as PNG files. Example of rays after 15 iterations for profile "HUL1_p02", model "R1_V1_A0" is shown in Figure 3.3.

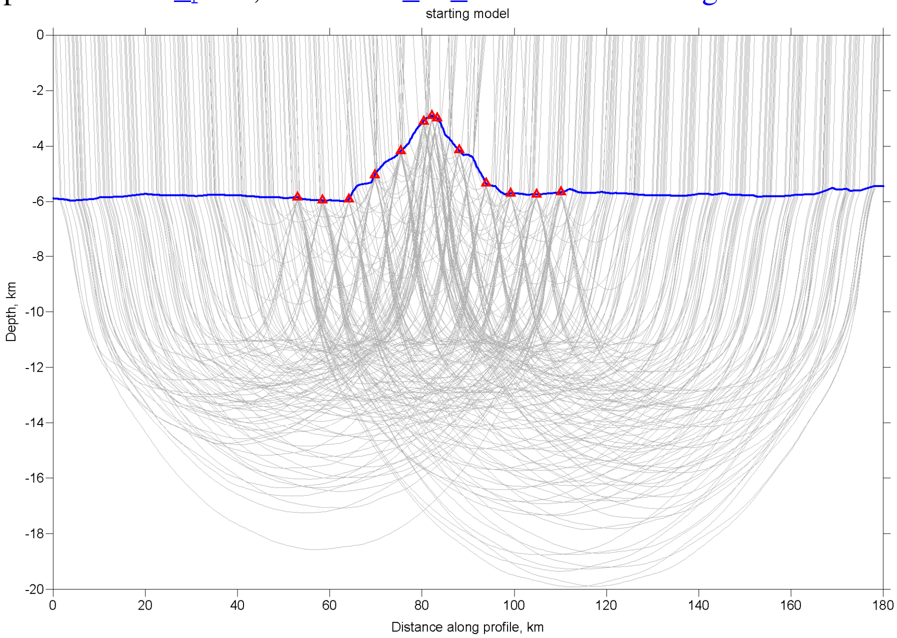


Figure 3.2. Ray paths (grey lines) after performing of ray tracing in iteration 1 (file /FIG_files/rays/rays1.bln).
Red triangles are the stations (file /DATA/HULA_p02/inidata/stat.dat).
Blue line is the sea bottom (file /DATA/HULA_p02/inidata/bottom.bln).

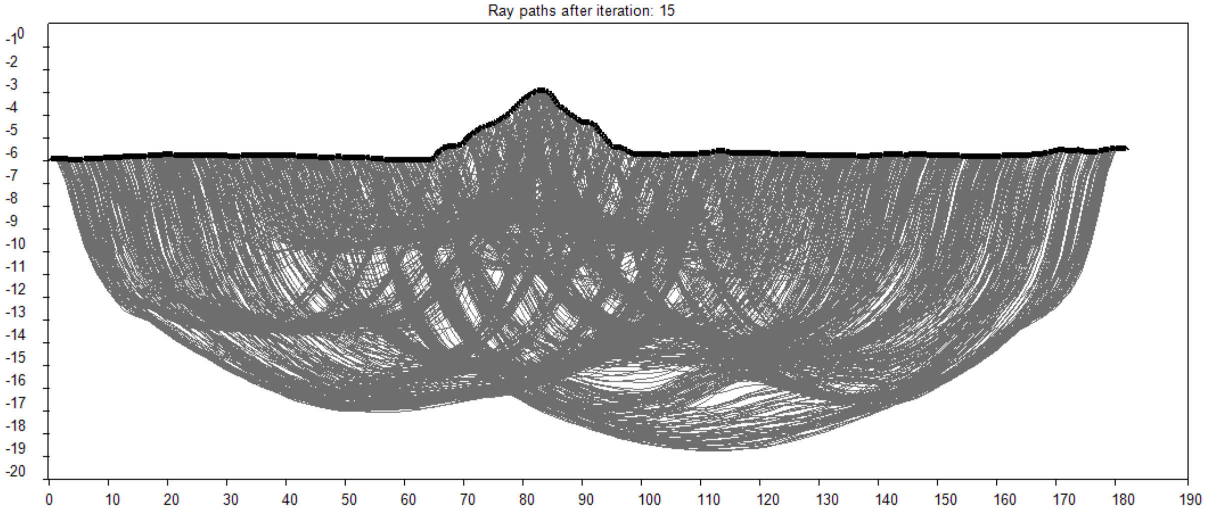


Figure 3.3. Ray paths after performing of ray tracing in iteration 15
(file: PICS\HUL1_p02\R1_V1_A0\rays15.png)

3.2. Construction of the parameterization grid:

Executed Project:
`\PROG\3_grid\`

The calculations are controlled by parameters in the file:
`/DATA/HULA_p02/R2_V5_A0/MAJOR_PARAM.DAT`

```
********************************************************
Parameters for grid construction
********************************************************
GRID_PARAMETERS:
0 180 2.            x1,x2,dx for ray density calculation
1 -20 -0.5          y1,y2,dy for ray density calculation
5                   divide the segments in smaller parts
1                   ! Grid type: 1: nodes, 2: blocks
0.5                 dy_min: min distance between nodes in y. direction
0.001 3.0     !plotmin, plotmax= maximal ray density, relative to average
0.01                !dy= step of movement along y
0 kod_surf: if not 0, allow sharp contrast on interfaces
```

Selected are the most important parameters, which determine the vertical and horizontal spacing of the grid. The other parameters are less important.

Description of the main principle of grid construction

We define the 2D velocity distribution using the node parameterization, which was previously developed for 3D tomographic inversion using the *LOTOS-07* code (Koulakov et al., 2007; Koulakov, 2008). The values of velocity anomalies are interpolated bilinearly between the nodes. The nodes are defined in a set of vertical lines with a fixed predefined spacing (between **x1** and **x2** with step **dx**). Along each line (between **y1** and **y2** with step **dy**) we compute the values of the ray density (summary ray length in a unit volume). The nodes then are distributed according to the ray density. To avoid excessive node fluctuations, we define the minimal spacing between the nodes in the vertical direction (**dy_min**). In areas with a lower ray density the distance between nodes is larger. We do not install the nodes in areas where the ray density is less than a predefined value (**plotmin**). It should be noted that for wide-angle observations, the node spacing in horizontal and vertical directions is not equivalent (e.g. **dx=2 km** and **dy_min=0.5 km**) as we expect a different vertical and horizontal resolution. Figure 3.4 presents an example of node distributions according to the ray paths. The nodes are installed only in the first iteration according to the ray distribution traced in the starting model. During later iterations, velocity variations are updated based on the same nodes. The nodes are previewed automatically as PNG files as shown in Figure 3.5.

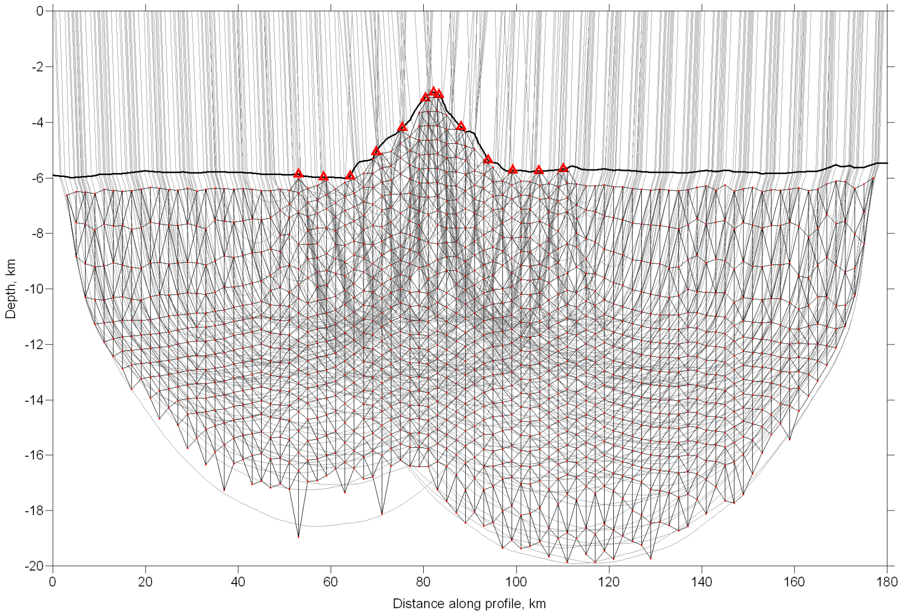


Figure 3.4. Ray paths and parameterization grid. The rays (grey lines) and stations (red triangles) are the same as in Figure 3.2. Parameterization nodes are shown with red dots
(file: /FIG_files/rays/nodes.dat).
Segments indicate links between neighboring nodes which are used for smoothing during inversion
(file: /FIG_files/rays/otr.bln).

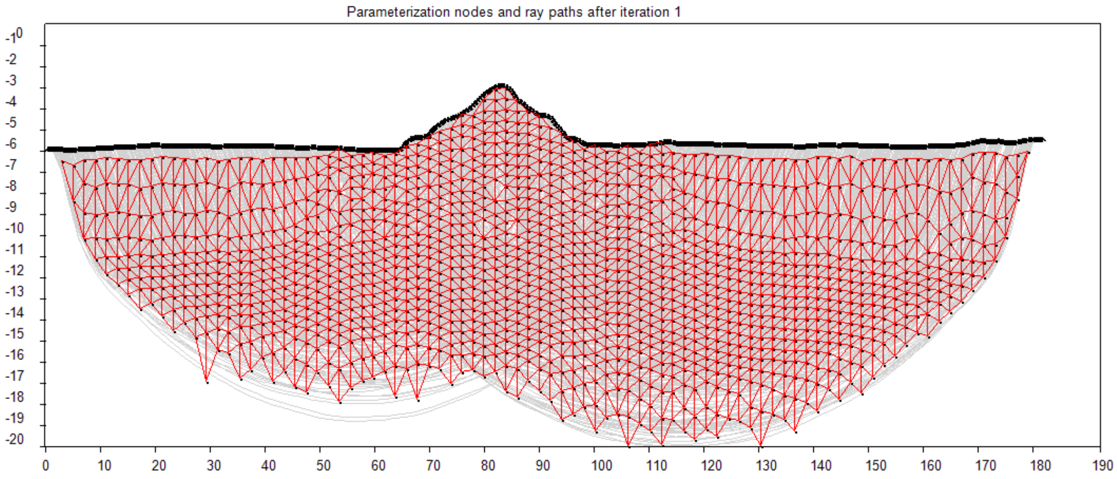


Figure 3.5. Example of PNG file with ray paths and parameterization grid.
(file: PICS\HUL1_p02\R1_V1_A0\nodes_rays 1.png).

3.3. Calculation of matrix:

Project: \PROG\4_matr\

Matrix calculation, is performed along the rays obtained after the ray tracing described in Section 3.1. The effect of velocity variation at each node on the travel time of each ray (∂t/∂V) is computed numerically, as in Koulakov et al., 2006. The data vector corresponding to this matrix consists of residuals obtained after the step of ray tracing.

3.4. Inversion:

Project \PROG\5_invers\
The parameters for the inversion are contained in the file:

/DATA/HULA_p02/R2_V5_A0/MAJOR_PARAM.DAT

```
********************************************************
Inversion parameters
********************************************************
INVERSION PARAMETERS:
80          LSQR iterations
5.          SM, smoothing
10.0        AM, amplitude regularization
```

Inversion of the entire sparse *A* matrix is performed using an iterative LSQR code (Page, Saunders, 1982, Van der Sluis, van der Vorst, 1987). Number of iterations for inversion is **LSQR**. Amplitude and smoothness of the solution is controlled by two additional blocks. The first block is a diagonal matrix with only one element in each line and zero in the data vector. Increasing the weight of this block, **AM**, causes a reduction of the amplitude of the derived velocity anomalies. The second block controls the smoothing of the solution. Each line of this block contains two equal nonzero elements of opposite signs, which correspond to all combinations of neighboring nodes in the parameterization grid. The data vector in this block is also zero. Increasing the weight of this block, **SM**, causes a reduction of the difference between solutions in neighboring nodes, which results in smoothing of the computed velocity fields.

The optimum values of these parameters depend on several factors. For example, when increasing the data amount, the damping parameters should be increased, while in the case of increasing the numbers of nodes due to smaller spacing, the damping should be decreased. In the case of larger noise level in the data, damping should be stronger to keep the solution stable. The process of finding the damping coefficients is not formalized yet. The relationships between the number of parameters, rays and values of amplitude and smoothing coefficients are not linear. For example, when the number of rays doubles, the same amplitude of the solution is obtained by increasing the damping coefficient to 1.2. For each dataset, these values should be identified individually using a variation of trials. The first hint for finding the damping weights is considering the evolution of RMS residuals contingent on iterations. When damping is not sufficient, the amplitude of the solution becomes too strong. As a result, deviations of rays with respect to the previous iteration are too strong. In this case, tracing in the next iteration might lead to a non-improved solution. On the other hand, over-damping might provide a too large final RMS. The role of damping parameters is illustrated in the next section. An alternative method for determining the optimal values of damping parameters is synthetic modeling.

In some cases, the times of shots are given with uncertainty. In this case it is recommended to introduce time corrections related to sources (source corrections) which are determined as independent unknown parameters. In this case we should include an optional rubric to MAJOR_PARAM.DAT:

/DATA/HULA_p02/R2_V5_A0/MAJOR_PARAM.DAT

```
   ********************************************************
   Parameters for source correction (optional)
   ********************************************************
   SOURCE CORRECTION:
   1.0 W_srce: weight for source correction in inversion
```

If this rubric does not exist or if it contains zero value, source corrections are not computed.

3.5. Calculation of 2D model in a regular grid

Project `\PROG\6_2dmodel\`

The velocity anomalies obtained after inversion are re-computed in a regular grid and added to the velocity model obtained during the previous iteration. Regular representation of the velocity field is more convenient for performing the ray tracing in the next iteration. Parameters of the calculation are defined in

`/DATA/HULA_p02/R2_V5_A0/MAJOR_PARAM.DAT`

```
********************************************************
Parameters for 2d regilar grid
********************************************************
2D GRID PARAMETERS:
0 180 2.          x1,x2,dx limits and step in horizontal direction
 1 -20 -0.2       y1,y2,dy limits and step in horizontal direction
5 0.7             dxmax, dymax: max distance to node in X and Y directions
```

Limits of the volume for interpolation and grid spacing along X, and Y are defined in the first two lines (**x1,x2,dx**) and (**y1,y2,dy**). **dxmax, dymax** define the maximal distance to the nearest node of the parameterization grids along X and Y directions. If the distance is larger, this point lies outside the resolved area and the value there is presumed 0.

4. Presentation of the results

4.1. Visualization tool for previewing

The *PROFIT* code contains a tool for automatic visualization of the results after each iteration. The images are created as PNG bitmap files and stored in a special folder.

NOTE! Prompt work of the visualization tools requires installing dotNetFramework (dotnetfx.exe). In most Windows operation systems it is installed *a-priori*.

Visualization is performed using a simple program which is written in C-sharp. The executable file is located in **\COMMON\visual_exe\visual.exe.**
This EXE file can be moved to any location and renamed.

The program contains three major tools which are required for visualization:

- imaging 2D fields using colored contour lines (GRD format);
- drawing polylines (BLN format);
- drawing dots (DAT format) either as circles or squares.

The input files are of the same format as used for SURFER (GDR, BLN and DAT). This program can visualize any order of layers with one of theses three information sources. The format of the layers is defined in file **\COMMON\visual_exe\config.txt,** which should be located in the same directory as the EXE file. Example of this file is presented below:

COMMON\visual_exe\config.txt

```
400 600
_______ Size of the picture in pixels (nx,ny)
-72.50000      -69.50000
_______ Physical coordinates along X (xmin,xmax)
-22.50000      -18.50000
_______ Physical coordinates along Y (ymin,ymax)
1 1
_______ Spacing of ticks on axes (dx,dy)
PICS\dv15 3.png
_______ Path of the output picture
 P anomalies, depth=   30 km
_______ Title of the plot on the upper axe
4
_______ Number of layers
********************************************
1
_______ Key of the layer (1: contour, 2: line, 3:dots)
DATA\dv15 3.grd
_______ Location of the GRD file:
SCALES\blue_red.scl
_______ Scale for visualization
-10 10
_______ scale diapason:
********************************************
2
_______ Key of the layer (1: contour, 2: line, 3:dots)
DATA\coastal_line.bln
_______ Location of the BLN file
2
_______ Thickness of line in pixels
0 130 255
_______ RGB color:
********************************************
3
_______ Key of the layer (1: contour, 2: line, 3:dots)
```

```
DATA\ztr 1.dat
________ Location of the DAT file
2
________ Symbol (1: circle, 2: square)
10
________ Size of dots in pixels
0 0 0
________ RGB color:
```

This file example contains four data groups. The 1st group contains general information about the plot: size of the plot in pixels, physical coordinates, properties of axes, name of the PNG file, title of the plot.

The next four groups contain information about different layers (from back to front). In this example, the GRD, BLN and DAT files are taken from DATA subfolder. Scale is taken from SCALES subfolder. The output picture is written to PICS subfolder.

The derived image is presented below:

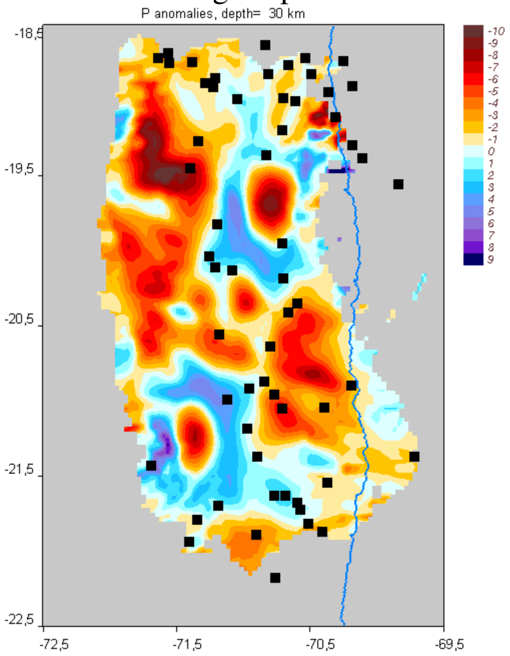


Figure 4.1. Resulting image (\COMMON\visual_exe\PICS\dv15 3.png) obtained as a result of running the file \COMMON\visual_exe\visual.exe using the configuration from \COMMON\visual_exe\config.txt.

Drawing 2D functions (key 1) requires using the color scales with indicated path (e.g. ..\..\COMMON\scales_scl\blue_red.scl). This file contains three columns which correspond to RGB coding. The first line can be ignored. For example:

COMMON\scales_scl/blue_red.scl

```
-1 1
102 51 51
129 24 24
159 0 0
208 0 0
255 3 0
255 64 0
255 117 0
255 157 0
255 196 0
```

```
255 235 158
222 255 255
156 255 255
60 224 255
25 192 255
89 160 255
119 136 238
141 114 216
141 77 204
112 19 204
0 0 102
```

The example in **\COMMON\visual_exe** can be used for immediate control of the visualization tool.

4.2. Preview of the intermediate and final results as bitmap images in PNG files

Final and intermediate results are visualized automatically and stored in the folder PICS in corresponding subfolders. In order to activate this option, the file preview_key.txt in the root directory should contain only one number (1 or any other nonzero integer number). In case of absence of this file, or if it contains 0, previewing is not performed.

The parameters of the previewing are defined in the file config.txt:

/DATA/HUL1_p02/config.txt

```
1000 400    npix_X,npix_Y, size of the picture in pixels
10 1  Ticks on the axes along X and Y
blue_red.scl
-20 20      Amplitudes of anomalies
rainbow_small.scl
4 8.1       Absolute velocity range
************* to draw travel times
1000 800    npix_X,npix_Y, size of the picture in pixels
20 5        Ticks on the axes along X and Y
3     MAX Distance along X to draw lines
1     way of picking: 1- fixed st, many shots; 2- fixed shot, many stations
```

The color scales should exist in the folder /COMMON/scales_scl.

The main steps of performing the data inversion can be seen in the PNG files produced in folder /PICS/'//ar//'/'//md//'/

Below are the main files which are produced:

1. Ray paths after tracing in each iteration:

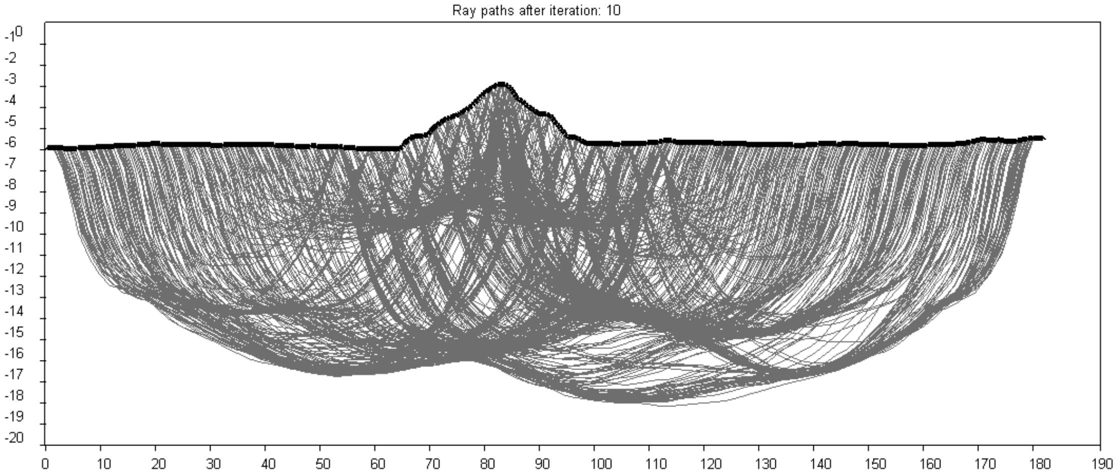


Figure 4.2. Picture in file `/PICS/HUL1_p02/R1_V1_A0/rays10.png,` which corresponds to ray paths in the 10-th iteration.

2. Parameterization grid constructed according to the ray density.

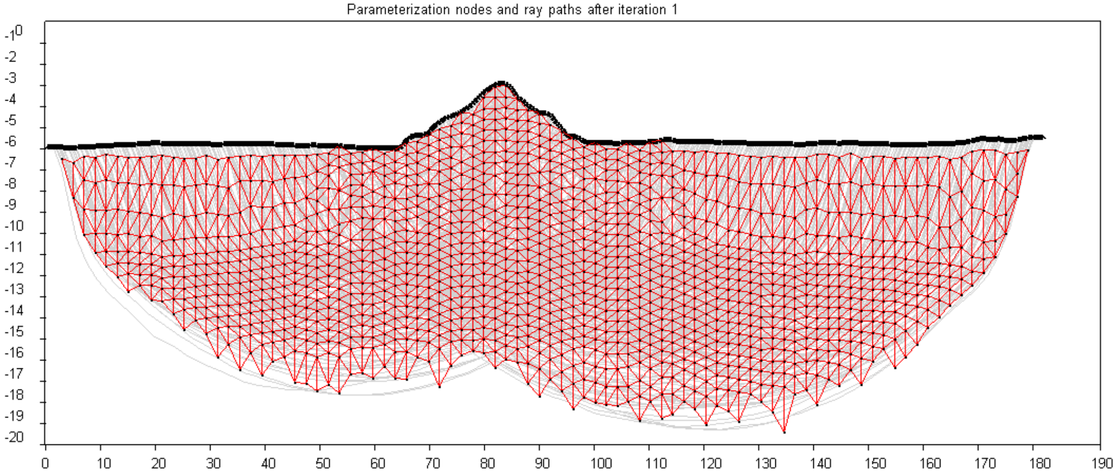


Figure 4.3. Picture in file `/PICS/HUL1_p02/R1_V1_A0/nodes_rays1.png,` which shows the parameterization grid constructed according to the ray density in the 1st iteration.

3. Starting velocity distribution and inversion results presented as absolute velocities and relative anomalies with respect to the starting model are shown in Figures 4.4-4.6.

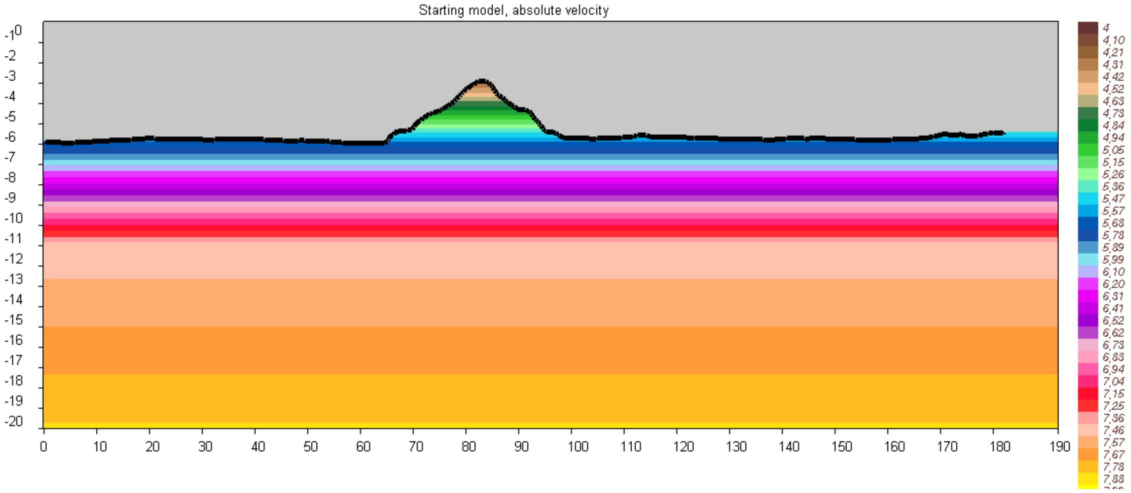


Figure 4.4. Picture in file /PICS/HUL1_p02/R1_V1_A0/start_abs.png, which corresponds to the starting velocity distribution in model R2_V5_A0.

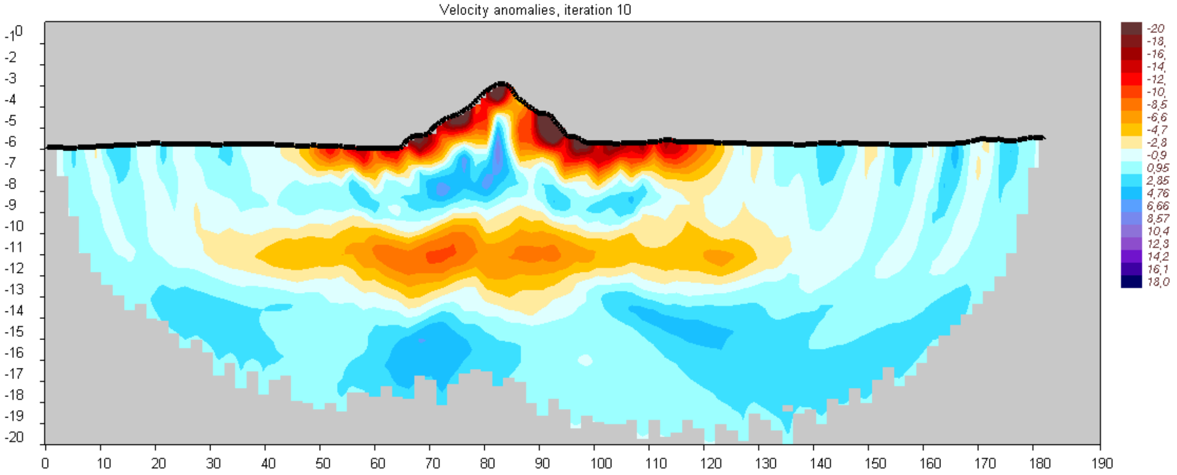


Figure 4.5. Picture in file /PICS/HUL1_p02/R1_V1_A0/anom10.png, which corresponds to resulting velocity anomalies after the 10th iteration.

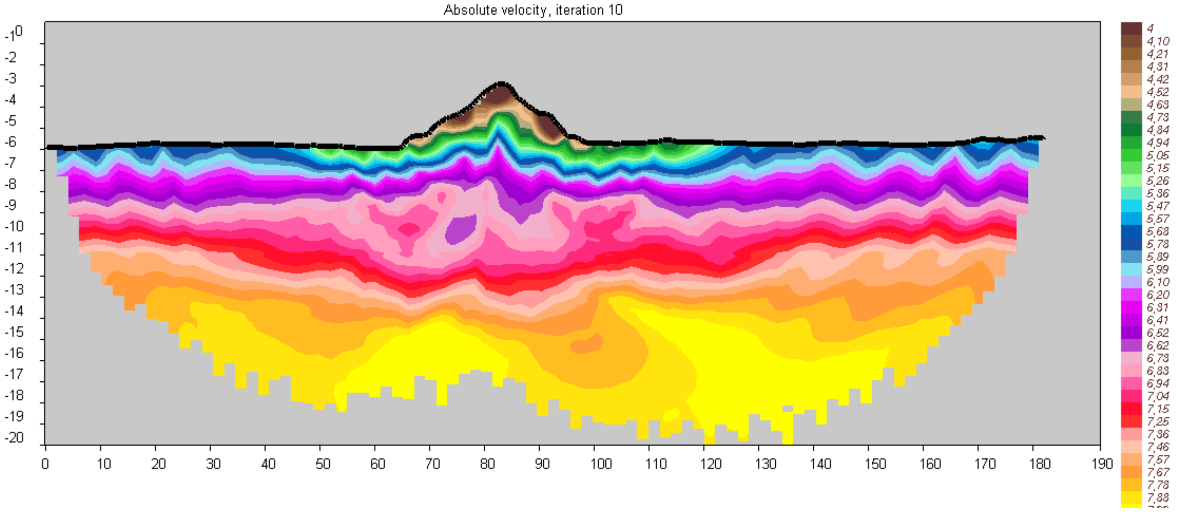


Figure 4.6. Picture in file /PICS/HUL1_p02/R1_V1_A0/v_abs10.png, which corresponds to resulting absolute velocities after the 10th iteration.

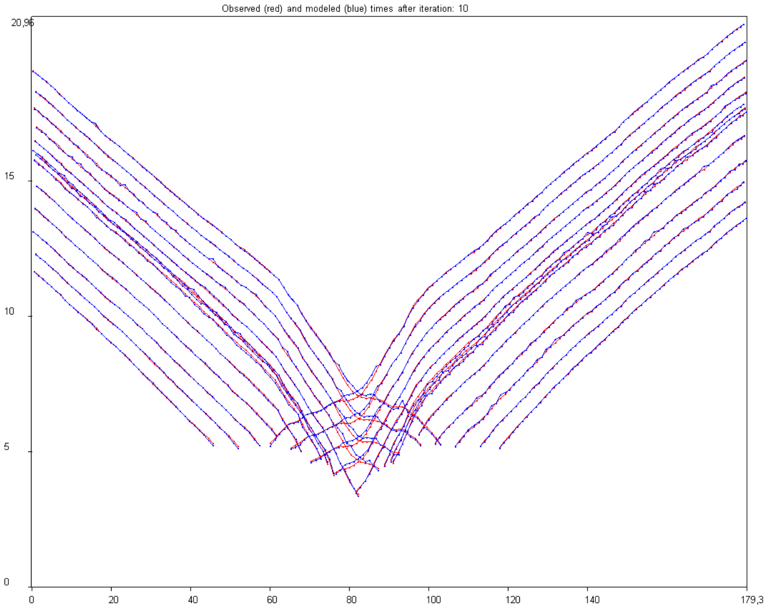


Figure 4.7. Picture in file /PICS/HUL1_p02/R1_V1_A0/times10.png, which corresponds to the observed (red) and computed (blue) travel times after the 10th iteration.

We recommend using these tools just for previewing. Publication quality files should be created in Surfer or other commercial visualization tools. The files with corresponding formats are stored to FIG_files folders (overwritten in a case of running a new model).

4.3. Visualization of the resulting anomalies and absolute velocities

The program which produces the GRD files of the resulting velocity anomalies and absolute velocity is located in the Project: /PROG/vis_result.

The general information about the visualized model is presented in file /model.dat

```
HULA_p02     name of the area
R2_V5_A0     name of the model
1            Iteration
```

Other parameters for visualization are defined in the file presented below:
Location: \DATA\HUL1_p02\set_vis.dat

```
0 190 1.           x1,x2,dx for ray density calculation
-20 0 0.1          y1,y2,dy for ray density calculation
2 1.0              dxmax, dymax: max distance to node
```

In Figure 4.8, visualization is performed for the model model_02 from the area “R1_V1_A0” iterations 1, 4 and 9 according to the parameters indicated above. Limits of the volume for interpolation and grid spacing along X and Y are defined in the first two lines (**x1,x2,dx**) and (**y1,y2,dy**). **dxmax, dymax** denote the maximum distance to the nearest node of the parameterization grids along X and Y directions. If the distance is larger, this point lies outside the resolved area and the value there is presumed -999.

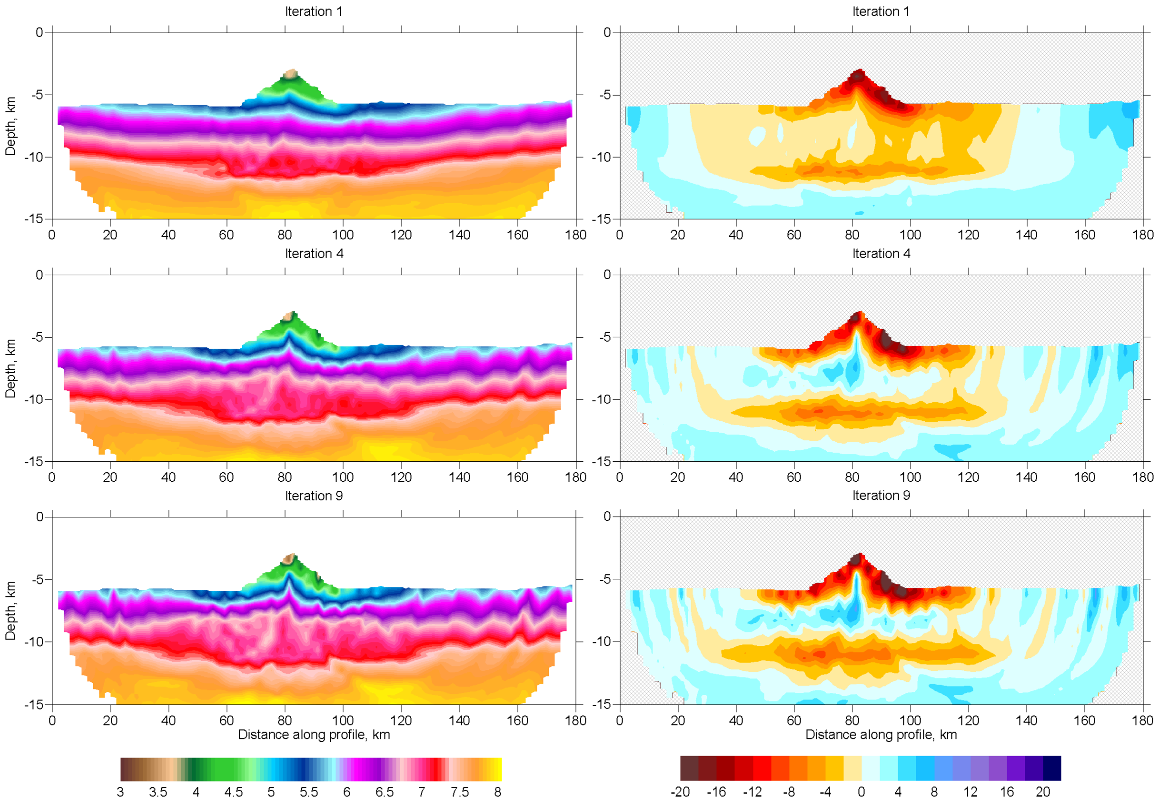


Figure 4.8 Resulting velocity anomalies and absolute velocities obtained for the area "`HULA_p02`", model "`model_02`" and iterations 1, 4 and 9.

The output of this program:

`/FIG_files/result/dv_result'//it//'.grd`: relative anomalies in percent computed according to the information in "`set_vis.dat`". This file can be directly visualized in the SURFER Software as a contour line plot.
`/FIG_files/result/v_abs'//it//'.grd`: absolute resulting velocities.

4.4. Report about variance reduction

The program which produces a report about deviation of time residuals and variance reduction can be found in Project:
`/PROG/var_reduct.`

The general information about the visualized model is defined in file `/model.dat`

```
HULA_p02      area
R1_V1_A0      model
9             Iterations
```

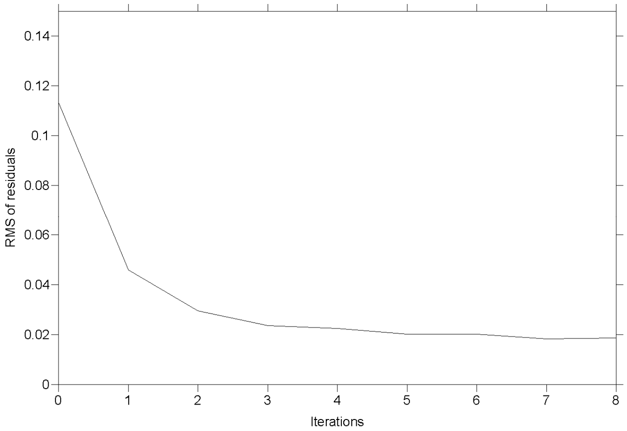


Figure 4.9. Value of variance reduction (area “HULA_p02”, model “R1_V1_A0”).

The report is given for all iterations from the 1st one to the iteration indicated in file “`model.dat`”.

The output of this program is displayed on the console as:

```
 iter=           1  disp=  0.1133814      red=  0.0000000E+00
 iter=           2  disp=  4.5967747E-02  red=   59.45742
 iter=           3  disp=  2.9482935E-02  red=   73.99668
 iter=           4  disp=  2.3665704E-02  red=   79.12735
 iter=           5  disp=  2.2531653E-02  red=   80.12756
 iter=           6  disp=  2.0127479E-02  red=   82.24799
 iter=           7  disp=  2.0253871E-02  red=   82.13651
 iter=           8  disp=  1.8275457E-02  red=   83.88144
 iter=           9  disp=  1.8597504E-02  red=   83.59740
```

A curve of the variance reduction (Figure 4.9) can be created based on the file: `/FIG_files/stat/HULA_p02&R1_V1_A0.bln.`

In addition a plot with observed and modeled travel times shown in Figure 4.7 is created by this program.

5. Velocity model definition

5.1. General remarks

All the programs are designed for a 2D case. Here we consider only defining the isotropic velocity models, although anisotropic version is also available. In all indications we use *X*-*Y* coordinates. *X* is lateral; *Y* is vertical (downward is negative). For description of starting and synthetic models, the same algorithm of velocity definition is used. The velocity model is determined as a superposition of basic velocity distribution and relative anomalies, which are defined in files "`velocity_start.dat`" and "`anomaly_start.dat`" (for starting model) and "`velocity_syn.dat`" and "`anomaly_syn.dat`" (for synthetic model). Each of these files presumes several options which are described below.

5.2. Basic absolute velocity definition

The basic absolute velocity distribution is defined in files "`velocity_start.dat`" and "`velocity_syn.dat`". There are several options for definition of the basic velocity, which depend on the index in the first line of the file according to Table 1:

Table 1. Options of basic velocity definition according to key index (first line in "`velocity_start.dat`" and "`velocity_syn.dat`")

Key Index	Definition
1	1D velocity distribution along the Y coordinate
2	Velocity definition with free-shaped polygons
3	Velocity definition as 2D matrix in a regular grid
4	Velocity definition with interfaces
5	Velocity definition with constant gradient between interfaces (method 1)
6	Velocity definition with constant gradient between interfaces (method 2)

More details about each of these cases are given below:

5.2.1. INDEX "1": 1D velocity distribution along the Y coordinate

The basic velocity is 1D along Y coordinate. An example is given below:

File 1: Example of "`velocity_start.dat`" for the case of the 1D basic velocity:

```
1       INDEX of Abs velocity. If 1: 1D velocity
********************************************
8               number of depth levels
-0 2400         y, v
-300 2500       y, v
-300 2700       y, v
-700 2700       y, v
-700 2500       y, v
-1200 2700      y, v
```

```
-1250 2800     y, v
-2000 3000     y, v
```

The velocity is defined at several depth levels (top to bottom); between these levels, the velocity is interpolated linearly. Above the first level, the velocity is constant (e.g. 2400 in our example). Below the last level the velocity is continued linearly according to the velocity gradient in the last segment. If necessary, sharp velocity jumps are allowed (as at 300 and 700 m depth).

5.2.2. INDEX "2": Velocity definition with free-shaped polygons

The basic velocity is defined with free-shaped polygons. An example is given below:

File 2: Example of "FREE_000/velocity_syn.dat" for the case of definition with free-shaped polygons:

```
2       2 - definition with polygons

2350    background velocity
4               number of patterns
*******************************
lay_4   Figure
0 -1500 3050   x1,y1,v1
0 -2000 3050   x2,y2,v2
*******************************
lay_3   Figure
0 -1500 2900   x1,y1,v1
0 -2000 2900   x2,y2,v2
*******************************
lay_2   Figure
0 -1500 2750   x1,y1,v1
0 -2000 2750   x2,y2,v2
*******************************
lay_1   Figure
0 -1500 2600   x1,y1,v1
0 -2000 2600   x2,y2,v2
```

First line (brown) is the index, which indicates the type of the velocity definition. "2" means that the following format corresponds to description with polygons. The background velocity is indicated in "blue". Velocity in the points, which are outside all defined polygons is equal to this background value (2350 m/s, in this example). Number of patterns is defined in the "violet" line. In this example, value "4" means that there are four patterns. Description of polygons is contained in the following blocks separated with stars (or any other symbols). Each block consists of three lines. First line in the pattern block (red) indicates the name of file with polygon coordinates which is located in a subfolder "forms". In these examples, the basic model is defined with four polygons indicated with different colors: "*lay_1*", "*lay_2*", "*lay_3*", "*lay_4*". Second and third lines (green) define the velocity within the polygon. The program allows defining inclined velocity gradients inside each polygon. x1, y1, v1 and x2, y2, v2 are coordinates and velocity values in two points, which determine constant velocity gradients. In all presented cases, the velocity is constant, so the coordinates are not important. In case of overlapping the figures, the polygon, which appear first is placed above the later one. For example, "*lay_4*" is above the "*lay_3*" figure.

The polygons are defined as *.bln* files. It is recommended to use *SURFER software* for digitizing (select the plot, then "Menu->Map->Digitize"). The digitized file should be saved to “forms” folder. The name of the file should contain five characters. An example of polygon is presented below:

File 3: Example of “FREE_000/forms/lay_4.bln” for the case of definition with polygons:

```
18,1
-1358.983185, -2093.259815
-1275.46989, -1835.128505
-1146.40487, -1668.103185
-986.970975, -1645.327005
-880.6815, -1797.168205
-736.43236, -1986.97034
-584.590525, -1546.62959
-326.45985, -1941.41798
-174.618015, -1599.77401
98.69678, -1842.720565
303.683035, -1652.919065
645.32637, -1873.08944
812.352325, -1576.99783
1085.66712, -1850.312625
1024.93064, -2275.469255
220.16974, -2526.00787
-690.88, -2510.82375
-1176.77311, -2389.35079
```

File 2 corresponds to the model “FREE_000” and is shown in Figure 5.1.

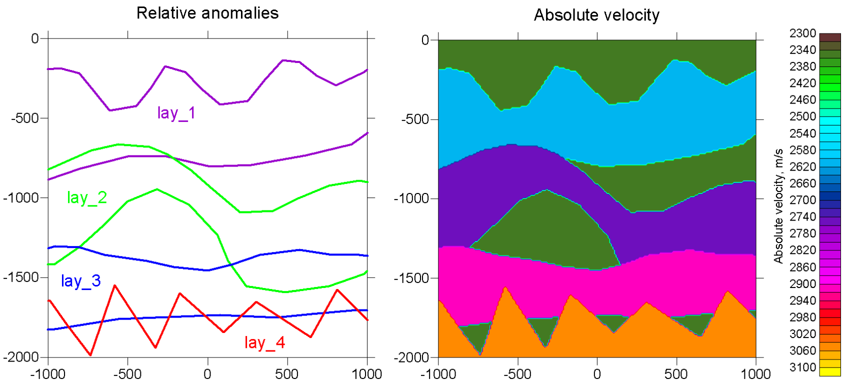


Figure 5.1. Scheme for definition of absolute velocity model polygons (Model: “FREE_000”). Left is the scheme of polygons; right is the resulting absolute velocity model.

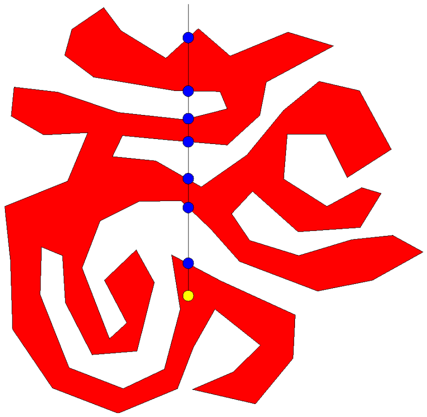


Figure 5.2. A scheme for explaining the algorithm for determining the points inside the polygon. Yellow dot is the current point to be investigated. Blue points are intersection points with the borders of the polygon above the current point. If number of these points is odd (1, 3, 5 etc), the current point is located inside the polygon.

The algorithm for determining the location of a point inside or outside the current polygon is illustrated in Figure 5.2. In case of any complexity figure, we move from a current point (yellow point) along a fixed direction (e.g. upward, as shown in the Figure) and count the number of the intersection points with the polygon (blue points). In case of odd numbers (1,3,5,7 etc) the current point is located inside the polygon.

5.2.3. INDEX “3”: Velocity definition as 2D matrix in a regular grid

The basic velocity model is defined by a 2D matrix. In this case, the file “`abs_vel_syn.dat`” contains only one line with index “3”. The matrix is taken from the file “`forms/vel_2D.dat`”, which is in binary format and can be read by the following program block:

```
Program block 1:

open(2,file='../../data/'//md//'/forms/vel_2D.dat',form='binary')
read(2)xmin,xmax,dx,nx
read(2)ymin,ymax,dy,ny
do ix=1,nx
      do iy=ny,1,-1
            read(2)vel_2d(ix,iy)
      end do
end do
close(2)
```

where “**md”** is the name of the model (8 characters)
xmin, xmax are left and right sides of the defined area;
dx, and **nx** are spacing and number of nodes along the X coordinate;
ymin, ymax are upper and lower sides of the defined area;
dy, and **ny** are spacing (normally, negative) and number of nodes along the Y coordinate.

In this case, the velocity value at any point of the defined area is determined by four matrix values and computed by bi-linear interpolation.

5.2.4. INDEX “4”: Velocity definition with interfaces

This method allows definition of separate velocity distributions in different polygons. The polygons are parameterized using a set of lines as shown in Figure 4. In this example six lines are defined. Yellow contour is composed of four segments corresponding to lines 1-2-4-3 (in this example the order is clockwise. However, counterclockwise order is also allowed). The algorithm allows also the definition as shown for the pink area where one line is mentioned two times: 2-5-2-3-6. If two lines do not intersect with each other, the program constructs their prolongation (for example, lines 1 and 2, for yellow polygon).

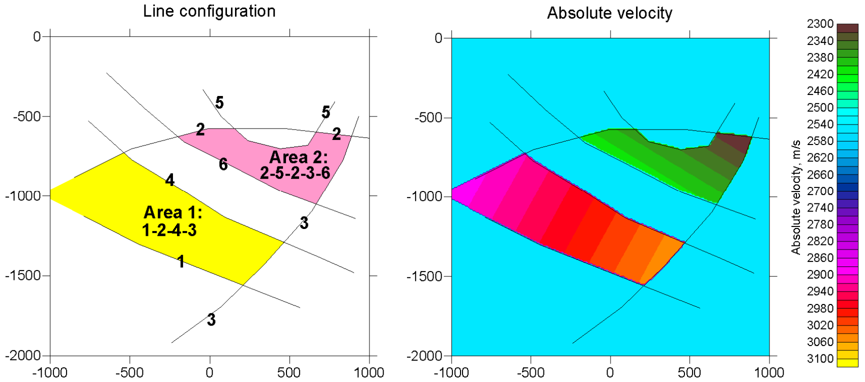


Figure 5.3. Scheme for definition of polygons using separate lines (Model: `LINE_000`). Left is the scheme of lines; right is the resulting absolute velocity model.

The lines are defined as .bln files. It is recommended to use SURFER software for digitizing (select the plot, then “Menu->Map->Digitize”). The digitized file should be saved to “forms” folder. The name of the file should contain five characters. An example of line is presented below:

File 4: Example of “`LINE_000/forms/line1.bln`” for the case of definition with free-shaped lines:

```
4,1
-789.371161883, -1126.45871423
-449.359745056, -1300.23443837
49.3240033379, -1496.67647888
563.119974137, -1700.67395818
```

In the first line, “4” means number of nodes. Second number is not important. Then 4 lines follow with X and Y pairs.

An example which corresponds to Figure 5.3 (Model “`LINE_000`”) is defined using the following file:

File 5: Example of “`LINE_000/abs_vel_syn.dat`” for the case of definition with free-shaped lines:

```
4       4 - definition with lines
____________________________________
2550 background velocity
2       number of patterns
***************************** layer 2 *
4       number of lines
line1
line2
line4
line3
-766.8006, -832.974045 2900          x1,y1,v1
326.459215, -1432.74869 3050         x2,y2,v2
***************************** layer 2 *
5       number of lines
line2
line5
line2
line3
line6
789.576145, -597.61955 2300          x1,y1,v1
546.62959, -1000 2400                x2,y2,v2
```

First line (brown) is the index which indicates the type of the velocity definition. “4” means that the following format corresponds to description with lines. The background velocity is indicated in “blue”. Velocity in the points which are outside all defined areas is equal to this background value (2550 m/s, in this example). Number of areas is defined in the “violet” line. In this example, value “2” means that there are two areas. Description of areas is contained in the following blocks separated with stars (or any other symbols). Each block contains list of lines indicated with red which surround the current area. The coordinates of lines are located in the subfolder “forms”. Green lines define the velocity within the area. The program allows defining inclined velocity gradients inside each area. x1, y1, v1 and x2, y2, v2 are coordinates and velocity values in two points which determine constant velocity gradient.

An example of creating a realistic synthetic model using this algorithm for the considered dataset (HULA_p02) and model `S1_V4_A0` is shown in Figure 5.4. In this case the basic velocity is defined with the key 4 and anomalies are zero.

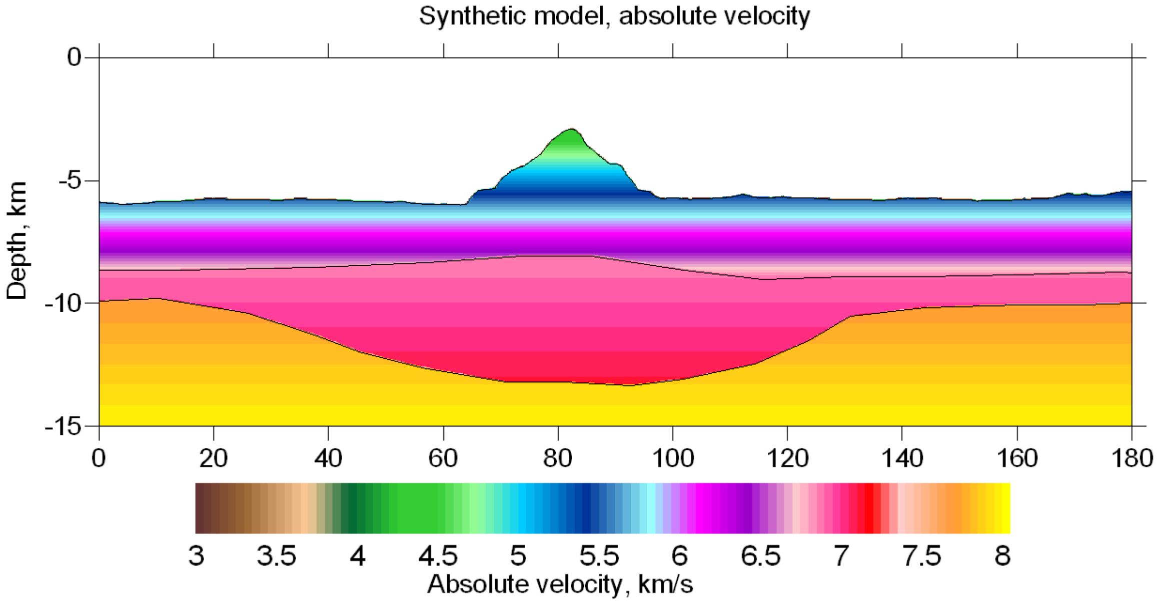


Figure 5.4. Absolute velocity model constructed using file `\DATA\HULA_p02\S1_V4_A0\velocity_syn.dat` and zero anomalies.

5.2.5. INDEX “5”: Velocity definition by vertical gradient between interfaces (method 1)

In this method we define several interfaces of free shape. They should be located one below another, and they should not intersect each other. In each interface we define a fixed velocity. Between the interfaces, the velocity is computed by linear interpolation between velocity values in the nearest levels. The nodes on the interfaces are located one below another.

An example which corresponds to Figure 5.5 (Model “`R1_V5_A0`”) is defined using the following file:

File 6: Example of "R1_V5_A0/velocity_start" for the case of definition with vertical gradients between lines:

```
5       5 - definition with vertical gradients between interfaces (method 1)
____________________________________
5              number of vertical lines
3              number of level (nlev)
4.0 7.7 8.2
***************************** line 1 *
0              xvert, X-coordinate
-3 -11 -40  yvert_i, Y coordinates in the vertical line with X=0
***************************** line 2 *
50             xvert, X-coordinate
-3 -11 -40  yvert_i, Y coordinates in the vertical line with X=50
***************************** line 3 *
80             xvert, X-coordinate
-2 -12 -40  yvert_i, Y coordinates in the vertical line with X=80
***************************** line 4 *
100            xvert, X-coordinate
-3 -11 -40  yvert_i, Y coordinates in the vertical line with X=100
***************************** line 5 *
200            xvert, X-coordinate
-3 -11 -40  yvert_i, Y coordinates in the vertical line with X=200
```

First line (brown) is the index which indicates the type of the velocity definition. "5" is a key which means that the following format corresponds to description with vertical gradients between interfaces. The interfaces are defined in nodes with the same X coordinates. Number of values along X is indicated in the "blue" line (5). Number of interfaces is fixed in the "magenta" line (3). Velocity on each interface is fixed in the "green" line. Then, for each X coordinate (5 times in this case) we define the value of X ("black" lines) and depths of the interfaces ("red" lines). The resulting model, corresponding to this definition is shown in Figure 5.5.

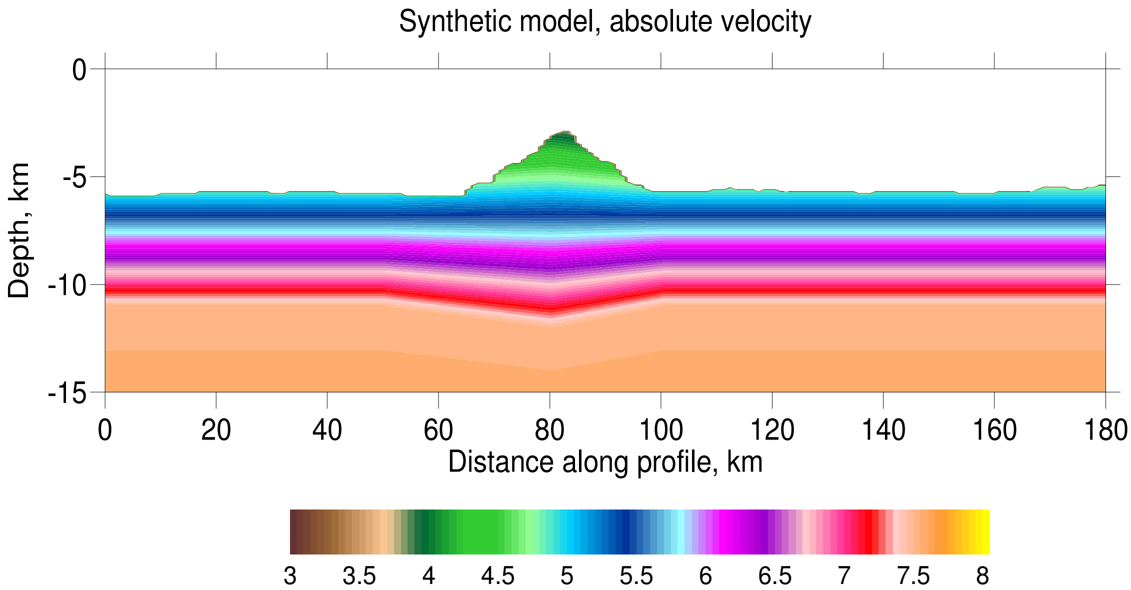


Figure 5.5. Absolute velocity model constructed using file \DATA\HULA_p02\R1_V5_A0\velocity_start.dat and zero anomalies.

5.2.6. INDEX "6": Velocity definition by vertical gradient between interfaces (method 2)

In this method we define several interfaces of free shape. They should be located one below another, and they should not intersect each other. In each interface we define a fixed velocity. Between the interfaces, the velocity is computed by linear interpolation between velocity values in the nearest levels.

An example which corresponds to Figure 5.5 (Model "`R1_V6_A0`") is defined using the following file:

File 7: Example of "`R1_V6_A0/velocity_start`" for the case of definition with vertical gradients between lines (method 2):

```
6              6: Gradient between the interfaces (method 2)
*****************************
6              number of interfaces
***********************************
relif          Name of the interface
4              Velocity on the interface
***********************************
blue_          Name of the interface
5.8            Velocity on the interface
***********************************
pink_          Name of the interface
6.8            Velocity on the interface
***********************************
red__          Name of the interface
7.6            Velocity on the interface
***********************************
lower          Name of the interface
8.2            Velocity on the interface
```

First line (brown) is the index which indicates the type of the velocity definition. "6" is a key which means that the following format corresponds to description with vertical gradients between interfaces (Method 2). The interfaces are defined in nodes with the same X coordinates. Number of interfaces is indicated in the "blue" line (6). Names of the interfaces from the subfolder "forms" is indicated in the "green" line. Velocities in the interfaces is defined in "red" lines. The resulting model, corresponding to this definition is shown in Figure 5.6.

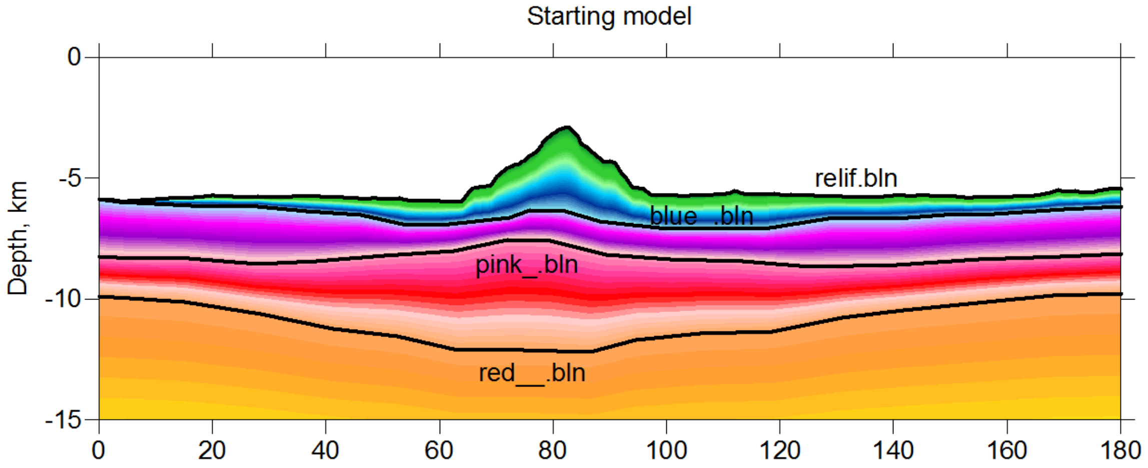


Figure 5.6. Absolute velocity model constructed using file `\DATA\HUL1_p02\R1_V6_A0\velocity_start.dat` and zero anomalies. Names of the used interfaces is shown.

5.3. Definition of velocity anomalies

Velocity anomalies are overlapped onto the basic velocity model and are defined in file “anomaly_start.dat” for the starting model, and “anomaly_start.dat” for the synthetic model. They are given in percent with respect to the velocity value according to the basic model. There are several options for the definition of the anomalies, which depend on the index in the first line of the file according to Table 2:

Table 2. Options for velocity anomalies according to key index (first line in “anomaly_iso.dat”)

Key Index	Definition
0	No anomalies
1	Checkerboard anomalies
2	Velocity anomalies defined with free-shaped polygons
3	Primary and secondary anomalies (e.g. for time-lapse tomography)
4	Velocity anomalies defined with free-shaped polygons in selected areas

More details about each of these cases are given below:

5.3.1. INDEX “0”: No anomalies

In case of “0” in the index line, or in case of absence of the “anomaly_start.dat” or “anomaly_start.dat” file, the velocity model is defined only as basic model with “velocity_start.dat” or “velocity_syn” file.

5.3.2. INDEX “1”: Checkerboard anomalies

Case of “1” means checkerboard anomalies, which represent alternating positive and negative rectangular anomalies. If necessary they can be separated of each other with zero intervals. The parameters of the checkerboard are defined in the file “anomaly_syn.dat”

File 7: Example of “R1_V1_A1/anomaly_syn.dat” for the checkerboard anomalies (index 1):

```
1       1 - board

3.00    Amplitude, %
0. 200. 10. 0  x_left, x_right, dx1, dx2
-15. 0. 2. 0.0 z_lower, z_upper, dz1, dz2
```

First line (brown) is the index which indicates the type of the anomaly (checkerboard).
Third line (blue) is the amplitude of the anomalies (±5%).
4-5 lines (red) define the parameters of the checkerboard:
`x_left, x_right, z_lower, z_upper`: limits of the area where the checkerboard is defined;
`dx1, dz1`: size of the checkerboard blocks;
`dx2, dz2` : empty space between the blocks.

An example of velocity anomalies constructed using File 6 is shown in Figure 5.6.

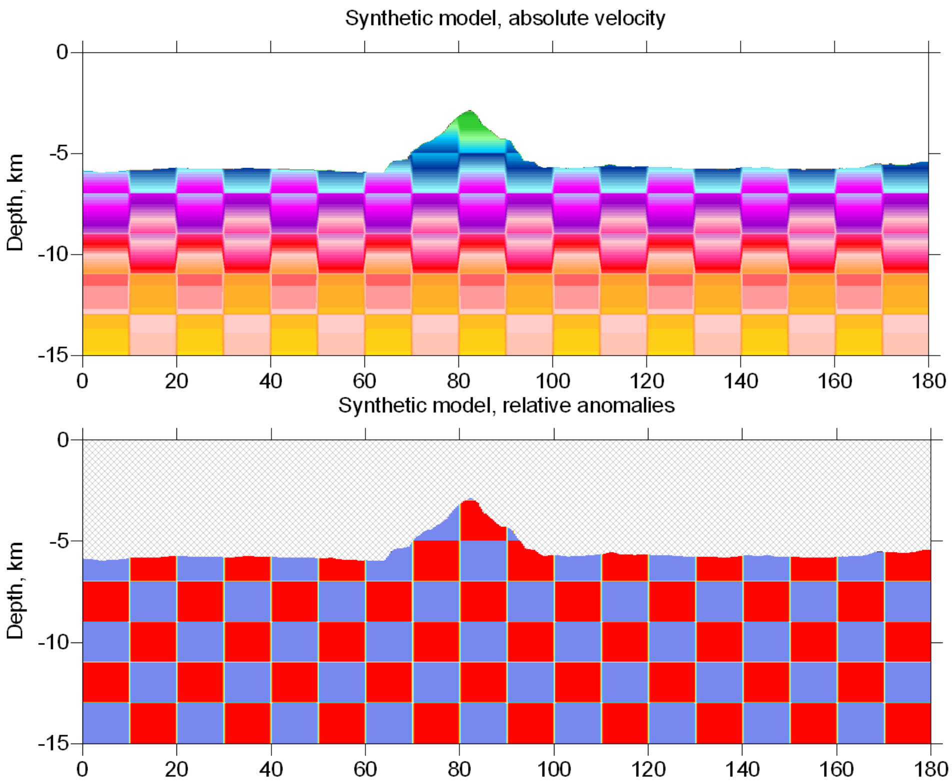


Figure 5.6. Absolute velocity (upper) and anomalies (lower) constructed using files `\DATA\HUL1_p02\S1_V1_A1\velocity_syn.dat` and `\DATA\HUL1_p02\S1_V1_A1\anomaly_syn.dat`.

5.3.3. INDEX "2": Velocity anomalies defined with free-shaped polygons

Case of "2" means that the anomalies are defined with free-shaped polygons. Detecting the points inside the polygon is performed using the same algorithm as described in Section 5.2.3. A set of polygons is defined in the file "`anomaly_syn.dat`" as shown in the following example:

File 8: Example of "1D+FREE1/anomaly_syn.dat" for the free-shaped polygons (index 2):

```
2       1 - board, 2 - free anom
_______________________________________________
2       number of anomalies
*******************************
anom1   Figure
0. 0. 0. 0.
7
*******************************
anom2   Figure
0. 0. 0. 0.
-7
```

First line (brown) is the index, which indicates the type of the anomaly. "2" means that the following format corresponds to description of free-shaped anomalies.

Number of patterns is defined in the “violet” line. In this example, value “2” means that there are two patterns. Description of anomalies is contained in the following blocks separated with stars (or any other symbols). Each block consists of three lines. First line in the block (red) indicates the name of file with polygon coordinates which is located in a subfolder “forms”. Second line (blue) indicates the area for the polygon. In case of all zero values, the coordinates from the file are used. Otherwise, the polygon is defined within the rectangle with coordinates indicated in this line (opposite corners of the rectangle: x1, y1, x2, y2).

Third line (green) is the amplitude of the anomaly, in percent, with respect to the basic absolute velocity model (7% and -7%).

An example description of one polygon (“anom2”) is presented below:

File 9: Example of polygon definition in file “1D+FREE1/forms/anom2.bln”:

```
21,1
-645.81105272, -1307.78987716
-222.685256517, -1270.0126832
-79.1245153868, -877.128602179
79.5478161811, -1118.90327544
457.338209103, -907.350357344
321.333895159, -544.688663426
585.787359794, -605.132173756
850.24082443, -778.907897893
820.017643553, -1217.12461166
804.906053115, -1768.6729073
585.787359794, -1413.56665217
631.122131109, -1882.0051211
351.557076035, -1557.11998921
336.445485597, -1874.44968231
170.21735881, -1542.00911162
19.1008224615, -1904.67143748
-33.7897440722, -1526.89823404
-373.801160899, -1821.56097884
-320.910594365, -1489.12104008
-736.480595349, -1700.67395818
-547.585082905, -1519.34279525
```

In this case the file is a standard output of the SURFER digitizing tool. First line indicates number of nodes (21; second number can be ignored). The following lines provide X - Y coordinates of nodes in the polygons. Note that the polygon is not necessarily closed. The first node is automatically joined with the last node.

An example of anomaly definition using File 8 is shown in Figure 5.7.

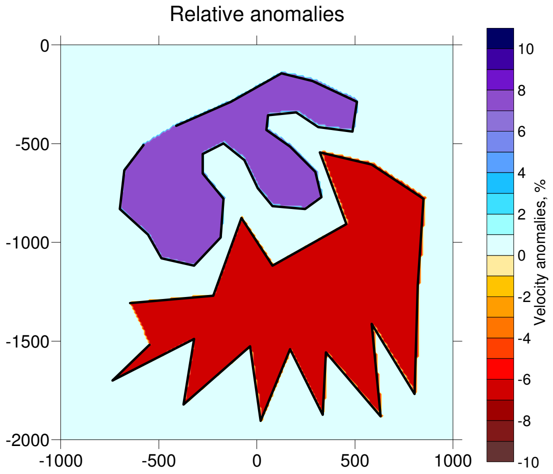


Figure 5.7. Example of definition of anomalies with polygons according to the File 7 (Model "1D+FREE1"). Black lines show polygons "anom1" and "anom2".

An example of creating a realistic synthetic model using this algorithm for the considered dataset (HULA_p02) is shown in Figure 5.8. In this case the basic velocity is defined with the key 4 and anomalies use the key 2.

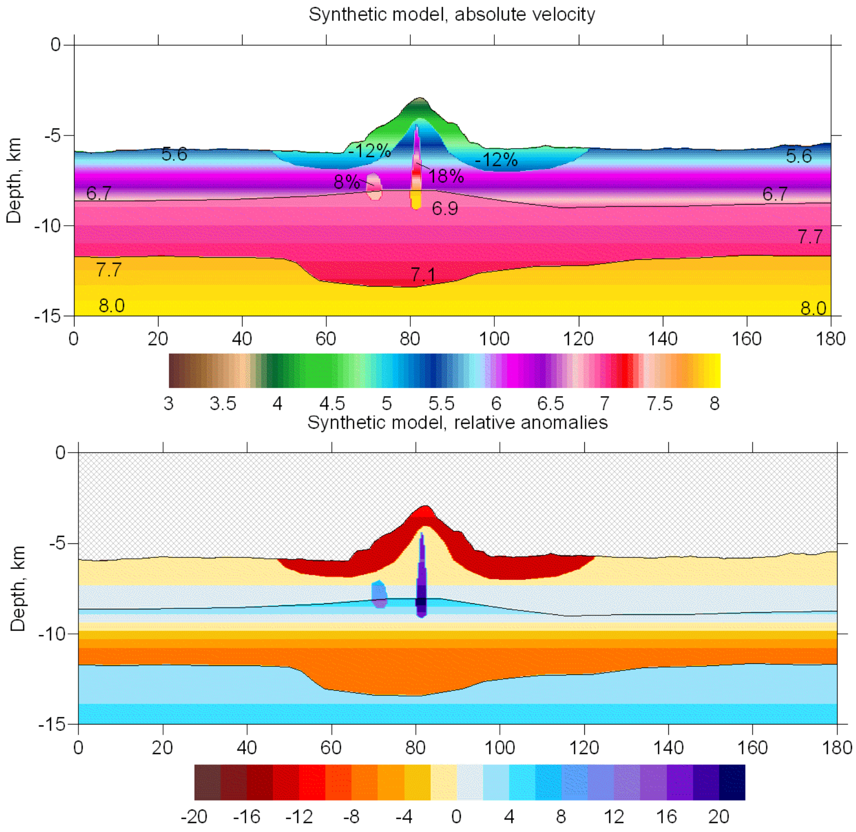


Figure 5.8. Absolute velocity (upper) and anomalies (lower) constructed using files `\DATA\HULA_p02\S1_V4_A2\velocity_syn.dat` and `\DATA\HULA_p02\S1_V4_A2\anomaly_syn.dat`.

5.3.4. INDEX “4”: Velocity anomalies defined with free-shaped polygons in selected areas

Case of “4” is the same as case “2”, but the anomalies are defined inside selected areas. This case is valid only if absolute velocities are defined with line (option “4”). The indicated value of anomaly is valid only in intersection of the anomaly and the selected area. A set of polygons for anomalies is defined in the file “anomaly_syn.dat” as shown in the following example:

File 10: Example of “LINE_FR1/anomaly_syn.dat” for the free-shaped anomalies in selected areas(index 4):

```
4       4: free-shaped inside selected areas
_______________________________________________
3       number of anomalies
*******************************
anom1   Figure
1       number of a layer
0. 0. 0. 0.
-5      Amplitude of isotropic anomaly, %
*******************************
anom1   Figure
2       number of a layer
0. 0. 0. 0.
-7      Amplitude of isotropic anomaly, %
*******************************
anom2   Figure
1       number of a layer
0. 0. 0. 0.
5       Amplitude of isotropic anomaly, %
```

First line (brown) is the index which indicates the type of the anomaly. “4” means that the following format corresponds to description of free-shaped anomalies in selected areas.

Number of patterns is defined in the “violet” line. In this example, value “3” means that there are three patterns. Description of anomalies is contained in the following blocks separated with stars (or any other symbols). Each block consists of four lines. First line in the block (red) indicates the name of file with polygon coordinates which is located in a subfolder “forms”. Second line (blue) indicates the number of area where this anomaly is valid (only in case if absolute velocity is defined with free anomalies or lines). Fourth line (green) is the amplitude of anomaly, in percent, with respect to the basic absolute velocity model.

Figure 5.9 shows an example of definition using this method based on File 10. The absolute velocity is defined using the line description using the key “4”. Two anomalies are defined as polygons as shown in left plot of Figure 5.9. Anomaly 1 is defined in both areas 1 and 2 and has the amplitudes of -5% and -7%, respectively. Anomaly 2 is defined only in Area 1 and has amplitude of +5%.

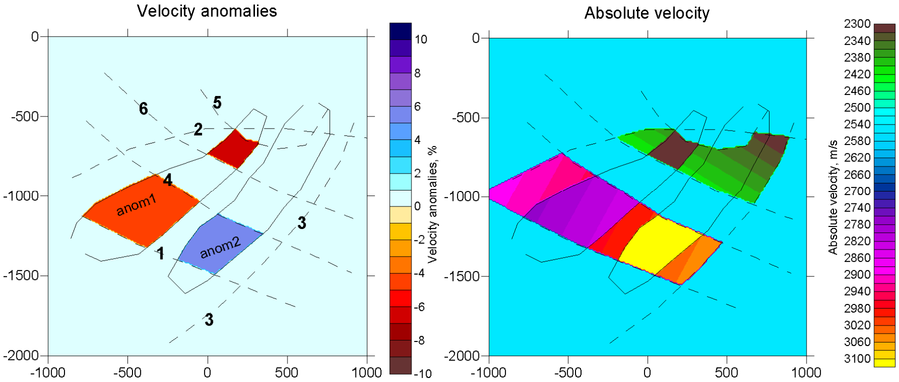


Figure 5.9. Scheme for definition of anomalies in selected areas according to File 11 (Model: "LINE_FR1"). Left is the relative velocity anomalies with scheme of lines for absolute velocity and relative anomaly definition; right is the resulting absolute velocity model.

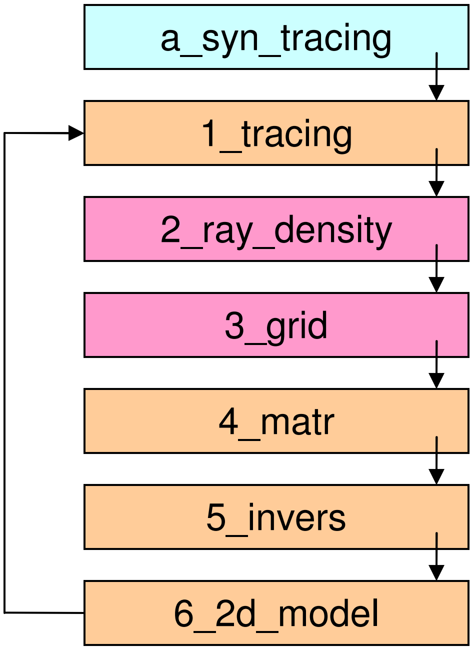
6. Synthetic modeling

6.1. General remarks

The *PROFIT* software provides a wide range of possibilities for performing various synthetic tests. The synthetic model is defined in two files according to the algorithms defined in Section 5:

1. File with description of absolute velocity:
`/DATA/'//ar//'/'//md//'/velocity_syn.dat`
2. File with description of anomalies with respect to the absolute velocity:
`/DATA/'//ar//'/'//md//'/anomaly_syn.dat`
where
`'//ar//'` is the area folder (e.g. "HULA_p02")
`'//md//'` is the model folder (e.g. "S1_V1_A1" or "S1_V4_A2")

Below is the block-scheme for performing the synthetic modeling



- Program for ray tracing in the synthetic model

- Program for ray tracing in a current velocity model

- Program for computing the ray density (only in 1st iteration)

- Program for constructing the parameterization grid (only in 1st iteration)

- Program for computing the matrix of first derivatives

- Program for matrix inversion

- Program for computing the velocity model in regular 2D grid which is used as basic model in the next iteration

The synthetic modeling starts with computing synthetic times (blue box). Then the full iterative procedure of data inversion is executed in the same way as in the case of observed data processing (Section 3). Programs indicated with pink are executed only during the first iteration.

6.2. Visualizing the starting or synthetic velocity models

When defining the starting or synthetic velocity models, it is recommended to control the definition using the programs in the project
`\PROG\a_define_model`

The names of the area and models are taken from file “/`model.dat`” in the root directory. The parameters for visualizing are defined in file “SET.DAT” located in the same directory as the program:

`\PROG\a_define_model\SET.DAT`

```
1     1: synthetic model; 2: starting velocity model
3           refining the visualization grid with respect to regular
```

It is recommended to refine the grid for presenting the synthetic model (3rd parameter) to avoid rough display of interfaces.

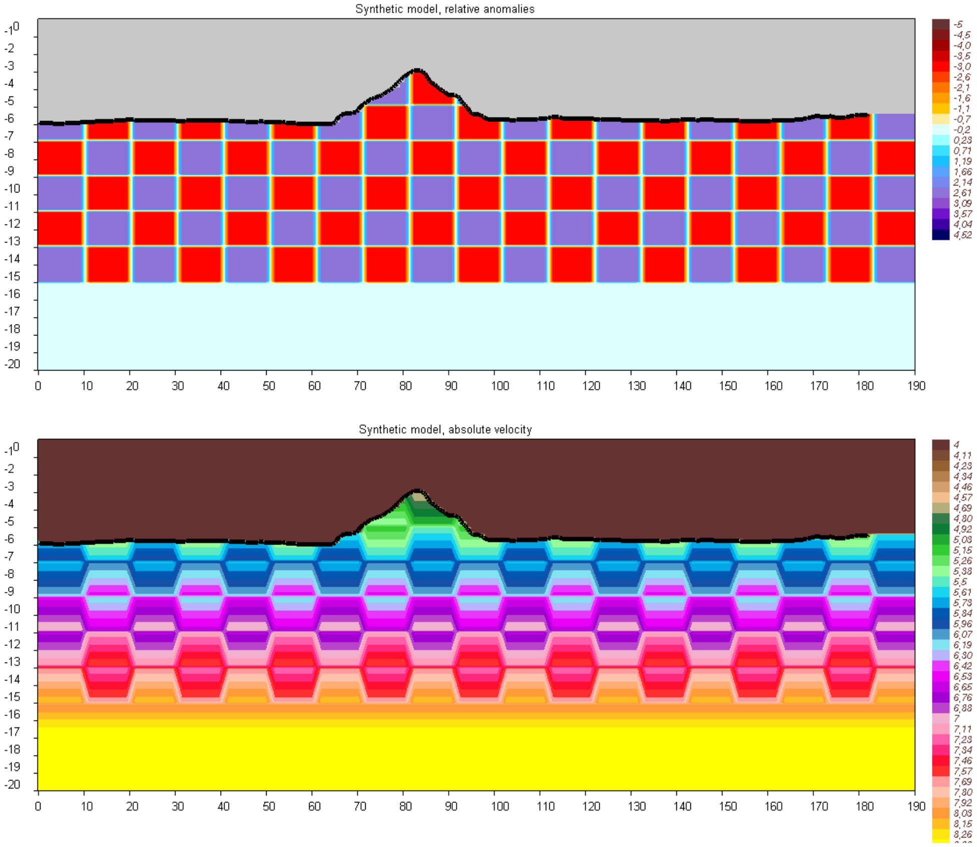


Figure 6.1. Picture in file /PICS/HUL1_p02/S1_V1_A1/syn_dv.png and /PICS/HULA_p02/S1_V1_A1/syn_abs.png which corresponds to synthetic velocity distribution in model HULA_p02/S1_V1_A1.

Other parameters for visualization are defined in the file presented below:
\DATA\HULA_p02\set_vis.dat

```
0 220 1.            x1,x2,dx for ray density calculation
-20 0 0.1           y1,y2,dy for ray density calculation
2 1.0               dxmax, dymax: max distance to node
```

The output of this program:

/FIG_files/syn_mod/dv_ini.grd: relative anomalies in percent with respect to model in vref_start.dat computed according to the information in “set_vis.dat“.**/FIG_files/syn_mod/abs_ini.grd:** absolute starting velocities. **/FIG_files/syn_mod/all_contours :** polylines used for definition of absolute velocity.

These files can be directly visualized in the Surfer Software as a contour line plot.

If the key in the file preview_key.txt is nonzero, the program produces the PNG files in the corresponding directory which can be used for previewing. More details about previewing can be found in Section 4.2.

Example of a preview file for model “HULA_p02/S1_V1_A1” is presented in Figure 6.1.

7. Running the code

7.1. Running the data inversion or synthetic modeling using “START” program

To perform a successful run of the *PROFIT* code, the data structure should be created as described in Section 2. The execution of each program can be implemented steps-by-step manually. For this, it is necessary to run an exe-file in a corresponding program folder. For example, to run the tracing program for the current iteration, one can execute the file `PROG/1_tracing/tracing.exe`.

However, the *PROFIT* code contains a program, which performs automatic managing of all steps for the cases of real data inversion and synthetic modeling. The source of this program is presented below:

Program for automatic managing of the *PROFIT* steps:
Program: `\PROG\START\start.f90`
(the executable program steps are highlighted in red)

```
USE DFPORT
character*8 md,ar,line,md_all(200),ar_all(200)
integer it_all(200)
character*40 line40(10)

nmodel=0
open(1, file='../../model_all.dat')
read(1,*)
read(1,*)
2       read(1,'(a8,1x,a8)',end=1) ar,md
        nmodel=nmodel+1
        ar_all(nmodel)=ar
        md_all(nmodel)=md
        write(*,*)nmodel,' area=',ar,' model=',md
        goto 2
1 close(1)

i=system('mkdir ..\..\tmp')
i=system('mkdir ..\..\FIG_files\rays')
i=system('mkdir ..\..\FIG_files\result')
i=system('mkdir ..\..\FIG_files\syn_mod')
i=system('mkdir ..\..\FIG_files\stat')

do imodel=1,nmodel
        ar=ar_all(imodel)
        md=md_all(imodel)

        open(1,file='../../data/'//ar//'/'//md//'/MAJOR_PARAM.DAT')
        do i=1,10000
                read(1,'(a8)',end=563)line
                if(line.eq.'GENERAL ') goto 564
        end do
        563 continue
        write(*,*)' cannot find GENERAL INFORMATION in MAJOR_PARAM.DAT!!!'
        pause
        564 continue
        read(1,*)key_1real_2syn
        read(1,*)niter
        close(1)

        if(key_1real_2syn.eq.1) write(*,*)' REAL DATA INVERSION, iterations:',niter
        if(key_1real_2syn.eq.2) write(*,*)' SYNTHETIC MODELING, iterations:',niter
        open(11,file='../../model.dat')
        write(11,'(a8)')ar
        write(11,'(a8)')md
        close(11)
        write(*,*)' model:',md
        write(*,*)' iterations:',niter
        i=system('mkdir ..\..\DATA\'//ar//'\'//md//'\DATA')

        open(1,file='../../PROG/a_define_model/SET.DAT')
        read(1,'(i1,a40)')i,line40(1)
        read(1,'(i1,a40)')i_small,line40(2)
        close(1)

        if(key_1real_2syn.eq.2) then
                open(11,file='../../PROG/a_define_model/SET.DAT')
                write(11,'(i1,a40)')2,line40(1)
                write(11,'(i1,a40)')i_small,line40(2)
                close(11)

                write(*,*)'     Visualizing synthetic model: '
                i=system('..\a_define_model\define_model.exe')

                write(*,*)'     Tracing synthetic rays: '
                i=system('..\a_syn_tracing\syn_tracing.exe')
        end if
        open(11,file='../../PROG/a_define_model/SET.DAT')
        write(11,'(i1,a40)')1,line40(1)
        write(11,'(i1,a40)')i_small,line40(2)
        close(11)

        write(*,*)'     Visualizing Starting model: '
        i=system('..\a_define_model\define_model.exe')

        do iter=1,niter
                write(*,*)' ITERATION:',iter
                open(11,file='../../model.dat')
                write(11,'(a8)')ar
                write(11,'(a8)')md
                write(11,'(i2)')iter
                close(11)
                write(*,*)'     Tracing direct rays: iter=',iter
                i=system('..\1_tracing\tracing.exe')

                if(iter.eq.1) then
                        write(*,*)'     Compute the ray density'
                        i=system('..\2_ray_density\ray_dens.exe')

                        write(*,*)'     Compute the parameterization grid:'
                        i=system('..\3_grid\grid.exe')
                end if

                write(*,*)'     Compute the matrix, iter=',iter
                i=system('..\4_matr\matr.exe')

                write(*,*)'     Perform the inversion '
                i=system('..\5_invers\invers.exe')

                write(*,*)'     Computing the velocity model in a regular grid '
                i=system('..\6_2d_model\model_2d.exe')

                write(*,*)'     Visualizing the results after current iteration '
                i=system('..\vis_result\vis_result.exe')
        end do
        write(*,*)' Prepare report about variance reduction and travel times '
        i=system('..\var_reduct\resid_norm.exe')
end do

stop
end
```

This program allows running all the *PROFIT* steps for one or several models. The list of models is defined in file "/model_all.dat". An example of this file is presented below:

/model_all.dat

```
1: name of the area
2: name of the model
HUL1_p02 R1_V1_A0
HUL1_p02 S1_V4_A4
COMPLEX_ R1_V6_A0
JAVA_P18 R1_V1_A0
```

In the presented example, three models are defined. Two of them are from AREA folder "HULA_P02", and the other two are from "COMPLEX_" and "JAVA_P18" AREAS, indicated in the 1st column. For all areas, the names of the models are: **"**R1_V1_A0**"**, **"**R1_V1_A0**"**, and **"**model_01**"**, that is indicated in the 2nd column. It is important to define all the parameters in the file "all_areas.dat" according to a fixed format: (a8,1x,a8) and they should start from line 3. Any number of different models up to 200 can be defined. They will run successively one after another.

A scenario of real data inversion or synthetic modeling and number of iteration is defined in MAJOR_PARAM.DAT in a rubric "GENERAL INFORMATION" as shown below.

```
    *************************************************************
    GENERAL INFORMATION :
    1   KEY 1: REAL; KEY 2: SYNTHETIC
    6   Number of iterations
```

Any number of iterations from 1 to 99 is allowed.

7.2. Running the data inversion using the BATCH file

The easiest way to run the data inversion is to start the BATCH file START.BAT, which is located in the root directory. This file runs the start.exe described in the previous section. Before running this file it is necessary to organize the file structure as described in Section 2 and define the names of areas and models in file model_all.dat to be computed.

File: \START.BAT

```
cd PROG
cd START
start.exe
pause
```

8. Closing remarks

This manual presents the *PROFIT* code based mainly on marine profiling data in the Pacific Ocean. At the same time, the *PROFIT* code is rather universal, and the same version can be applied on scales from dozens meters to hundreds of kilometers for marine, amphibious, and land data. Besides the deep seismic sounding marine data presented in this manual, many other datasets have already been processed using the *PROFIT* code, such as amphibious data in subduction zones in Chile and Central Java (papers in preparation), profiles in the Precaspian province aimed at detecting salt domes, some engineering profiles of ~1 km length with the purpose of planning tunnel constructions through topography, and numerous other applied and fundamental work. Further development of the *PROFIT* code is planned. In particular, we are working on including reflected and head waves in addition to the first arrivals. These data will be used for simultaneous inversion of velocity structures and geometry of interfaces.

Some examples of dataset processing are presented in file “**data_examples.pdf**”

Detailed instruction how to start working with the code are given in the text file “**to_get_started.pdf**”.

We wish you a successful application of the *PROFIT* code and bright results. We would appreciate any help and suggestions on improving the code. In case of any inconsistencies and errors, please address the author, Ivan Koulakov (KoulakovIY@ipgg.nsc.ru). We are planning to prepare new versions of the *PROFIT* code with friendlier interface.

Many thanks to **Heidrun Kopp, Kirill Gadylshin** and **Tatyana Stupina** for taking part at preparing the code, datasets and the code description.

This study is supported by Russian Foundation for Basic Researches (08-05-00276-a), Heimholtz Society and RFBR joint research project 09-05-91321-SIG_a, multidisciplinary projects SB RAS 44, 21 and SB-UrO-DVO RAS 96, and project ONZ RAS 7.4.